Oracle FLEXCUBE Direct Banking

Mobile Client Non Invasive Development
Release 12.0.3.0.0

Part No. E52543-01

April 2014

ORACLE®

Mobile Client Non Invasive Development
April 2014

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/

Contents

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

Structure

Following mobile clients are released as part of current release of Oracle FLEXCUBE Direct Banking.

- J2ME Plain – Java client with simple UI. Targeted for low end mobile phones. Any Java enabled phone supporting MIDP 2.0 and CLDC 1.0 is suitable for this client.
- J2ME Rich – Java client with rich UI. Targeted for high end mobile phones. Any Java enabled phone supporting MIDP 2.0 and CLDC 1.1 is suitable for this client.
- I-Phone
- I-Pad

- Android – This client is developed for Google Android based phones.
- Android Tablet – This client is developed for Google Android based tablet devices.

These clients come out from the development factory with default configurations. These configurations will undergo changes as part of the implementation phase. Mobile clients will typically undergo changes such as:

1. Logo image change
2. Property file changes for application URL, Vendor name etc.

This document provides guidelines on implementing these changes.

Customizing and Repackaging the J2ME client

Oracle FLEXCUBE Direct Banking J2ME client is delivered in the form of JAR and JAD files. These files are available under *<FCDB BASE DIR>/deploy/mobile/j2merich* for the Rich client and *<FCDB BASE DIR>/deploy/mobile/j2meplain* for the Plain client. The Bank should obtain valid certificate and sign the JAD/JAR before distributing it to their customers. The JAR typically consists of following components:

- Java byte code (.class files)
- MANIFEST.MF file
- Resources such as images, property file
- Theme file (.res) in case of the Rich version of client

During the implementation phase of the software, components like property files, images, themes can be changed. A typical example is to change the default Oracle logo with the Bank's logo.

Such a change in the J2ME client requires re-packaging of the client application.

Subsequent sections describe various steps required for customizations of the J2ME client. Please note that this document provides only guidelines. Actual steps can be achieved by different methods.

There can me two modes of customization:

- Manual Mode
- Automated Mode (Using Apache Ant)

Manual Mode

Changing JAD File

The JAD file can be changed in any text editor. Please note that the properties 'MicroEdition-Profile' and 'MicroEdition-Configuration' should not be changed. Also some properties should be updated carefully. For example the property 'MIDlet-Jar-URL' must point to the correct JAR file. Another example would be the property 'MIDlet-Jar-Size', which must contain the exact size of the JAD file. The installation of the software may fail, if these values are not correct. Details of all the properties are provided in the parameter sheet.

Changing images in JAR

Images are used for the Splash logo, menu icons, transaction icons etc. The process to change the images is as follows:

1. Keep the new image in the same folder as that of the JAR
2. Update the JAR with new image. Simple JAR command, which comes with JDK, can be used for this purpose. An example would be as follows. This command replaces the oracle.png image inside the J2ME Rich client

   jar uvf <jar-name> <image-name>

   Same command can be used to update other resource files as well.

3. Calculate the exact size of the JAR file in bytes and update the 'MIDlet-Jar-Size' property in JAD file accordingly

Changing Property files in JAR

The JAR contains two property files – MANIFEST.MF and midletui.properties.

One might need to change the MANIFEST.MF file for updating the handler classes or to change any other application property like the application name etc.

The midletui.properties file would undergo change for implementation in a language other than English or if the Bank wishes to use English words different from the defaults mentioned in this file.

Below is the process to change midletui.properties file:

1. Keep the updated midletui.properties file in the same folder as that of the JAR
2. Update the JAR with new file using JAR command.

   jar uvf <fcdb_kernel>.jar midletui.properties

3. Calculate the exact size of the JAR file in bytes and update the 'MIDlet-Jar-Size' property in JAD file accordingly

For MANIFEST.MF, the steps would be as follows:

1. Extract the MANIFEST.MF file (.res) from the jar using following command
   jar xvf <fcdb_kernel>.jar META-INF/MANIFEST.MF

   This command will create a folder META-INF in the same folder as that of the JAR and will extract MANIFEST.MF in this META-INF folder

2. Make the necessary changes in MANIFEST.MF
3. Update the JAR with new file using JAR command.
   jar uvf <fcdb_kernel>.jar META-INF/MANIFET.MF

4. Calculate the exact size of the JAR file in bytes and update the 'MIDlet-Jar-Size' property in JAD file accordingly

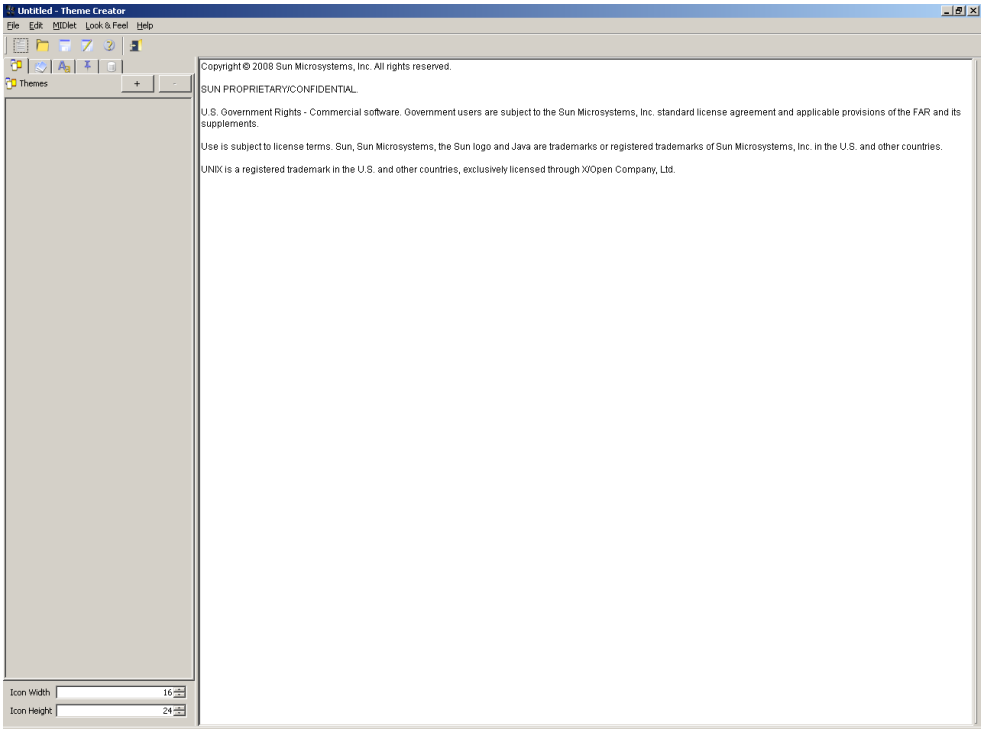
Changing LWUIT Resource file (.res)

Application uses LWUIT for UI rendering. The theme file (.res) file is used for generating theme for the application. This file can be updated using the Theme Creator (or Resource Editor) utility provided by LWUIT. Below are the steps to be followed for updating the theme (.res) file:

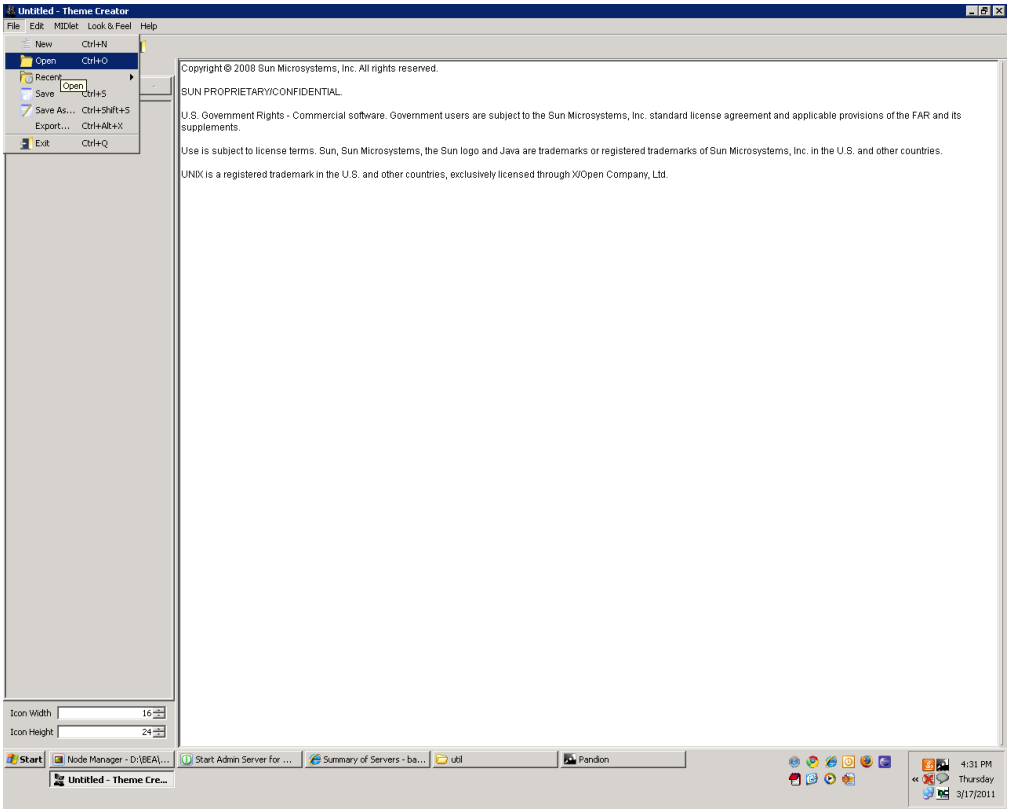
1. Extract the theme file (.res) from the jar using following command

   jar xvf <fcdb_kernel>.jar j2merich.res

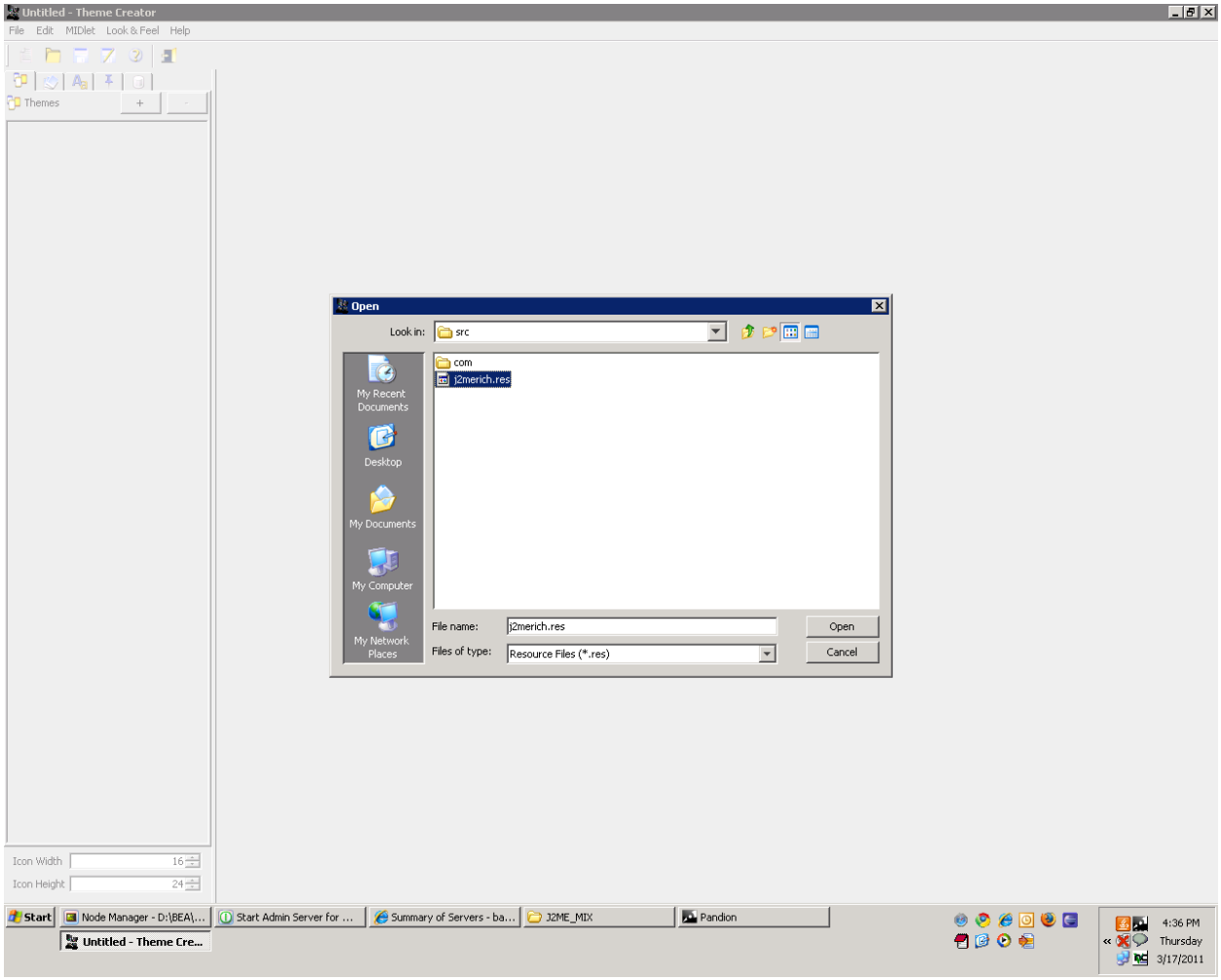
2. Open ResourceEdit.exe



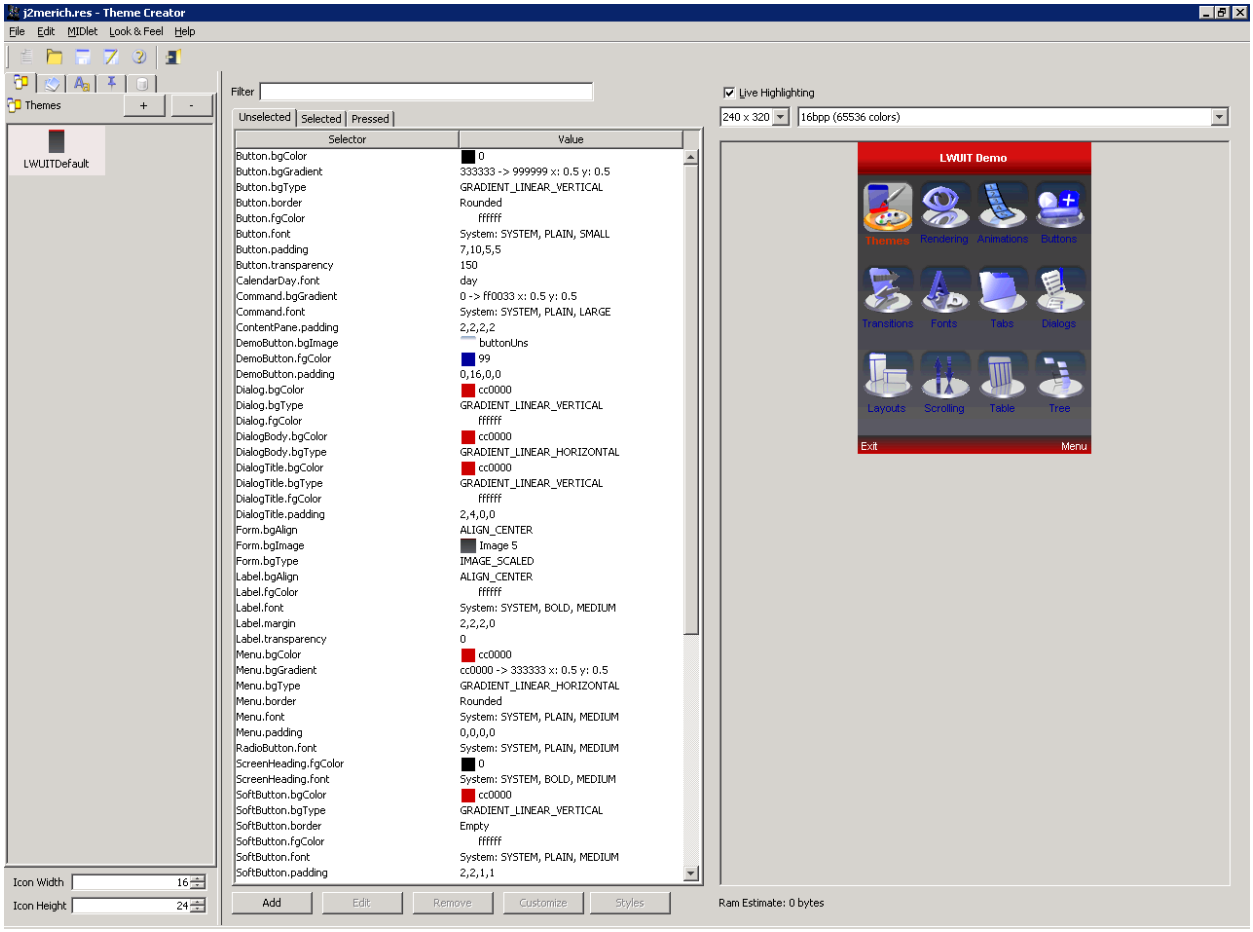
3. Click on File -> Open



4. Browse and select the theme file extracted above.



5. Screen like below showing various screen properties will be loaded.



6. Make necessary changes and then click on File -> Save. Please note that the name of the theme file should not be changed as it needs to be referred from within the application.
7. Update the JAR with new file using JAR command.

   jar uvf <fcdb_kernel>.jar j2merich.res

8. Calculate the exact size of the JAR file in bytes and update the 'MIDlet-Jar-Size' property in JAD file accordingly

Please refer to the documentation provided along with LWUIT destributable. The 'Theming' section of this document would provide useful information with respect to the theming and look and feel attributes used in the J2ME client.

Automated Mode

Prerequisites:

- The automated mode works with Ant Apache 1.8.0.Therefore Apache ant version 1.8.0 should be installed on the machine.

- The files required are
  - client_mobile.xml
  - client_mobile.properties
  - client_mobile.bat
  - <fcdb_kernel>.jar

These files are available at location FCDB/deploy/mobile/tools after installation of the application.

Follow the steps mentioned below :

- Put FCDBMobileNID.jar in lib folder of apache ant.
  - The path would be: ${base.dir}/apache-ant-1.8.0/lib where ${ base.dir } is the directory where apache ant is installed.
    e.g. D:\ProgramFiles

- Update client_mobile.bat
  - Replace ${build.directory} with the directory that has all three build files (client_mobile.xml, client_mobile.properties, client_mobile.bat)
    - e.g. D:\MyFolder\build
  - Replace ${ base.dir } that contains the apache-ant-1.8.0 folder
    - e.g. D:\ ProgramFiles

- Update client_mobile.properties
  - Replace :
    - *%%folder.dir%%* with the directory where jar, that is to be updated, is kept.
    - *%% org_jar%%* with the name of the jar that is to be updated/modified.
    - *%% images_path%%* with the path of the images folder that contains the images that are to be replaced/overwritten or added in the new jar.
    - *%% jar_name%%* with the name of the final jar. The jar and jad name will be the same. The jar name should be same as the property MIDlet-Jar-URL in the property file.
    - *%% prop_prefix%%* - uncomment the property prefix that has to be used.

- To change the property in JAD file or Manifest file, update the same in client_mobile.properties.
- To change images in jar , put the new images in *%% images_path%%* folder.
- To change the property file midletui.properties, extract it from the *%% org_jar%%* as explained in Manual Mode section, update it and keep it in *%% images_path%%* folder.
- To change the Lwuit resource file (.res), extract it from the *%% org_jar%%* as explained in Manual Mode section, update it and keep it in *%% images_path%%* folder.
- Click on the client_mobile.bat file. The new jar and jad will be created in *%%folder.dir%%* /new folder.

Customizing and repackaging the i-Phone/i-pad client

iPhone/iPad client is provided in the form of a workspace that needs to be imported on Macintosh machine and then customized as per the requirement. The workspace is available at location. This workspace is available under *<FCDB BASE DIR>/deploy/mobile/iphone* for i-Phone client and under *<FCDB BASE DIR>/deploy/mobile/ipad for ipad client*.

i-Phone/iPad client consists of application binaries, images and property files. In a typical customization scenario images and property files will undergo changes. Change in the binaries is not allowed. The property files available are app.plist(stores the server URL and other customizable properties) and Localisable.strings (stores the language specific keywords).

Pre-requisites

For customizing i-Phone client, developer needs to have:

1. Apple Mac PC with Leopard or Snow Leopard Operating Software
2. Xcode IDE for development which has an inbuilt iPhone simulator.

For customizing i-Pad client, developer needs to have:

1. Apple Mac PC with Snow Leopard Operating Software
3. Xcode IDE for development which has an inbuilt iPad simulator.

Installation of XCode

Below are the steps to be followed for installation of XCode on the Mac machine. Oracle FLEXCUBE Direct Banking client has been developed using iOS 3.1.3 with Xcode 3.1.4 or higher.

1. Depending on the OS available on the Mac machine, install the corresponding XCode installer
2. Click on the Apple logo in the menu bar and click > About this Mac
3. If OS is 10.5.x(Leopard) install - iphone_sdk_3.1.3_with_xcode_3.1.4__leopard__9m2809a.dmg

   OR if its 10.6.x(Snow Leopard) install

   iphone_sdk_3.1.2_with_xcode_3.2.1__snow_leopard__10m2003.dmg

The installers can be downloaded from – http://developer.apple.com/iphone/index.action

The PDF in the .dmg file explains in detail XCode Installation.

For **iPad** Snow Leopard OS with Xcode

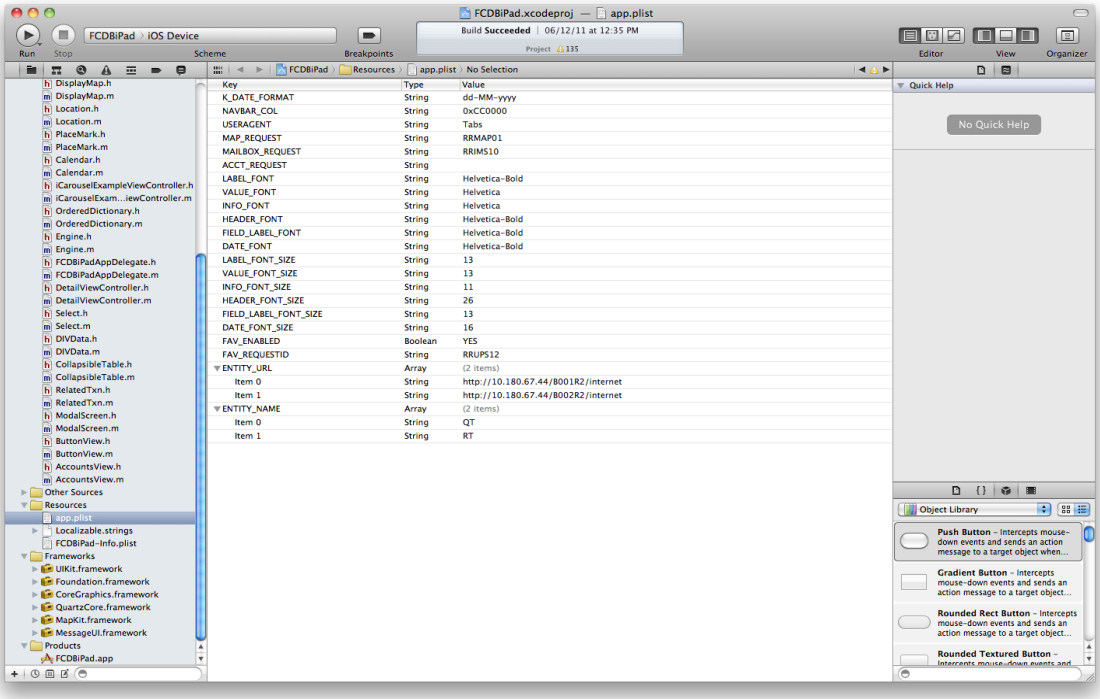
iphone_sdk_4.0 _with_xcode_3.2__snow_leopard__10m2003.dmg

is the minimum requirement.

Customizing the client in XCode

Customizable components in i-Phone/iPad client are

1. App.plist – Contains the application URL and other application properties.

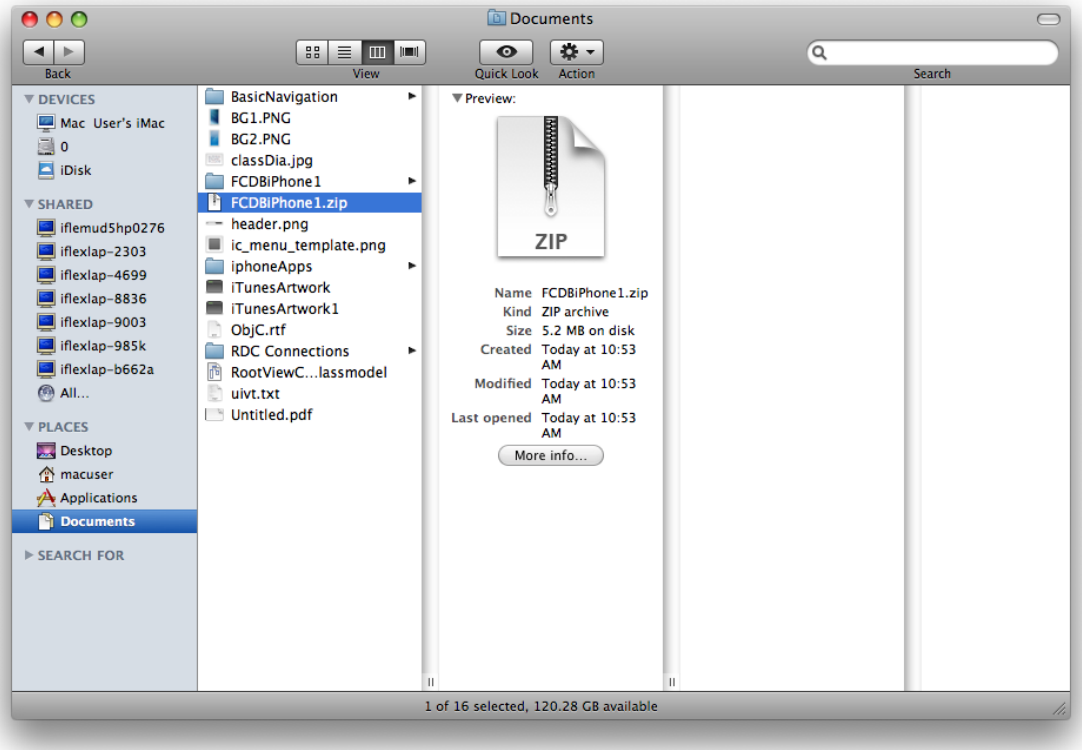


2. Images – A complete list of images along with their usage is mentioned in the parameter sheet *Oracle_FLEXCUBE_Direct_Banking_Parameter_Sheet.xls*.
3. Localisable.strings – Contains the language specific mapping for keywords used in the application. Below is the list of keywords and their significance. Refer to the Oracle_FLEXCUBE_Direct_Banking_Mobile_iPhone_Client_Developer_Guide.docx or Oracle_FLEXCUBE_Direct_Banking_Mobile_iPad_Client_Developer_Guide.docx for changes to be done for language dependent strings.
4. Settings Bundle – This file contains application version and copyright info. Updating of About.plist and Root.plist is required with language strings in root.strings.
5. FCDBiPhone-Info.plist / FCDBiPad-Info.plist - File for storing application name, code signing details, iTunesArtwork details and bundle/bundle display name. Below is the list of properties available in this file.

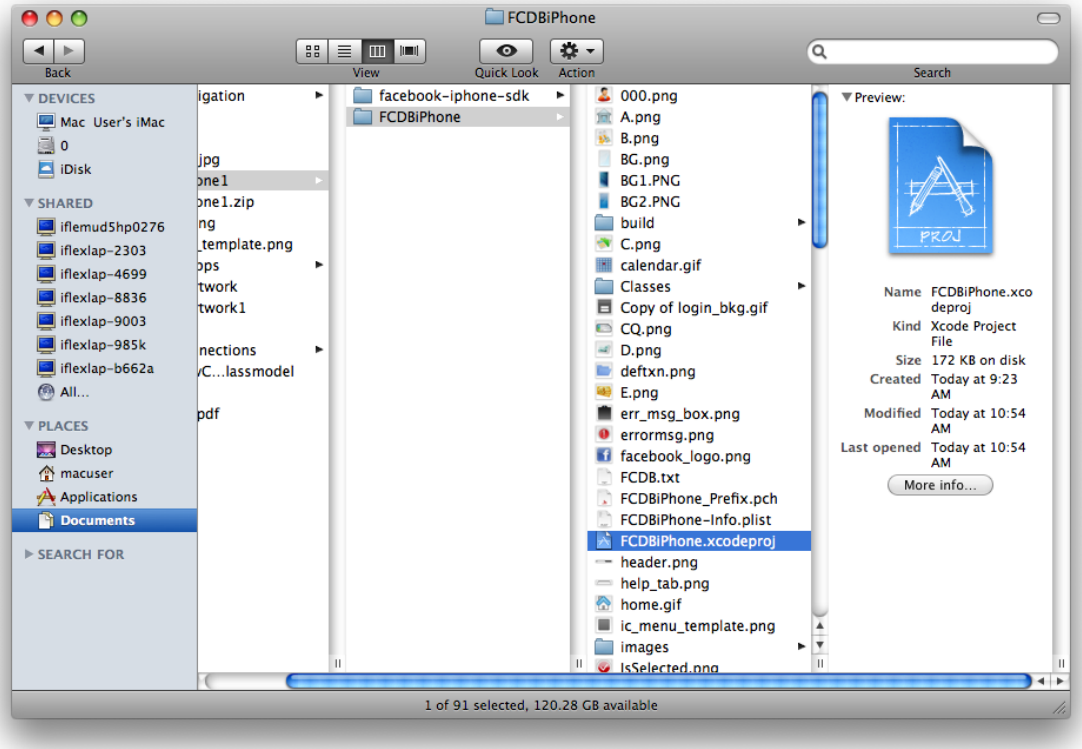
CFBundleDisplayName	Name of application in Springboard
CFBundleIconFile	iTunesArtwork – Name of application icon in Springboard
CFBundleVersion	Version of application

Below are the steps to be followed for updating i-Phone/iPad client

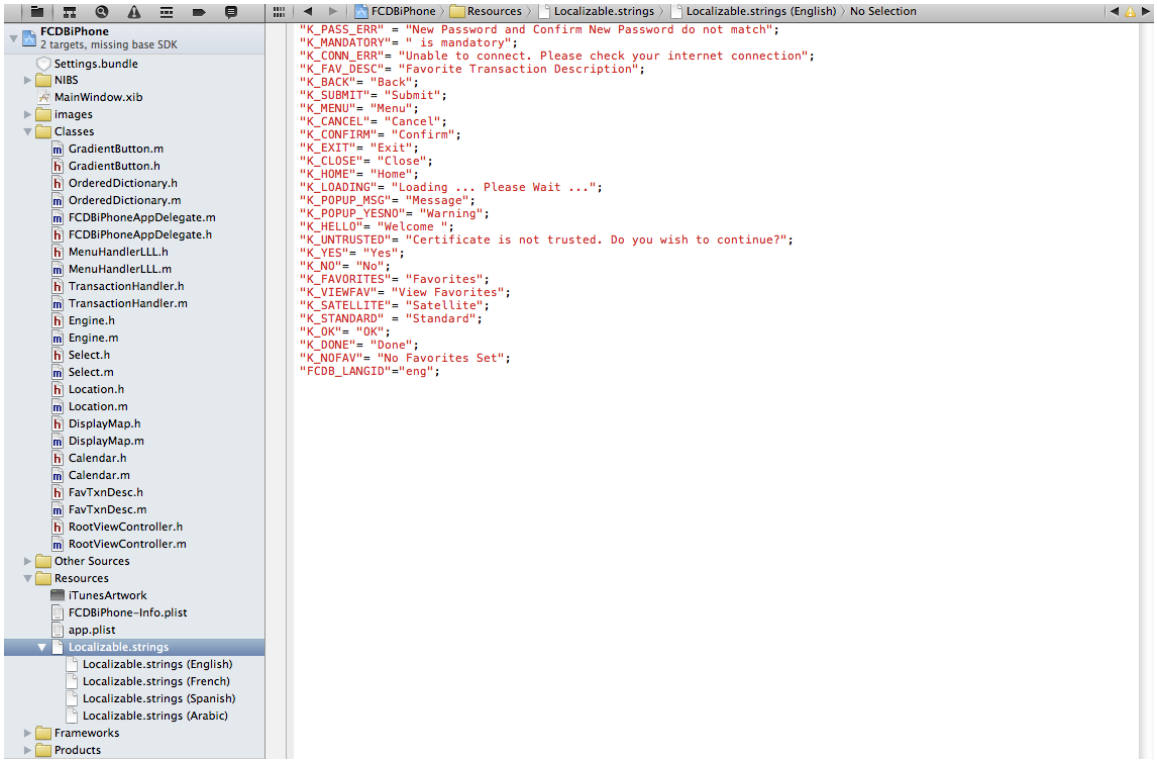
1. Take the FCDBiPhone1.zip/FCDBiPad.zip from the distributable and copy to the Mac. Sample screen is as shown below:



2. Double-click on the zip to decompress the files. Then double click on FCDBiPhone.xcodeproj/ FCDBiPad.xcodeproj to launch XCode



3. Make necessary changes. Typical changes would involve change to URL in app.plist in the other sources folder. Another example would be change in the application logo etc. Please note that any change in the image has to be done using XCode and not through the file system. Also two image files should not have same name (excluding extension also).



4. Click “Build and Go” to launch the simulator.

Customizing and Repackaging the Android Phone / Android Tablet clients

Android clients are available for phones and tablet devices. These clients are available at

<FCDB BASE DIR>/deploy/mobile/android (for android Phones) and

<FCDB BASE DIR>/deploy/mobile/androidtab (for android Tablets)

in the distribution. The Android client is in the form of unsigned APK. Bank needs to obtain a valid certificate and sign the APK before distributing it to the customers. Please refer to below website from Google for details on signing and publishing the Android application http://developer.android.com/guide/publishing/app-signing.html

Android Development is widely done using Eclipse with the android sdk plug-in added into it. For more information on how to setup the workspace, please read the document - Oracle_FLEXCUBE_Direct_Banking_Mobile_Android_Workspace_Configuration.pdf

The Android client consists of application binaries and other resources like images, property files and layout XML files. In a typical customization scenario images and property files will undergo changes. Change in the binaries is not allowed. The folder structure is as below:

Resource Type	Folder	File Name
Images	/res/drawable	*.png
Property files	/res/raw	android.txt, customproperties.txt
Layout XMLs	/res/layout	*.xml
Localizable strings	/res/values	strings.xml

Pre-requisites

For testing the Android client after customization, Android SDK is required along with the necessary platform version.

Android SDK is available for download from Google at location: http://developer.android.com/sdk/index.html

Changing images and property files

Below are the steps to customize the Android client:

1. Open the APK available in shipment using any archive application (like WinRAR).
2. Update the images/property files with new ones.

All the configurable properties and images are listed in the parameter sheet (Oracle_FLEXCUBE_Direct_Banking_Parameter_Sheet.xls).

Customizing and Repackaging the Blackberry client

The Blackberry client is available at location *<FCDB BASE DIR>/deploy/mobile/nativeblackberry* in the distribution. The Blackberry client is in the form of .cod. Bank needs to obtain a valid certificate and sign the cod file before distributing it to the customers. Please refer to below website for details on signing and publishing the blackberry application

http://supportforums.blackberry.com/t5/Testing-and-Deployment/BlackBerry-Applications-and-code-signing-Start-to-Finish/ta-p/445848

The Blackberry client consists of application binaries and other resources like images, .res files. In a typical customization scenario images and .res files will undergo changes. Change in the binaries is not allowed. The folder structure is as below:

Resource Type	Folder	File Name
Images	/res/ img	*.png
Resource file	/res/ img	*.rrc

For customization purpose user needs to add the workspace of blackberry in eclipse with blackberry plug-in.