Oracle FLEXCUBE Direct Banking

Android Client Developer Guide
Release 12.0.3.0.0

Part No. E52543-01

April 2014

ORACLE®

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway

Goregaon (East)

Mumbai, Maharashtra 400 063

India

Worldwide Inquiries:

Phone: +91 22 6718 3000

Fax:+91 22 6718 3001

www.oracle.com/financialservices/

Table of Contents

1. **Preface**

Intended Audience

Any interested party working on the delivery of Oracle FLEXCUBE Direct Banking may read this document. The following profile of users would find this document useful:

- Application Architects
- End to End Designers
- Business Service Detailed Designers and Developers
- Implementation Partners

Specifically, however, this document is targeted at:
Implementation Partners, Customization Development Teams or Vendors providing customization, configuration and implementation services around the Oracle FLEXCUBE Direct Banking product.

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 OFSS Support

https://support.us.oracle.com

2. Introduction

FLEXCUBE Direct Banking mobile client applications are supposed to be designed in order to understand this XML structure and render it on the mobile device screen appropriately. This stands as the basic requirement for Android client as well. The Android mobile client is a client application specifically targeted for Google's Android OS platform. This document is a generic guide for development of Android mobile client.

FLEXCUBE Direct Banking's mLEAP framework is the entity responsible for generating the content for mobile clients. This content is represented in a pre-defined XML format. For details on the XML structure, please refer to the document Oracle_FLEXCUBE_Direct_Banking_Mobile_App_XML_structure.docx.

Some key points about developing an application on Android platform are highlighted below:

- Android applications are written in the Java programming language.
- The compiled Java code along with any data and resource files required by the application — is bundled by the Android Asset Packaging Tool into an Android package, an archive file marked by an .apk suffix.
- This file is the vehicle for distributing the application and installing it on mobile devices; it's the file users download to their devices.
- All the code in a single APK file is considered to be one application.
- The APK file is signed with a certificate before deployment.
- The client receives the response from the server in XML and parses it to render the screen.
- The screen rendering logic is written in client who majorly uses components of the Android SDK framework.

3. Acronyms and Abbreviations

FCDB	FLEXCUBE Direct Banking
AVD	Android Virtual Device
IDE	Integrated Development Environment
SDK	Software Development Kit
ADT	Android Development Tools
UI	User Interface
APK	Android Package

4. Scope

This document speaks about various design and development level considerations for the FLEXCUBE Direct Banking mobile client for Android platform. It describes the high level design, process flow, various key components involved, screen layout design etc. This mobile client is targeted at Google's Android platform. Some of the terms used in this document are specific to the Android platform. This document does not go in detail of the Android platform features but speaks only with relevance to the FLEXCUBE Direct Banking mobile client architecture. Reference sites for Android are mentioned wherever applicable.

1. Eclipse IDE Setup

1.1 ANDROID SDK ECLIPSE PLUGIN

Android Development Tools (ADT) is a plugin for the Eclipse IDE which is an integrated environment to build Android applications. Before you can install or use ADT, you must have a compatible version of Eclipse installed on your development computer. For this purpose, Eclipse 3.4 (Ganymede) or greater is required.

1.2 DOWNLOADING THE SDK STARTER PACKAGE

The SDK starter package is not a full development environment—it includes only the core SDK Tools, which you can use to download the rest of the SDK components (such as the latest Android platform). The starter package can be obtained from:

http://developer.android.com/sdk/index.html

Make a note of the name and location of the SDK directory on your system—you will need to refer to the SDK directory later, when setting up the ADT plugin and when using the SDK tools from command line.

1.3 INSTALLING THE ADT PLUGIN FOR ECLIPSE

Use Update Manager feature of your Eclipse installation to install the latest revision of ADT on your development computer. The ADT Plugin URL location is:

https://dl-ssl.google.com/android/eclipse/

Note: If you have trouble acquiring the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons).

If you are having trouble downloading the ADT plugin because of the network firewall, you can configure proxy information from the main Eclipse menu:
Window > Preferences > General > Network Connections

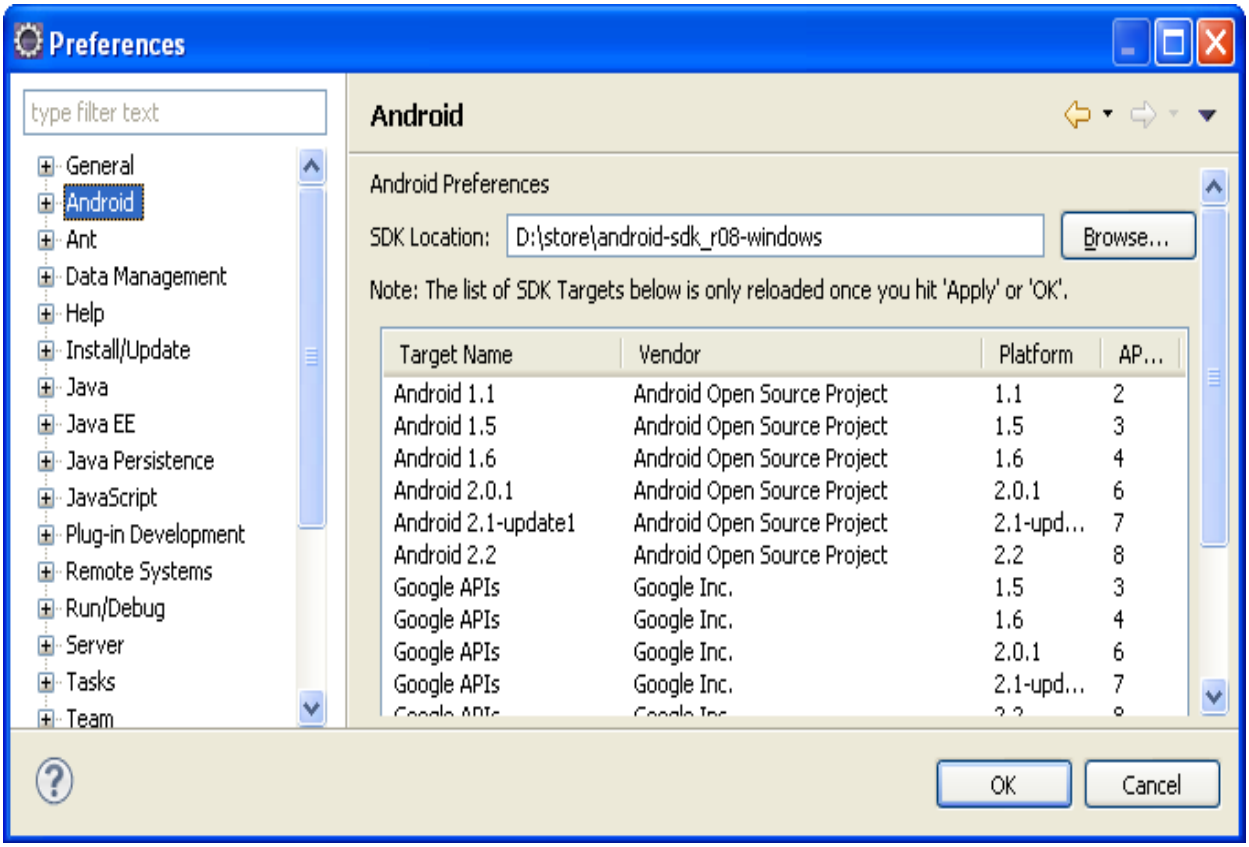
For further information kindly use the following link:

http://developer.android.com/sdk/eclipse-adt.html#installing

1.4 CONFIGURING THE ADT PLUGIN IN ECLIPSE

Once you've successfully downloaded ADT as described above, the next step is to modify your ADT preferences in Eclipse to point to the Android SDK directory:

- In Eclipse menu bar, select **Window > Preferences** to open the Preferences panel.
- Select Android from the left panel.
- For the SDK Location in the main panel, click the Browse button and locate your downloaded SDK directory.
- Click the OK button.
- Eclipse restart is required.



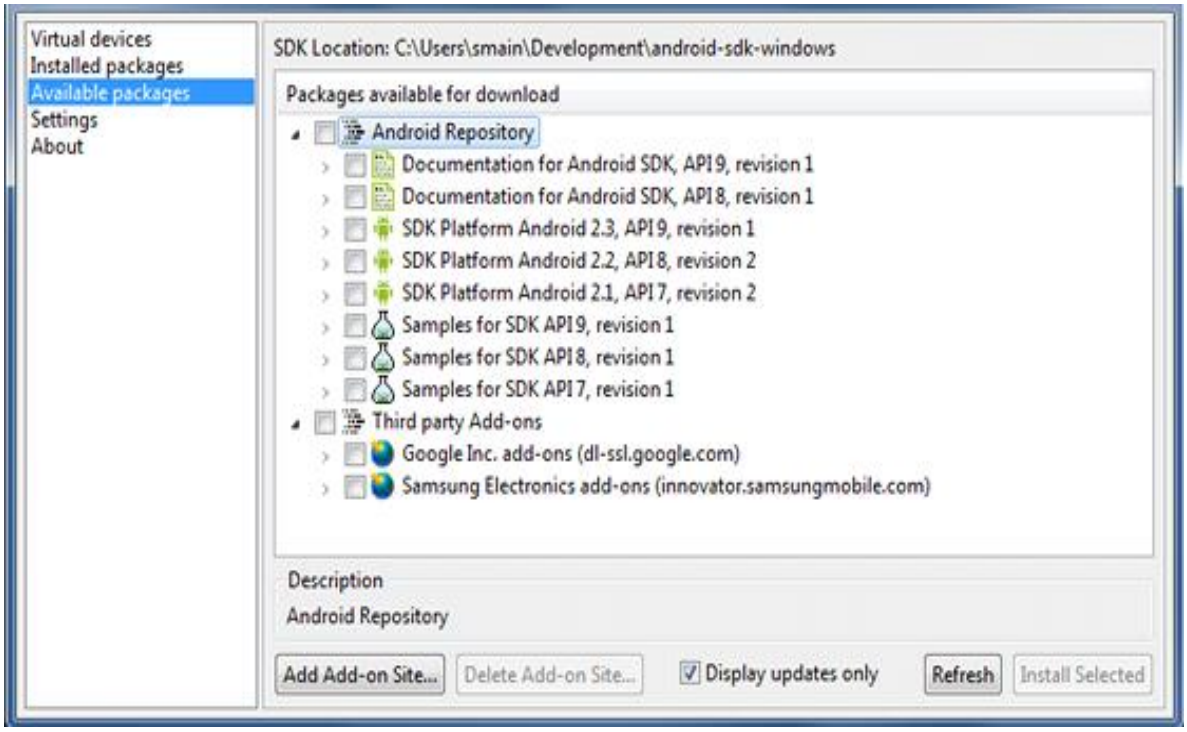
1.5 ADDING PLATFORMS AND OTHER IN ECLIPSE

The last step in setting up your SDK is using the **Android SDK and AVD Manager** (a tool included in the SDK starter package) to download essential SDK components into your development environment.

You can start the Android SDK and AVD Manager in one of the following ways:

- From within Eclipse menu, select **Window > Android SDK and AVD Manager**.
- Run **android.bat** in **<Android SDK>\tools** folder.

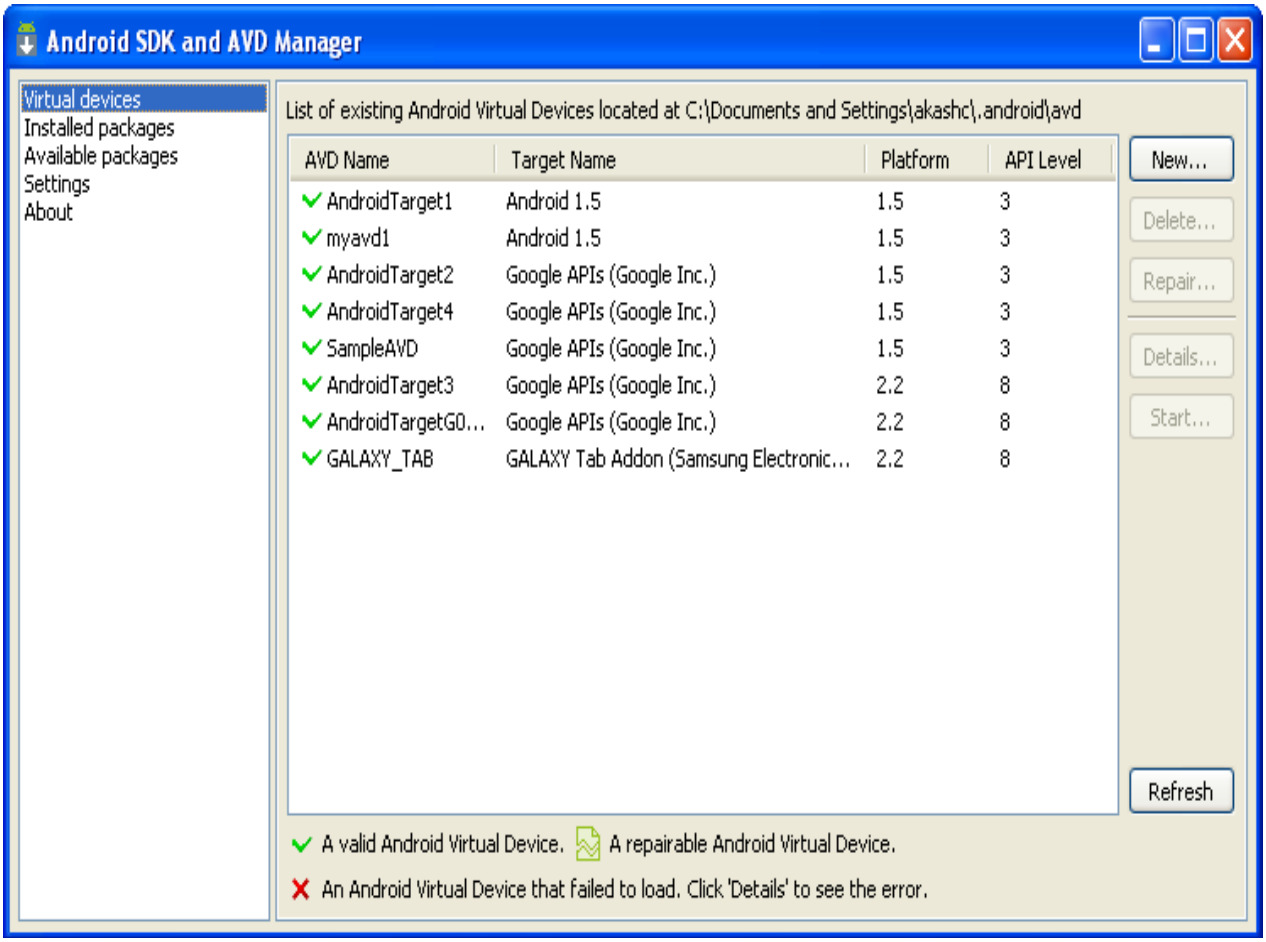
To download components, use the graphical UI of the Android SDK and AVD Manager to browse the SDK repository and select new or updated components.



Note: Kindly note that the **Android Repository** node must have an API level 3 and above.
Also, **Third party Add-ons** node should have a Google Inc. component, as this will be required for Google Map support.
If either of the above is missing, then it must be downloaded using **section 1.3** of this document.

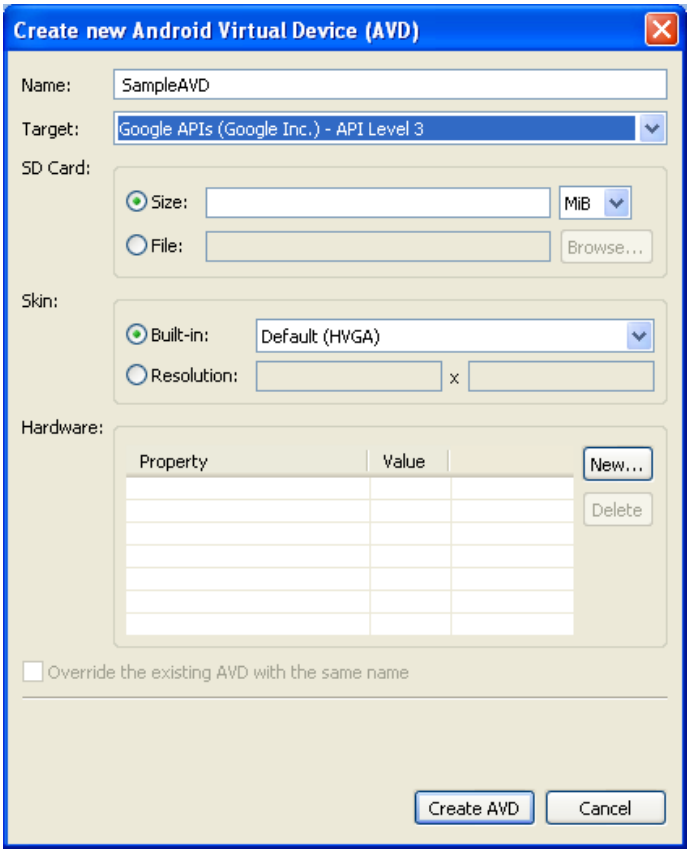
1.6 CREATING A NEW AVD

- Goto the “tools” folder in Android SDK
  e.g. D:\store\android-sdk_r08-windows\tools
- Run “android.bat”.
- “Android SDK and AVD Manager” will open in a new window

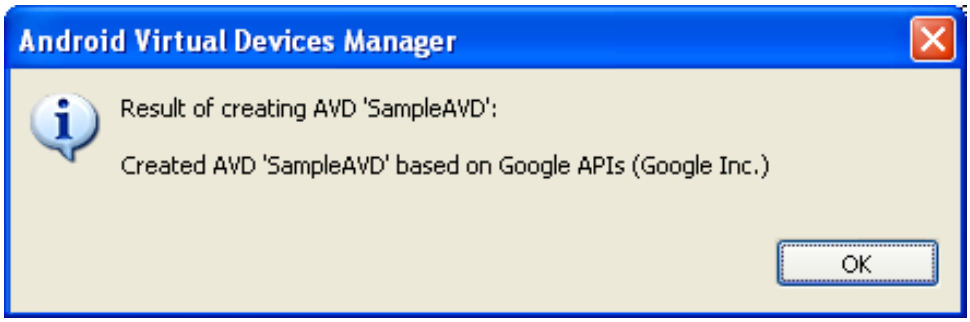


- From the panel on the left side, select “Virtual Devices”.
- A list of previously created AVDs will be displayed.
- From the right side of the window, click on “New” button.
- A new window will appear.
- Enter the name for the new AVD, e.g. “SampleAVD”.
- Select the “Target” as “Google APIs (Google Inc.) – API Level 3”

  Note: Kindly note that the API level can be 3 and above, but the API should be of Google and not any other, as this will be required for Google Map support



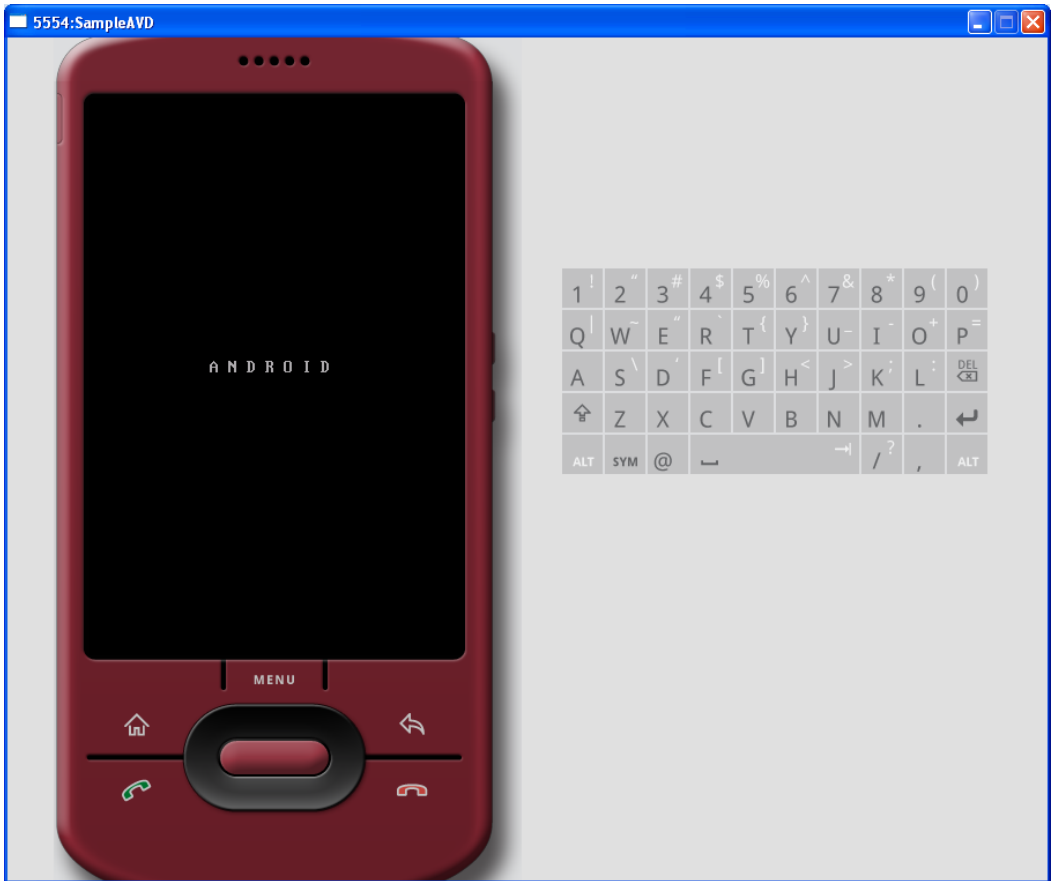
- Click on “Create AVD” button. A message box will be shown confirming the successful creation of the AVD.



- Close the “Android SDK and AVD Manager”.

1.7 RUNNING AVD

- Open a command prompt and goto “tools” folder in Android SDK
  e.g. D:\store\android-sdk_r08-windows\tools
- Run the following command “**emulator -avd SampleAVD”**
  where “SampleAVD” is the name of the AVD created.
- This will start the AVD:



1.8 INSTALL APPLICATION ON AVD

- Open a command prompt and goto “tools” folder in Android SDK
  e.g. D:\store\android-sdk_r08-windows\tools
- Run the following command:
  If “DIRECT_BANKING.apk” is in the "tools" folder:
  D:\store\android-sdk_r08-windows\tools>adb install DIRECT_BANKING.apk

  If the DIRECT_BANKING.apk is in the "D:\FCDB\" folder:
  D:\store\android-sdk_r08-windows\tools>adb install D:\FCDB\DIRECT_BANKING.apk
  (Where "DIRECT_BANKING.apk" is the name of our application file.)

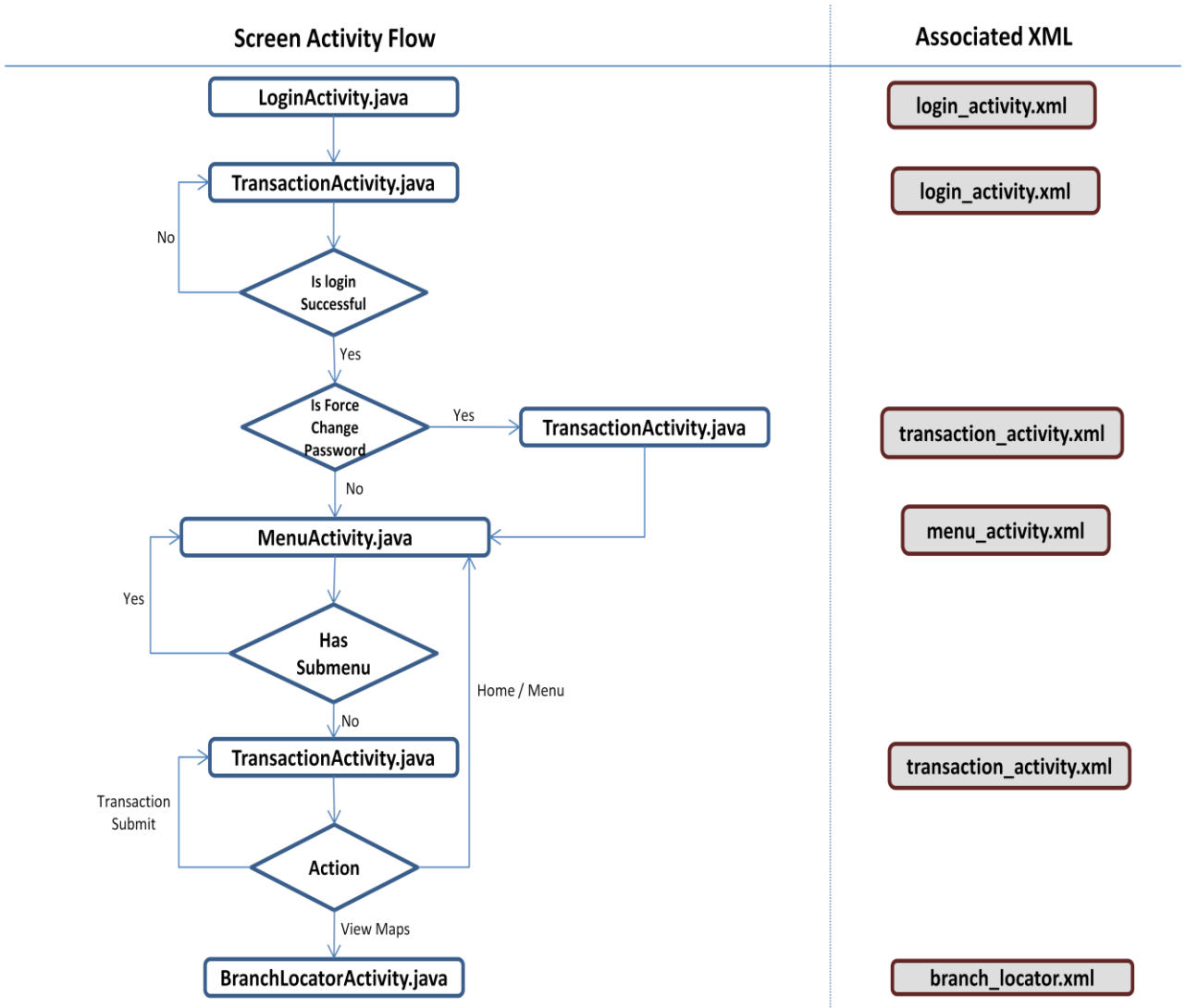
- If an earlier version of "DIRECT_BANKING.apk" exists in the AVD, then before installing the new version, the older "DIRECT_BANKING.apk" will have to be uninstalled

2. High Level Design

For the ease of development, the classes in FCDB Android application have been categorized into 2 groups:

- Activity
  - All classes that define the on screen activity for Android is termed as an Activity.
  - An activity is used to create a window to place your UI elements like buttons, text box etc.
  - Each activity is supported by an associated layout XML file.
- Helper classes.
  - Classes which support the Activity classes from background

Various key components involved are outlined below:



2.1 FAVORITE-TRANSACTION BAR

Favorite-Transaction activity is enabled for the user if `FAVENABLED flag in customproperties.txt file is set to 'Y'.`

- The **Favorite-Transaction bar** is placed at the bottom of the screen(IF MENUSTLE='L' in `customproperties.txt`).
  - This UI component has been added in both menu_activity.xml and the transaction_activity.xml and controlled through the respective classes.
  - If a transaction available in a role related to application based mobile banking is mapped as favorite for the user, then it will be visible across the application after successful login.
- Favorite button is provided on first screen of transaction.Transaction can be added or removed from favorite transactions on clicking that button.'RRUPS12' Is fired on clicking Favorite button.

- The common dropdown UI components in are replaced with **Spinners** in Android applications.
  - When a spinner component is clicked, then another UI view is invoked.
  - This new list view is modal.
  - The user can then select a value from this list view, which is then passed to the spinner in the main view.

2.2 DATEPICKER WIDGET

- **Datepicker widget** is used to add calendar control.
  - The calendar icon is placed next to the textbox, which when clicked calls the widget.
  - The date can be set manually also by direct input in the textbox.

2.3 ATM/BRANCH LOCATER

- The **ATM/Branch Locater** is added to the android application.
  - **BranchLocatorActivity.java** and **branch_locator.xml** are used to render a map based on the Google Map API.
  - As Android supports reverse geocoding, the coordinates (latitude & longitude) are passed to the BranchLocatorActivity.java class to put the place mark at the exact location.
  - This class is currently invoked for RRMAP01 requestID only.
  - Red and green place marks are rendered for ATM and branch respectively.

2.4 TRANSACTION LEVEL HELP

- **Transaction level help** support is also available for every screen. If FLEXCUBE Direct Banking is sending across any information / help for given request, then the same would be shown on screen. In this case, a Help button appears on the screen, clicking on which help content is displayed using a slider.

3. System Overview

3.1 CLASS DIAGRAM

3.2 SCREEN LAYOUT DESIGN

In Android, the screen can be defined in two ways:

- **Declare UI elements in XML**. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.
- **Instantiate layout elements at runtime**. Application can create View and ViewGroup objects (and manipulate their properties) programmatically.

The FLEXCUBE Direct Banking mobile client uses both of these methods for declaring and managing application's UI. For example, declare application's default layouts in XML, including the screen elements that will appear in them and their properties. Then add code in the application that would modify the state of the screen objects, including those declared in XML, at run time.

The advantage to declaring your UI in XML is that it enables us to better separate the presentation of application from the code that controls its behavior. The UI descriptions remain external to the application code, which means that they can be modified or having to modify source code and recompile. Additionally, declaring the layout in XML makes it easier to visualize the structure of UI, so it's easier to debug problems.

More details on the UI design from Google can be found at:
http://developer.android.com/guide/topics/ui/declaring-layout.html

3.3 INTEGRATION WITH MLEAP

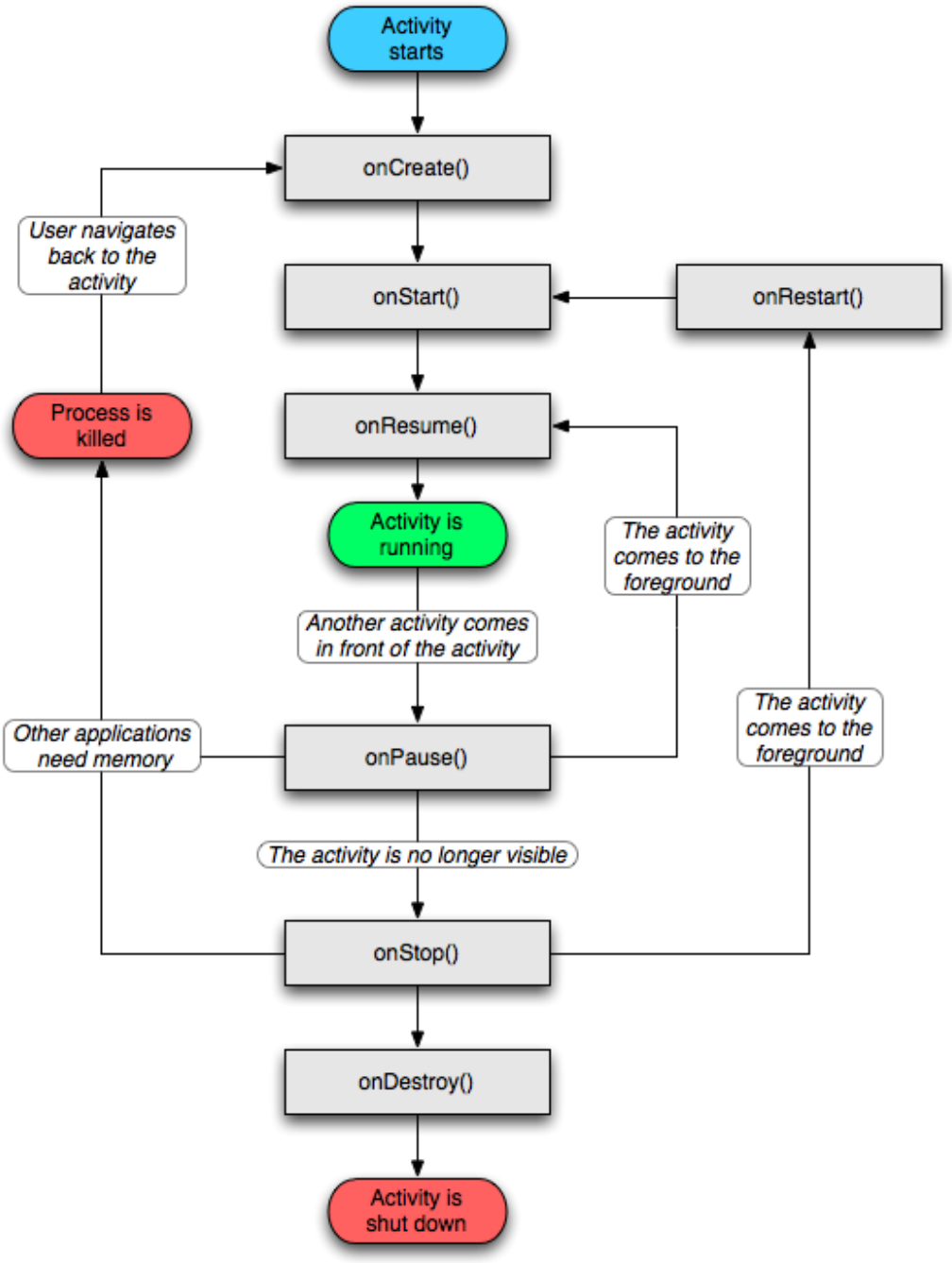
The client is designed to understand the XML tags generated by mLEAP and render them appropriately on the screen. Please refer to the document Oracle_FLEXCUBE_Direct_Banking_Mobile_App_XML_structure.docx for list of supported tags and their meanings.

4. Activity Class Overview

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI components.

Note: Each activity class has to be manually in the manifest file.

4.1 ACTIVITY LIFECYCLE



4.2 COMMON METHODS USED IN FCDB ANDROID CLIENT ACTIVITY

Method Name	Return Type	Parameters Type	Description
onCreate ( )	Void	• (android.os.Bundle) p_savedInstanceState	•Overridden method where you initialize your activity. •Most importantly, here you will usually call setContentView(int) with a layout resource defining your UI. •You can retrieve various XML layouts and other resources or add them programmatically to paint the screen UI.
onCreateOptionsMenu ( )	boolean	• (android.view.Menu) p_menu	•Adds Option Menu items buttons using FCDB global instance hash tables and custom property file.
onMenuItemSelected ( )	boolean	• (Int) p_featureId • (android.view.MenuItem) p_item	•This method acts as an action listener for menu items.
showError ( )	Void		•Picks up error messages from the global instance and paints them in a dialog box. •The dialog box title and button value can be modified from the custom property files.
onStop ( )	Void		•Overridden method which forcefully stops the current activity and its super class. •This method is added to implement the exit logic for the application.

4.3 LOGINACTIVITY.JAVA

- Initiates the application.
- UI layout file associate with LoginActivity.java is login_activity.xml.
- LoginActivity also creates a global instance for the application which contains all the runtime variables.
- LoginActivity populates all the parameters required for FCDB and sends a prepare login request to the server and calls TransactionActivity.

4.4 TRANSACTIONACTIVITY.JAVA

- It receives the response XML from the server and parses it.
- Paints the UI based on the response tags.
  - The UI layout file used during login for TransactionActivity is login_activity.xml.
  - After a successful login, only transaction_activity.xml is used as the new UI layout file for TransactionActivity.java.
- Other than the specific menu requests, TransactionActivity is recursively called.

4.5 MENUACTIVITY.JAVA

- Once the user successfully logins into the application, the MenuActivity is called, which uses menu_activity.xml.
- MenuActivity.java is invoked for RRMNU00 and RRMNU01 only.
- Currently supports 3 level menu sub grouping.
- There are two types of menu style supported 'L' and 'G'.'L' is for giving list style menu and 'G' is for grid style menu.These values are stored in custom properties.txt under `MENUTYPE tag`.
- Contains an inner class EfficientAdapter
  - EfficientAdapter extends BaseAdapter.
  - Populates the menu list items into the listview.

4.6 BRANCHLOCATORACTIVITY.JAVA

- The sole purpose of this class is to populate Google maps based on the data received from the FCDB branch locator service.
- BranchLocatorActivity contains an inner classe MapItemizedOverlay.
  - MapItemizedOverlay class is used to populate multiple map markers on the map.
  - This inner class also contains the logic of obtaining screen coordinates of the location during an onTap event and showing the predefined address of the touched marker.
- The Google Maps service requires that each MapView identify itself to the service using a Maps API Key.
- **The generation of API key requires a google account.**
- Before providing Maps tiles to a MapView, the service checks the Maps API Key supplied by the MapView to ensure that it:
  - References a certificate/developer registered with the service, and
  - References a certificate that matches the certificate with which the application (containing the MapView) was signed.
- To register for a Maps API Key, you need to provide an MD5 fingerprint of the certificate that you will use to sign your application.
- When you run the application in emulator through Eclipse, the Android SDK build tool automatically signs the APK using the default debug certificate.
- Hence, we use the same keystore's certificate for Google maps also.
- To generate an MD5 fingerprint of the debug certificate, first locate the debug keystore. By default, build tools create the debug keystore in the active AVD directory. The location of the AVD directories varies by platform:
  - Windows Vista: C:\Users\<user>\.android\debug.keystore
  - Windows XP: C:\Documents and Settings\<user>\.android\debug.keystore
- An example of a Keytool command that generates an MD5 certificate fingerprint for the key alias_name in the keystore my-release-key.keystore:
  keytool -list -alias alias_name -keystore my-release-key.keystore

- Keytool will prompt you to enter passwords for the keystore and key. As output of the command, Keytool prints the fingerprint to the shell. E.g:
  Certificate fingerprint (MD5): 94:1E:43:49:87:73:BB:E6:A6:88:D7:20:F1:8E:B5:98

- When you are ready to register for a Maps API Key, load this page in a browser: http://code.google. com/android/maps-api-signup.html

- To register for a Maps API Key, follow these steps:

  - If you don't have a Google account, use the link on the page to set one up.
  - Read the Android Maps API Terms of Service carefully. If you agree to the terms, indicate so using the checkbox on the screen.
  - Paste the MD5 certificate fingerprint of the certificate that you are registering into the appropriate form field.
  - Click "Generate API Key"
- Place the Google API key in the Google map's layout XML. Eg.

```
<com.google.android.maps.MapView
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:enabled="true"
 android:clickable="true"
 android:apiKey="Place your Google Maps API Key String here" />
```

- Make sure that you added a <uses-library> element referencing the external com.google.android.maps library in the application's manifest. Also, check the permissions granted to the application are added in the manifest. These element must be a child of the <application> element. E.g.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.package.name">
 ...
 <application android:name="MyApplication" >
   <uses-library android:name="com.google.android.maps" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
 ...
 </application>
```

5. Helper Classes

5.1 GLOBAL.JAVA

- This class is used to store all global level parameters.
- Contains methods to set, get and clear the global instance.

5.2 HTTPWORKER.JAVA

- This class is used to establish connection with the server.

5.3 APPHELPER.JAVA

- This class is used to maintain various methods which can be used across all activities.
- **doClientSideValidation**(String p_requestId) is one such method, which has been hard coded to perform validation of password fields during 'Force Change Password' transaction.

6. FEATURES

6.1 LANGUAGE PROPERTIES FILE FOR LABELS AND ERROR MESSAGES

- Certain labels like Menu, Exit, Submit, any error messages are hardcoded in the application. These are loaded from the properties file strings.xml in the values folder and a values folder is made for each language support. In each string file FCDB_LANGID value should be set for that language. Example: for French language it should be FCDB_LANGID=fre.

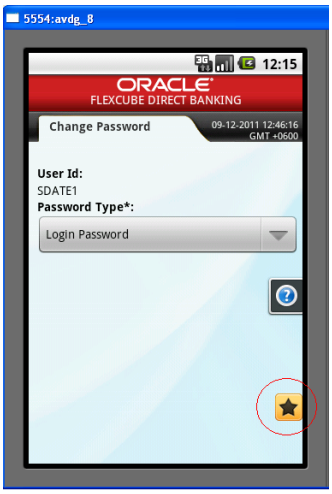
6.2 STANDBY MODE

- If user is using this application and an incoming call comes then the application will go in standby mode.

6.3 ADD/REMOVE FAVORITES

Favorite transactions can be added/removed from the client side itself. This can be done by setting FAVENABLED=Y in customproperties.xml. The facility to add remove favorite transactions is added by placing a star icon on the first screen of the transaction. If the idrequest currently painted is part of menu, then on that screen, the star icon appears. If the transaction is already there in favorites, then green star is shown. If the star is clicked the transaction is removed from favorites. Else to add a txn to favorites black star is shown.
On clicking of the star and idrequest is fired. Currently it is RRUPS12 is fired. This idrequest is configurable in app.plist file. Along this, another parameter fldsetfav is passed with value Y/N to add/remove transactions respectively. This parameter is in app.plist file.
The response of this request should be as below –

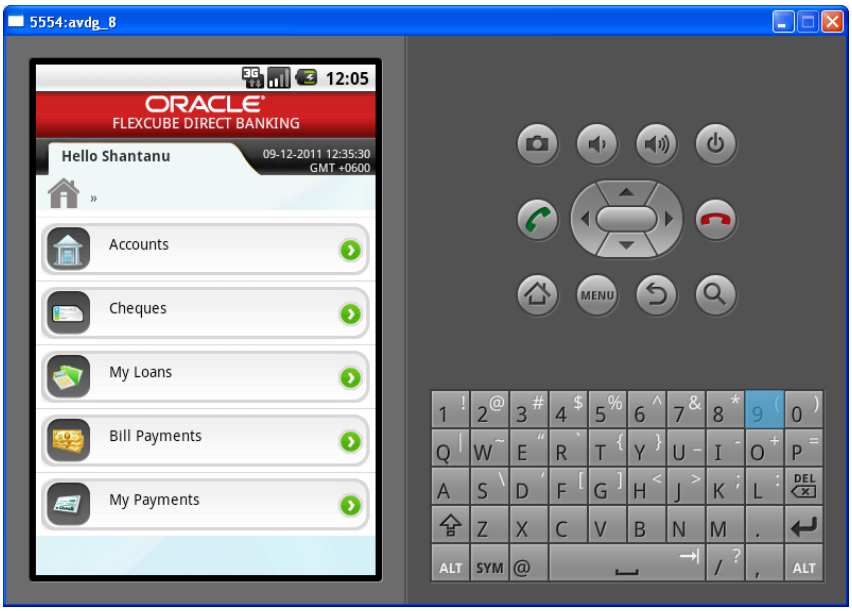
```
<?xml version="1.0" encoding="UTF-8"?>
<F xmlns:str="http://exslt.org/strings">
<H ln="" fn="" tz="14-06-2011 10:46:05" t="0"/>
<B t="e"/>
<M l='Transaction added to favorites' t='s'/>
</F>
```



6.4 MENUSTYLES

This version supports two menu styles. L for list and G for grid as specified in custom properties.xml

MENUSTYLE=L

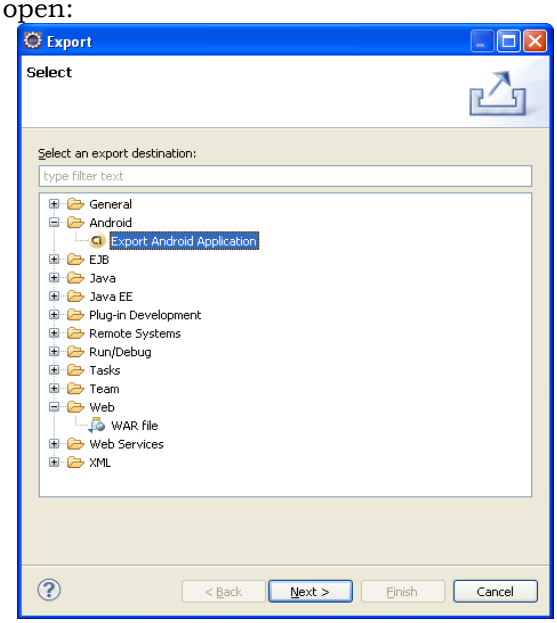


MENUSTYLE=G

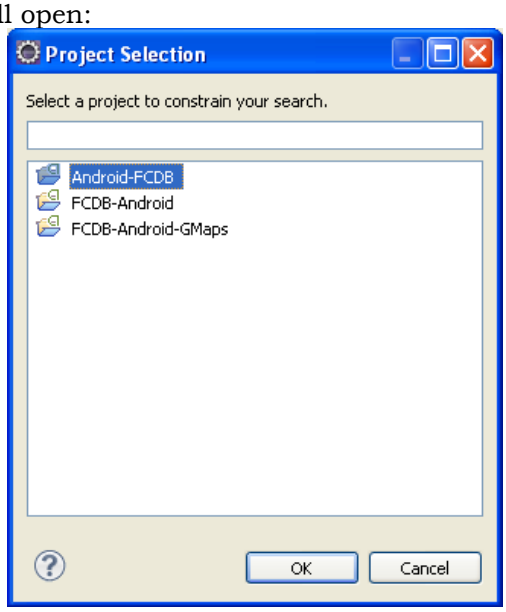
7. Exporting Final Output APK File

7.1 EXPORT WIZARD

- In Eclipse menu, go to **File > Export**.
- A new window will open:

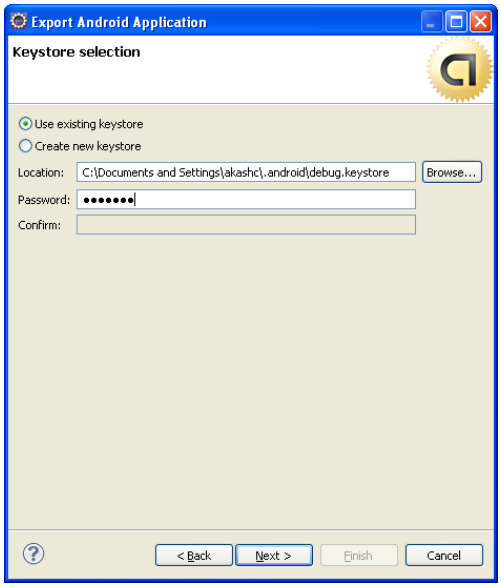


- Select **Android > Export Android Application**.
- Click on 'Next >'.
- In the next screen, click on 'Browse'.
- A new window will open:

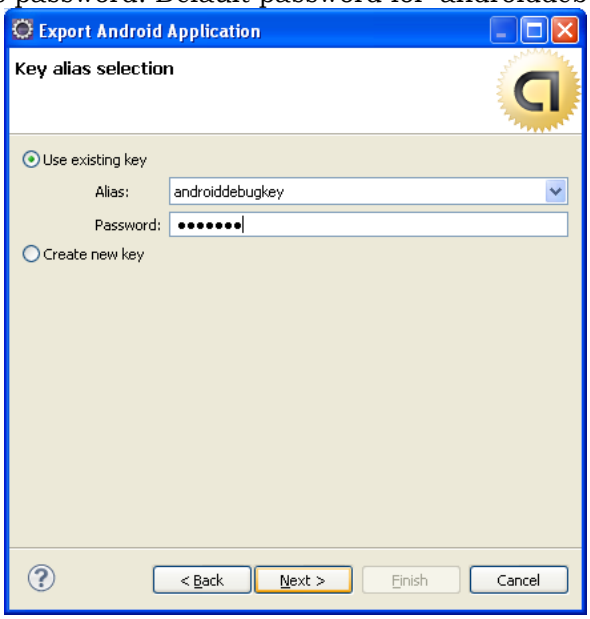


- Select your project and click on 'OK'.
- Click 'Next' in the parent window.
- Select 'Use existing keystore'.
- Give the location of your keystore and the keystore password.

- You can create a new keystore by selectin 'Create new keystore'.
- For development purpose, you can also use the default keystore. The default keystore password is 'android'.



- Click on 'Next >'.
- Select 'Use existing key'.
- Select your Alias from the dropdown. Default alias for default keystore is 'androiddebugkey'.
- Enter the alias password. Default password for 'androiddebugkey' is 'android'.



- Click on 'Next'.
- Give the complete path of the destination APK file. E.g.

  D:\store\Android-FCDB\output\FCDB_ANDROID_6.2.0.apk

- Click on 'Finish'.
- The wizard will create the above mentioned APK.

8. UI Resource Guide

Below is the list of various resources like images, property files, XML layout files used within the application. Usage of each resource is also provided. This list can be referred while customizing the application. Please keep in mind below limitations on the naming conventions of images:

- Images with name in capital letters do not work
- Image name should not have white space
- Only numerical name of image files is not allowed

The name of two image files (excluding extension) cannot be same.

Resource Type	Resource Name	Resource uses at
Images(Menu Grouping)	a.png	1st level menu grouping of accounts transaction
	a_.png	1st level menu grouping of accounts transaction
	b.png	1st level menu grouping of My Deposits transaction
	c.png	1st level menu grouping of My Loans transaction
	cq.png	1st level menu grouping of Cheques transaction
	d.png	1st level menu grouping of My Payments transaction
	e.png	1st level menu grouping of Bill Payments transaction
	r.png	1st level menu grouping of My Cards transaction
	t.png	1st level menu grouping of Customer Services transaction
	x.png	1st level menu grouping of Mutual Funds transaction
Images(Others)	oraclefavs.png	Application icon
	oraclelogo.png	Used in place of title
	titlebg.png	Application Header background
	rightalign.png	Used in case of subheading
	recordbg.png	Translucent background for “D” tag box.
	header.png	Used in case of transaction heading
	atm_marker.png	Marker for ATM in maps.
	branch_marker.png	Marker for branch in maps.
	mapview.png	Satellite map – Street map View flipper button.
	pushpin.png	Default marker for maps.
	back.png	Default icon for Back button.
	confirm.png	Default icon for Confirm button.
	exit.png	Default icon for Exit button.
	menu.png	Default icon for Menu button.
	login.png	Default icon for Login button.
	submit.png	Default icon for Submit button.
	menuhome.png	Home icon in breadcrumbs.

	bg.jpg	Application background image.
	boxbg.png	Background image for Transaction Help Box.
	calendar.png	Date picker icon.
	iconbg.png	Transaction Icon background in Menu list items.
	menu_arrow.png	Arrow on right side of Menu list items.
	menubg.png	Background for Menu list item's text.
	s1.png	Favorite transaction icon 1.
	s2.png	Favorite transaction icon 2.
	s3.png	Favorite transaction icon 3.
	s4.png	Favorite transaction icon 4.
	s5.png	Favorite transaction icon 5.
	f1.png	Add to favorites button
	f2.png	Remove from favorites button
	uparrow.png	Favorite transaction information flipper button.
	txn_info.png	Transaction Help Box button.
	errormsg.png	Error message button.
Images(Transactions)	rraac01.png	Account Activity icon
	rradt01.png	Account Details icon
	rrasm01.png	My Accounts icon
	rrasr01.png	Adhoc Statement icon
	rrbmf01.png	Buy Mutual Fund icon
	rrbpa01.png	Bill Payment icon
	rrcbr01.png	New Cheque Book icon
	rrcpw01.png	Change Password icon
	rrcsi01.png	My Cheques icon
	rrdbr01.png	Delete Biller icon
	rrdtf01.png	Domestic Transfer icon
	rrfer01.png	Forex rate icon
	rrfrx01.png	Forex rate icon
	rrifd01.png	Financing Details icon
	rrims03.png	Mail Box icon
	rritg01.png	Internal transfer icon
	rrlad01.png	Loan Details icon
	rrlob00.png	ATM Branch Locator icon
	rroat01.png	Own Account Transfer icon
	rrost01.png	Order Status icon

	rrptf01.png	portfolio icon
	rrrbr01.png	Register Biller icon
	rrrmf01.png	Redeem Fund icon
	rrrtd01.png	Deposit Redemption icon
	rrsuc01.png	Stop Cheque icon
	rrswt01.png	Switch Fund icon
	rrtcv01.png	Contract Deposit icon
	rrtdf01.png	Deposit Details icon
	rrvat08.png	Pending Authorization icon
	rrvcd01.png	Credit Card Details icon
	rrvst01.png	Credit Card statement icon
Layout XMLs	login_activity.xml	Login screen layout.
	menu_activity.xml	Menu screen layout.
	transaction_activity.xml	Transaction screen layout.
	branch_locator.xml	Map screen layout.
	listview.xml	Menu item list layout.
	spinnerlayout.xml	Spinner list layout.
	textviewbox.xml	Rounded bordered box shape layout.
	white_box.xml	Rounded bordered box shape layout with white background.
Properties file	android.txt	Contain application server specific information.
	customproperties.txt	Contains custom values.
	AndroidManifest.xml	The manifest presents essential information about the application (The name of the file cannot be changed).
Properties Handled in customproperties.txt	TEXTVIEW.TEXTCOLOR	Label text color.
	TEXTVIEW.TYPEFACE	Label text weight(e.g. Bold, Italic)
	BREAD.CRUMB.SEPERATOR	Character for breadcrumb text separator.
	PARAM.NAME.REQUESTID	Request id field name.
	PARAM.NAME.SESSIONID	Session id field name.
	PARAM.VALUE.DEVICE	Device id field value.
	PARAM.VALUE.LANG	Language id field value.
	PARAM.VALUE.REQUESTID.LGN	Login request id field value.
	PARAM.VALUE.REQUESTID.MNU	Menu request id field value.
	PARAM.VALUE.REQUESTID.LGF	Log off request id field value.
	PARAM.VALUE.REQUESTID.MAP	Branch locator map request id field value.
	PARAM.VALUE.REQUESTID.LOB	Branch locator request id field value.

	PARAM.VALUE.REQUESTID.FAV	Add/Remove from favorites request id
	FAVENABLED	Y/N as per Add/Remove favorite txn is required.
	MENUTYPE	L/G as per List/Grid menu style required.

Reference Documents

Sr.No	*Name of Document*
1	*Oracle_FLEXCUBE_Direct_Banking_Mobile_J2ME_Clients_Developer_Guide*
2	*Oracle_FLEXCUBE_Direct_Banking_Mobile_App_XML_structure*
3	*Oracle_FLEXCUBE_Direct_Banking_Mobile_Banking_User_Interface_Guide*
4	*Oracle_FLEXCUBE_Direct_Banking_Parameter_Sheet.xls*