Oracle FLEXCUBE Direct Banking

System Handbook – Volume I – Core and Architecture
Release 12.0.2.0.0

Part No. E50108-01

September 2013

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://flexsupp.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 12.0.2.0.0, refer to the following documents:

- Oracle FLEXCUBE Direct Banking System Handbook – Volume II
- 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. Installation and Configuration

Oracle FLEXCUBE Direct Banking provides the installation documents which will allow the seamless installation of the application on any popular Java EE platform. Oracle FLEXCUBE Direct Banking is supported on the following platforms.

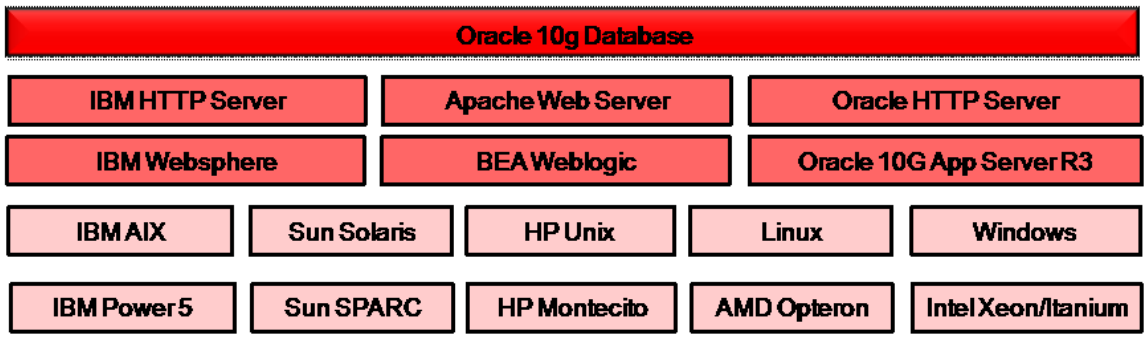
Database

- Oracle 10g / 11g Database (11.2.0.2)

Java EE Servers

- Oracle Application Server (BEA Weblogic Server) 10.3.5

The web server as supported by the corresponding Java EE server is used in most cases with Apache Web Server being the primary we server of choice.



While Oracle FLEXCUBE Direct Banking follows standard Java SE and Java EE principles, the environment is restricted for these servers to provide strong support and features as required for business critical applications like Internet Banking and Mobile Banking.

ⓘ Refer the 'Oracle_FLEXCUBE_Direct_Banking_Environment' document for the pre-requisites and supported environments for the specific release being installed. The document provides the necessary infrastructure requirements to install and operate Oracle FLEXCUBE Direct Banking.

The installation procedures are standard deployment procedures for Java EE applications to be deployed on any of the Java EE servers on both the Windows and the UNIX / LINUX platforms.

The installation documents explain the deployment of the solution from a Production / Testing perspective and do not include setting up the development environment with the IDE, Workspace etc. These are included in the **Extensibility Features → Developer Tools** section in this document separately.

ⓘ Refer the 'Oracle_FLEXCUBE_Direct_Banking_Installation_Steps' document for installation and configuration procedures for the Oracle FLEXCUBE Direct Banking platform.

3.1. Installer

Oracle FLEXCUBE Direct Banking Installer is Java based package used to perform complete or partial setup of the application. The installer consists of product archive and configuration files and is shipped to the client as a single deliverable CD. The Installer is configurable and OS independent software which can be executed on Windows as well as UNIX based environments.

Oracle FCDB Installer – This package is a Swing based interactive tool to install Oracle FCDB application as well the related Day Zero database. This unit will be used by FCDB implementation team.

ⓘ Refer the ***Oracle_FLEXCUBE_Direct_Banking_Installer_UserGuide*** document for basic usage and input required for executing Oracle FCDB Installer.

Oracle FCDB Installer is an interactive (Swing based) executable used to deploy the application on the client side. The complete deliverable FCDB Installer is created using FCDB Packager routine using an interactive build session

This executable can be used in two modes depending on the invocation parameter.

- Command prompt based
- Swing UI based

The functionality, language and workflow of the Oracle FCDB Installer are configurable and based on property files (buildConfig.xml and userMsg_en.xml). These files can be encrypted before publishing to the client. Oracle FCDB Installer also provides inbuilt security like CRC check to verify against tampering and logs the activity.

One of the main activities of Oracle FCDB Installer is database day zero setup. This is done by altering the existing Day zero DB scripts in such a way that backward compatibility with SQL*Plus is preserved.

The SQL scripts, with configurable delimiters mentioned in the Installer configuration property file is then executed with JDBC connection. The SQL script names and their order are also configurable within the configuration property file hence completely extensible.

These Day zero scripts create the entire working database along with all the required DB objects.

4. System Configurations and Setup

4.1. Day Zero (0) and DAY (1) Setup

Oracle FLEXCUBE Direct Banking provides UI based/command-line based installer toolkit.

Post installation, Day zero configurations needs to be completed. These include updating properties files, properties table and master tables.

ⓘ Refer document **Oracle_FLEXCUBE_Direct_Banking_Parameter_Sheet** to find the Day Zero parameters. The sheet mentions all parameters which need to be updated as part of Day Zero activity.

ⓘ Refer document **Oracle Flexcube Direct Banking Day One Configurations** find data-model briefing about Day Zero tables.

Following Day Zero Setup exercise should be carried:

1) Presentation tier configurations: Parameters residing in configuration file for presentation tier <<entity.xml>> should be updated.
2) Channel tier configurations: Parameters residing in configuration file for channel tier `fcat-config.xml` should be updated.
3) Business tier configurations: Business tier use `fcat.properties` as property file. The property file contains information to locate database. JNDI details to fetch the local database datasource are available in `fcat.properties` and should be updated during Day Zero Setup. Other Day Zero configuration parameters reside in table `MSTPROPERTIES` should also be updated as per Day Zero configuration. Refer document Oracle_FLEXCUBE_Direct_Banking_Database_Data_Model to find data-model briefing about Day Zero tables.

ⓘ Refer document **Oracle_FLEXCUBE_Direct_Banking_Parameter_Sheet** to find the Day Zero parameters. The sheet mentions all parameters which need to be updated as part of Day Zero activity.

4) Entity setup needs to update as per expected entities topology. Please refer to appendix section for details on how to setup a new entity.
5) User Setup needs to be completed as per expected user segments. Oracle FLEXCUBE Direct banking provides predefined user model as out-of-box solutions. Also it provides highly extensible user-model which can be customized to add new user type or modify the pre-defined user-types. Please refer to appendix section for details on pre-defined user-types and steps to add a new user type.

5. Technical Architecture

Oracle FLEXCUBE Direct Banking is a multi channel platform supporting Internet and Mobile Banking channels. The Oracle FLEXCUBE Direct Banking platform can also provide business integration services, leveraging its SOA enabled architecture, to expose the requisite business services to other channels or customer touch-points not implemented by Oracle FLEXCUBE Direct Banking.

For banks implementing other applications which may need to leverage the same business services as used by Oracle FLEXCUBE Direct Banking. The business services exposed by the Business Tier can be consumed by the external applications based on appropriate access control.

Oracle FLEXCUBE Direct Banking provides an extensible architecture which is aligned towards quick and easy development of internet and mobile banking functions to keep itself abreast of the latest trends and solutions required.

ⓘ The Architecture of the Oracle FLEXCUBE Direct Banking is based on a Rapid Application Development (RAD) framework based on the Java SE and Java EE platforms.

The framework provides the core guiding principles for any features and functionality within the solution and allows many common features and core services to be enabled by configurations rather than development.

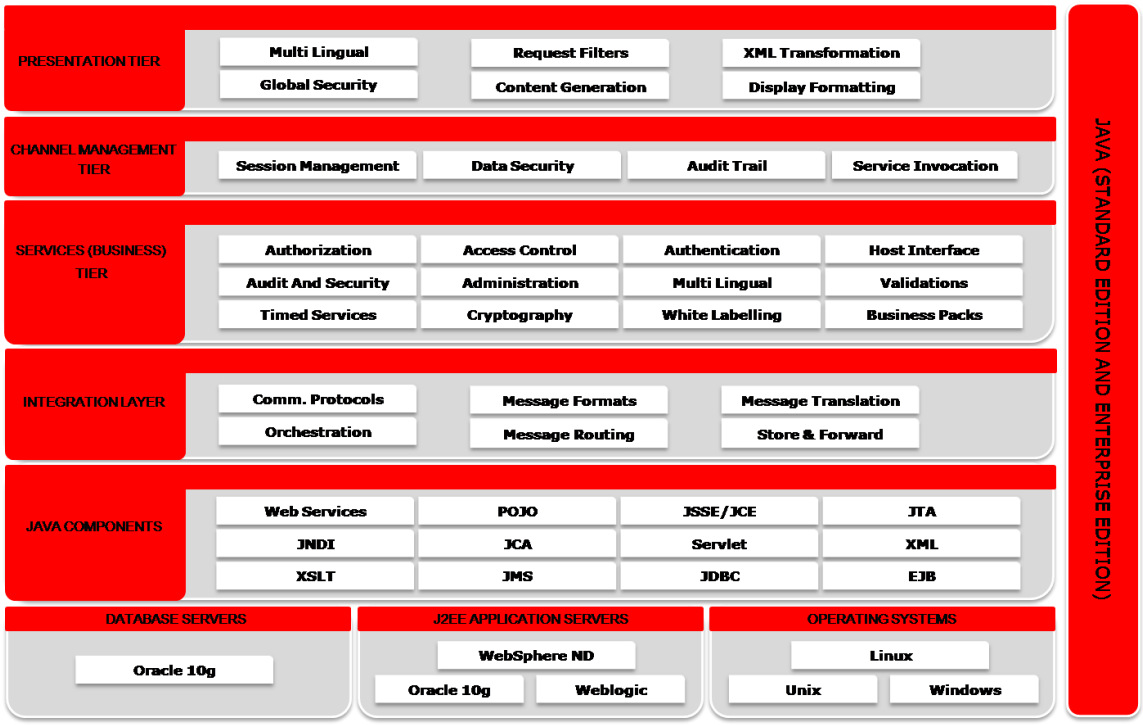
The following sections provide some insights into the architecture and design which is essential to understand the extensibility options provided by Oracle FLEXCUBE Direct Banking in later sections on the document.

5.1. Architecture Components

Oracle FLEXCUBE Direct Banking is a multi-tiered architecture with the ability to segregate deployment at the following tiers

1. Presentation Tier
2. Channel Management Tier
3. Service Orchestration Tier
4. Business Tier
5. Database Tier

Each of the tiers is a logically separate tier which can also be located physically separate on a different infrastructure. The architecture is geared towards a highly scalable deployment with the option of scaling each tier as appropriate.



The architecture is fine tuned or web application development over various releases and leverages on standard design patterns as well as uses some time proven proprietary techniques for various aspects of the solution. The architecture is geared towards simplification and standardization of the transaction development and deployment on the Oracle FLEXCUBE Direct Banking platform making it easily extensible and flexible to define and derive custom components easily.

The key features of each of the architecture tiers are explained in the sections below:

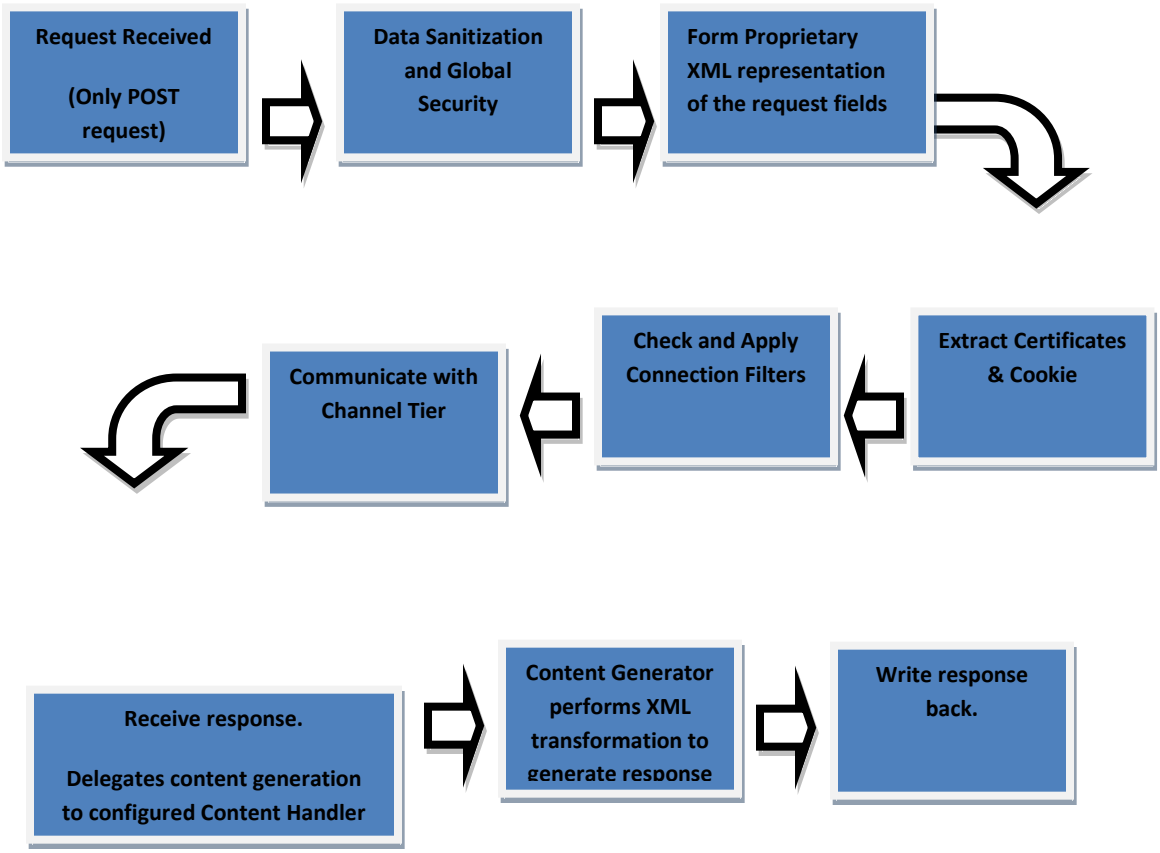
5.1.1. Presentation Tier

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.

Some of the key features within the

- Oracle FLEXCUBE Direct Banking only supports the processing of POST requests and rejects GET or any other requests outright within the application. The other HTTP requests like TRACE, HEAD etc should be rejected at the Web Server itself via appropriate configurations since these are not required to be processed. Refer
- Use of a UTF-8 character set to
- Ability to perform key security checks and reject the request outright if the basic global checks are not successfully completed.
- The JSessionID from the browser is captured from the



The Presentation Tier communicates with the Channel Management Tier for the required Request Processing and Session Management actions.

The Presentation Tier can be collapsed into the Channel Management Tier using a tightly coupled deployment option and hence resulting into a simpler deployment option if required.

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.

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.

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.

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.

5.1.2. Channel Management Tier

The Channel Management tier performs the Session Management activities, and controls the lifecycle of any channel session. The Tier has access to the Local Database (LDB) used within the Oracle FLEXCUBE Direct Banking solution and uses the same for session related activities.

Audit Trail logging and Data Level Security are features of this layer before the request it passed to the Business Services for transaction processing.

The Channel Management Tier communicates with the Business Tier using Web Services by default. The options are available in Oracle FLEXCUBE Direct Banking to configure the Channel Management Tier to point to Service Orchestration Engine like FLEXCUBE Connect or a BPEL Engine to choreograph the transaction processing using simple services developed in Oracle FLEXCUBE Direct Banking or external systems.

The default deployment option does not require FLEXCUBE Connect or any BPEL Engine for the solution to be implemented.

SESSION MANAGEMENT

The USERSESSION table stores the primary session of the user. This table stores session information which is retrieved by Channel tier before accessing Business tier.

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 by business tier.
5) If validated, verification screen is displayed showing data as submitted by user. User needs to verify the data & choose to confirm/change it.
6) 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.
7) This feature ensures user that data cannot be tampered on confirmation.

AUDIT TRAIL

Channel tier features highly useful & configurable auditing capability. Both request into the channel tier & response by channel tier can be audited. 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. Please refer to extension section for detail.

SERVICE INVOCATION

5.1.3. Service Orchestration Tier

The service orchestration tier provides orchestration across multiple business services. The service orchestration tier can be implemented in multiple ways

1. EJB/Web Services – In this mode of implementation, only one business service is invoked. This is typically used for internet banking functions related to inquiries and financial transactions where the request/ response are online.

5.1.4. Business Tier

The Business Tier is a collection of services which are one atomic transaction by default. Each service is a clearly defined REQUEST-RESPONSE pair which defines the transaction performed for the services.

Services are reusable and can be linked to each other within the same transaction context to support complex business functions. The business services provide a consistent experience to all channels within the solution and can be extended to adapt to specific variations to the requirements.

5.1.5. Integration Tier

The Integration Tier comprises of the FLEXCUBE Connect application integration platform. The Integration tier will interface with the backend systems or middleware synchronously, asynchronously or via batch mode of integration.

The integration tier can perform has support for multiple communication protocols like MQ/JMS, Web Services, RMI/IIOP, TCP/IP and message formats like XML, ISO 8583, ASCII etc.

In case of point to point communication with the backend systems the integration tier can perform message transformation, message routing and orchestration. In case the integration with backend systems is via EAI platform / middleware then message transformation, message routing, orchestration can be performed by EAI platform / middleware.

5.1.6. Database Tier

Oracle FLEXCUBE Direct Banking uses a database for its configurations, data and audit information called Local Database (LDB). The LDB contains the user related information and all configurations and parameters that are required by the solution.

JDBC is used to connect to the database and appropriate database specific JDBC drivers are used based on the choice of deployment and database.

ⓘ Please refer to document ***Oracle_FLEXCUBE_Direct_Banking_Database_Setup*** for detail information on local database schema setup.

5.2. Technology Platform

5.2.1. Security

Oracle FLEXCUBE Direct Banking provides for a number of security features and options within the platform for configuration. Oracle FLEXCUBE Direct Banking provides a Security Best Practices Guide for the IMPLEMENTERS to configure the application for secure use. These are expected to be followed for all environments in which Oracle FLEXCUBE Direct Banking has been deployed.

ⓘ Refer the ***Oracle_FLEXCUBE_Direct_Banking_Security_Guide*** for application hardening and security configurations to be done post the installation of the software.

💣 It is the responsibility of the IMPLEMENTER and the LICENSEE together to ensure the security configurations are completed for hardening the environment as required.

5.2.2. Scalability and Performance

Oracle FLEXCUBE Direct Banking recommends a clustered environment to provide for failover as well as performance of the application. The IMPLEMENTER should perform the appropriate configurations of the underlying infrastructure to manage the performance of the application.

The recommended tuning parameters for Oracle FLEXCUBE Direct Banking are provided in the Oracle FLEXCUBE Direct Banking Performance Tuning Guide. This guide can be referred for specific configurations and pointers to implement highly scalable configurations for Oracle FLEXCUBE Direct Banking.

ⓘ Refer the ***Oracle_FLEXCUBE_Direct_Banking_Security_Guide*** document for monitoring, tracking and measuring the performance of Oracle FLEXCUBE Direct Banking.

💣 It is the responsibility of the IMPLEMENTER and the LICENSEE together to ensure the performance checklists are verified regularly and the system maintained for optimal performance.

5.3. Release Nomenclatures

Oracle FLEXCUBE Direct Banking follows the following Release Nomenclature for naming the product releases. Every release of Oracle FLEXCUBE Direct Banking is classified as a Kernel or a Cluster release.

5.3.1. Kernel

The Oracle FLEXCUBE Direct Banking Kernel Releases are main releases which are delivered with primary changes. Any changes to the Base or Core module are taken up within these Releases and LICENSEEs have the choice of implementing

5.3.2. Cluster

Oracle FLEXCUBE Direct Banking releases Cluster Packs which may be targeted towards specific Regional requirements.

5.4. Developer Tools

Oracle FLEXCUBE Direct Banking is a Java SE and Java EE based platform and any standard development environment for Java applications can be used. The following are the recommended developer tools for developing on Oracle FLEXCUBE Direct Banking.

Java Editors and Tools	- Eclipse IDE 3.3 (Europa) - Eclipse IDE 3.4 (Ganymede)
Browsers	- Any industry standard browsers supported by Oracle FLEXCUBE Direct Banking.
Database	- Oracle 10g - Oracle 11g
Database Tools	- Oracle 11g Client - Oracle SQL Developer - PL / SQL Developer - Any industry standard database tools for Oracle 10g / 11g
JDBC Driver	- 11g
Operating System	Windows XP, Windows Vista Professional Workstations

6. Database – Extensions and Configurations

Oracle FLEXCUBE Direct Banking uses a Local Database for all its configurations and data.

ⓘ Refer the Oracle FLEXCUBE Direct Banking Database Design Guide for detailed information on the database objects and artifacts.

6.1. Database Jobs

6.1.1. Purge MANAGE MAINTAIN Log

This job is used to purge no. of table configured in Purge_Duration table.

This is the generic purging activity. This job will invoke the procedure which in turn takes each table's configuration from Purge_Duration table and does archive & delete the records depending on the duration specified in the same table. This job archives the data in offline tables.

The offline tables are available in tablespace "TBLS_OFFLINE" to ease the backup policy

6.1.2. Purge File Upload table

This scheduled job is used to purge the table used to temporarily hold the uploaded files.

The job invokes procedure "PURGE_FILEUPLOAD" which purges the data.

6.1.3. Purge Timed out User Sessions

This scheduled job is used to purge the user sessions which have timeout.

The job invokes procedure "PURGE_USERSESSION" which purges the data. The procedure purges the session data & audits termination of the session.

6.1.4. Purge Temp Tables

This scheduled job is used to purge the temporary tables. This job does not purge the log data.

The job invokes procedure "PURGE_TMPTBL" which maintains the user.

6.1.5. Maintain Hibernated User

This scheduled job is used to maintain the users who have exceeded the hibernation period. "Passwordpolicy" allows setting hibernation period for users. If the user does not successfully logs into the system before the hibernation period, the user is locked.

The job invokes procedure "MAINTAIN_HIBERNATEDUSER" which maintains the user.

6.2. Extending Database Jobs

Oracle FLEXCUBE Direct Banking allows the IMPLEMENTERS to configure purging job as per requirement and plug them into the Oracle database.

7. Core – FEATURES

7.1. JDBC Engine

JDBC Engine is available in package `com.iflex.fcat.xjava.jdbc`.

7.1.1. JDBCEngine

The `com.iflex.fcat.xjava.jdbc.JDBCEngine` class provides utility methods to execute `Statements` or `PreparedStatements` on the database. The utility methods

1) `executeUpdate`: This method should be used to execute all INSERT, UPDATE and DELETE queries. It returns the number of records affected by the query. The overridden method provides ability to execute predefined query from `MSTQUERY` table.
2) `executeQuery`: The method executes a SELECT statement to return the results of the query. The overridden method provides ability to execute predefined query from `MSTQUERY` table.

ⓘ Refer the Java Documentation for additional documentation on the component.

7.1.2. JDBCResultSet

JDBC Engine is implemented by class `com.iflex.fcat.xjava.jdbc.JDBCResultSet`.

The class provides a data object to hold the results of the query execution for all SELECT queries to the database. The class encapsulates the results retrieved from the `java.lang.ResultSet` object and stores the same as an instance of the class.

ⓘ Refer the Java Documentation for additional documentation on the component.

7.2. Validation Engine

The Validation Services or the Validation Engine as it is commonly referred is the key feature of Oracle FLEXCUBE Direct Banking. This engine provides support for request and data field validations to be put in place for any requests that are processed by Oracle FLEXCUBE Direct Banking. This includes the web based requests originated as part of the Oracle FLEXCUBE Direct Banking internet and mobile banking solution or any other requests processed as a part of business integration services offered by Oracle FLEXCUBE Direct Banking.

The Validation Engine provides a pluggable and configurable mechanism to verify for data integrity checks and validations to be performed before the request is passed on to the actual business processing layer which completes the transaction. This ensures that data elements are strongly typed and validated against a pre-defined set of rules as well as the ability to plug-in custom validation rules for verification of any specific field values.

The Validation Engine also acts as a security mechanism by which data could be introspected for specific data patterns that can results in SQL injection or Cross Scripting attacks or any other security threats. The Validation Engine segregates the validation functions from the business processing layer allows flexibility and extensibility to modify the required validation rules without impacting or with a minimal impact to the business processing layer.

ⓘ The Validation Engine is one of the Core Services in the Business Tier and provides extensibility features for validations to be adapted, modified, updated and custom configured for specific installations and implementation of Oracle FLEXCUBE Direct Banking. Volume IV of System Handbook discusses the Validation Engine in detail.

7.3. Exchangers

7.3.1. Message Queue Exchangers

MESSAGE WAITERS

The class `com.iflex.fcat.gateway.MessageWaiter` defines the contract between a class that wants to receive messages from a queue and the class that reads a message queue (class `QueueReceiver`). Any class that wants to receive messages over a message queue needs to implement this interface.

An application must take following steps to send a request message over

a message queue to a host and receive its response over a message queue.

- Get a `com.iflex.fcat.gateway.QueueSender` the host.
- Get a `com.iflex.fcat.gateway.QueueReceiver` that can receive messages from the host.
- For each message pair to be exchanged,
    1. Register a message waiter with the receiver.
    2. Call an appropriate write method of the sender.
    3. Notify the sender.
    4. Wait for the receiver to receive the response (or time out). The receiver will provide the response and notify the waiting instance.

ⓘ Refer the Java Documentation for additional documentation on the component.

ANONYMOUS MESSAGE WAITERS

The interface `com.iflex.fcat.gateway.AnonymousMessageWaiter` extends the `MessageWaiter` interface providing anonymous message processing capability. This interface is a type identifier (or a tag) interface and does not add any variable or method.

A class implementing this interface may be used for following purposes:

- If message type is immaterial to message processing then all the messages received in a synchronous messages exchange mode may be given to a registered `AnonymousMessageWaiter`.

- In asynchronous message exchange a message received for a timed out `MessageWaiter` is given to a registered `AnonymousMessageWaiter`.

7.4. 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

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>**

ⓘ Refer the Oracle_FLEXCUBE_Direct_Banking_Security_Guide for additional documentation on the security aspects of the Application.

7.5. Types of Mobile clients

7.5.1. 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.docx

7.5.2. 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

7.5.3. 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

7.5.4. 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_iPhone_Client_Developer_Guide.pdf

7.5.5. Android phone client

This client is targeted at android based phones. Detailed architecture and customization guidelines are provided in the developer guide

Oracle_FLEXCUBE_Direct_Banking_Mobile_Android_Client_Developer_Guide.pdf

7.5.6. Android Tabs client

This client is targeted at android based Tablets. Detailed architecture and customization guidelines are provided in the developer guide

Oracle_FLEXCUBE_Direct_Banking_Mobile_Android_Client_Developer_Guide.pdf

7.5.7. Blackberry Native client

This client is targeted at Blackberry phones. Detailed architecture and customization guidelines are provided in the developer guide

Oracle_FLEXCUBE_Direct_Banking_Mobile_Blackberry_ Native_Client_Developer_Guide.pdf

7.5.8. 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.

8. Single Sign On for Retail and Corporate users

A user can log in to Oracle FLEXCUBE Direct Banking through a variety of channels available out of the box. Until version 12.0.0, each channel had its own set of security credentials including the user id, login password and the transaction PIN. Starting from version 12.0.1, these channels can be grouped together to form a channel group such that a single set of security credentials can be maintained across all channels.

This eliminates the need to maintain multiple user ids and passwords to login and transact on the application from various available channels. The channel grouping is made available only for following channels:

1. Internet Banking
2. Browser Based Mobile Banking
3. Application Based Mobile Banking

SMS Banking would always continue to be considered as a separate group having just the SMS banking channel.

The bank can configure the groups such that any one or more of the above three (3) channels can for a group. However, it is mandatory that no two (2) groups would have the same channel in them for the same entity-user type combination.

The out-of-box configuration available in the current version is grouping of all non-SMS banking channels for Retail and Corporate Users both. The following table illustrates the current configurations.

Entity	*User Type*	*Channel*	*Group*
B001	Retail (EN1)	Internet (01)	Internet and Mobile Banking (ALL)
		Mobile Browser (42)	
		Mobile Application (43)	
		SMS Banking (41)	SMS Banking (SMS)
	Corporate (ECU)	Internet (01)	Internet and Mobile Banking (ALL)
		Mobile Browser (42)	
		Mobile Application (43)	
		SMS Banking (41)	SMS Banking (SMS)
	Corporate Administrator (CA2)	Internet (01)	Internet Banking (INT)

	Virtual Banking User (ENV)	Internet (01)	Internet Banking (INT)
T001	Retail (EN1)	Internet (01)	Internet and Mobile Banking (ALL)
		Mobile Browser (42)	
		Mobile Application (43)	
		SMS Banking (41)	SMS Banking (SMS)
	Corporate (ECU)	Internet (01)	Internet and Mobile Banking (ALL)
		Mobile Browser (42)	
		Mobile Application (43)	
		SMS Banking (41)	SMS Banking (SMS)
	Corporate Administrator (ECU)	Internet (01)	Internet Banking (INT)
	Virtual Banking User (ENV)	Internet (01)	Internet Banking (INT)
F001	Bank Administrator (INA)	Intranet (11)	Intranet (INTRA)
	Helpdesk (HDU)	Internet (01)	Helpdesk (HLP)

For detailed configurations available for grouping multiple channels, please refer the Day One configuration Guide (Oracle FLEXCUBE Direct Banking Day One Configurations).

The following transactions available in the application for User Maintenance and related activity are impacted.

1. Bank Administration Tasks
    a. Create User
    b. Modify User
    c. Reset Password
    d. Print Password
    e. Manage Policies
    f. View User
2. Business User Tasks
    a. Change Password
    b. Subscribe / Unsubscribe User Channel
    c. Preferences (Change User Id)
    d. Lock Transaction Password
    e. Reissue Transaction Password

Following tables are added / modified for supporting channel grouping.

Table: MSTENTITYUSERCHANNELGRP	**Action:** [New]		**Remark:** This table holds the channel grouping information at entity-user type level.	
Column	Data Type	Null	Default	Remark
ID_ENTITY	VARCHAR2(5)	No		Entity Identifier

TYPEUSER	VARCHAR2(3)	No		User Type
IDGROUP	VARCHAR2(5)	No		Group ID for grouping multiple channels
POLICYIDPASSWORD	VARCHAR2(50)	No		The Login Password Policy applicable to the Channel Group
POLICYIDTRANSACTION	VARCHAR2(50)	Yes		The Transaction Password Policy applicable to the Channel Group
POLICYIDUSER	VARCHAR2(50)	No		The User ID Policy applicable to the Channel Group
POLICYCODEACTIVATION	VARCHAR2(50)	Yes		The Activation Code Policy applicable to the Channel Group

Table: MSTENTITYUSERCHANNEL	**Action:** [Modify]		**Remark:** This table has been modified to add grouping information for channels.	
Column	Data Type	Null	Default	Remark
IDGROUP	VARCHAR2(5)	No		Group ID for grouping multiple channels

Table: MSTCHANNELUSER	**Action:** [Modify]		**Remark:** This table has been modified to add grouping information for channel user.	
Column	Data Type	Null	Default	Remark
IDGROUP	VARCHAR2(5)	No		Group ID for grouping multiple channels
FAILEDGRPATTEMPTCNT	NUMBER	No	0	Count of failed login attempt across all channels in the group. This is reset after success login. This count can be used for user lock.

Table: PASSWORDHISTORY	**Action:** [Modify]		**Remark:** This table has been modified so that channel information is not used and instead channel group information is used.	
Column	Data Type	Null	Default	Remark
IDGROUP	VARCHAR2(5)	No		Group ID for grouping multiple channels
IDCHANNEL	VARCHAR2(2)	Yes		This field has been modified to accept null values. The channel id would not be stored in this table. Rather the

				Group ID would be stored.
Constraint: P_KEY_PASSHISTORY	**Action:** [Modify] **Type:** [Primary Key]		**Remark:** This primary key has been modified from (ID_ENTITY, IDUSER, IDCHANNELUSER, IDCHANNEL, DATCREATION) to (ID_ENTITY, IDUSER, IDCHANNELUSER, IDGROUP, DATCREATION)	

Table: USERPASSWORDPRINT	**Action:** [Modify]		**Remark:** This table has been modified so that channel information is not used and instead channel group information is used.	
Column	Data Type	Null	Default	Remark
IDGROUP	VARCHAR2(5)	Yes		Group ID for grouping multiple channels

Table: USERPINHISTORY	**Action:** [Modify]		**Remark:** This table has been modified so that channel information is not used and instead channel group information is used.	
Column	Data Type	Null	Default	Remark
IDGROUP	VARCHAR2(5)	No		Group ID for grouping multiple channels
IDCHANNEL	VARCHAR2(2)	Yes		This field is modified to accept null values. The channel id would not be stored in this table. Rather the Group ID is stored.

Table: USERSESSION	**Action:** [Modify]		**Remark:** This table is modified so that channel information is not used and instead channel group information is used.	
Column	Data Type	Null	Default	Remark
IDGROUP	VARCHAR2(5)	No		Group ID for grouping multiple channels

For further details on these functionalities please refer the respective User Manuals.

9. Configurations

Service tier uses properties file `fcat.properties` and table `mstproperties` to house application properties. This properties file should typically contain only database information. All the other application configurations are available in table `MSTPROPERTIES` in local database. UI based Administrative services are available to manage properties in the table.

Sample `fcat.properties` file

```
FCAT.LDB.DATABASE.NAME=ORACLE


FCON.A1.JNDI.NAME=A1

FCON.A1.LDB.DRIVER=oracle.jdbc.driver.OracleDriver

FCON.A1.LDB.URL=jdbc:oracle:thin:name/pwd@dbserver:1521:SR1

FNDI.A1.ABCD=


FNDI.TX.java.naming.factory.initial= weblogic.jndi.WLInitialContextFactory

FNDI.TX.java.naming.provider.url= t3s://servhostname:7003
```

`MSTPROPERTIES`: This table residing in local database schema is main location to hold properties used by service tier.

Column Name	Column Type	Description
IDSERVER	Varchar2(59)	Server Identifier. Default id “ZZ” should be used.
PROPNAME	Varchar2(255)	Property name. Entity specific property is prefixed by “<entityID>.”
PROPVALUE	Varchar2(4000)	Property value.

ENABLED	Char(1)	Y: Property enabled and available N: Property not enabled & so not available
ISGUIENABLED	Char(1)	Y: This property can be managed using UI administrator service. N: This property cannot be managed using UI.

ⓘ User Interface in administrative service is available to manage properties available in table MSTPROPERTIES.

💣 Changing any property in fcat.properties/mstproperties will require server restart for change to take effect.

💣 Since this file contains database information, this file should be encrypted using tool provided for same. Please refer Installation documents for same.

Application Message

The Error, Information, Warning, Success, Debug messages displayed by the Oracle FLEXCUBE Direct Banking application are parameterized and maintained in the database table `APPLICATIONMESSAGES`. Entity-specific messages can be used to override default messages.

Column Name	Column Type	Description
IDAPP	Char (2)	Application Identifier. Default application id “A1”

		should be used.
IDMESSAGE	Varchar2(20)	Message identifier. Entity specific message identifier can be used to override default message. Prefix the idmessage with "<entity>." to add entity-specific message.
IDDEVICE	Char (2)	Device/Channel Identifier. **: Any device
IDLANG	Char(3)	Language Identifier.
TXTMESSAGE	Varchar2(4000)	Locale specific text message.
ISGUIENABLED	CHAR(1)	The possible values in this

ⓘ User Interface in administrative service is available to manage application messages.

9.1. External Payment Interface

External Payment Interface (EPI) functionality enables the business users to make merchant payments by accessing the FCDB application through a merchant site. A business user would log into any on-line e-commerce site and selects the items/bills to be purchased / paid. The e-commerce site shall display the options through which the amount can be paid by the user. The business user can invoke FCDB application upon selecting the appropriate option on the merchant's site and pay the amount by debit to their account. User will have to login to FCDB application through their login credentials and make a payment from accounts mapped to the user id. FCDB shall send the debit request to the Host and shall receive the response from the Host. Depending on the host response, FCDB shall send the success or failure status back to the merchant online.

Business user shall only be able to make the payment to the merchant. No other menu / transaction (except logout) shall be accessible / displayed to the user.

ⓘ Refer the **Oracle_FLEXCUBE_Direct_Banking_External_Payment_Interface** document for further details on configurations of the External Payment Interface.

9.2. Error Code Configuration in APPLICATIONMESSAGE Table under NID Mode

If you need to define generic messages across all entities then use message format as CM.10000 (i.e. starting from 10000).
If override is needed then use the format as <<USER_ENTITY>>.CM.10000 (starting from 10000)
If base product message to be overridden then use the format as <<ENTITY>>.<<base product message id>>

10. User Logging

Flexcube Direct Banking (FCDB) is a multi-tier application consisting of UI Tier→Channel Tier→Service Tier. Loggers are available at each tier of FCDB to enable debugging whenever required, Loggers are also separately available for certain important business modules like Authorization Engine which when enabled can give the logs of the data going through the Authorization Engine. All these loggers when enabled require a server restart and will write the logs in separate files as per the Apache log4j configurations. Also these loggers don't have the option to segregate the logs into a single file as per the FCDB logged in channel user.

Horizontal Logging in FCDB is something which will enable the logging of data across all tiers and all modules in a single file specific to a FCDB channel user. Logging of data specific to a user at a single location is very useful for debugging. It ensures that, to get the logs you are not restricting other users from accessing the application and also you need not go to different log files to collect the complete log for a particular use-case.

To enable Horizontal Logging a new transaction "User Logging" has been provided to admin user. Admin User can search for the FCDB user's who are currently logged-in into the application. Admin user can then enable the logging for that user, once enabled all fcdb logs for that user will start getting updated at a single location. Once the user has done the respective transactions for which logs are required admin user can disable the logging for that user. On clicking the disable button the logs will get downloaded as text file.

Limitations

1. Logging can only be enabled for user having valid session i.e. this feature can be enabled for the user only after user has successfully logged-in into the application.

2. Logging starts from channel Tier and in return journey ends at channel tier itself, hence it doesn't cover the GUI tier logs, so one will not get the name of the XSL getting used for a transaction and other similar info which are being logged at GUI tier.

11. APPENDIX: References

The following material is useful for any documentation

System Handbook

Document Name	Description
Oracle_FLEXCUBE_Direct_Banking_System_Handbook_Volume II	The document provides the various extensibility features available for the **Presentation Layer**
Oracle_FLEXCUBE_Direct_Banking_System_Handbook_Volume III	The document provides the various extensibility features available for the **Channel Layer**
Oracle_FLEXCUBE_Direct_Banking_System_Handbook_Volume IV	The document provides the various extensibility features available for the **Business Services Layer.**
Oracle_FLEXCUBE_Direct_Banking_System_Handbook_Volume V	The document provides the various extensibility features available for the **Host Interfacing Layer**

Installation

Document Name	Description
Oracle_FLEXCUBE_Direct_Banking_Environment	The document provides the list of hardware and software environments on which the platform is available.
Oracle_FLEXCUBE_Direct_Banking_Installer_UserGuide	The document provides guide on using Installer toolkit for installation.
Oracle_FLEXCUBE_Direct_Banking_Installation_Steps	The document provides step-by-step installation guide.

Database

Document Name	Description
Oracle_FLEXCUBE_Direct_Banking_Database_Setup	The document provides detail information on database setup.
Oracle_FLEXCUBE_Direct_Banking_Database_Design	The document provides database object design.
Oracle_FLEXCUBE_Direct_Banking_Database_Data_Model	The document provides data model for local database schema.

Reports/BI Publisher

Document Name	Description
Oracle_FLEXCUBE_Direct_Banking_Reports_Setup_and_Configuration	The document provides installations, setup and Configuration guidelines for integrating the reports available within the Oracle FLEXCUBE Direct Banking platform.
Oracle_FLEXCUBE_Direct_Banking_Customer_Reports_Definition	Provides details on the various canned reports and the details of the fields.

Development

Document Name	Description
Oracle_FLEXCUBE_Direct_Banking_User_Interface_Guide.doc	This document provides the guidelines for the User Interface of Oracle FLEXCUBE Direct Banking with the options of choosing the correct User Interface Layout, Theme or updating the
Oracle_FLEXCUBE_Direct_Banking_Security_Guide	Oracle FLEXCUBE Direct Banking Security Best Practices Guide
Oracle_FLEXCUBE_Direct_Banking_Developers_Guide	This document provides sample scripts to create or modify a new functionality.

12. APPENDIX: Third Party Libraries

Please refer document "*Oracle_FLEXCUBE_Direct_Banking_Software_Stack*" to find detail information on third party libraries used with Oracle FLEXCUBE Direct Banking application.

13. APPENDIX: Creating New Service Request

Service request transaction is created like normal transaction which will have unique IDTXN in MSTTXN table. But there are some additional configurations which need to be done to create new service request transaction in Oracle FLEXCUBE Direct Banking.

1. Entry will be done in MSTTXN table with FLAGSERVICEREQUEST column value as 'Y' and ISMENUTXN column value as "N".
2. Entry needs to be done in MSTUSERTYPETXN table for each user type who can make this particular service request. E.g. EN1, ENU, ECU, etc…In MSTUSERTYPETXN table also FLAGSERVICEREQUEST column will be set to "Y" and ISMENUTXN column will be set to "N".
3. Each service request goes through Release Transaction Flow. To enable Release flow "RELEASEREQUIRED=A" has to be configured for each service request in ADTNL_PARAMS column of MSTTXN table.
4. In MSTUSERTYPETXN table the column INITAUTHID has to be assigned value "ZERO_AUTH_CHECKER".
5. This service request needs to be configured for admin user also. So entry has to be made in MSTUSERTYPETXN table for F001/AD1 and F001/INA.
6. Each service request goes through authorization engine. I.e. after Retail/Corporate User makes service request, status of that request becomes released. At this time there won't be any service call. There will only be entry made in ADMINTXNUNAUTHDATA table. Now admin user will be able to see this service request in his Dash Board transaction. Admin user can either Dispatch or Reject this Service Request. When admin user will dispatch this service request then actual service will be invoked and processing will be done. After successful response from the service authorization engine will change the status of service request to Accepted.
7. In ROLETXN this particular service request transaction should have FLGAUTH as 'Y' for the role of the admin user

ⓘ Refer the **Oracle_FLEXCUBE_Direct_Banking_Transaction_Release_Workflow** document for further details on configurations of the Transaction Release Workflow.

14. APPENDIX: Payments Design

All payments modules (except Multiple Internal Transfer) and their Beneficiary Maintenance Transactions have been developed through LEAP Framework.

> ⓘ Refer the **User Manual Oracle FLEXCUBE Development Workbench for Direct and Mobile Banking** document for further details on configurations of Generic Payments Design.

15. APPENDIX: Supported Alerts In FCDB

There are different types of alerts supported in FCDB. List of different alert type is given below.

- Default Alerts
- User Based Alerts
- Customer Based Alerts
- Account Based Alerts
- Authorization Alerts
- Transaction Alerts

ⓘ Refer the **Oracle_FLEXCUBE_Direct_Banking_Alerts** document for further details on supported alerts in FCDB.