

Oracle FLEXCUBE Direct Banking

System Handbook - Mobile Enabler
Release 12.0.3.0.0

Part No. E52543-01

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/

Copyright © [2008], [2014], Oracle and/or its affiliates. All rights reserved.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

1. Preface	1
1.1. Intended Audience	1
1.2. Documentation Accessibility	1
1.3. Access to OFSS Support	1
1.4. Structure	1
1.5. Related Information Sources	1
Overview	2
About this Document	3
Technical Architecture	5
GUI (Presentation) Tier	7
1.6. XML Transformation.....	8
1.7. Multi Lingual.....	9
1.8. Request Filters	10
1.9. Content Generation.....	11
1.10. Cross Site Scripting Attack Prevention Configuration	12
Channel Management Tier	13
1.11. Session Management.....	14
1.12. Data Security.....	15
Mobile Enabler Interfacing Tier	17
Multi Entity	19
Installation	21
System Configurations	23
Application Configurations	29
1.13. Transaction Access Control.....	30
1.14. User Access Control.....	31
1.15. Transaction Black Out for Mobile Channel.....	32
Transaction Configurations	33
1.16. Transaction Mapping.....	34
1.17. Request Field Mapping	35

1.18. Response XPATH mapping.....	37
Validation Framework.....	39
1.19. Data Types	40
1.20. Data Dictionary	43
1.21. Validation Templates.....	44
1.22. Validation Configurations	45
1.23. Validators.....	46
1.24. Error Handling.....	48
Auditing.....	49
Data Masking.....	51
Mobile Clients.....	52
1.25. J2ME Plain client.....	53
1.26. J2ME Rich client.....	54
1.27. iPhone client.....	55
1.28. IPad client	56
1.29. Blackberry client	57
1.30. Browser Based Mobile Banking.....	58

Intended Audience

This document is primarily targeted at

- Oracle FLEXCUBE Direct Banking Development Teams
- Oracle FLEXCUBE Direct Banking Implementation Teams
- Oracle FLEXCUBE Direct Banking Implementation Partners

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

- This handbook is organized into the following categories:
- Preface gives information on the intended audience. It also describes the overall structure of the System handbook.
- Abbreviations provides details of abbreviations used in document.
- Application server configuration explains deployment and configuration of Oracle FLEXCUBE Direct Banking ME

Related Information Sources

For more information on Oracle FLEXCUBE Direct Banking Release 12.0.3.0.0, refer to the following documents:

- Oracle FLEXCUBE Direct Banking Licensing Guide

Overview

Mobile Enabler Framework is a part of FCDB 12.0.1.0.0 release. This Framework will help the older versions of FCDB/FC@ to use only the mobile banking features of FCDB 12.0.1.0.0 without changing their existing Internet Banking setup. Older versions of FCDB/FC@ which either don't have mobile banking features or have very small footprint of mobile banking in them can use this framework to enable their sites with mobile banking functionality.

About this Document

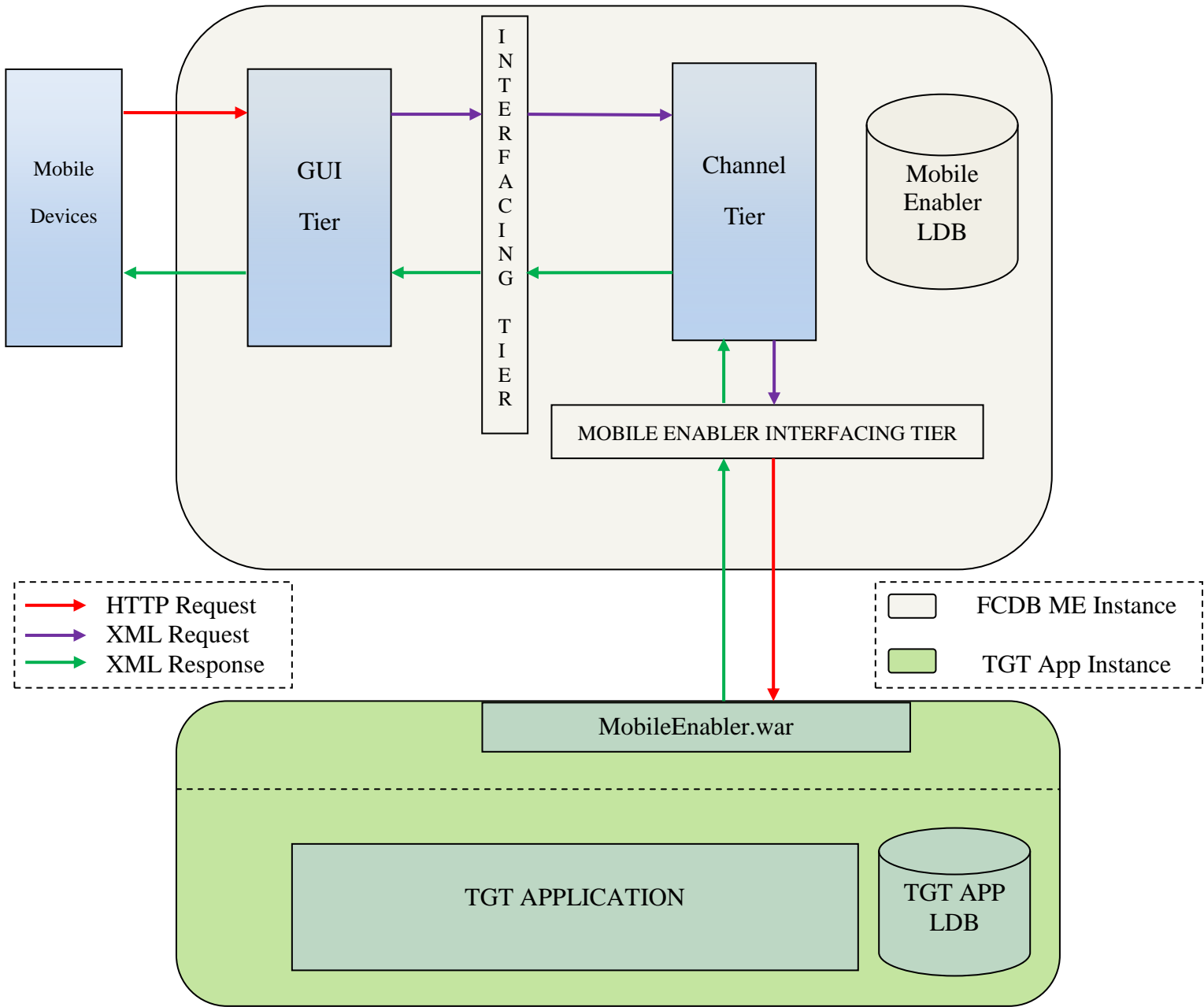
This document “**Oracle Flexcube Direct Banking Mobile Enabler System Handbook**” is a reference document for Oracle Flexcube Direct Banking Mobile Enabler application. This document is meant to explain the architecture and design of the application and contains the basic guidelines to configure the application during implementation. This document also discusses the key features of the application which can be used to configure mobile banking for different transaction of the existing Internet Banking application.

Abbreviations

FCDB / FC DB / FC Direct Banking / Direct Banking	Oracle FLEXCUBE Direct Banking
ME	Mobile Enabler
FCDB ME	Oracle Flexcube Direct Banking Mobile Enabler
App	Application
TGT App	Target Internet Banking Application
LDB	Local Data Base
Java EE / JEE	Java Enterprise Edition
Java SE / JSE	Java Standard Edition
Java ME / JME	Java Mobile Edition
DBA	Database Administrator
XML	Extensible Markup Language
XSL	XML Stylesheets
TCP	Transmission Control Protocol
HTTP	Hypertext Transmission Protocol
HTTPS	Secured Hypertext Transmission Protocol
SSL	Secured Socket Layer

Technical Architecture

Oracle Flexcube Direct Banking Mobile Enabler (FCDB ME) is a multi- tier application which connects to the existing **Internet Banking Application** (TGT App) through an interface. This interface uses HTTP URL Connection to send HTTP request to TGT App and expects XML response for the corresponding request sent.



GUI (Presentation) Tier

The Presentation Tier is the entry point for Flexcube Direct Banking Mobile Enabler application. This tier accept HTTP(S) request coming from the mobile devices. This tier converts the request from HTTP request format to a predefined XML format which is used by this tier and Channel Management Tier.

1.1. XML Transformation

The HTTP request is converted into a predefined XML format as shown below. All the element values are added in a CDATA section as indicated below to ensure that the values are interpreted correctly.

```
<?xml version="1.0" encoding="UTF-8" ?>
<faml>
  <request>
    <fldLoginUserId><![CDATA[DEMOUSER]]></fldLoginUserId>
    <fldLangId><![CDATA[eng]]></fldLangId>
    <fldDeviceId><![CDATA[eng]]></fldDeviceId>
  </request>
  <response/>
</faml>
```

The data undergoes appropriate global validations, before being added to the XML. There are basic XML injection and SQL injection checks that are done on the data with the facility of using additional regular expression patterns which allow the request to be trapped and validated before being accepted within the system.

The response for the transaction gets appended within the response node in the XML as shown in the sample XML extract.

The default XML structure used within the Presentation and Channel Management Tiers cannot be changed in anyway.

1.2. Multi Lingual

Flexcube Direct Banking Mobile Enabler is a complete end-to-end multi lingual solution and has the appropriate support to ensure that different character sets can be supported for reading the input received by the presentation tier. The default character set used is “**UTF-8**” and can be configured differently if the request encoding is different. The parameter is defined for each instance of the servlet using the **FCAT.REQUEST.CHARSET** property in the <<[*entity.xml*](#)>> configuration file.

1.3. Request Filters

Request Filtering is another feature provided at presentation tier. This feature is configurable using an interface “**com.iflex.fcat.gui.filters.RequestFilter**”.

```
package com.iflex.fcat.gui.filters;

public interface RequestFilter {

    public int accept (
        HttpServletRequest    p_request
        ,   HttpServletResponse    p_response
        ,   InternetServletConfig    p_config
        ,   InternetInstanceData    p_data
    ) throws Exception;
}
```

This class abstracts the basic security and validation services to be supported with the instance of the servlet. One can implement this class and provide their logic whether the request should be accepted for processing or should be rejected. The servlet handles the processing of the request if the return value is 0 (“CONNECTION_ACCEPTED”) and rejects request if the return value is 1 (“CONNECTION_REJECTED”). By default application uses “**com.iflex.fcat.gui.filters.DefaultRequestFilter**”, if a separate implementation needs it be configured it can be done using property “**FCAT.REQUEST.FILTER**” in <<[entity.xml](#)>> configuration file.

1.4. Content Generation

This tier also takes care of the content generation of the response xml in return journey. It takes care of the transformation of the response XML into desired format (for e.g. html, .pdf, .xls). This feature is also pluggable and configurable using an abstract class “**com.iflex.fcat.gui.content.ContentGenerator**”.

```
package com.iflex.fcat.gui.content;

public abstract class ContentGenerator {

public abstract void handleContent (
    InternetServletConfig p_config
,    HttpServletRequest    p_request
,    HttpServletResponse   p_response
,    InternetInstanceData  p_instancedata
) throws Exception;

}
```

One can implement this class to define a new Content Generator and assign a unique ID to that. This id is mapped to each request and thus denotes the content handler to be invoked for the corresponding request. The configuration resides in column CONTENTSTYLE of table mstchannelats of ME LDB.

A new Content Generator needs to be defined in <<[entity.xml](#)>> configuration file similar to example shown below

```
<FAML>

<CONTENT.GENERATORS>
  <CONTENT.GENERATOR ID="HTML"
CLASS.NAME="com.iflex.fcat.gui.content.HTMLContentGenerator">
  <ATTRIBUTE NAME="CONTENT.TYPE" VALUE="text/html;charset=UTF-8"/>
  </CONTENT.GENERATOR>
</CONTENT.GENERATORS>

</FAML>
```

1.5. Cross Site Scripting Attack Prevention Configuration

At this tier, Configuration can be done to prevent Cross Site Scripting Attack. For this <<[entity.xml](#)>> file needs to be updated with the following properties

FCAT.FILTER.EXP	Characters to filter can be added here comma separated
FCAT.REPLACE.EXP	Character to be placed in place of character getting replaced. There is a one to one mapping.

If above mapping is not specified in <<[entity.xml](#)>> file, then default replacement filtering will be done as follows:-

```
<FCAT.FILTER.EXP><![CDATA[<,>,&#,\" ,alert(,alert  
(,javascript)]></FCAT.FILTER.EXP>
```

```
<FCAT.REPLACE.EXP><![CDATA[[,],##,' , ,)]></FCAT.REPLACE.EXP>
```

Channel Management Tier

The Channel Management Tier takes care of session management and data security. This tier also takes care of audit trail logging. This tier then passes the XML request to the Mobile Enabler Interfacing tier.

1.6. Session Management

This tier has the access to Flexcube Direct Banking Mobile Enabler LDB and uses the same for session related activities. This tier does the session management by storing the session id of the user in “usersession” table. This table contains the session id of both the application (FCDB ME and TGT App).

1.7. Data Security

Channel tier Framework provides developer capability to use predefined processing features for data storage which can be used to provide simple data-security during a transactional workflow.

Application allows configuration capability to store a request & read the request back before proceeding with business processing. This serves as a data-security feature as follows:

- 1) Typically a transaction workflow consists of data-entry screen, verification screen and confirmation screen.
- 2) On data-entry screen, user keys-in intended data & submits the same.
- 3) The submitted data is persisted by channel tier. This is based on configuration.
- 4) The data is validated, if valid, verification screen is displayed showing data as submitted by user. User needs to verify the data & choose to confirm/change it.
- 5) If user chooses to confirm, no user data is submitted from verification screen. The user request data is rather picked from the persisted storage & forwarded for processing.

This feature ensures user that data cannot be tampered on confirmation. To use this feature of Data Security one need to configure columns “FLAGPREPROCESS/FLAGPOSTPROCESS” in table “mtchannelats” for a transaction workflow.

Column FLAGPREPROCESS: Enumerations mentioned below. The processing is applied on the request path before passing the request XML to Mobile Enabler Interfacing Tier.

Pre Process	Description
1	Fetches only request data stored in table “ <u>usersessiondata</u> ” and add it to current request data. The request data to be fetched from “ <u>usersessiondata</u> ” is identified using request parameters “fldSectionId” & “fldDataId”
2	Persist the current request data in “ <u>usersessiondata</u> ” & return fldDataId (identifier to the data) in response.
5	Fetches only response data stored in table “ <u>usersessiondata</u> ” and add it to current request data. The request data to be fetched from “ <u>usersessiondata</u> ” is identified using request parameters “fldSectionId” & “fldDataId”
6	Fetches Both request and response data stored in table “ <u>usersessiondata</u> ” and add it to current request data. The request data to be fetched from “ <u>usersessiondata</u> ” is identified using request parameters “fldSectionId” & “fldDataId”

Column FLAGPOSTPROCESS: The processing is applied on the response path before response is passed to Presentation tier. Enumerations mentioned below.

Post Process	Description
1	Fetch request and response data stored in table " <u>usersessiondata</u> " and add it to current response. The request and response data to be fetched from usersessiondata is identified using request parameters "fldSectionId" & "fldDataId"
2	Persists the current request and response data in " <u>usersessiondata</u> " & return fldDataId (identifier to the data) in response.
4.	Updates entire SectionID Node with Response (Does not retain request if it exists in " <u>usersessiondata</u> ") and add it to current response. The Response data to be updated in usersessiondata is identified using request parameters "fldSectionId" & "fldDataId".
5.	Fetches only response data stored in table usersessiondata and add it to current response. The response data to be fetched from " <u>usersessiondata</u> " is identified using request parameters "fldSectionId" & "fldDataId"
6.	Updates only Response (Retains request if exists) data in " <u>usersessiondata</u> " and selects updated response and previous request (if exists) to current response being send to Presentation tier. The Response data to be updated in " <u>usersessiondata</u> " is identified using request parameters "fldSectionId" & "fldDataId".

Mobile Enabler Interfacing Tier

The Mobile Enabler Interfacing Tier takes the request XML from Channel tier and converts it into HTTP request format to be passed to TGT App. Before converting it into HTTP request format this tier performs the following operations

1. **User Access Check**: This operation is performed to check whether the user has the access for mobile banking. Please refer to section [User Access Control](#).
2. **Transaction Access Check**: This operation is performed to check whether a particular transaction is available for a particular entity, usertype and channel combination. Please refer to section [Transaction Access Control](#).
3. **Transaction Blackout Check**: This operation is performed to check whether a transaction has been blacked out from being used through Mobile Banking channels. Please refer to section [Transaction Black Out for Mobile Channel](#).
4. **Invoke FCDB Validation Engine Framework**: This operation is performed to check field data validation and enrichments before sending the request to TGT App. This operation allows validating the field in FCDB ME App itself and hence the data validation is done before sending it to TGT App, this operation also helps to enrich data using JAVA/SQL based enricher. For more details please refer to section [Validation Framework](#).

Once the above operations are successful this tier converts the FCDB ME XML request into HTTP request format of TGT App. It uses FCDB ME LDB to map FCDB ME fieldname value to TGT App fieldname. Tables where these field mappings are stored are “[mobileenablerdefaultfieldmap](#)” and “[mobileenablerfieldmap](#)”.

mobileenablerdefaultfieldmap: This table will contain the mapping of fields of FCDB ME App to those HTTP request field of TGT App which are always required by the TGT App for all requests irrespective of transaction id. Mapping done here can be overridden for a particular request by doing entry in table “[mobileenablerfieldmap](#)”.

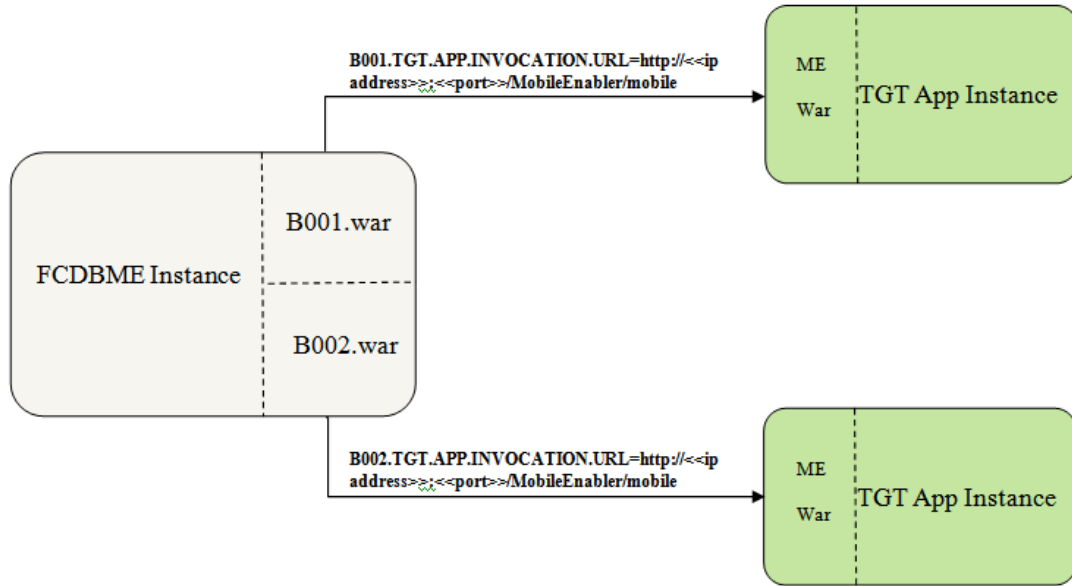
mobileenablerfieldmap: This table will contain the mapping of the form fields of FCDB 6.2.0 to the HTTP request field name required by TGT App for a particular transaction.

This tier also takes care of auditing the HTTP request going to TGT App and the corresponding XML response given by the TGT App. For auditing the request/response data this tier uses “**mobileenablerauditlog**” table available in FCDB ME LDB.

Multi Entity

Flexcube Direct Banking has the potential to represent multiple entities i.e. Business units, through a single setup. In this way having single Oracle FCDB Binaries and Single Database one can configure two different Business Units. The same feature has been envisaged in Mobile Enabler where one can configure multiple entities to point to different TGT App. As a part of the release one default entity has been provided called as “B001”. In order to configure another entity one needs to follow the following steps

1. In FCDBME application server, deploy another web-application (.war) to serve the requests. Make a copy of existing B001.war and rename it to **<<identity>>.war** and deploy the same. The web.xml of the application should be modified to change the parameter **“FCAT.INTERNETSERVLET.DAEMON.NAME”** to entity identifier **<<identity>>**.
2. A new presentation tier property file should be created named as **<<FCAT.INTERNETSERVLET.DAEMON.NAME>>.xml** in system/home folder of FCDBME environment setup. Refer to property **<<[entity.xml](#)>>** in System Configurations section.
3. Use **“entityclone.sql”** and execute in FCDME LDB to create a clone of DB entries required for new entity.
4. Deploy mobileenabler.war in another TGT App and configure the URL for the property **<<identity>>TGT.APP.INVOCATION.URL** in **[mobileenabler.properties](#)** file.



Installation

The implementation team doesn't need to do any changes in the existing container/database setup of Internet Banking application. All they need to do is as follows

- Create a new environment setup of FCDB Mobile Enabler Application using Flexcube Direct Banking Mobile Enabler Installer.
- Deploy FCDB ME application (B001.war) on the Mobile Enabler Application Server environment.
- Deploy MobileEnabler.war in their existing Internet Banking server environment.

For the above mentioned steps please refer to Oracle FLEXCUBE Direct Banking ME Installation document for the corresponding Application Server.

[MOBILEENABLER.WAR](#)

This war file needs to be deployed on the TGT App server environment. A detailed description of web.xml is shown below.

```

<servlet>
  <description>Servlet For Enabling Mobile Devices</description>
  <display-name>MobileEnablerServlet</display-name>
  <servlet-name>MobileEnablerServlet</servlet-name>
  <servlet-class>
    com.iflex.fcat.gui.mobileenabler.MobileEnablerServlet
  </servlet-class>
  <init-param>
    <param-name>style-id</param-name>
    <!--for FC@2.9 -->
    <param-value>
      com.iflex.fcat.servlet.XMLContentGenerator
    </param-value>
    <!--for FCDB 5.3.X, 5.0.X -->
    <param-value>XML</param-value>
  </init-param>
  <init-param>
    <param-name>debug-mode</param-name>
    <param-value>D</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>MobileEnablerServlet</servlet-name>
  <url-pattern>/mobile</url-pattern>
</servlet-mapping>

```

As the main Servlet of TGT App and override the `writeResponse ()` method to generate the XML response.

Two additional initialization parameters needs to be defined as discussed below:

style-id: This is a mandatory parameter to decide the content generator class to be called by MobileEnablerServlet to generate XML response.

debug-mode: This is an optional parameter to enable logging of request/response for MobileEnablerServlet in the TGT App server environment.

The “**com.iflex.fcat.gui.mobileenabler.MobileEnablerServlet**” .class file and .java file are both packaged inside the **MobileEnabler.war**, one can do the necessary customization in the java file to meet their requirements for any change required in response XML originating from TGT App.

System Configurations

Once the installation is complete, following Day Zero configurations needs to be completed.

- **<<entity>>.xml**: This file contains the presentation tier configurations. After successful installation one can find this file in **system\home** folder of the installed FCDB ME application. As part of release one will get the default entity i.e. B001.xml file. In case of Multi-Entity scenario one needs to make a copy of this xml and rename it to respective <<entity>>.xml and update the properties as per the requirements. Following are the important parameters in this file to be configured at Day Zero.
 1. **FCAT.ID.ENTITY**: This parameter should contain the name of the entity.
 2. **FCAT.GUIXSL.ZIP.NAME**: This parameter to provide the name of the gui-xsl zip to be used, specify this property if zip is to be used.
 3. **FCAT.XSL.ISCACHE.ENABLED**: This property determines if the GUI xsl's should be cached.

- **fcac-config.xml**: This file contains the configurations for channel tier. This file also contains the details of local database datasource to be used by channel tier to take the database connections. Following are the important parameters in this file to be configured at Day Zero.
 1. **FCON.A2.USER**: This parameter to contain username of the database app user id to be used by the application when connection to specified database is established.
 2. **FCON.A2.PASS**: This parameter to contain password of the database app user id to be used by the application when connection to specified database is established.
 3. **FCON.A2.LDB.URL**: This parameter to contain URL of the database to which the application will be connecting in the format <DB Server IP or Machine Name>:<Listener Port>:<DB Service Name>.

- **fcac.properties**: The property file contains information to locate database. JNDI details to fetch the local database datasource are available in fcac.properties and should be updated during Day Zero Setup. Following are the important parameters in this file to be configured at Day Zero.

1. **FCON.A1.USER**: This parameter to contain username of the database app user id to be used by the application when connection to specified database is established.
2. **FCON.A1.PASS**: This parameter to contain password of the database app user id to be used by the application when connection to specified database is established.
3. **FCON.A1.LDB.URL**: This parameter to contain URL of the database to which the application will be connecting in the format <DB Server IP or Machine Name>:<Listener Port>:<DB Service Name>.
4. **ORACLE.UTILS.IMPL.CLASS**: This parameter to be commented if the application server is WebSphere.

➤ **mobileenabler.properties**: This file contains the property required by mobile enabler framework to do the handshake with the TGT App. This file also contains the day zero property to understand the response XPATH of TGT App for some common application behaviour. **All the properties in this file are entity agnostic, so in case of multi entity scenario one can define the same property for different entity by appending “<<identity>>.” prefix.**

1. **TGT.APP.INVOCATION.URL**: This property will contain the URL of the TGT App to which HTTP request will be posted. This URL will be as per the configurations done during the deployment of **MobileEnabler.war** on TGT App server environment. For e.g.

TGT.APP.INVOCATION.URL =
http://<<ip address>>:<<port>>/MobileEnabler/mobile

2. **TGT.APP.MENURESP.TXNID.XPATH**: This property will contain the XPATH (under faml tag) containing id of all the transactions coming as part of menu in login success response XML of TGT App. For e.g.

TGT.APP.MENURESP.TXNID.XPATH =
response/menu/menuitem/menuitem/menuitem/@txnid

This property will be default configured changes to be done if required.

3. **TGT.APP.RETURN.CODE.XPATH**: This property will contain the XPATH (under faml tag) to get the return code value from the response XML of TGT App. This property will be default configured changes to be done if required. The possible return code value of TGT App should be mapped to FCDB return codes in *mobileenablerreturncodemap* table as day zero configuration.
4. **TGT.APP.HOST.REF.NO.XPATH**: This property will contain the XPATH (under faml tag) to get the host reference number value from the

response XML of TGT App, if any. This property will be default configured changes to be done if required.

5. **TGT.APP.ERROR.NODE.XPATH**: This property will contain the XPATH (under faml tag) to get the error details from the response XML of TGT App, if any.
Framework can handle multiple error nodes if coming from the TGT APP. To handle such scenario “#” separated XPATH should be configured for the above property name, as shown below, after completely analyzing the possible error node of TGT APP.

TGT.APP.ERROR.NODE.XPATH =
rc/@errorcode,@errormessage#response/txndata/txn/errorlist/error/@errorcode,@errordesc

This property will be default configured changes to be done if required.

6. **TGT.APP.WARNING.NODE.XPATH**: This property will contain the XPATH (under faml tag) to get the warning message details from the response XML of TGT App, if any. Similar to property explained above one can give a “#” separated XPATH after completely analyzing the possible warning message node of TGT APP. This property will be default configured changes to be done if required.
7. **TGT.APP.RESULT.NODE.XPATH**: This property will contain the XPATH (under faml tag) to get the result message details from the response XML of TGT App, if any. Similar to property explained above one can give a “#” separated XPATH after completely analyzing the possible result message node of TGT APP. This property will be default configured, changes to be done if required.
8. **TGT.APP.SESSION.RESPONSE.XPATH**: This property will contain the XPATH (under faml tag) to get the session id from the response XML of TGT App. This property will be default configured changes to be done if required.
9. **TGT.APP.SESSION.REQUEST.FIELD**: This property will contain the name of the HTTP request field in which TGT App expects the session id value. This property will be default configured changes to be done if required.

10. **TGT.APP.USERID.RESPONSE.XPATH**: This property will contain the XPATH (under faml tag) to get the user id from the response XML of TGT App after successful login. This property will be default configured changes to be done if required.
11. **TGT.APP.CHUSERID.RESPONSE.XPATH**: This property will contain the XPATH (under faml tag) to get the channel user id from the response XML of TGT App after successful login. By default framework will set user id value to the channel user id field, if this property is commented.
12. **TGT.APP.USERID.FNAME.XPATH**: This property will contain the XPATH (under faml tag) to get the channel user id first name from the response XML of TGT App after successful login. This property will be default configured changes to be done if required.
13. **TGT.APP.USERID.LNAME.XPATH**: This property will contain the XPATH (under faml tag) to get the channel user id last name from the response XML of TGT App after successful login. This property will be default configured changes to be done if required.
14. **TGT.APP.USERID.LSUCC.XPATH**: This property will contain the XPATH (under faml tag) to get the channel user id last successful login date timestamp from the response XML of TGT App after successful login. This property will be default configured changes to be done if required.
15. **TGT.APP.USERID.LSUCC.DATE.FORMAT**: This property will contain the XPATH (under faml tag) to get the channel user id last successful login date timestamp format from the response XML of TGT App after successful login. This property will be default configured changes to be done if required.
16. **TGT.APP.USERID.LFAIL.XPATH**: This property will contain the XPATH (under faml tag) to get the channel user id last failed login date timestamp from the response XML of TGT App after successful login. This property will be default configured changes to be done if required.
17. **TGT.APP.USERID.LFAIL.DATE.FORMAT**: This property will contain the XPATH (under faml tag) to get the channel user id last failed login date timestamp format from the response XML of TGT App after

successful login. This property will be default configured changes to be done if required.

18. **TGT.APP.NBRLOGIN.RESPONSE.XPATH**: This property will contain the XPATH (under faml tag) to get the number of login from the response XML of TGT App after successful login. This property will be default configured changes to be done if required.
19. **TGT.APP.USERENTITY.RESPONSE.XPATH**: This property will contain the XPATH (under faml tag) to get the entity of user from the response XML of TGT App after successful login. This property will be default configured changes to be done if required.
20. **TGT.APP.USERTYPE.RESPONSE.XPATH**: This property will contain the XPATH (under faml tag) for retrieving the user type of the user after successful login. This XPATH will be evaluated by the framework and the evaluated value should have a mapping with FCDB usertype in *mobileenablerusertypemap* table. For e.g.

**TGT.APP.USERENTITY.RESPONSE.XPATH =
concat(chkUserSegmentType/@segmentId,'~',chkUserSegmentType/
@customerType)**

After evaluating the above XPATH one will get a value for retail user as “RETAIL~I”, so this value should be mapped to FCDB usertype “EN1” in *mobileenablerusertypemap* table as day zero configuration.

21. **TGT.APP.FCP.CONDITION.XPATH**: This property will contain the XPATH (under faml tag) for retrieving the force change password condition of the user after successful login. This XPATH will be evaluated by the framework and the evaluated value should return true or false. For e.g.

**TGT.APP.FCP.CONDITION.XPATH = (user/@nbrlogins = '0' or
user/@nbrlogins = '')**

22. **TGT.APP.TXNPIN.CONDITION.XPATH**: This property will contain the XPATH (under faml tag) for retrieving the transaction pin authorization condition of the user while doing any transaction. This XPATH will be evaluated by the framework and the evaluated value should return true or false. For e.g.

TGT.APP.TXNPIN.CONDITION.XPATH = (txnpinrequired = 'Y')

23. **TGT.APP.HEADER.NAME**: This property will contain a comma separated list of HTTP Header names which needs to be passed from FCDB Mobile Enabler application to the target app.For e.g.

TGT.APP.HEADER.NAME = Host, Accept, User-Agent, Accept-Language, Accept-Encoding, Content-Type, Connection, Referer

24. **CLIENT.USERAGENT.NAME**: This property will contain a comma separated list of mobile user agents which client uses to invoke the target app. For e.g.

CLIENT.USERAGENT.NAME = AndPhone,AndTabs,iPhone,iPad

25. **CLIENT.SYSTEM.DATE**: This property will contain system date of client application which will be used in FCDB Mobile Enabler application without timestamp.For e.g.

CLIENT.SYSTEM.DATE = systemdate

- **logger.properties**: This property file needs to be updated to enable logging of data, if required, at each tier of the application.

To check the HTTP request coming from the mobile channel, the HTTP request going to TGT App and the XML response coming from the TGT App, one need to configure logging level to DEBUG at Mobile Enabler Interface tier as shown below.

```
log4j.category.com.iflex.fcat.channels.plugins.FCDBMobileEnablerPlugin=DEBUG,
LOGFILE_FCMEP
log4j.appender.LOGFILE_FCMEP=org.apache.log4j.RollingFileAppender
log4j.appender.LOGFILE_FCMEP.File=%%FCAT.APPLOGS%%/FLEXCUBEMobileEnablerPlugin.log
log4j.appender.LOGFILE_FCMEP.layout=org.apache.log4j.PatternLayout
log4j.appender.LOGFILE_FCMEP.layout.ConversionPattern=%d %-5p %c - %m%n
log4j.appender.LOGFILE_FCMEP.MaxFileSize=9099KB
log4j.appender.LOGFILE_FCMEP.MaxBackupIndex=6
```

Application Configurations

Implementation team can control the behavior of Mobile Enabler Framework using following application configurations.

1.8. Transaction Access Control

This framework provides a facility to configure transactions to be displayed on mobile devices for a particular entity, usertype and channel combination. It will also make sure that the subset of, transaction configured in this framework and transaction assigned to the user in TGT App, is displayed.

A table “*mobileenablertxnlist*” has been provided in this framework to configure the Transaction Access Control. The structure of the table is as shown below

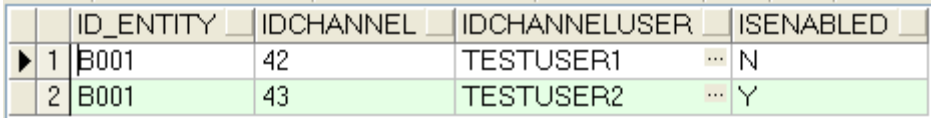
Column Name	Type	Nullable	Default	Comments
ID_ENTITY	VARCHAR2(5)	No		Entity ID of FCDB
USERTYPE	VARCHAR2(3)	No		User Type of FCDB
FCDBIDCHANNEL	VARCHAR2(3)	No		Channel ID of FCDB
FCDBIDTXN	CHAR(3)	No		Transaction ID of FCDB
TGTAPPIDTXN	VARCHAR2(3)	No		Transaction ID of Target Application, for e.g. 2.9.X,5.3.X
TGTIDCHANNEL	VARCHAR2(2)	No	'01'	Channel ID of Target Application, for e.g. 2.9.X,5.3.X
ISENABLED	VARCHAR2(1)	No	'Y'	Transaction is enabled or not for Mobile Banking.
TXNDESCRIPTION	VARCHAR2(100)	No		Description of Transaction.

1.9. User Access Control

This framework provides a facility to restrict a particular user from using Mobile Banking without restricting their accessibility to Internet Banking. A view **“*fcattvwmobileaccesscontrol*”** has been provided which has been created using a table **“*mobileenableruseraccesscontrol*”**.

This framework checks for the value of the column “ISENABLED” for a particular user id, if any entry exists and the values of this column is “N”, user would not be able to use Mobile Banking.

If the value of this column is “Y” or if there is no entry found for the user then application will allow the user to use Mobile Banking. For e.g. as per the entry shown below



	ID_ENTITY	IDCHANNEL	IDCHANNELUSER	ISENABLED
1	B001	42	TESTUSER1	N
2	B001	43	TESTUSER2	Y

“TESTUSER1” will not be allowed to use Browser based Mobile Banking (id channel 42) and since there is no entry for this user for idchannel 43 (App Based Mobile Banking) “TESTUSER1” will have full access to use App Based Mobile Banking.

Similarly for “TESTUSER2”, since the value of “ISENABLED” is “Y” for idchannel 43, so this user has full access to App Based Mobile Banking, and since there is no entry for idchannel 42, so this user has again full access for Browser Based Mobile Banking.

Implementation team can change this view **“*fcattvwmobileaccesscontrol*”** so that they can refer their own database table through database link to create this view, only contract to be followed is, this view must have the column “ISENABLED” for which the value can be only “Y” or “N”.

1.10. Transaction Black Out for Mobile Channel

This framework provides a facility to configure Black Out for a particular transaction only for mobile channels. A table “txnblackout” has been provided to configure the same.

Column Name	Type	Nullable	Default	Comments
TYPEUSER	VARCHAR2(3)	No		User Type of FCDB.
IDTXN	CHAR(3)	No		Transaction ID of FCDB.
STARTDATE	DATE	No		Blackout start date.
STARTTIME	NUMBER(6)	No	0	Blackout start time in “hhmiss” format for daily black out.
ENDDATE	DATE	No		Blackout end date.
ENDTIME	NUMBER(6)	No	0	Blackout end time in hhmmss format for daily black out.
BLACKOUTFLAG	CHAR(1)	No	'F'	'F' indicates a flat black out, or 'D' for a daily black out.
ID_ENTITY	VARCHAR2(5)	No		Entity ID of FCDB.
IDSEQUENCE	NUMBER	No		A unique Number.
IDAPP	CHAR(2)	No	'A1'	Id of the application whose transaction needs to be disabled. Currently always to be set 'A1'.

As per the configuration shown below

	TYPEUSER	IDTXN	STARTDATE	STARTTIME	ENDDATE	ENDTIME	BLACKOUTFLAG	ID_ENTITY	IDSEQUENCE	IDAPP
▶ 1	EN1	AAC	1/29/2012	1400	2/3/2012	1600	D	B001	12	A1
2	EN1	ASM	1/29/2012	1400	2/3/2012	1600	F	B001	13	A1

Account Activity (idtxn ‘AAC’) has been configured for black out for retail user (usertype ‘EN1’) from 29th Jan 2012 to 3rd Feb 2012 from 1400 hrs to 1600 hrs daily.

Account Summary (idtxn ‘ASM’) has been configured for full blackout starting from 1400 hrs of 29th Jan 2012 till 1600 hrs of 3rd Feb 2012.

Transaction Configurations

This section discusses the approach for configuring any new transaction on mobile devices, if the corresponding transaction is available in TGT Internet Banking App. A simple use case of “Cheque Book Request” transaction has been discussed below.

First one needs to identify the transaction id of “Cheque Book Request” transaction in TGT App. For this use case assume it as “CQR”.

CQR : Transaction id of Cheque Book Request in TGT App.

One need to then identify similar transaction in FCDB ME App. To get the list of transaction one can refer to following tables in FCDB ME Local Data Base.

- **msttxn**: This is the parent table of FCDB ME app where all the transactions are defined/registered in the app.
- **mstusertypetxn**: This is a child table of **msttxn** where transactions are defined/registered at user segment level (for e.g. ECU i.e. corporate user). To get the list of available usertype/segment refer to table **mstentityusertypes**.

Once it has been identified that a similar transaction exists in FCDB ME App (for this use case assume it as “CBR”), one need to follow the following steps to map the transaction from FCDB ME App to TGT App

CBR : Transaction id of Cheque Book Request in FCDB ME App.

1.11. Transaction Mapping

To map a transaction in FCDB ME App to TGT App one need to make an entry in table *mobileenablertxnlist* as shown below

	ID_ENTITY	USERTYPE	FCDBIDCHANNEL	FCDBIDTXN	TGTAPPIDTXN	TGTIDCHANNEL	ISENABLED	TXNDESCRIPTION
▶ 1	B001	ECU	42	CBR	CQR	01	Y	Cheque Book Request ...
2	B001	ECU	43	CBR	CQR	01	Y	Cheque Book Request ...
3	B001	EN1	42	CBR	CQR	01	Y	Cheque Book Request ...
4	B001	EN1	43	CBR	CQR	01	Y	Cheque Book Request ...

As shown in the screenshot above column FCDBIDTXN contains the transaction id ('CBR') of FCDB ME App and TGTAPPIDTXN contains the transaction id ('CQR') of TGT App. The above mapping needs to be done for the relevant user segment (refer to USERTYPE column of this table) and channel id (refer to FCDBIDCHANNEL column of this table). To get the list of channel id/device id refer to table *mstdevice* in FCDB ME Local Data Base.

After doing the entry in the above table FCDB ME server is required to be restarted, after the server restart the transaction "Cheque Book Request" should be available in the menu of the mobile devices.

Once the transaction is available in menu, on clicking the transaction one may get following message

"Illegal or Invalid Request. Validation Failed." This message signifies that the corresponding idrequest is not registered with the validation engine. Please refer to section [Validation Templates](#) to do the same.

1.12. Request Field Mapping

On clicking the transaction a HTTP request gets posted to FCDB ME App (please refer to [Technical Architecture](#) section), one can check the request fields coming from the mobile devices in “**FLEXCUBEMobileEnablerPlugin.log**” (To enable debug logs please refer to section [logger.properties](#)). The debug log will contain the HTTP request for each request submitted as shown below.

```
“DEBUG com.iflex.fcat.channels.plugins.FCDBMobileEnablerPlugin - HTTP request
Coming from Mobile App >>>>>>>>
fldDataId=Yr1Thbnlc/RMvfIt/8PN2Ixq0/duhx5c38bJ78+ER78=&fldjsessionid=DK0vP1
ZB8h154f0T4sR0ZHnjbnt2N9PKv0Bj0z12QQRvyjGnXQTf&fldServiceType=CBR&ids
equence=g114LuKAXf9wuNza7YnlpxMKCo4pvYfiLggijibPJaU=&datetime=18-05-
2012
10:38:49&fldEntityId=B001&fldDeviceId=42&fldSessionId=KkINPDUjaAdo8o+u2x/K
72j3cBNF2aize6bq4/zTH/w=&fldRequestId=RRCBR01&”
```

fldRequestId: This variable is used to identify the step of workflow, its nomenclature is “RR+<FCDBME IDTXN>+screen sequence id”. So for CBR it will be RRCBR01 for the first screen of Cheque Book Request.

After analyzing the FCDB ME request field’s one need to map the required fields to the relevant field names in TGT App. Mapping of the fields is to be done in **mobileenablerfieldmap** table as shown below

ID_ENTITY	USERTYPE	IDCHANNEL	USERAGENT	FCDBIDTXN	FCDBIDREQUEST	FCDBFIELDNAME	TGTAPPFIELDNAME	TGTFIELDDEFAULTVALUE
1	B001	ECU	43	*	CBR	RRCBR01	fldScrnSeqNbr	01
2	B001	ECU	43	*	CBR	RRCBR01	fldTxnId	CQR
3	B001	ECU	43	*	CBR	RRCBR03	fldacntnumber	
4	B001	ECU	43	*	CBR	RRCBR03	fldbranchcode	
5	B001	ECU	43	*	CBR	RRCBR03	fldcurrcode	
6	B001	ECU	43	*	CBR	RRCBR03	fldacntnumber	
7	B001	ECU	43	*	CBR	RRCBR03	fldchq	
8	B001	ECU	43	*	CBR	RRCBR03	fldScrnSeqNbr	02
9	B001	ECU	43	*	CBR	RRCBR03	fldTxnId	CQR
10	B001	ECU	43	*	CBR	RRCBR03	fldTypeOfBook	C

As shown in the screen shot above “FCDBFIELDNAME” column will contain the name of the request field of FCDB ME App, “TGTAPPFIELDNAME” column will contain the name of TGT App field, mapping these two fields will ensure that field value from FCDME App gets assigned to TGT App field. In few cases “FCDBFIELDNAME” column is blank, in such scenarios one can set a default value for TGT App field using “TGTFIELDDEFAULTVALUE” column.

“FCDBIDREQUEST” column will contain the value of field “**fldRequestId**”. “USERAGENT” column can be used to identify the fields for different devices if required, else it should be defaulted to “*”.

If TGT App expects certain field which is default required for all the transactions then such field mapping can be configured in **mobileenablerdefaultfieldmap** as shown below

	ID_ENTITY	USERTYPE	IDCHANNEL	USERAGENT	FCDBFIELDNAME	TGTAPPFIELDNAME	TGTFIELDDEFAULTVALUE
1	B001	ECU	42	*		fldAppId	CO
2	B001	ECU	42	*		fldDeviceId	01
3	B001	ECU	42	*	fldLangId	fldLangId	eng
4	B001	ECU	43	*		fldAppId	CO
5	B001	ECU	43	*		fldDeviceId	01
6	B001	ECU	43	*	fldLangId	fldLangId	eng

As per the screenshot shown above for each HTTP request going to TGT App, three fields “fldAppId”, “fldDeviceId” and “fldLangId” will always get passed.

Once the field mapping is done one can check the HTTP request going to TGT App in “**FLEXCUBEMobileEnablerPlugin.log**”. The debug log will contain the HTTP request going to TGT App as shown below.

```

“DEBUG com.iflex.fcat.channels.plugins.FCDBMobileEnablerPlugin - HTTP request
Going to Target App >>>>>>>>>
fldScrnSeqNbr=01&fldTxnId=CQR&fldAppId=CO&fldDeviceId=01&fldLangId=eng&f
ldSessionId=P0jg%2B7vlfMIPQj4v6g
”

```

One can also see the HTTP request going to TGT App in “HTTPREQTOTGTAPP” column of table **mobileenablerauditlog**.

1.13. Response XPATH mapping

Once the transaction is successful in TGT App, it will return a XML response. This XML response is captured in “XMLRESPONSEFROMTGTAPP” column of table *mobileenablerauditlog*. This XML response is included in FCDB ME response xml inside the tag <mobileEnablerResp></mobileEnablerResp>. The final XML will be available in the debug log as shown below.

“DEBUG com.iflex.fcat.channels.plugins.FCDBMobileEnablerPlugin - HTTP response in mobile enabler >>>>>>>>> <?xml version="1.0" encoding="UTF-8" ?><faml><header></header><request></request><response><sessioninfo></sessioninfo><mobileenableresp></mobileenableresp></response></faml>”

One needs to refer this response XML to update the XPATH in respective user interface component of the screen getting painted to show the response data on the screen.

To identify a user interface component and update the response XPATH being used to paint the screen, one needs to follow the following steps

- For particular requestId check the entries in table “mstchannelats” table. For e.g. for idrequest “RRCBR01” i.e. first screen of Cheque Book request transaction the entries are as shown below

	IDREQUEST	TYPPLUGIN	AUDITREQUIRED	IDCHANNEL	IDTXN	REQUIRESLOGIN	CONTENTSTYLE	NAMRESOURCE
▶ 1	RRCBR01	C	N	42	CBR	Y	DHTML	genericscreentemplate.xml
2	RRCBR01	C	N	43	CBR	Y	MPXML	genericscreentemplate.xml

- Refer to column “NAMRESOURCE”, this column contain the UI xsl being used to paint the screen. If the namresource is “genericscreentemplate.xml” then the screen is being painted using screen design XML else if the namresource is other than “genericscreentemplate.xml” then the XSL itself contains the screen design code.
- In case namresource is other than “genericscreentemplate.xml” one can find that XSL in FCDB ME setup environment in system/home/datafiles/gui folder. This folder contains all the gui files in gui.zip. One needs to open the zip and find out the respective XSL file in <usertype or parentusertype>/<channel> folder and update the XPATH. As shown below

	IDREQUEST	TYPPLUGIN	AUDITREQUIRED	IDCHANNEL	IDTXN	REQUIRESLOGIN	CONTENTSTYLE	NAMRESOURCE
▶ 1	RRICR01	C	N	42	ICR	Y	HTML	interestratesinquiry.xml
2	RRICR01	C	N	43	ICR	Y	HTML	interestratesinquiry.xml

“interestratesinquiry.xml” can be located in “system\datafiles\gui\ECU\43\template” folder, if not found it can be located in

“system\datafiles\gui\ENU\43\template” folder as ENU is parent of ECU user type.

- In case namresource is “genericsscreentemplate.xml” one can find the respective screen design xml in <usertype or parentusertype>/<channel> folder and update the XPATH. The nomenclature of screen design xml is < idrequest >.xml As shown below

	IDREQUEST	TYPPLUGIN	AUDITREQUIRED	IDCHANNEL	IDTXN	REQUIRESLOGIN	CONTENTSTYLE	NAMRESOURCE
▶ 1	RRCBR01	C	N	42	CBR	Y	DHTML	genericsscreentemplate.xml
2	RRCBR01	C	N	43	CBR	Y	MPXML	genericsscreentemplate.xml

for Cheque Book request first screen design XML “RRCBR01.xml” can be located in “system\datafiles\gui\ECU\43\template” folder, if not found it can be located in “system\datafiles\gui\ENU\43\template” folder as ENU is parent of ECU user type. For more details on how to update the screen design XML please refer to “*User Manual Oracle FLEXCUBE Development Workbench for Direct and Mobile Banking.pdf*”.

Validation Framework

Validation Engine of FCDB ME has the potential to validate the data field values coming from the mobile devices. This framework has the capability of enriching these data fields using custom JAVA/SQL based enricher. This framework can be used to plug custom java validation classes. The key features of validation engine can be summarized as follows:

1. Data Type validations allowing each incoming field to confirm to a predefined data type.
2. Data Length validations using the minimum and maximum lengths defined for the data field.
3. Data Integrity checks disallowing invalid characters in the input (for e.g.: Special Characters allowed for String values)
4. Data Input Format Checks which allows certain data to be imported only in the predefined format (for e.g.: Date Formats etc.)
5. Pluggable JAVA/SQL based enricher to enrich the HTTP request field before sending it to TGT App.
6. Validations for all data fields before error is returned allowing all errors to be returned in a single validation call rather than returning separate errors.
7. Validations against enumeration of values to allow only a certain set of valid values for a given data field. The Validation Engine supports enumerations to check for a list of valid values.
8. One or more Custom Validation can be plugged-in to allow for additional functional validations over and above the required data verification checks.

1.14. Data Types

Data Types identify the supported types of the various data field elements in a request. The data types are defined in the database (FCDB ME LDB) table *data dictionary types* and are associated with an appropriate Java Validator component.

FCDB ME supports a list of standard data types to be used and have the flexibility to support custom data types if required for new data types. For e.g. : The Account Number can be treated as a new data type with specific validator associated with the data type to only validate for the rules for account numbers.

Each validator also returns a typed object, appropriate for the data type, after a successful validation which is the parsed form of the data that has been received. For e.g.: For a DATE data field, the DATE validator shall return an instance of java.lang.Date representing the date value after validating the incoming string date value.

Some of the commonly used data types that are currently supported are as follows.

ALPHABETS (A)

This data type allows values which contain only Alphabets in upper case and lower case. Special characters and Numeric digits are not allowed for this data type.

STRING (S)

This data type allows values which contain Alphabets and Numeric Digits. Special characters are not allowed, by default, for this data type.

The validator for the STRING data type returns the actual value as a String value as the typed object.

NUMERIC (N)

This data type allows all numeric values for a given data field. This includes all integer and floating point values.

The validator for the NUMERIC data type returns the actual value as a java.math.BigDecimal value as the typed object.

FLOATING POINT (N)

This data type allows all numeric values for a given data field. This includes all integer and floating point values.

The functionality is the same as the NUMERIC data type but the only difference is the FLOATING POINT data type validator returns an instance of java.lang.Double as a typed object than a java.math.BigDecimal instance returned by the Numeric Validator.

The validator for the FLOATING POINT data type returns the actual value as a java.lang.Double value as the typed object.

INTEGER (N)

This data type allows all numeric integer values for a given data field.

The validator for the INTEGER data type returns the actual value as a java.lang.Long value as the typed object.

EMAIL (E)

This data type is used to validate for data fields used for email addresses. The validator checks for the required valid fields (e.g: @ and .) at appropriate locations and allows only characters allowed as per the SMTP specification in the email address.

The validator for the EMAIL data type returns the actual value as a String value as the typed object.

DATE (D)

The date type supports validations for date fields. The default incoming date format is expected to be dd/MM/yyyy.

The validator for the DATE data type returns the actual value as a java.util.Date instance as a typed object.

TIMESTAMP (T)

The date type supports validations for date time fields. The default incoming date format is expected to be dd/MM/yyyy.

This data type is exactly similar to the DATE data type but differs only in the instance of the typed object returned by the validator. The validator for the TIMESTAMP data type returns the actual value as a java.sql.Timestamp instance while the DATE validator returns an instance of java.util.Date.

UPPERCASE ALPHABETS (UA)

This data type allows values which contain only Alphabets in UpperCase. Special characters and Numeric digits are not allowed for this data type.

The validator for the UPPERCASE ALPHABETS data type returns the actual value as a String value, as a typed object.

UPPERCASE STRING (US)

This data type allows string values which contain Alphabets in UpperCase only along with numeric digits. Special characters are not allowed by default for this data type.

The validator for the UPPERCASE STRING data type returns the actual value as a String value, as a typed object.

POSITIVE NUMBER (PN)

This data type supports only positive integral values (Numeric Values > 0). Numeric values less than zero shall result in a validation error.

The validator for the POSITIVE NUMBER data type returns the actual value as a java.lang.Long value, as a typed object.

FREETEXT (FT)

This data type allows all characters to be accepted for any given data field including all special characters. The validator associated with this data type is a flow through validator which does not perform any specific data validations. The validator for the FREETEXT data type returns the actual value as a String value, as a typed object.

PATTERN (P)

This datatype allows characters to be accepted for any given data field according to given pattern.

1.15. Data Dictionary

The Data Dictionary is the master set of data elements used within the application. The Data Dictionary identifies the base data type of the data element and also the attributes like the minimum and maximum length of the data. The data dictionary can be defined in *data dictionary* table of FCDB ME LDB.

The Data Dictionary should be defined prior to defining the Validation Templates. Data Dictionary elements can be reused for multiple fields hence applying common validations to fields across various requests.

This table has a column FIELDFORMAT, one can define the allowed formats for following data types

- STRING (S): The FIELDFORMAT can be used to indicate the special characters to be allowed with the Upper Alphabets and the Numeric Digits.
- DATE (D): The FIELDFORMAT can be used to override the default format and provide the required incoming date format.
- TIMESTAMP (T): The FIELDFORMAT can be used to override the default format and provide the required incoming date time format.
- PATTERN (P): The FIELDFORMAT can be used to define the REGEX pattern to be applied.
- POSITIVE NUMBER (PN): The FIELDFORMAT can be used to identify the maximum value that can be accepted by the data field.

Similar column FIELDFORMAT is available in table *txn data* where one can define the similar format as explained above. The format defined in *txn data* will get applied on a particular field and hence will override the value defined in *data dictionary*.

1.16. Validation Templates

The Validation Templates identify the request and its associated fields for any given request. The Validation Templates are defined in the *txn data master* and the *txn data* database tables. *txn data* is the child table of *txn data master*.

“IDREQUEST” column in *txn data master* is the unique identifier for the validation template for each idrequest. FCDB ME App expects for each idrequest an entry in *txn data master* table **even if no data validation is required**. Defining an entry in *txn data master* for each idrequest ensures that the incoming request is registered with the Validation Engine of the application.

If any data validation is to be done entries needs to be done in *txn data* for the fields. The template identifies the request data fields and tags them to the Data Dictionary and the Data Types defined. The template also identifies whether a field is mandatory in the request or not and also whether a field requires data validation or not. One can plug custom java validator, enrichments and define specific error codes for field validations.

The format for idrequest can be

- <<identity>>.<<usertype>>.<<idchannel>>.<<idrequest>> or
- <<identity>>.<<idchannel>>.<<idrequest>> or
- <<identity>>.<<idrequest>> or
- <<usertype>>.<<idchannel>>.<<idrequest>> or
- <<idchannel>>.<<idrequest>> or
- <<idrequest>>

The one found first in the sequence mentioned will be considered by the application. The need for providing the above options is to define different validation templates, if required.

1.17. Validation Configurations

The Validation Engine component drives the complete validation for a given request. The request identifier and the required data values are passed to the Validation Engine to be validated.

The Validation Engine loads the required Validation Template to be used and checks for the data fields defined in the Validation Template. The Validation Engine checks whether a given field is mandatory or required validation before the validation is invoked on the required fields. The engine also checks for any custom validators defined for the data fields and use the same, if required. All errors returned by the validators are accumulated and returned to the caller. The engine indicates a success or failure of the validation and provides details on the errors.

The Validation Engine is invoked inside Mobile Enabler Interfacing tier before creating the HTTP request for TGT App.

1.18. Validators

All Validator components extend the validator interface and provide the required validation logic for the data type and data field validations. Each data type defined in the Data Types section has a corresponding standard validator associated with the same.

The validator uses the configuration and the runtime (data values) passed to it to perform the required validations on the data field. The validators use the FIELDFORMAT field from the Data Dictionary definition or from the Data Element level (if overridden). The interpretation of the FIELDFORMAT field in the definition is as per the data type and the validator for the data types interprets that field for completing the validations.

Custom Validators

The Validation Services support the mechanism of extending the default validators or developing custom validators which may be used for specific scenarios.

The custom validators can be plugged-in for individual data fields defined for the request in the *txn data* table using the CUSTOM_VALIDATOR column. The fully qualified class name of the Validator should be maintained in this location. The Validation Engine uses the Java Reflection API to instantiate the custom validator classes and execute the same. Similarly, the custom validators can be plugged-in for the entire request using the CUSTOM_VALIDATOR field in the *txn data master* database table, for the required Request Identifier.

The basic data type validations are ignored while using a custom validator but the basic field length validations are still performed before passing the request / data field to the custom validator.

The guidelines governing any validation using custom validators are the same as the guidelines for defining standard or generic validations.

The abstract class **com.iflex.fcat.xjava.data.Validator** provides the model for extending validation behavior. This class models the validators to be plugged into the ValidationEngine. All implementing validators should confirm to this interface.

```
public int validate (  
    Hashtable          p_table  
,    String           p_value  
,    TxnData          p_txndata  
,    TxnDataElement   p_txndata_elem  
,    Connection        p_con  
,    ValidationResult p_result  
)
```

This method should be implemented by the subclasses to provide for the actual validation. The method shall return zero (0) in case of a successful validation else it shall return the actual error code. The implementing validator can also add custom error messages to ValidationResult reference available.

Custom Enrichments

Validation Framework supports JAVA/SQL based enrichment for data fields. Enrichment needs to be defined in txn data enrichment table. Once defined the corresponding enrichment needs to be plugged against the corresponding field in txn data table in the column ENRICHMENT.

Most commonly used enrichment in FCDB ME application is “**MOBILEENABLERSPLITTER**”. It’s a java based enricher mapped to java file “**com.iflex.fcat.services.apps.enricher.MobileEnablerSplitter**” as shown in screen shot below for txn data enrichment.

	ENNAME	ENTYPE	ENNUMPARAMS	ENPARAMFIELDS	ENJAVACLASS
▶ 1	MOBILEENABLERSPLITTER	J	0	dummy	com.iflex.fcat.services.apps.enricher.MobileEnablerSplitter

This enricher can be used for any transaction. This enricher split’s the value of the field defined in FREETEXT column and assign it to the field on which this enrichment is plugged. In FREETEXT column, one needs to define the <<field to be splitted>>^<<the delimiter>>^<<position of the value to be picked>>.

	IDREQUEST	FIELDNAME	REFFIELDNAME	ENRICHMENT	FREETEXT
1	RRCBR03	F0009TEMPLATEE	fldacntnumber	MOBILEENABLERSPLITTER	fldacct^^^0
▶ 2	RRCBR03	F0009TEMPLATEE	fldbranchcode	MOBILEENABLERSPLITTER	fldacct^^^2
3	RRCBR03	F0009TEMPLATEE	fldcurrcode	MOBILEENABLERSPLITTER	fldacct^^^4

As shown in the screen shot above for txn data table for idrequest “RRCBR03” “fldbranchcode” value will be enriched by validation engine using **MOBILEENABLERSPLITTER** enricher. This enricher will split the value of “fldacct” having value coming from the screen and assign the token value to variable “fldbranchcode”.

1.19. Error Handling

The Validation Engine provides support for complete validations of all data elements in a request before a validation error is returned. This mechanism allows all the validation errors to be returned to the caller which can be rectified.

Validation Engine will return the predefined error codes and error messages for validation failure of each field configured to go through the validation engine, but one can configure their own error codes also. To configure a custom error message, one can plug their own error code in ERRORCODE column of *txn_data* table against a particular field. For the corresponding error code one can define the error message in *applicationmessage* table.

In FCDB Mobile Enabler currently auditing of data is being done at two stages

1. Channel tier

Auditing at this tier is configurable. The auditing can be controlled at workflow level. The auditing component can be extended to enrich the auditing behavior. As out-of-box, channel tier auditing component allows auditing to table “auditlog” in local database schema.

Auditing component implements interface “**com.iflex.fcat.channels.audtrl.AuditTrailHandler**”. This allows extension and provides capability to extend & develop new auditing behavior.

The audit handler to be used by Channel tier for auditing is configured in XML property file “fcats-config.xml”.

Auditing can be configured at workflow level in column “mstchannelats.auditrequired”.

“Y”: Auditing required

“N”: No Auditing.

2. Mobile Enabler Interfacing Tier

Auditing at this tier is **not** configurable and is **mandatory**. All the data in form of HTTP request going to the TGT App from FCDB MobileEnabler App gets stored in “mobileenablerauditlog” table. The corresponding XML response also gets

stored in the same table. This table also takes care of storing the session id of both the applications.

Data Masking

Data masking for the HTTP request and XML response is available and can be configured for a particular workflow (i.e. idrequest). Using this feature one can disable logging of important data fields in the relevant audit log tables.

To configure masking of sensitive data one need to enter the comma separated list of field names they want to mask, in “*appldata*” table of mobile enabler LDB, for a particular idrequest and for dataname “MASKFIELD”.

As per the configuration shown below

	IDAPP	DATANAME	DATAVALUE	IDLANG	IDDEVICE	VALUESTRING	LOADFLAG
▶ 1	A1	MASKFIELD ...	RRLGN01 ...	eng	***	fldPassword, fldpassword, password, Password ...	Y

data masking has been configure for field name

“fldPassword, fldpassword, password, Password” for idrequest “RRLGN01”.

Mobile Clients

1.20. J2ME Plain client

This client is targeted at lower end mobile phones and provides a very basic user interface and navigations. Any Java enabled Phone that supports MIDP 2.0 and CLDC 1.0 profiles can run this client. Detailed architecture and customization guidelines are provided in the developer guide

[Oracle_FLEXCUBE_Direct_Banking_Mobile_J2ME_Clients_Developer_Guide.pdf](#)

1.21. J2ME Rich client

This client is targeted at higher end mobile phones and provides rich user interface. Any Java enabled Phone that supports MIDP 2.0 and CLDC 1.1 profiles can run this client. Detailed architecture and customization guidelines are provided in the developer guide [Oracle_FLEXCUBE_Direct_Banking_Mobile_J2ME_Clients_Developer_Guide.pdf](#)

1.22. iPhone client

This client is targeted at i-Phones. Detailed architecture and customization guidelines are provided in the developer guide

Oracle_FLEXCUBE_Direct_Banking_Mobile_iPhone_Client_Developer_Guide.pdf

1.23. iPad client

This client is targeted at i-Pads. Detailed architecture and customization guidelines are provided in the developer guide

Oracle_FLEXCUBE_Direct_Banking_Mobile_iPad_Client_Developer_Guide.pdf

1.24. Blackberry client

This client is targeted at Blackberry devices. Detailed architecture and customization guidelines are provided in the developer guide Oracle
Oracle_FLEXCUBE_Direct_Banking_Mobile_Blackberry_Native_Client_Developer_Guide.pdf

1.25. Browser Based Mobile Banking

Oracle FLEXCUBE Direct Banking features can also be accessed over the Browser from a mobile device. This Web based channel provides user interface optimized for mobile browsers. To access browser based mobile banking one need to login using channel.jsp available in FCDB ME deployment B001.war

<https://<<ipaddress of FCDB ME server>>:<<port>>/B001/channel.jsp>