Oracle FLEXCUBE Direct Banking

System Handbook – Volume II – Presentation Layer
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/

Contents

1. Preface

1.1. Intended Audience

This System Handbook (Volume I – Core and Architechture) is intended for the following audience:

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

1.2. 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.

1.3. Access to OFSS Support

https://support.us.oracle.com

1.4. Structure

This document, termed Oracle FLEXCUBE Direct Banking System Handbook, is a single reference for the product information which can be managed, configured, extended, by external parties, to implement, customize or rollout the product to a financial institution.

This is not an Implementation Guide but a System Handbook to explain low level details of how certain key features are implemented within the solution and how these could be extended, customized as appropriate to meet the requirements of the implementation.

This document is intended to provide a set of principles, guidelines and parameters for configuration and extending Oracle FLEXCUBE Direct Banking to meet the . As such, this document does not go into detail regarding the context and background of a number of design decisions but explains the extensibility features and provides insight into the design guidelines and principles for external parties to leverage and develop the required extensions in a non invasive way to the primary features and functionality of the application.

This document is segregated into five Volumes

1	Volume I – Core and Architecture
2	Volume II – Presentation Layer
3	Volume III – Channel Layer
4	Volume IV – Business Service Layer
5	Volume V – Host Interfacing Layer
6	Volume VI – Origination and Peer-to-Peer Payments

1.5. 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 System Handbook – Volume I
- Oracle FLEXCUBE Direct Banking System Handbook – Volume III
- Oracle FLEXCUBE Direct Banking System Handbook – Volume IV
- Oracle FLEXCUBE Direct Banking System Handbook – Volume V
- Oracle FLEXCUBE Direct Banking System Handbook – Volume VI

2. About This Document

2.1. Glossary Of Terms

The following terms are some of the key terms used within the document for identifying the actor for the various actions mentioned within this document.

2.1.1. LICENSEE

The LICENSEE is the Financial Institution, Application Services Provider or the Bank which has licensed the Oracle FLEXCUBE Direct Banking application and shall rollout the solution to its customers as an internet and / or mobile banking channel.

2.1.2. IMPLEMENTER

The IMPLEMENTER is the Implementation Partner, Vendor, Application Service Provider or the LICENSEE themselves who is responsible for rolling out, configuring, extending or developing on Oracle FLEXCUBE Direct Banking.

2.2. TERMINOLOGY

The following terms and terminology is used within the documents to explain underlying processes, components, actions, actors etc.

Term	Definition
Business Service	A Business Service or a Transaction Service is a coarse-grained component that delivers a particular service contract. The Service Interfaces and that make up the contract are each implemented by their particular Service Endpoints.
POJO	A Plain Old Java Object (POJO) is exactly what it says. The term is used to differentiate these simple objects from more specific or complex types such as EJB classes. For example, when creating an EJB, a specific class must implement the SessionBean interface. However, that class will often delegate much of its functionality to one or more POJOs to aid maintainability and reuse of functionality.
Service Implementation or Service Endpoint	A Service Implementation is a concrete implementation of a Service Interface.
Service Interface	A Service Interface is a cohesive set of Service Methods that are grouped together in the anticipation that they will be commonly used together by a

	consumer. For example, the Service Interface for the FundsTransferService would contain a set of Service Methods that perform different types of immediate money transfer between two accounts.
Service Method	A Service Method takes the form of a Java method implemented by the Service Implementation and the Service Delegate. The consumer of the service will invoke one or more Service Methods to help perform part of a business process.
Extension Schema	The **Extension Schema** is a term used for the separate database schema as deployed by Oracle FLEXCUBE Direct Banking to allow IMPLEMENTERS to extend the Oracle FLEXCUBE Direct Banking application as per their needs.

2.3. Abbreviations

Acronyms	Description
FCDB / FC DB / FC Direct Banking / Direct Banking	Oracle FLEXCUBE Direct Banking
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
IDS	Intrusion Detection System

2.4. Conventions

- The diagrams and / or text in this document may contain colour to communicate or highlight additional information. However, the content of this document is retained when rendered without colour. Specific references to colour can be ignored if necessary.

- The technical terminology relating to the Oracle FLEXCUBE Direct Banking solution is aligned as much as possible to standard definitions or should be defined in the Glossary of Terms. Any deviations from standard terminology are either noted in the Terminology Section, or in context of usage.

- Some sections may contain additional notes and caveats included with the body text. For general and contextual information, these notes are contained within document footnotes. Any notes that have important implications or detailed recommendations are denoted by the information symbol (ⓘ). Important caveats are denoted with the warning symbol (💣).

- Some sections may contain examples included with the body text. Such examples are denoted by the use of shading and the introductory word “EXAMPLE”.

3. Presentation Tier

The Presentation Tier acts as the communication end point for the HTTP(S) traffic for both the Internet and Mobile Banking channels. The Presentation Tier converts the request from the HTTP request format to the XML format which is used internally within the Presentation Tier and the Channel Management tier.

3.1. XML Transformation

Oracle FLEXCUBE Direct Banking converts the HTTP request received into an XML representation for internal use. The conversion is basically done in the following structure. The character set of the XML is based on the character set in which the request has been read.

Refer Multi Lingual section for specifics on the character set used in the request processing.

```
<?xml version="1.0" encoding="UTF-8" ?>
<faml>
<header>
<<All HTTP Header Fields Go Here as XML Elements>>
</header>

<request>
<<All HTTP Request Fields Go Here as XML Elements>>
<request>
<response/>
</faml>
```

The HTTP Headers are read and added to the XML request. These can be used, if required later, in the transaction processing.

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>

<header>

<Referer><![CDATA[https://netbanking.demobank.com]]></Referer>

</header>


<request>

<fldLoginUserId><![CDATA[DEMOUSER]]></fldLoginUserId>

<fldLangId><![CDATA[eng]]></fldLangId>

<fldDeviceId><![CDATA[eng]]></fldDeviceId>


<request>

<response/>

</faml>
```

The data undergoes appropriate global validations, as indicated in the Global Security section, 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.

3.2. Global Security

Presentation tier allows configuring data pattern that can be used to detect data injection. The pattern is configured in property file <<entity.xml>>. Presentation tier also performs data sanitization before forming the XML representation from the fields.

3.3. Multi Lingual

Oracle FLEXCUBE Direct Banking 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 following method is used before the request is completely read by the Internet Servlet.

javax.servlet.http.HttpServletRequest.setCharacterEncoding(<<CHARACTERSET>>);

This sets the character encoding of the request to the required value before the request is read by the component.

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.

The Presentation Tier performs global security check for known security vulnerabilities and performs the content generation functions within the solution.

3.4. Request Filters

The Request Filters are abstractions provided within the Presentation Tier architecture to provide for any filtering capability of the requests submitted to the Oracle FLEXCUBE Direct Banking application. Please refer to section on Extension capabilities for detail.

4. Presentation Tier – Extensions and Configurations

The Presentation Tier of Oracle FLEXCUBE Direct Banking provides the communication end point for the internet and mobile banking transactions to be processed.

4.1. Configurations

This tier uses XML based property file <<entity.xml>>. The property file typically contains plug-in configuration & content generator configuration.

Sample property `file F001.xml`

```
<FAML>

<FCAT.ID.ENTITY>F001</FCAT.ID.ENTITY>

.

.

.

<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>
```

💣 Changing any property in <<entity.xml>> will require server restart for change to take effect.

4.2. Extension Request Filters

This class abstracts the basic security and validation services to be supported with the instance of the servlet. The implementation components are expected to provide 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 true and rejects request if the return value is false.

```
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;
}
```

4.3. Extension Certificate Extractor

Oracle FLEXCUBE Direct allows the Certificate Extraction logic, from the HTTP request, to be extended and provided by the implementers of the application. The interface `com.iflex.fcat.gui.CertificateExtractor` can be inherited and extended to provide for any non-standard techniques to be implemented for retrieving

The Certificate Extractor interface provides the following methods which are expected to be implemented by the custom Certificate Extractor implementation.

```
public Object [] getCertificates (

      HttpServletRequest         p_request

);
```

ⓘ Refer the Java API documentation for Oracle FLEXCUBE Direct Banking application for detail information.

Weblogic Certificate Extractor

The Weblogic Certification Extractor `(com.iflex.fcat.gui.WeblogicCertificateExtractor)` is an implementation for the Oracle Application Server (previously BEA Weblogic Server) for retrieving the X509 certificates delivered within the HTTP request.

WebSphere Certificate Extractor

The WebSphere Certification Extractor `(com.iflex.fcat.gui.WebSphereCertificateExtractor)` is an implementation for the IBM WebSphere Application Server for retrieving the X509 certificates delivered within the HTTP request.

4.4. Extension Content Generator

This class abstracts the content generation capability. Each Content generator is attached to a style id. Single content generator can be attached to more than 1 style ids. This configuration is available in <<entity.xml>> file. The style id is mapped to each request and thus denotes the content handler to be invoked for the corresponding request. The configuration resides in column `MSTCHANNELATS.CONTENTSTYLE`.

```
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;


}
```

<<entity.xml>>

```
<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>
```

ⓘ Refer the Java API documentation for Oracle FLEXCUBE Direct Banking application for detail information.

ⓘ Refer to document "**Oracle_FLEXCUBE_Direct_Banking_Parameter_Sheet"** for details on properties to be configured.

4.5. EXTENSION Logger

The Presentation Tier, being a logically and physically separate component uses its own logger for logging information and debug as required. This logger is different from any other loggers implemented in other tiers of Oracle FLEXCUBE Direct Banking.

The property to configure this logger is `FCAT.INTERNETSERVLET.LOGGER` in the <<entity.xml>>. The fully qualified class name of the Logger should be provided as the property value in the <<entity.xml>> configurations file of Presentation Tier.

The default logger used is the Standard Output and Error Logger if this configuration property is not specified.

Oracle FLEXCUBE Direct allows the logging facility to be extended and provided by the implementers of the application. The com.iflex.fcat.gui.Logger component can be inherited and extended to provide for standard logging.

The Logger interface provides the following methods which are expected to be implemented by the custom Logger implementation.

```
public abstract void logInfo (
       String p_component_name
,      String p_msg
);


public abstract void logDebug (
       String p_component_name
,      String p_msg
);


public abstract void logWarning (
       String p_component_name
,      String p_msg
);


public abstract void logError (
```

```
        String p_component_name
,       String p_msg
);
```

ⓘ Refer the Java Documentation for the `com.iflex.fcat.gui.Logger` class for additional documentation on the component.

The Null Logger, Standard Output and Error Logger and Log4JLogger are standard loggers provided by Oracle FLEXCUBE Direct Banking to enabled appropriate logging facilities within the Presentation Tier.

Null Logger

This is a simple logger `(com.iflex.fcat.gui.Logger)` which ignores all logging preformed by Oracle FLEXCUBE Direct Banking. This logger can be configured as a simple way of switching off any logging completely from the Standard Output and Standard Error stream

Standard Output and Error Logger

Log4J Logger

Oracle FLEXCUBE Direct Banking provides a Log4J implementation of the Logger allowing the uses of standard logging facilities and control as per the Apache Log4J implementation. This logger is an optional component which can be configured to leverage on the appender name is GUI for the Presentation Tier Logger implementation using Log4J. The log4j.properties file located in the application home folder.

4.6. Cross Site Scripting Attack Prevention Configuration

At the Presentation Layer, Configuration can be done to prevent Cross Site Scripting Attack. For this **<entity>.xml** file needs to be updated with the following properties

Key	Value
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(,<script,<img
src,javascript,<object]]></FCAT.FILTER.EXP>
```

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

4.7. Widgets

In Dashboard Layout, the user is presented with a default landing page which display a multitude of information. The information is displayed as small snippets of information. These individual snippets are referred to as “Widgets” in Oracle FLEXCUBE Direct Banking. This section describes the configurations required for enabling information to be displayed as “Widgets”.

Database Configurations

MSTWIDGETLAYOUT

This table holds master information about dashboard layout. This table holds information about alignment of Big widgets, width of widget columns. This table holds configurations using which dashboard will be logically divided into Big Area and a Small Area.

Following are the columns and expected values:

Column Name	Description	Expected Value
ID_ENTITY	The Entity ID for which the dashboard layout is to be set.	F001, B001, B002…. (Must be a correct entity configured in MSTENTITYUSERCHANNELS table)
USERTYPE	The User Type for which the dashboard layout is to be set.	ECU, EN1… (Must be a proper user type as configured in MSTENTITYUSERCHANNELS table)
IDCHANNEL	The id channel for which the dashboard layout is to be set.	01, 11… (Must be a proper id channel as configured in MSTENTITYUSERCHANNELS table)
FIXCOLWIDTH	This column indicates the width (in percentage) of Fix column. (Fix column represents Small Area and it cannot contain big widgets).	30, 40… (This must be a positive number less than 100)

BIGCOLWIDTH	This column indicates the width (in percentage) of Big column (Big column represents Big Area and it can contain big widgets).	70, 60… (This must be a positive number less than 100) Total of 'FIXCOLWIDTH' and 'BIGCOLWIDTH' should be less than equal to 100
BIGLEFTCOLWIDTH	This column indicates the width of the left column. This column is present inside Big column. (Here width is in percentage and with respect to Big column)	40, 50… (This must be a positive number less than 100)
BIGRIGHTCOLWIDTH	This column indicates the width of the right column. This column is present inside Big column. (Here width is in percentage and with respect to Big column)	60, 50… (This must be a positive number less than 100) Total of 'BIGLEFTCOLWIDTH' and 'BIGRIGHTCOLWIDTH' should be less than equal to 100.
FIXCOL_LOC	This column determines location of Fix column on a dashboard layout.	Either of 'L' or 'R'.

MSTWIDGET

This table holds the information of the location and sequence in which the Widgets are to be displayed. This table depends on MSTWIDGETLAYOUT for primary details.

Following are the columns and the expected values:

Column Name	Description	Expected Value
ID_ENTITY	The Entity ID for which the widget is to be configured.	F001, B001, B002…. (Must be a correct entity configured in MSTENTITYUSERCHANNELS table)
USERTYPE	The User Type for which the widget is to be configured.	ECU, EN1… (Must be a proper user type as configured in MSTENTITYUSERCHANNELS table)
WIDGET_LOCATION	Identified the column under which the widget is to be displayed. All 'B' will sequentially appear in 'Big Area'. Case 1: MSTWIDGETLAYOUT – FIXCOL_LOC value is 'R' then 'L' and 'C' will come under 'B' in left and right column respectively. Case 2: MSTWIDGETLAYOUT – FIXCOL_LOC value is 'L' then 'C' and 'R' will come under 'B' in left and right column respectively.	The values currently supported are: 1. 'L' – Left Panel 2. 'C' – Center Panel 3. 'R' – Right Panel 4. 'B' – Big Panel 5. 'F' – Bottom Panel

	All 'F' will sequentially come at the bottom.	
WIDGET_TYPE	The type of widget. Specifies whether the widget is an internal widget or an external widget.	The values currently supported are: 1. 'I' – Internal transactions (available in FCDB) 2. 'E' – External links (Currently only supports static URLs) 3. 'S' – Script based widgets (typically embeddable Google Gadgets)
WIDGET_SEQ	A unique sequence number in which the widgets would be displayed.	Integer specifying position in the appropriate WIDGET_LOCATION.
IDREQUEST	The IDREQUEST to be invoked in case of WIDGET_TYPE='I'. Must be non-null value if WIDGET_TYPE is 'I'.	Must hold a valid IDREQUEST from MSTACHANNELATS if WIDGET_TYPE is 'I'
EXTERNAL_INFO	Holds the URL information for WIDGET_TYPE='E' or the script tag information for WIDGET_TYPE='S'	Must hold a proper URL in case WIDGET_TYPE is 'E' Must hold a script tag if WIDGET_TYPE is 'S'
HASNAVIGATION	Indicates whether the top navigation bar on a widget (including the title bar) is required.	Possible values 'Y' and 'N'
MINIMIZABLE	Indicates whether the widget can be minimized.	Possible values 'Y' and 'N'
REFRESHABLE	Indicates whether the widget can be refreshed.	Possible values 'Y' and 'N'
CUSTOM_CONTENT_CLASS	Allows assigning custom class to the content pane of the widget.	Text specifying CSS class
UDF1	User defined field to be used in the future.	Free text
UDF2	User defined field to be used in the future.	Free text
UDF3	User defined field to be used in the future.	Free text
UDF4	User defined field to be used in the future.	Free text
UDF5	User defined field to be used in the future.	Free text
ISENABLED	Determines if the widget is enabled.	The values currently supported are: 1. 'Y' – Widget is enabled and user will get this widget on dashboard at first time login. 2. 'Z' – Widget is enabled but it will not be displayed on dashboard

		at first time login of the user. User can select this widget using Dashboard Widget Management Transaction. 3. 'N' – Widget is disabled.
IDTXN	The transaction id of the widget.	Must hold a valid IDTXN from MSTTXN if WIDGET_TYPE is 'I'
IDCHANNEL	The channel id for which widget is enabled.	Must hold a valid IDCHANNEL from MSTDEVICE.
WIDGET_HEIGHT	This column indicates height of the widget (in pixel).	A positive number.
MORELINK	This column indicates a flag for enabling 'More' link on a widget.	Possible values 'Y' and 'N'
ISMANDATORY	Determines if the widget is mandatory. Bank can set a particular widget as mandatory. For e.g. Advertisements.	Possible values 'Y' and 'N'

Widget Names

APPLDATA ENTRIES FOR WIDGET_DESC

The widget name is fetched from APPLDATA for DATANAME 'WIDGET_DESC'. The key for fetching widgets' name is '<USERTYPE>.<REQUEST_ID>'. Entity can be specialized for particular entity using key as '<ENTITY_ID>.<USERTYPE>.<REQUEST_ID>'.

The below table explains the values to be entered for each column in APPLDATA:

APPLDATA Table Column Name	Value for WIDGET_DESC
IDAPP	A1
DATANAME	WIDGET_DESC
DATAVALUE	Default value:- <USERTYPE>.<REQUEST_ID> Entity Specific value:- <ENTITY_ID>.<USERTYPE>.<REQUEST_ID>
IDLANG	Value as per requirement from mstlang.idlang column.
IDDEVICE	Value as per requirement from mstdevice.iddevice column. This column may contain value "**" if the description is same for all devices.
VALUESTRING	The name of the widget to be displayed to the user.
LOADFLAG	Y

APPLDATA ENTRIES FOR WIDGET_SUMMARY

The widget description is fetched from APPLDATA for DATANAME 'WIDGET_SUMMARY'. The key for fetching widgets' description is '<USERTYPE>.<REQUEST_ID>'. Entity can be specialized for particular entity using key as '<ENTITY_ID>.<USERTYPE>.<REQUEST_ID>'.

The below table explains the values to be entered for each column in APPLDATA:

APPLDATA Table Column Name	Value for WIDGET_SUMMARY
IDAPP	A1
DATANAME	WIDGET_SUMMARY
DATAVALUE	Default value:- <USERTYPE>.<REQUEST_ID> Entity Specific value:- <ENTITY_ID>.<USERTYPE>.<REQUEST_ID>
IDLANG	Value as per requirement from mstlang.idlang column.
IDDEVICE	Value as per requirement from mstdevice.iddevice column. This column may contain value "**" if the description is same for all devices.
VALUESTRING	The summary of the widget to be displayed to the user.
LOADFLAG	Y

TransacTion Changes

This section outlines the transaction level changes required to design a new Widget.

For designing a new Widget, the following points need to be ensured:

1. The existing transactional services can be used as is without any changes as the service to be called for Widgets.
2. Widgets can be Internal or External
    a. Internal Widgets: An internal widget is a widget developed using the existing FCDB framework (MSTWIDGET.WIDGET_TYPE='I'). Following are the design guidelines to be followed for developing a new widget:
        i. An entry in MSTWIDGET is required with WIDGET_TYPE='I',
        ii. An additional MSTCHANNELATS entry is required which specifies the custom XSL to be used for rendering the widget. The service XSL may or may not be same as the standard transaction's XSL based on the requirement.
        iii. The service must not differentiate its behavior for Widgets. If differential processing is required, a new service must be introduced for the same.
        iv. All requests fired from within a widget must always fire an AJAX request.
        v. The MSTCHANNELATS.NAMEOTRESOURCE for all widget transactions must be 'widgeteot.xsl' unless a specific EOT XSL is required.
        vi. All Internal Widget transactions must have unique MSTCHANNELATS entries and each entry must have MSTCHANNELATS.TYPE_REQUEST as 'W'.
    b. External Widgets: An External Widget is a widget that is not developed in FCDB framework, however, is simply a pointer to an external service. An External Widget may be further classified as under.
        i. Script Based: External Widgets that provide a script tag to be embedded in the web page for displaying the widget are supported via Script Based External Widgets (MSTWIDGET.WIDGET_TYPE='S')
        ii. Static URL Based: External Widgets that are rendered in an IFRAME such that the IFRAME's URL is provided as a parameter of the widget are supported by Static URL Based External Widgets (MSTWIDGET.WIDGET_TYPE='E')

Dashboard widget management transaction

This section outlines the transaction for user's widget preferences to be shown on dashboard.

There is one transaction (defaulted at application level) under customer services named '**Dashboard Widget Management**', which will help users to choose the widgets they want to show on their dashboards. Users can set their widget preferences for different channels using this transaction. Once saved user need to re-login to get the widgets on the dashboard as per their preference.

DATABASE CONFIGURATIONS

MSTWIDGETUSERPREF

This table holds user preferences for dashboard widgets. Following are the columns and expected values:

Column Name	Description	Expected Value
IDUSER	The User ID for which the preferences are being set.	Must be a proper IDUSER configured in MSTCHANNELUSER table
ID_ENTITY	The Entity ID for which the widget is to be configured.	F001, B001, B002…. (Must be a correct entity configured in MSTENTITYUSERCHANNELS table)
USERTYPE	The User Type for which the widget is to be configured.	ECU, EN1… (Must be a proper user type as configured in MSTENTITYUSERCHANNELS table)
IDCHANNEL	The channel id for which widget is enabled.	Must hold a valid IDCHANNEL from MSTDEVICE.
IDTXN	The transaction id of the widget.	Must hold a valid IDTXN from MSTTXN if WIDGET_TYPE is 'I'
IDREQUEST	The IDREQUEST to be invoked in case of WIDGET_TYPE='I'. Must be non-null value if WIDGET_TYPE is 'I'.	Must hold a valid IDREQUEST from MSTACHANNELATS if WIDGET_TYPE is 'I'

WIDGET_LOCATION	Identified the column under which the widget is to be displayed. All 'B' will sequentially appear in 'Big Area'. Case 1: MSTWIDGETLAYOUT – FIXCOL_LOC value is 'R' then 'L' and 'C' will come under 'B' in left and right column respectively. Case 2: MSTWIDGETLAYOUT – FIXCOL_LOC value is 'L' then 'C' and 'R' will come under 'B' in left and right column respectively. All 'F' will sequentially come at the bottom.	The values currently supported are: 1. 'L' – Left Panel 2. 'C' – Center Panel 3. 'R' – Right Panel 4. 'B' – Big Panel 5. 'F' – Bottom Panel
WIDGET_SEQ	A unique sequence number in which the widgets would be displayed.	Integer specifying position in the appropriate WIDGET_LOCATION.
ISSELECTED	Determines if the widget is selected.	Possible values 'Y' and 'N'

This table is meant for FCDB application for storing the widget preferences set by the user using the menu transaction '**Dashboard Widget Management**'. This table is not meant for any operation through backend.

TRANSACTION CONFIGURATION

This section outlines the configurations required for this transaction.

MSTPROPERTIES FOR MAX WIDGETS ALLOWED

There is one check of maximum number of widgets a user can select. This limit is fetched from MSTPROPERTIES for PROPNAME 'MAX_USER_PREF_WIDGETS'. The key for fetching the limit is 'MAX_USER_PREF_WIDGETS'. Entity/User Type/Channel can be specialized for particular entity/usertype/channel using key as '<ENTITY_ID>.<USERTYPE>.<IDCHANNEL>.MAX_USER_PREF_WIDGETS'.

The below table explains the values to be entered for each column in MSTPROPERTIES:

MSTPROPERTIES Table Column Name	Value for PROPNAME
IDSERVER	ZZ
PROPNAME	Default name:- MAX_USER_PREF_WIDGETS Entity Specific value:- <ENTITY_ID>.<USERTYPE>.<IDCHANNEL>.MAX_USER_PREF_WIDGETS
PROPVALUE	Some numeric value (Maximum widgets user can select including mandatory widgets set by Bank).
ENABLED	Possible values 'Y' and 'N'
ISMODIFIED	Possible values 'Y' and 'N'
DATEFFECTIVE	System Date.
DATMODIFIED	System Date.
ISGUIENABLED	Possible values 'Y' and 'N'

4.8. Transaction Quick Access

Oracle FLEXCUBE Direct Banking application allows its user to have quick and easy access to the transactions using the following three default functionalities.

1. Quick Tasks

2. Quick Tools
3. Default Navigation

Transactions to be made available using these functionalities can be configured at day zero. Transaction access will be followed while displaying the transactions.

QUICK TASKS

This functionality is available in form of dashboard widget. 'Quick Tasks' widget is default transaction. This widget need to be configured for 'Entity-Usertype' in 'mstwidgets' table. (Please refer section '9.8.1.2' for widget configuration).

'Quick Tasks' widget contains transaction list ('Tasks'). The user is allowed to click on name of the 'Task' and he will be navigated to parent transaction corresponding to that 'Task'.

A transactions access mechanism has been followed while displaying the 'Tasks' in 'Quick Task' widget.

DATABASE CONFIGURATIONS

MSTUSERTYPETXN

This table makes 'Task' available in 'Quick Tasks' widget.

Column Name	Description	Expected Value
QUICKTASK	Using this column a transaction can be made available for a 'Quick Tasks' widget for respective Entity, Usertype and Channel.	'Y' and 'N'. Y: Transaction will be considered as 'Task'. N: Transaction will not be considered as 'Task'.

APPLDATA

The 'Task' name is fetched from APPLDATA for DATANAME 'TXN_DESC_QK'. The key for fetching 'Task' name is '<ID_TXN>'

APPLDATA Table Column Name	Value for WIDGET_DESC
IDAPP	A1
DATANAME	TXN_DESC_QK
DATAVALUE	ID_TXN
IDLANG	Value as per requirement from mstlang.idlang column.
IDDEVICE	Value as per requirement from mstdevice.iddevice column. This column may contain value "**" if the description is same for all devices.
VALUESTRING	The name of the 'Task' to be displayed to the user.

LOADFLAG	Y

** If specific ‘Task’ name is not configured then ‘Task’ name will be same as ‘Transaction Name’.

QUICK TOOLS

This functionality is available in form of dashboard widget. ‘Quick Tools’ widget is default transaction. This widget need to be configured for ‘Entity-Usertype’ in ‘mstwidgets’ table. (Please refer section ‘9.8.1.2’ for widget configuration).

‘Quick Tools’ widget contains transaction (‘Tools’) in form of a drop down list. The user is allowed to select the ‘Tool’ from a list and on clicking the ‘Go’ button he will be navigated to parent transaction corresponding to that ‘Tool’.

A transactions access mechanism has been followed while displaying the ‘Tool’ in ‘Quick Tools’ widget.

DATABASE CONFIGURATIONS

MSTUSERTYPETXN

This table makes ‘Tool’ available in ‘Quick Tools’ widget.

Column Name	Description	Expected Value
ISTOOL	Using this column a transaction can be made available for a ‘Quick Tools’ widget for respective Entity, Usertype and Channel.	‘Y’ and ‘N’. Y: Transaction will be considered as ‘Tool’. N: Transaction will not be considered as ‘Tool’.

APPLDATA

The ‘Tool’ name is fetched from APPLDATA for DATANAME ‘TXN_DESC_TL’. The key for fetching ‘Tool’ name is ‘<ID_TXN>’

APPLDATA Table Column Name	Value for WIDGET_DESC
IDAPP	A1
DATANAME	TXN_DESC_TL
DATAVALUE	ID_TXN
IDLANG	Value as per requirement from mstlang.idlang column.
IDDEVICE	Value as per requirement from mstdevice.iddevice column. This

	column may contain value "**" if the description is same for all devices.
VALUESTRING	The name of the 'Tool' to be displayed to the user.
LOADFLAG	Y

** If a specific 'Tool' name is not configured then 'Tool' name will be same as 'Transaction Name'.

Default navigation

'Default Navigation' functionality is available on post login window of Oracle FLEXCUBE Direct Banking application. Using this functionality a quick and easy access to transaction will be possible. Such transactions will be available in form of hyperlink at the top of post login window.

A transactions access mechanism has been followed while displaying the transactions in 'Default Navigation' list.

DATABASE CONFIGURATIONS

MSTUSERTYPETXN

This table makes a transaction available for 'Default Navigation' in post login window.

Column Name	Description	Expected Value
ISDEFAULTNAV	Using this column a transaction can be made available for a 'Default Navigation' for respective Entity, Usertype and Channel.	'Y' and 'N'. Y: Transaction will be considered in 'Default Navigation'. N: Transaction will not be considered in 'Default Navigation'.

APPLDATA

The transaction name is fetched from APPLDATA for DATANAME 'TXN_DESC_DN'. The key for fetching transaction name is '<ID_TXN>'

APPLDATA Table Column Name	Value for WIDGET_DESC
IDAPP	A1
DATANAME	TXN_DESC_DN
DATAVALUE	ID_TXN
IDLANG	Value as per requirement from mstlang.idlang column.
IDDEVICE	Value as per requirement from mstdevice.iddevice column. This column may contain value "**" if the description is same for all devices.
VALUESTRING	The name of the transaction to be displayed to the user.

LOADFLAG	Y

** If a specific transaction name is not configured (for the ‘Default Navigation’ functionality) then transaction name will be as same as it present in ‘Menu’.

5. User Interface

Oracle FLEXCUBE Direct Banking provides a simple, yet rich and flexible User Interface to meet the internet banking and mobile banking requirements of financial institutions.

The *Oracle_FLEXCUBE_Direct_Banking_UserInterface_Guide* provides details on configuring, customizing the user interface to meet the branding and user interface guidelines. The document is a reference guide to all implementers to understand and configure the user interface to meet their branding needs as well as guidelines to extend the user interface

ⓘ Refer the **Oracle FLEXCUBE Direct Banking User Interface Guide** document for managing the branding, user interface layout and the various user interface changes within Oracle FLEXCUBE Direct Banking.

💣 It is the responsibility of the IMPLEMENTER and the LICENSEE together to support the User Interface in case of changes done to any of the components within the User Interface. Oracle FLEXCUBE Direct Banking shall provide the required sources that are used for rendering the User Interface or Look-n-Feel etc. and any changes to the same shall require to be maintained by the IMPLEMENTER or LICENSEE themselves.

5.1. Login Screen

The login screen is to be designed by the Bank. The snippets provided can be included to provide the "login box" on the Bank's login page.

Following snippet should be inserted in the HEAD section of the Login page HTML:

```
	<script language="JavaScript" type="text/JavaScript"
src="jsdir/virtualkeyboard.js"></script>

	<script language="JavaScript" type="text/JavaScript"
src='jsdir/security.js'></script>

	<script language="JavaScript" type="text/JavaScript"
src="jsdir/common.js"></script>

	<script Language="JavaScript">
```

```
	function fLogon() {


		var l_encPass = encrypt (document.frmLogon.fldPassword.value,
document.frmLogon.fldEncrKey.value);
		document.frmLogon.fldPassword.value = l_encPass;
		l_encPass = null;


		var scr_w, scr_w1;
		var scr_h, scr_h1;
		scr_w1 = 1015;
		scr_h1 = 740;


		if (scr_w == '800') {
			scr_w1 = 785;
			scr_h1 = 500;
		}
		if (document.frmLogon.fldLoginUserId.value == "") {
			alert ("User Id must be entered");
			document.frmLogon.fldLoginUserId.focus ();
			return false;
		}
		if (document.frmLogon.fldPassword.value == "") {
			alert ("Password must be entered");
			document.frmLogon.fldPassword.focus ();
			return false;
		}
		if(typeof loginWindow != 'undefined'){
			try {
				loginWindow.close();
			} catch(e) {
```

```
                        }

                }


                var scrW = screen.availWidth-10+"px";

                var scrH = screen.availHeight-60+"px";

                var windowName =
document.frmLogon.fldLoginUserId.value.replace(/[^a-zA-z0-9]+/g,"a");

                loginWindow = window.open ("", windowName ,

                        "dependant=no,directories=no,location=no,menubar=yes"

                +       ",resizable=no,scrollbars=yes,titlebar=no,toolbar=no,"

                + "0, 0, top=0,left=0,status=1,"

                + "width=" + (scrW)

                + ",height=" + (scrH));


                document.frmLogon.target = windowName ;

                document.frmLogon.submit ();

                loginWindow.focus();


                document.frmLogon.fldLoginUserId.value = "";

                document.frmLogon.fldPassword.value = "";


                return false;

        }

//-----------------------------------------------------------------------------

        function DeletePwd()

        {

                if (document.frmLogon.chksecmod.checked) {

                        var strNumField = new
String(document.frmLogon.fldPassword.value);

                        document.frmLogon.fldPassword.value =
strNumField.substring(0,strNumField.length-1);
```

```
                        document.frmLogon.fldPassword.id="";
                        doRandomize();
                }
                return false;
        }
//-------------------------------------------------------------------------
        function ClearPwd()
        {
                if (document.frmLogon.chksecmod.checked) {
                        document.frmLogon.fldPassword.value = "";
                        doRandomize();
                }
                return false;
        }
//-------------------------------------------------------------------------
        function SecurityWarning() {
                var keyBoard = document.getElementById('login-
keyboard').getElementsByTagName("input");
                if (document.frmLogon.chksecmod.checked) {
                        document.frmLogon.elemC4.checked =true;
                        document.frmLogon.elemC4.disabled = false;
                        document.frmLogon.fldPassword.readOnly =true;
                        document.frmLogon.fldPassword.blur();
                        for(i=0; i<=keyBoard.length-1; i++){
                                keyBoard[i].disabled=false;
                                keyBoard[i].style.className='enableText';
                        }
                } else {
                        document.frmLogon.elemC4.checked =false;
                        document.frmLogon.elemC4.disabled = true;
```

```
                        document.frmLogon.fldPassword.readOnly = false;

                        for(i=0; i<=keyBoard.length-1; i++){

                                    keyBoard[i].disabled=true;

                                    keyBoard[i].style.className='disableText';

                        }

                        if (false == confirm("Use of virtual keyboard is
recommended to protect your password. \nChoose Cancel to use virtual keyboard
or choose OK to type.")) {

                              document.frmLogon.chksecmod.checked = true;




                        }

                }

                document.frmLogon.fldPassword.focus();

        }

//---------------------------------------------------------------------------

        function initialize() {


                getToken();

                document.frmLogon.fldPassword.focus ();

                document.frmLogon.fldLoginUserId.focus ();

                if (document.frmLogon.chksecmod.checked) {

                        document.frmLogon.fldPassword.readOnly =true;

                        document.frmLogon.fldPassword.blur();

                } else {

                        document.frmLogon.fldPassword.readOnly = false;

                        var keyBoard = document.getElementById('login-
keyboard').getElementsByTagName("BUTTON");

                        for(i=0; i<=keyBoard.length-1; i++){

                                    keyBoard[i].disabled=true;

                                    keyBoard[i].style.color='#AAAAAA';
```

```
                        }

                }

        }

//-------------------------------------------------------------------------------

        function getToken() {

                document.frmtoken.submit();

        }
```

Following snippet should be inserted in the HEAD section of the Login page HTML:

```
<form name="frmLogon" method="post" action="internet" AUTOCOMPLETE="off">

<table border="0" cellspacing="0" cellpadding="5" align="left" width="100%">

        <tr>

                <td width="40%" style="white-space:nowrap;"> </td>

                <td class="LoginTextBold white" nowrap="true" align="right"
style="text-align:left">User ID:</td>

                <td align="left" colspan="1" width="45%" style="text-align:left">

                        <input value="" class="inputbg" name="fldLoginUserId"
tabIndex="1">

                        <select name="fldlitever">

                                <option value="">Select</option>

                                <option value="C">Contemporary</option>

                                <option value="L">Classic</option>

                        </select>

                </td>

        </tr>

        <tr><td style="height:10px;"> </td></tr>

        <tr>

                <td style="text-align:left; padding-left:20px">

                        <input name="chksecmod" type="checkbox" checked='true'
```

```
id="securitykeyboard" onClick= "return SecurityWarning();"  tabIndex="4"/>

                    <label for="usekey" class="DataWhite">Use virtual
keyboard</label>

             </td>

             <td class="LoginTextBold" style="text-align:left">Password:</td>

             <td align="left" colspan="1" style="white-space:nowrap; text-
align:left">

                    <input type="password" class="inputbg" onpaste="return
false;" oncopy="return false;" value="" name="fldPassword"tabIndex="2">

                    <input name="imageField" type="submit" value=""
class="buttsignin" onClick=" return fLogon()" tabIndex="3">

             </td>

      </tr>

      <tr>

             <td style="text-align:left; padding-left:20px">

                    <input id="elemC4" name="elemC4" type="checkbox"/>

                    <nobr>

                           <label class="DataWhite"> Click here to enter by
hovering</label>

                    </nobr>

             </td>

             <td colspan="3"> </td>

      </tr>

      <tr>

             <td colspan="3" align="center">

                    <div class="virtualBox">

                           <div class="contentBox">

                                  <div id="login-keyboard"
style="position:relative; left:0px; top:0px; z-index:3">

                                         <table cellpadding="0" cellspacing="0"
align="center">

                                                <tr>

                                                       <td align="center"
```

```
align="center">
                                                                <div id="login-
keyboard-special">
                                                                    <table
cellpadding="1" cellspacing="1" width="75%">
                                                                        <tr>

        <SCRIPT LANGUAGE="JavaScript">

        for (var i=0; i < 13; i++) {

                document.write ('<td>');

                imageSpCharOnPage(i);

                document.write ('</td>');

        }

        </SCRIPT>

                                                                        </tr>
                                                                    </table>
                                                                </div>
                                                            </td>
                                                            <td rowspan="3"
style="vertical-align:top;">
                                                                <div id="login-
keyboard-numeric" style="">
                                                                    <table
cellpadding="1" cellspacing="1">
                                                                        <tr>


        <SCRIPT LANGUAGE="JavaScript">
```

```
for (var i=0; i < 3; i++) {

        document.write ('<td>');

        imageNumOnPage(i);

        document.write ('</td>');

}

</SCRIPT>
                                                                    </tr>

                                                                    <tr>

<SCRIPT LANGUAGE="JavaScript">

for (var i=3; i < 6; i++) {

        document.write ('<td>');

        imageNumOnPage(i);

        document.write ('</td>');

}

</SCRIPT>
                                                                    </tr>

                                                                    <tr>

<SCRIPT LANGUAGE="JavaScript">

for (var i=6; i < 9; i++) {

        document.write ('<td>');

        imageNumOnPage(i);
```

```
        document.write ('</td>');

}

</SCRIPT>

                                                                    </tr>

                                                                    <tr>

<td> </td>

<SCRIPT LANGUAGE="JavaScript">

for (var i=9; i < 10; i++) {

        document.write ('<td>');

        imageNumOnPage(i);

        document.write ('</td>');

}

</SCRIPT>

<td> </td>

                                                                    </tr>

                                                                </table>

                                                            </div>

                                                        </td>

                                                    </tr>

                                                    <tr>

                                                        <td valign="top"
```

```
style="text-align:center; width:75%">
                                                        <div id="login-
keyboard-alpha">
                                                            <table
cellpadding="1" cellspacing="1" align="center">
                                                                <tr>
                                                                <td
align="center">
                                                                <table
cellpadding="1" cellspacing="1" align="center">

                                                                <tr>

       <SCRIPT LANGUAGE="JavaScript">

       for (var i=0; i < 10; i++) {

              document.write ('<td>');

              imageAlphaOnPage(i);

              document.write ('</td>');

       }

       </SCRIPT>

       </tr></table></td>
                                                                </tr>
                                                                <tr>
                                                                <td
align="center" style="width:75%; background:none">
                                                                <table
cellpadding="1" cellspacing="1" align="center">
                                                                <tr>
```

```
	<SCRIPT LANGUAGE="JavaScript">

	for (var i=10; i < 19; i++) {

		document.write ('<td>');

		imageAlphaOnPage(i);

		document.write ('</td>');

	}

	</SCRIPT>

	</tr></table></td>

                                                                        </tr>

                                                                        <tr>

                                                                        <td
align="center" style="width:75%; background:none">

                                                                        <table
cellpadding="1" cellspacing="1" align="center">

                                                                        <tr>

	<SCRIPT LANGUAGE="JavaScript">

	for (var i=19; i < 26; i++) {

		document.write ('<td>');

		imageAlphaOnPage(i);

		document.write ('</td>');

	}

	</SCRIPT>
```

```
      </tr></table></td>
                                                                    </tr>
                                                                </table>
                                                            </div>
                                                        </td>
                                                    </tr>
                                                    <tr>
                                                        <td style="text-
align:center; width:80%;">
                                                            <table
cellpadding="1" cellspacing="1" align="center">
                                                                <tr>
                                                                    <td>

      <input type="button" class="buttons" value="Upper" id="elemC0"
onMouseDown="return changeToStar();" onMouseUp="return changeBack();"
onMouseOut="return changeBack();" OnClick="setCase();" />

                                                                    </td>
                                                                    <td>

      <input type="button" class="buttons" value="Delete" id="elemC1"
onMouseDown="return changeToStar();" onMouseUp="return changeBack();"
onMouseOut="return changeBack();" OnClick="DeletePwd();" />

                                                                    </td>
                                                                    <td>


      <input type="button" class="buttons" value="Clear All" id="elemC2"
onMouseDown="return changeToStar();" onMouseUp="return changeBack();"
onMouseOut="return changeBack();" OnClick="ClearPwd();" />

                                                                    </td>
                                                                    <td>

      <input type="button" class="buttons" value="Not Mixed" id="elemC3"
```

```
onMouseDown="return changeToStar();" onMouseUp="return changeBack();"
onMouseOut="return changeBack();" OnClick="setRandom();" />
                                                                                    </td>
                                                                                </tr>
                                                                            </table>
                                                                        </td>
                                                                    </tr>
                                                                </table>
                                                            </div>
                                                        </div>
                                                    </div>
                                                </td>
                                            </tr>
                                        </table>
                                        <input type="hidden" name="fldDeviceId" value="01" />
                                        <input type="hidden" name="fldLangId" value="eng" />
                                        <input type="hidden" name="fldRequestId" value="RRLGN01" />
                                        <input type="hidden" name="fldEncrKey" value="" />
                                    </form>
                                    <form name="frmtoken" method="post" action="internet" target="frmnew">
                                        <input type="hidden" name="fldDeviceId" value="01" />
                                        <input type="hidden" name="fldLangId" value="eng" />
                                        <input type="hidden" name="fldRequestId" value="RRLGN00"/>
                                        <input type="hidden" name="fldData" />
                                    </form>
                                    <iframe name="frmnew" width="0px" height="0px" class="iframe" marginwidth="0px"
                                    marginheight="0px" scrolling="no" frameborder="1" src="trans.htm">
                                    </iframe>
```

5.2. Help Integration

Oracle FLEXCUBE Direct Banking provides Transaction Specific Help for each transaction. To integrate the help all the Help files must be named as “<<idtxn>>.html” and must be maintained by the Bank in a static folder outside the application deployment. The path of the internet accessible help files must be provided in properties as described below

Property Name	Property Value
<<**entity id**>>.<<**user type**>>**.HELP.PATH**	The path at which the static HTML help files are available publicly. (ex.: /help/HTMLS/ if the out-of-the-box help.war is deployed)

5.3. User Interface Layout

The application supports “Contemporary” as well as “Classic” (no Dashboard) version. The following configurations are done to enable the option of selecting between “Contemporary” and “Classic” layouts in login JSP page:

```
<select name="fldlitever">

        <option value="">Select</option>

        <option value="C">Contemporary</option>

        <option value="L">Classic</option>

</select>
```

Available menu types for above modes are:

Mode	Menu Type	Description
Contemporary	D	Dashboard - Taskbar based
	DT	Dashboard - Top based
	DL	Dashboard - Left based
Classic	L	Left menu based
	T	Top menu based
	TL	Group menu based

User need to select classic or contemporary option to login into respective mode. If the user logs in with the default selection option (i.e. “Select”). The menu type will be selected as follows.

- The menu type user has set in the preferences.
- If the user has not set any option in preferences the menu type will be selected as per the value in ‘MENUTYPE’ column of the table ‘MSTENTITYUSERCHANNEL’.

For a particular user type, if Contemporary menu type is set in ‘MSTENTITYUSERCHANNEL’ (i.e. D, DT or DL) and if user of that user type selects “Classic” as login style then the following property needs to be added into MSTPROPERTIES to select menu type for classic version.

Property Name	Property Value
LOGIN.MODE.MAP.L	The value can either be ‘T’, ‘TL’ or ‘L’ based on which layout would be supported for classic version.

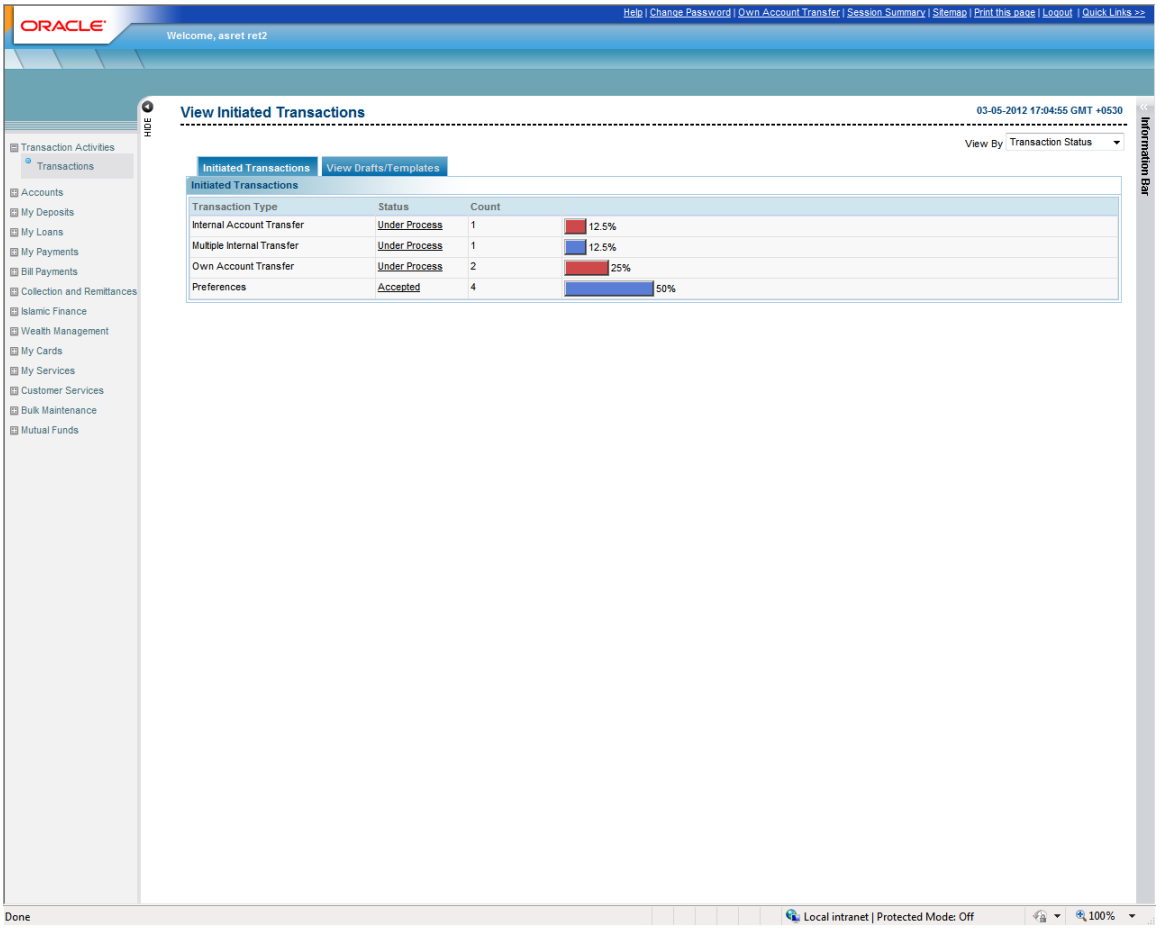
In case if the implementation for contemporary menu type is not required, then the option has to be removed from the dropdown in login page. (i.e. for particular user type if classic menu type is set in the table ‘MSTENTITYUSERCHANNEL’ then the contemporary option should not be available for user on login screen.)

Following is the description for all the available menu types mentioned above.

Left Menu Based

This menu style can be achieved by configuring ‘L’ in ‘MENUTYPE’ column of the table ‘MSTENTITYUSERCHANNEL’ at ‘entity-user type-channel’ level.

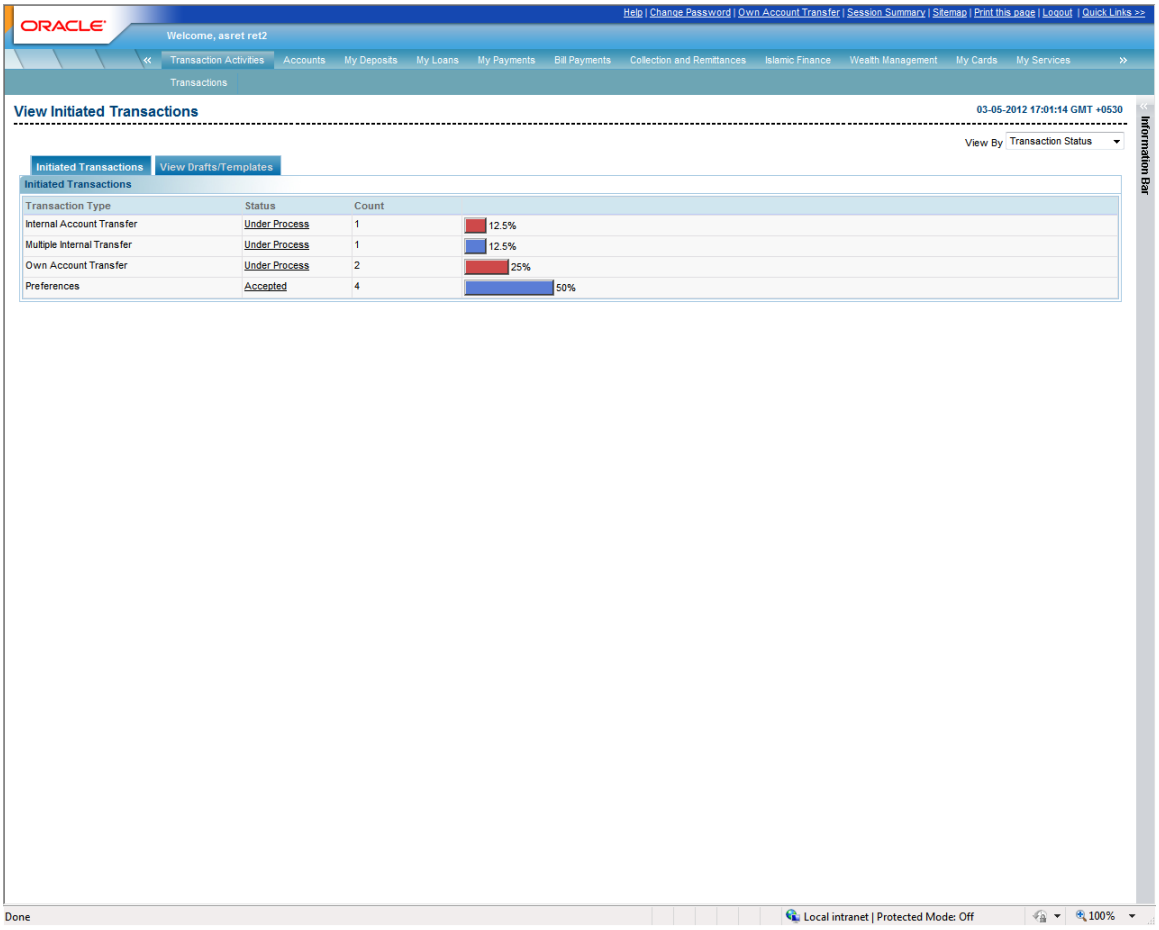
The User Interface is rendered in the following format wherein transaction menu appears on the top bar of the user interface. The menu is scrollable and any hidden menu items, in case of a large menu, can be viewed by scrolling the menu horizontally using the scroll buttons.



Top Menu Based

This menu style can be achieved by configuring ‘T’ in ‘MENUTYPE’ column of the table ‘MSTENTITYUSERCHANNEL’ at ‘entity-user type-channel’ level.

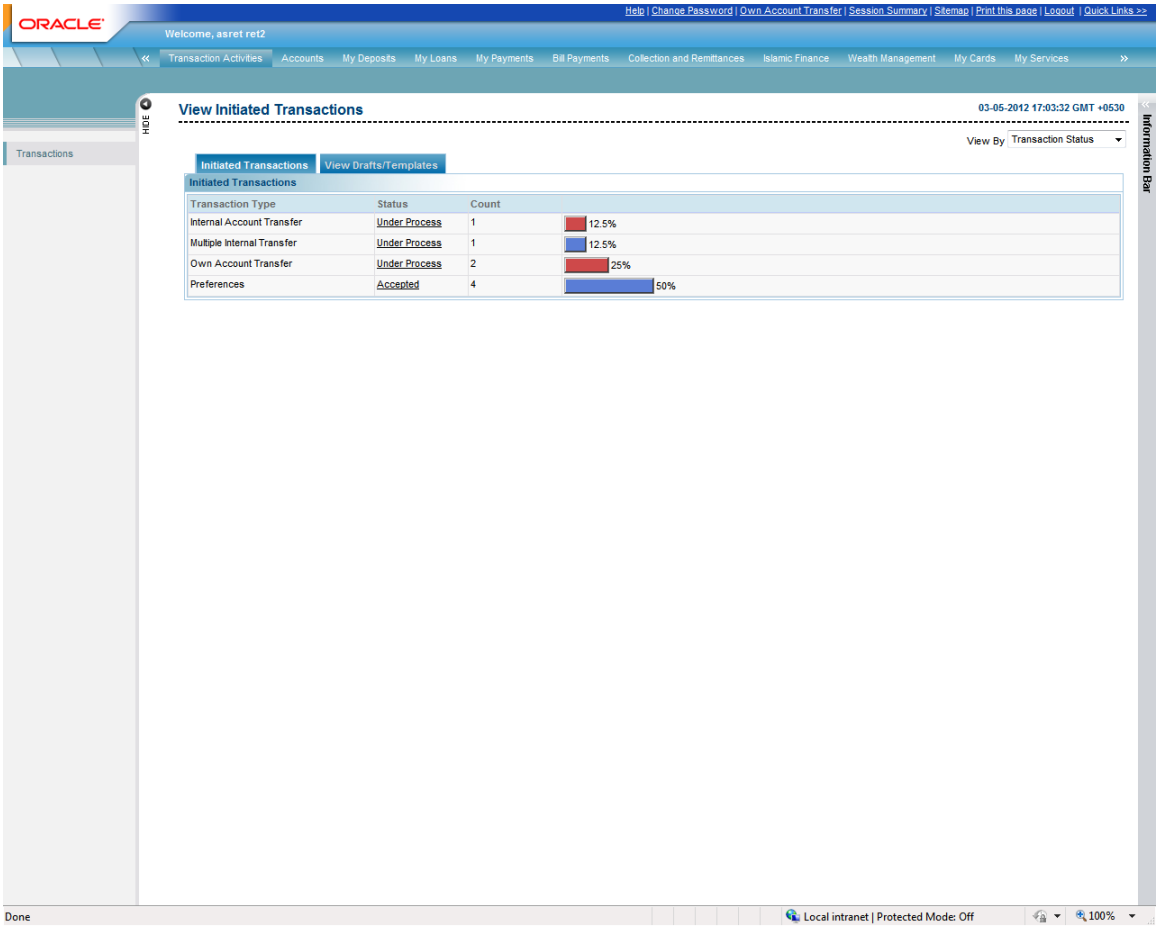
The User Interface is rendered in the following format wherein transaction menu appears on the top bar of the user interface. The menu is scrollable and any hidden menu items, in case of a large menu, can be viewed by scrolling the menu horizontally using the scroll buttons.



Group Menu Based

This menu style can be achieved by configuring ‘TL’ in ‘MENUTYPE’ column of the table ‘MSTENTITYUSERCHANNEL’ at ‘entity-user type-channel’ level.

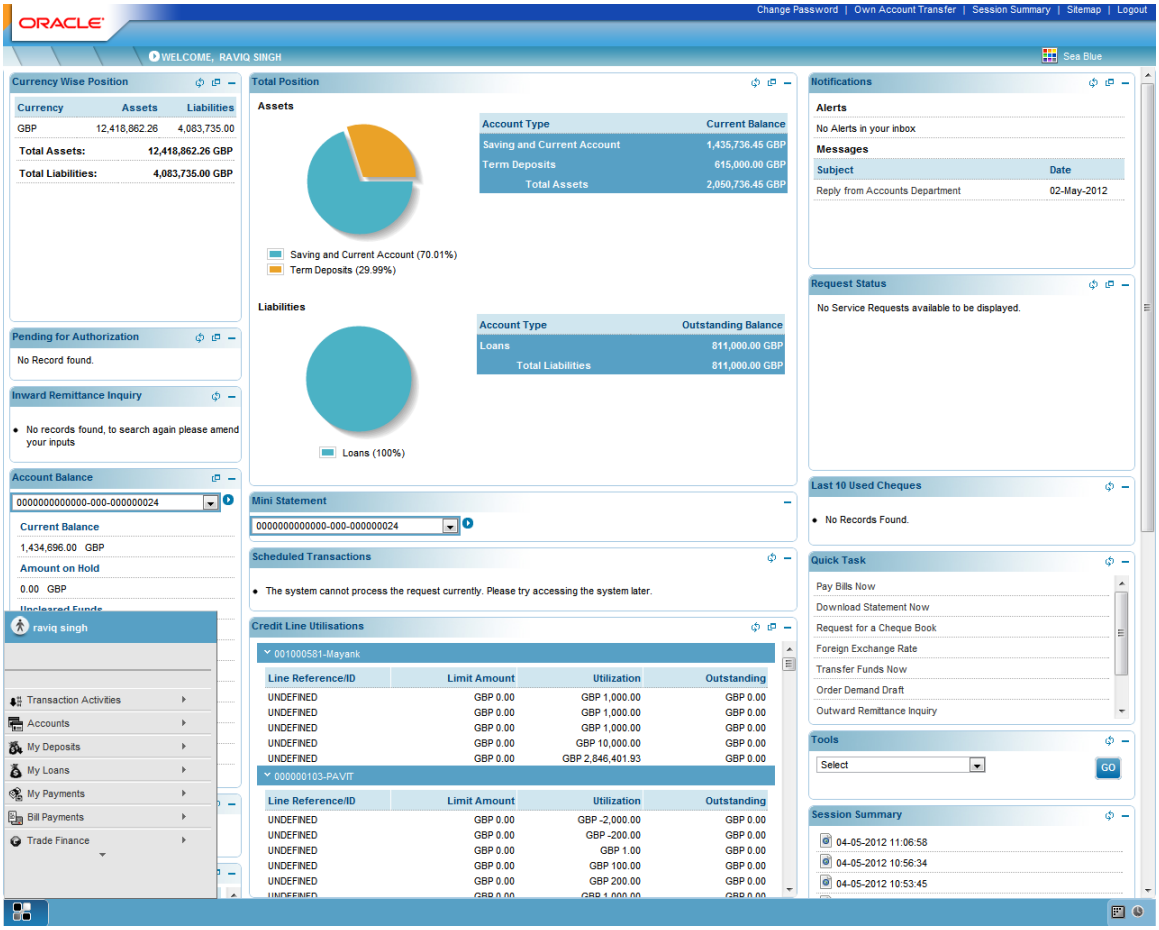
The User Interface is rendered in the following format wherein transaction group menu appear on the top bar whereas the menu within the groups continues to appear within the left menu frame of the user interface.



Dashboard – Taskbar based

This menu style can be achieved by configuring ‘D’ in ‘MENUTYPE’ column of the table ‘MSTENTITYUSERCHANNEL’ at ‘entity-user type-channel’ level.

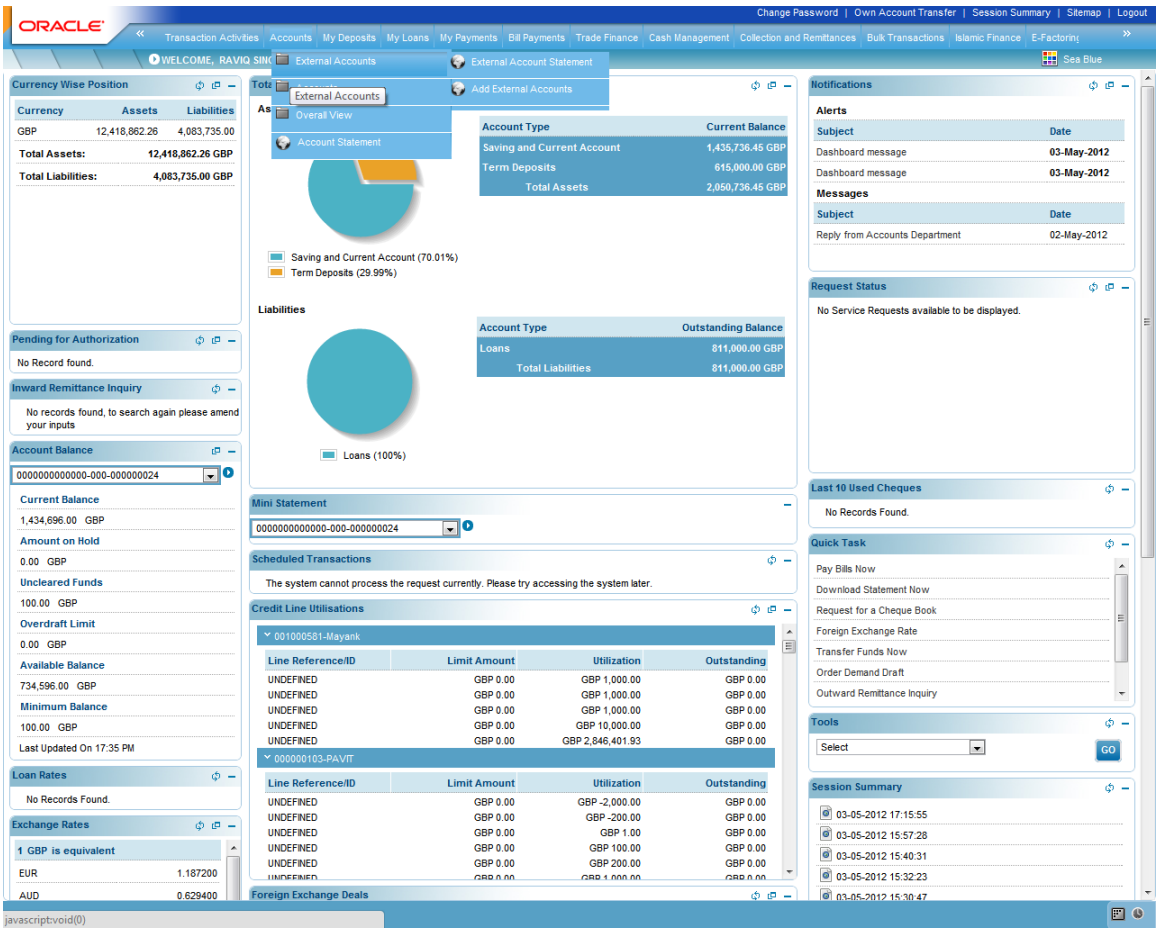
The User Interface is rendered in the following format wherein the menu icon is displayed at the bottom left corner of the screen. Clicking on the icon brings up the first level transaction groups. On clicking on the transaction group either second level transaction group is displayed or the Transaction would be displayed (depending on menu configurations).



Dashboard – Top based

This menu style can be achieved by configuring 'DT' in 'MENUTYPE' column of the table 'MSTENTITYUSERCHANNEL' at 'entity-user type-channel' level.

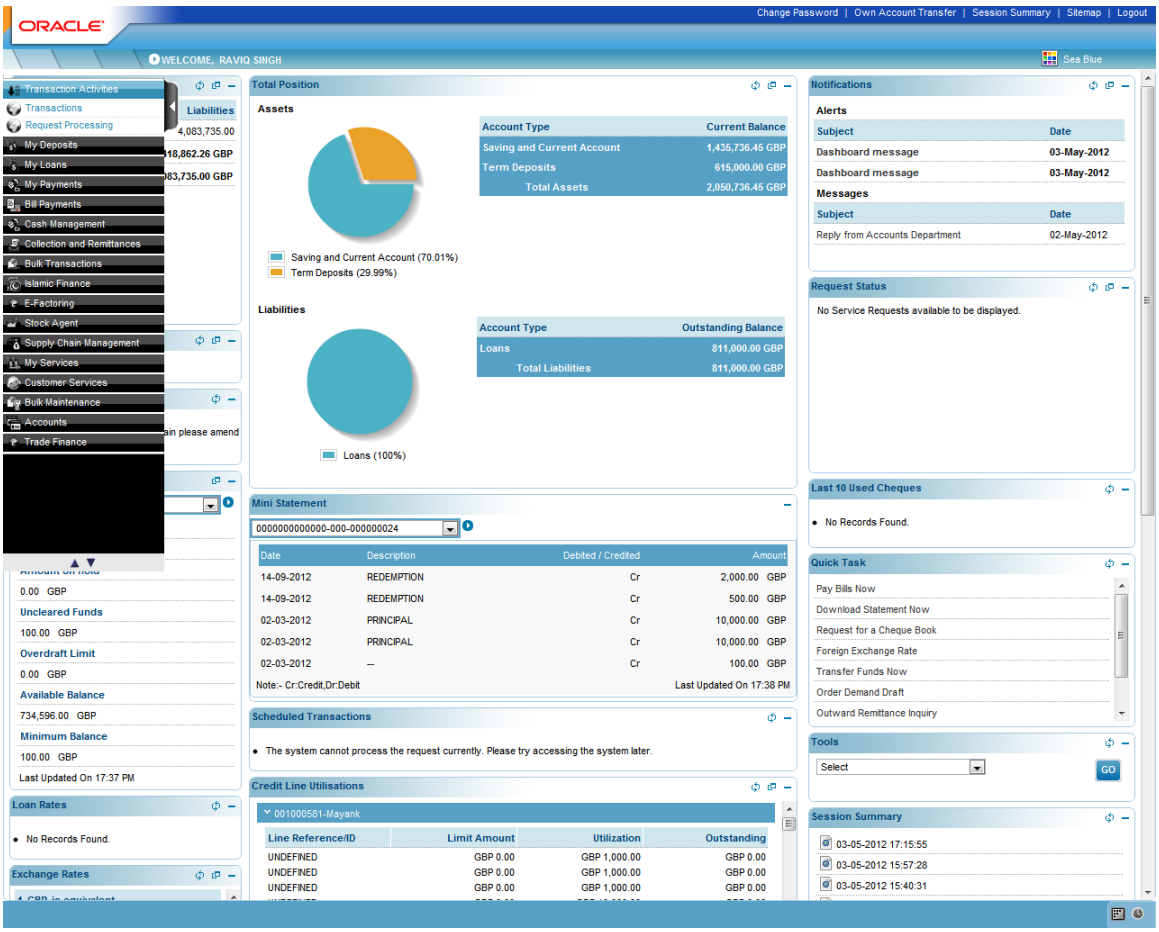
The User Interface is rendered in the following format wherein the first level menu grouping is displayed at the top. On hovering the mouse on a menu group, the transaction and/or second level transaction grouping is displayed (depending on menu configurations).



Dashboard – Left based

This menu style can be achieved by configuring ‘DL’ in ‘MENUTYPE’ column of the table ‘MSTENTITYUSERCHANNEL’ at ‘entity-user type-channel’ level.

The User Interface is rendered in the following format wherein a Menu Tab is displayed to the left edge of the screen. On hovering the mouse on this tab brings out the Menu in the 3 level structure as described for Taskbar and Top menu layouts.



Custom Layout

The value of `MENUTYPE` has to be “X” in case none of the above standard layouts are expected to be used for the implementation. If the IMPLEMENTER has chosen to go for a very custom layout, which is not supported within the layouts as provided by Oracle FLEXCUBE Direct Banking, then the indicator should be “X” for a custom layout to be used.

ⓘ The Sitemap hyperlink at the top of the user interface allows the complete menu to be viewed in a single page and the user can jump to any transaction from this page.

Dynamic Branding

Oracle FLEXCUBE Direct Banking, in addition to the standard Branding, provides facility to define UI Branding for a specific set of users (eg. Privilege Customers). It also provides facility to impose the Branding, disallowing other Branding options available to the rest of the users. To enable Dynamic Branding, the following configurations are required:

1. An entry in MSTHOSTINTERFACE with IDREQUEST 'CSS_FETCH' is required. The default out-of-box entry uses `com.iflex.fcat.services.hostinterface.impl.NullResponseAdapter`.
2. A new adapter will have to be implemented as per the requirements.

3. The following properties are needed to be enabled in in MSTPTOPERTIES

Property Name	Property Value
<ENTITY ID>.CSS_FETCH_REQUESTID	CSS_FETCH
<ENTITY ID>.CSS_FETCH_REQUEST_VERSION	0

Please note that the values of the property may change if the above MSTHOSTINTERFACE entry is different.