Oracle FLEXCUBE Direct Banking

Developers Guide

Release 12.0.2.0.0

Part No. E50108-01

September 2013

ORACLE®

Developers Guide
September 2013

Oracle Financial Services Software Limited
Oracle Park
Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India
Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax:+91 22 6718 3001
www.oracle.com/financialservices/
Copyright © 2008, 2013, Oracle and/or its affiliates. All rights reserved.

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

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

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

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.
The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

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

CONTENTS

1. Preface

1.1. Intended Audience

This document is primarily targeted at

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

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 Developers' Guide, 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 Developers Guide 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.

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 Licensing Guide

2 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 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.

2.2 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.

2.3 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.

2.4 ABBREVIATIONS

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
API	Application Programming Interface

2.5 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 GUIDE FOR CREATING APPLICATION COMPONENTS

This will provide the DNA components by which one would be able to do the following

Term	Definition
New Business Functionality	A developer or a implementer wants to create a new business functional service which is not existing as part of the Oracle FLEXCUBE Direct Banking product.
Modify Existing Business Functionality	A developer or a implementer wants to modify the existing business functional service which exists as part of the Oracle FLEXCUBE Direct Banking product. One would need to either modify the following ❖ GUI component ❖ Business component ❖ Host Interfacing component
Extend Existing Business Functionality	A developer or a implementer wants to extend the existing business functional service which exists as part of the Oracle FLEXCUBE Direct Banking product. One would need to either extend the following ❖ GUI component ❖ Business component ❖ Host Interfacing component
Plugins / Extensions Available	A developer or a implementer may want to extend the existing business functional service which exists as part of the Oracle FLEXCUBE Direct Banking product by taking advantage of the following extensions ❖ Gui Layers • CSS

	• LEAP data structures ❖ Channel Layer • Configure a specific Audit Handler ❖ Business Layer • Pre Business Service Helper • In Service Helper • Extended Handlers ❖ Host Interfacing Layer • Host Interfacing Adaptors • Host Mappers • Host Transforming GUI components

It can use the following core services to build this new functionality

Term	Reference Document
LEAP Data Structure for GUI rendering	Oracle FLEXCUBE Open Development Workbench for Direct and Mobile Banking.pdf
CSS modification	Oracle FLEXCUBE Direct Banking UI Developer Guide.pdf
Channel Extensions	Oracle FLEXCUBE Direct Banking System Handbook.pdf
Business Services	Oracle FLEXCUBE Direct Banking System Handbook.pdf
Host Services	Oracle FLEXCUBE Direct Banking Host Interfacing.pdf

4 NEW BUSINESS FUNCTIONALITY

Oracle FLEXCUBE Direct Banking product gives the facility add new business functionality by piggy banking on the core framework components mentioned above.

Typically new business functionality could be broken down into the following components

1. Create new Front Screen and Client Side Javascript.
2. To create a new Transaction in FCDB Application one needs to register the transaction in the application. Refer to section 3.2, Add new Transaction below.
3. Create new Java Endpoint Service and Interface and DTO.
4. Invocation of existing Business Service from Service Catalogue
5. Link the Front End Screens to the Business Service via Service XSL's as well as Validation Entries to validate these Business Services.
6. If Any Host Interfacing Component then Create Host Adaptors or use exiting Host Adaptors.

4.1 CREATE FRONT END SCREEN

Oracle FLEXCUBE Direct Banking uses XML-XSLT technology to generate the User Interface. The Framework follows a standard screen design XML (LEAP/mLEAP) to paint the screen. Oracle FLEXCUBE Direct Banking provides GUI designing tool (refer section above) for creating and modifying LEAP XML's for Internet and mLEAP XML's for mobile apps and browsers.

Each front screen would need the following components which would need to be created

4.1.1 OPEN DEVELOPMENT WORKBENCH

The developer would need to create the following

- LEAP and mLEAP Data Structures
- Javascript to be created

4.2 ADD NEW TRANSACTION (TXN)

Oracle FLEXCUBE Direct Banking needs the following entries to be updated for a successful transaction to be appearing for a particular user type and entity and channel combination.

4.2.1 SAMPLE DATABASE ENTRIES

These are the sample database entries which would need to be applied on the ADMIN schema

4.2.1.1 TABLE: MSTTXN

```
insert into msttxn (IDTXN, DESCRIPTION, TXNGROUP, IDSEQ, FLAGSERVICEREQUEST, GIFNAME, ISMENUTXN, TOKEN1, TOKEN2, TOKEN3, TOKEN4, TOKEN5, IDPROXYREQD, IDAPP, ENABLETXNBLACKOUT, LIMITSALLOWED, TYPECUST, CODTXN, IDMODULE, TYPETXN, AUTHPARAM, ADTNL_PARAMS, REF_ACCTTYPE, PROXYTXNID, LAYOUT_TYPE, ALLOWED_PRIVILEGES, ISEXTERNAL, LIVEHELPMODULEID)
values
('LOT', 'LOAN ORIGINATION', '000', 3, 'N', '', 'Y', '1', 'M', '05.08', '', '08', 'Y', 'RR', 'Y', 'N', '', 'LNIN', '', 'I', '00000', '', 'L', '', '', '', 'N', 'NON');
```

4.2.1.2 TABLE: MSTUSERTYPETXN

```
insert into mstusertypetxn (ID_ENTITY, USERTYPE, IDTXN, INITAUTHID, IDCHANNEL, TYPECUST, LIMITSALLOWED, TXNGROUP, IDSEQ, FLAGSERVICEREQUEST, GIFNAME, ISMENUTXN, TOKEN1, TOKEN2, TOKEN3, TOKEN4, TOKEN5, ENABLETXNBLACKOUT, INITREQUESTID, CODTXN, IDMODULE, AUTHALERTREQUIRED, REF_USERTYPE, ALERTPARAM, ISENABLED, CUTOFFALLOWED, ALLOWED_PRIVILEGES, ISEXTERNAL, QUICKTASK, ISTOOL, USERAGENTREQID, ISDEFAULTNAV)
values
('B001', 'ECU', 'LOT', '', '01', '', 'N', '000', null, 'N', '', 'Y', '', 'M', '', '', '1', 'Y', '', '', '', 'N', '', '', 'Y', 'N', 'I', 'N', 'N', 'Y', '', 'N');
```

4.2.1.3 TABLE: MSTCHANNELATS

```
insert into mstchannelats (IDREQUEST, TYPPLUGIN, AUDITREQUIRED, IDCHANNEL, IDTXN, REQUIRESLOGIN, CONTENTSTYLE, NAMRESOURCE, IDSERVICE, IDAPP, FLAGPREPROCESS, FLAGPOSTPROCESS, NAMEOTRESOURCE, NAMEOSRESOURCE,
```

```
ISONLYVALIDATEREQUEST, IDSERVICE_EOT, IDSERVICE_EOS, IDREQUEST_EOT, AUTHREQUIRED, TXNPWDREQUIRED, FLAGPAGINATION, ADTNL_PARAMS, FLGORCH, ISONLYAUTHVALIDATE, ISGENERICTEMPLATE, HASEXTENDEDRESPONSE, ALERTREQUIRED, VALID_IDREQUEST, TOKEN1, TOKEN2, TOKEN3, TYPE_REQUEST)
values
('RRLOT01', 'C', 'N', '01', 'LOT', 'Y', 'DHTML', 'genericscreentemplate.xsl', 'prepareloanorigination.xsl', 'RR', '', '', 'eot.xsl', 'eos.xsl', 'N', '', '', '', 'N', 'N', null, '', 'C', 'N', 'N', 'Y', 'N', '', '', '', '', 'T');

insert into mstchannelats (IDREQUEST, TYPPLUGIN, AUDITREQUIRED, IDCHANNEL, IDTXN, REQUIRESLOGIN, CONTENTSTYLE, NAMRESOURCE, IDSERVICE, IDAPP, FLAGPREPROCESS, FLAGPOSTPROCESS, NAMEOTRESOURCE, NAMEOSRESOURCE, ISONLYVALIDATEREQUEST, IDSERVICE_EOT, IDSERVICE_EOS, IDREQUEST_EOT, AUTHREQUIRED, TXNPWDREQUIRED, FLAGPAGINATION, ADTNL_PARAMS, FLGORCH, ISONLYAUTHVALIDATE, ISGENERICTEMPLATE, HASEXTENDEDRESPONSE, ALERTREQUIRED, VALID_IDREQUEST, TOKEN1, TOKEN2, TOKEN3, TYPE_REQUEST)
values
('RRLOT02', 'C', 'N', '01', 'LOT', 'Y', 'DHTML', 'genericscreentemplate.xsl', 'initiateloanorigination.xsl', 'RR', '', '', 'genericscreentemplate.xsl', 'eos.xsl', 'N', 'prepareloanorigination.xsl', '', 'RRLOT01', 'N', 'N', null, '', 'C', 'N', 'N', 'Y', 'N', '', '', '', '', 'T');
```

Following are the sample database entries which would need to be applied on the `ADMIN` schema for using UI Download feature of FCDB.

4.2.1.4 TABLE: MSTUIDOWNLOAD

```
insert into mstuidownload (ID_ENTITY, TYPEUSER, IDTXN, TYPEDOWNLOAD, TYPEFORMAT, PATH, IDREQUEST, RECORDSPERPAGE, SORTCOLUMN, SORTORDER, ADTNL_PARAMS, IDCHANNEL)
values
('B001', 'ECU', 'ITB', 'UD', 'PDF,XLS,HTML,RTF', '/faml/response/beneficiaryresponsedto/beneficiarydto/beneficiarydto', 'RRBTG04', 10, '2', 'A',
'FIXED.COLUMN.TYPE=C;FIXED.COLUMN.ID.PARAM=0;FIXED.COLUMN.ID.PARAM=1;FIXED.COLUMN.ID.PARAM=2;FIXED.COLUMN.ID.PARAM=3;FIXED.COLUMN.ID.PARAM=4;FIXED.COLUMN.ID.PARAM=5;FIXED.COLUMN.ID.PARAM=6;FIXED.COLUMN.ID.PARAM=7;FIXED.COLUMN.ID.PARAM=8;FIXED.COLUMN.ID.PARAM=9;FIXED.COLUMN.ID.PARAM=10;FIXED.COLUMN.ID.PARAM=11;FIXED.COLUMN.ID.PARAM=12;FIXED.COLUMN.DELIMITER=~;FIXED.COLUMN.DISABLE=//faml/response/beneficiaryresponsedto/extendedresponse/extendedresponsedto/user/userdto/iduser!=//faml/response/beneficiaryresponsedto/beneficiarydto/beneficiarydto/idcreator', '01');
```

4.2.1.5 TABLE: MSTUIDOWNLOADPARAM

```
insert into mstuidownloadparams (ID_ENTITY, TYPEUSER, IDTXN, IDPARAM, NAMPARAM, PARAMSEQ, TYPEPARAM, ISENABLED, ISFIXED, NAMPATH, ISLINK, NAMFUNCTION, ARGFUNCTION, ALIGN, IDREQUEST, FIELDLENGTH, DYNAMICNAMPATH, REFIDPARAM, ADTNL_PARAMS, TYPEFIELD, IDCHANNEL)
values
('B001', 'ECU', 'ITB', 0, 'K_BENEFICIARY_ID', 0, 'S', 'Y', 'Y', 'beneficiaryid', 'Y', 'viewRecords', '0~^5~^6', 'L', 'RRBTG04', null, '*', '', '', 'S', '01');
```

Following are the sample database entries which would need to be applied on the `ADMIN` schema for registering transaction as widgets in FCDB.

4.2.1.6 TABLE: MSTWIDGET

```
insert into mstwidget (ID_ENTITY, USERTYPE, WIDGET_LOCATION, WIDGET_TYPE, WIDGET_SEQ, IDREQUEST, EXTERNAL_INFO, HASNAVIGATION, MINIMIZABLE, REFRESHABLE, CUSTOM_CONTENT_CLASS, UDF1, UDF2, UDF3, UDF4, UDF5, ISENABLED, IDTXN, IDCHANNEL, WIDGET_HEIGHT, MORELINK, ISMANDATORY)
values
('B001', 'ECU', 'R', 'I', 1, 'RRIMS08', '', 'Y', 'Y', 'Y', '', '', '', '', '', '', 'Y', 'IMS', '01', 160, 'Y', 'N');
```

4.3 CREATE NEW JAVA ENDPOINT SERVICE AND INTERFACE

Each 'Service', within the Oracle FLEXCUBE Direct Banking Services Architecture, defines an interface or a method within an existing interface that provides the business definitions for that function. The business definitions always define and exchange of DTOS (Request DTO and Response DTO) that require to be implemented by the implementation component or the endpoint

The endpoint provides the implementations for the business definitions to access data from various 'host data sources' using appropriate interface mechanisms. Based on the functionality the developer would need to create the following

4.3.1 INTERFACES

The Business Definitions are represented as Java Interfaces. The business definitions method signatures that provide the required business definitions for data exchange.

4.3.2 DATA TRANSFER OBJECTS

The Data Transfer Objects or DTOs are the data carriers within and external to the system. The DTOs provide transparent access to the underlying data (either incoming in the request or outgoing in the response).

The DTOs within the Services Architecture support specific data types as indicated below. All DTOs should define only the following data types.

Data Types

- All Primitive Java Types are supported as data types.
- Arrays of all Primitive Java Types are supported
- Other than primitive data types, only `java.util.Date` and `java.lang.String` instances are allowed to be used.
- DTOs can contain other DTOs extending from the `BaseDTO`, `RequestDTO` or `ResponseDTO`.

4.3.3 ENDPOINTS

Service Endpoints are implementation components for the given service interface definitions. The endpoints are the plug-in components which adapt to the changing business requirements by host, region, implementation specific parameters etc.

Steps to create a new service

1) Create a new Interface with definition of all the services to implement.

2) The new Interface should extend `com.iflex.fcat.services.TransactionService` if the expected service will be transactional. If the expected service will be singleton, extend `com.iflex.fcat.services.SingletonService`

```
package com.iflex.fcat.services.apps.interfaces;

import com.iflex.fcat.services.TransactionContext;
package com.iflex.fcat.services.apps.interfaces;

import com.iflex.fcat.services.TransactionContext;
import com.iflex.fcat.services.TransactionService;
import com.iflex.fcat.services.apps.dtos.LoanOriginationRequestDTO;
import com.iflex.fcat.services.apps.dtos.LoanOriginationResponseDTO;
import com.iflex.fcat.services.apps.dtos.NullRequestDTO;
import com.iflex.fcat.services.apps.dtos.PrepareLoanOriginResponseDTO;

public interface
        LoanOriginationInterface
extends
        TransactionService
{

        public PrepareLoanOriginResponseDTO prepareLoanOrigin(
                TransactionContext      p_ctx
        ,       NullRequestDTO                  p_req
        )throws Exception;
        public LoanOriginationResponseDTO initiateLoanOrigin(
                TransactionContext      p_ctx
        ,       LoanOriginationRequestDTO               p_req
        )throws Exception;

}
```

3) Create required Request/Response DTOs.
4) New RequestDTOs should extend `com.iflex.fcat.services.RequestDTO`
5) New ResponseDTOs should extend `com.iflex.fcat.services.ResponseDTO`
6) Create a new endpoint implementing the above created Interface.
7) Implement the services in the endpoint.

```
package com.iflex.fcat.services.apps.endpoints;

import com.iflex.fcat.services.ServiceConstants;
import com.iflex.fcat.services.TransactionContext;
import com.iflex.fcat.services.apps.dtos.LoanOriginationRequestDTO;
import com.iflex.fcat.services.apps.dtos.LoanOriginationResponseDTO;
import com.iflex.fcat.services.apps.dtos.NullRequestDTO;
import com.iflex.fcat.services.apps.dtos.PrepareLoanOriginResponseDTO;
import com.iflex.fcat.services.apps.interfaces.LoanOriginationInterface;
import com.iflex.fcat.services.hostinterface.dtos.HostRequestDTO;
import com.iflex.fcat.services.hostinterface.dtos.HostResponseDTO;

public final class
      LoanOriginationService
implements
      LoanOriginationInterface
{

      public PrepareLoanOriginResponseDTO prepareLoanOrigin(
            TransactionContext      p_context
      ,     NullRequestDTO          p_request
      ) throws Exception {

            PrepareLoanOriginResponseDTO l_resp = new PrepareLoanOriginResponseDTO();

            //Implement your Business Logic for this method

            l_resp.result.returnCode = ServiceConstants.RESULT_CODE_SUCCESS;
            return l_resp;
      }
```

```
	public LoanOriginationResponseDTO initiateLoanOrigin(
		TransactionContext			p_context
	,	LoanOriginationRequestDTO	p_request
	) throws Exception {

		HostRequestDTO		l_hreq	= new HostRequestDTO();
		HostResponseDTO		l_hresp	= null;
		LoanOriginationResponseDTO l_resp = new LoanOriginationResponseDTO();

		//Implement your Business Logic for this method

		l_resp.result.returnCode = ServiceConstants.RESULT_CODE_SUCCESS;
		return l_resp;
	}

}
```

8) A service is made available in the application by registering it in table `MSTSERVICES`.
9) Add validation information for the service in table `TXN_DATA_MASTER` and `TXN_DATA`.
10) Add service mapping information if needed in table `MSTSERVICESMAP`.

4.3.4 DATA TRANSFER OBJECTS

The Data Transfer Objects or DTOs are the data carriers within and external to the system. The DTOs provide transparent access to the underlying data (either incoming in the request or outgoing in the response).

The DTOs within the Services Architecture support specific data types as indicated below. All DTOs should define only the following data types.

Data Types

- All Primitive Java Types are supported as data types.
- Arrays of all Primitive Java Types are supported
- Other than primitive data types, only `java.util.Date` and `java.lang.String` instances are allowed to be fined.

- DTOs can contain other DTOs extending from the `BaseDTO`, `RequestDTO` or `ResponseDTO`.

4.3.5 SAMPLE DATABASE ENTRIES

These are the sample database entries which would need to be applied on the `ADMIN` schema

4.3.5.1 TABLE: MSTSERVICES

```
insert into mstservices
(NAMSERVICE, INTERFACE, ENDPOINT, VERSION, ISENABLED, ISMULTIPHASE, ISROLLBACKONLY, CUSTOM_HELPER, NAMMETHOD, NAMTXNCONTEXT, ISAPI, ISAUDITREQUIRED, AUTHDTOMAPPER, TXNAUTHDTOMAPPER, VALIDATIONVERSION, EXTENDEDRESPONSEHANDLER)
values
('LoanOriginationInterface.LoanOriginationService.initiateLoanOrigin', 'com.iflex.fcat.services.apps.interfaces.LoanOriginationInterface', 'com.iflex.fcat.services.apps.endpoints.LoanOriginationService', 0, 'Y', 'N', 'N', '', 'initiateLoanOrigin', '', 'N', 'N', '', '', 0, '');
```

4.3.5.2 TABLE: MSTSERVICESMAP

```
insert into MSTSERVICESMAP
(ID_ENTITY, TYPEUSER, NAMSERVICE, ISENABLED, INPUTVERSION, OUTPUTVERSION)
values
('B001', 'ECU', 'InternationalAccountTransferServiceInterface.InternationalAccountTransferService.internationalTransfer', 'Y', 0, 100);
```

4.3.5.3 TABLE: TXN_DATA_MASTER

```
insert into txn_data_master
(IDREQUEST, TXN_DESC, CUSTOM_VALIDATOR, NAMSCHEMA, TYPEREQUEST, REFREQUEST, COMMENTS, ISFINREQUEST, TOTALFIELDS, MODEREQUEST, SEPSTRING, ENCODING, PREFIX, POSTFIX, TERMINATOR, ADJUSTMENT, BIZTYPE, FREETEXT, ISLENIENT)
values
```

```
('LOANORIGINATIONINTERFACE.LOANORIGINATIONSERVICE.INITIATELOANORIGIN', 'Initiate Loan Origination', '', '', 'S', '', '', 'N', 0, 0, '', '', '', '', '', null, '', '', 'N');

insert into txn_data_master (IDREQUEST, TXN_DESC, CUSTOM_VALIDATOR, NAMSCHEMA, TYPEREQUEST, REFREQUEST, COMMENTS, ISFINREQUEST, TOTALFIELDS, MODEREQUEST, SEPSTRING, ENCODING, PREFIX, POSTFIX, TERMINATOR, ADJUSTMENT, BIZTYPE, FREETEXT, ISLENIENT)
values
 ('LOANORIGINATIONINTERFACE.LOANORIGINATIONSERVICE.INITIATELOANORIGIN.CUSTDETAILS', 'Initiate Loan Origination', '', '', 'S', '', '', 'N', 0, 0, '', '', '', '', '', null, '', '', 'N');

insert into txn_data_master (IDREQUEST, TXN_DESC, CUSTOM_VALIDATOR, NAMSCHEMA, TYPEREQUEST, REFREQUEST, COMMENTS, ISFINREQUEST, TOTALFIELDS, MODEREQUEST, SEPSTRING, ENCODING, PREFIX, POSTFIX, TERMINATOR, ADJUSTMENT, BIZTYPE, FREETEXT, ISLENIENT)
values
('LOANORIGINATIONINTERFACE.LOANORIGINATIONSERVICE.INITIATELOANORIGIN.PRESENTADDR', 'Initiate Loan Origination', '', '', 'S', '', '', 'N', 0, 0, '', '', '', '', '', null, '', '', 'N');

insert into txn_data_master (IDREQUEST, TXN_DESC, CUSTOM_VALIDATOR, NAMSCHEMA, TYPEREQUEST, REFREQUEST, COMMENTS, ISFINREQUEST, TOTALFIELDS, MODEREQUEST, SEPSTRING, ENCODING, PREFIX, POSTFIX, TERMINATOR, ADJUSTMENT, BIZTYPE, FREETEXT, ISLENIENT)
values
 ('LOANORIGINATIONINTERFACE.LOANORIGINATIONSERVICE.INITIATELOANORIGIN.PERMADDR', 'Initiate Loan Origination', '', '', 'S', '', '', 'N', 0, 0, '', '', '', '', '', null, '', '', 'N');

insert into txn_data_master (IDREQUEST, TXN_DESC, CUSTOM_VALIDATOR, NAMSCHEMA, TYPEREQUEST, REFREQUEST, COMMENTS, ISFINREQUEST, TOTALFIELDS, MODEREQUEST, SEPSTRING, ENCODING, PREFIX, POSTFIX, TERMINATOR, ADJUSTMENT, BIZTYPE, FREETEXT, ISLENIENT)
values
 ('LOANORIGINATIONINTERFACE.LOANORIGINATIONSERVICE.INITIATELOANORIGIN.EMPADDR', 'Initiate Loan Origination', '', '', 'S', '', '', 'N', 0, 0, '', '', '', '', '', null, '', '', 'N');
```

4.3.5.4 TXN_DATA

```
insert into txn_data
(IDREQUEST, FIELDNAME, REFFIELDNAME, FIELDFORMAT, VAL_ENUM, DEFAULT_VALUE, CUSTOM_VALIDATOR, ISREQUESTFLD, VALIDATION_REQUIRED, ISMANDATORY, ERRORCODE, FLD_NUMSEQUENCE, TOKEN1, TOKEN2, TOKEN3, LENGTH, FIXEDLENFLAG, JUSTIFICATION, FILLCHAR, DELIMITER, PREFIXLEN, POSTFIXLEN, TYPFIELD, NUMOCC, IDOCCFIELD, ISPARAMFIELD, ISPLACEHOLDER, ENRICHMENT, ISAGGREGATE, MULTIPLIER, MINOCCUR, MAXOCCUR, ISHASHFIELD, FREETEXT, DIVIDER, ENPARAMFIELDS)
values
('LOANORIGINATIONINTERFACE.LOANORIGINATIONSERVICE.INITIATELOANORIGIN', 'AMOUNT', 'loanAmount', '', '', '', '', 'Y', 'Y', 'Y', '9100011', 0, 'fldloanamt', '', '', null, '', '', '', '', null, null, '', null, null, 'N', 'N', '', 'N', null, 1, 1, 'N', '', -1, '');
```

4.3.5.5 CONFIGURING NEW PRE ENDPOINT HELPER SERVICES

```
insert into mstservices (NAMSERVICE, INTERFACE, ENDPOINT, VERSION, ISENABLED, ISMULTIPHASE, ISROLLBACKONLY, CUSTOM_HELPER, NAMMETHOD, NAMTXNCONTEXT, ISAPI, ISAUDITREQUIRED, AUTHDTOMAPPER, TXNAUTHDTOMAPPER, VALIDATIONVERSION, EXTENDEDRESPONSEHANDLER)
values
('MultipleFundsTransferServiceInterface.MultipleFundsTransferService.doMultipleInternalTransfer',
'com.iflex.fcat.services.apps.interfaces.MultipleFundsTransferServiceInterface',
'com.iflex.fcat.services.apps.endpoints.MultipleFundsTransferService', 100, 'Y', 'N', 'N',
'com.iflex.fcat.services.MultiTransactionServiceHelper', 'doMultipleInternalTransfer', '', 'N', 'N',
'com.iflex.fcat.services.apps.auth.authdtomapper.MultipleFundTransferAuthDTOMapper', '', 0, '');
```

4.3.5.6 CONFIGURING IN MSTENTITYUSERTYPES USER LIFE CYCLE HANDLERS

```
insert into MSTENTITYUSERTYPES (ID_ENTITY, TYPEUSER, DESCRIPTION, ISENABLED, NAMAUTHENTICATOR, LIMITSALLOWED, IDSORT, REF_USERTYPE, IS_ROLE_CUSTPROF_ALLOWED, TXNINITFLOWHANDLER, IS_ROLE_DEFAULT_ALLOWED, TYPUSERTOKEN, USERLIFECYCLEEXTHANDLER, TYPEROLE, PROXYUSERTYPE, LOOKUPTYPE, HASCUSTOMER, HOSTTYPE, ISACTIVEFLAG, TOKEN1, TOKEN2, TOKEN3, TOKEN4, TOKEN5, TOKEN6, TOKEN7, TOKEN8, TOKEN9, TOKEN10)
values
('B001', 'ECU', 'CORPORATE USER ', 'Y', 'com.iflex.fcat.services.apps.DefaultPasswordAuthenticator', 'Y', 201, '', 'Y',
'com.iflex.fcat.services.apps.handlers.DefaultTransactionPinInitHandler', 'Y', 'S',
'com.iflex.fcat.services.apps.handlers.DefaultUserLifeCycleAlertHandler', 'B', '', 'N', 'A', '', 'Y', null, null, 'Y', 'N', 10, 'Y', '', 'Y', '', '');
```

4.3.5.7 CONFIGURING IN SERVICE ENDPOINT EXTENDED HANDLERS

```
insert into mstservices (NAMSERVICE, INTERFACE, ENDPOINT, VERSION, ISENABLED, ISMULTIPHASE, ISROLLBACKONLY, CUSTOM_HELPER, NAMMETHOD, NAMTXNCONTEXT, ISAPI, ISAUDITREQUIRED, AUTHDTOMAPPER, TXNAUTHDTOMAPPER, VALIDATIONVERSION, EXTENDEDRESPONSEHANDLER)
values
('UserServiceInterface.UserService.getListCustomer', 'com.iflex.fcat.services.apps.interfaces.UserServiceInterface', 'com.iflex.fcat.services.apps.endpoints.UserService', 0, 'Y', 'N', 'N', '', 'getListCustomer', '', 'N', 'N', '', '', 0, 'com.iflex.fcat.services.apps.CompleteExtResponseHandler');
```

4.3.5.8 USING GENERIC PAYMENT SERVICE AUTHORIZATION

```
insert into mstservices (NAMSERVICE, INTERFACE, ENDPOINT, VERSION, ISENABLED, ISMULTIPHASE, ISROLLBACKONLY, CUSTOM_HELPER, NAMMETHOD, NAMTXNCONTEXT, ISAPI, ISAUDITREQUIRED, AUTHDTOMAPPER, TXNAUTHDTOMAPPER, VALIDATIONVERSION, EXTENDEDRESPONSEHANDLER)
values
('GenericPaymentServiceInterface.GenericPaymentService.initiateGenericPaymentDetails',
'com.iflex.fcat.services.apps.interfaces.GenericPaymentServiceInterface', 'com.iflex.fcat.services.apps.endpoints.GenericPaymentService',
0, 'Y', 'N', 'N', '', 'initiateGenericPaymentDetails', '', 'N', 'N',
'com.iflex.fcat.services.apps.auth.authdtomapper.GenericPaymentsDTOMapper', '', 0, '');
```

1.1.1.1 CONFIGURING AUTHORIZATION

```
insert into mstchannelats (IDREQUEST, TYPPLUGIN, AUDITREQUIRED, IDCHANNEL, IDTXN, REQUIRESLOGIN, CONTENTSTYLE, NAMRESOURCE, IDSERVICE, IDAPP, FLAGPREPROCESS, FLAGPOSTPROCESS, NAMEOTRESOURCE, NAMEOSRESOURCE, ISONLYVALIDATEREQUEST, IDSERVICE_EOT, IDSERVICE_EOS, IDREQUEST_EOT, AUTHREQUIRED, TXNPWDREQUIRED, FLAGPAGINATION, ADTNL_PARAMS, FLGORCH, ISONLYAUTHVALIDATE, ISGENERICTEMPLATE, HASEXTENDEDRESPONSE, ALERTREQUIRED, VALID_IDREQUEST, TOKEN1, TOKEN2, TOKEN3, TYPE_REQUEST)
values
('RROAT01', 'C', 'N', '01', 'OAT', 'Y', 'DHTML', 'genericscreentemplate.xsl', 'genericpaymentprepare.xsl', 'RR', '', '', 'eot.xsl', 'eos.xsl', 'N', '', '', '', 'N', 'N', null, '', 'C', 'N', 'Y', 'Y', 'N', '', '', '', '', 'T');
```

```
insert into mstchannelats (IDREQUEST, TYPPLUGIN, AUDITREQUIRED, IDCHANNEL, IDTXN, REQUIRESLOGIN, CONTENTSTYLE, NAMRESOURCE, IDSERVICE, IDAPP, FLAGPREPROCESS, FLAGPOSTPROCESS, NAMEOTRESOURCE, NAMEOSRESOURCE, ISONLYVALIDATEREQUEST, IDSERVICE_EOT, IDSERVICE_EOS, IDREQUEST_EOT, AUTHREQUIRED, TXNPWDREQUIRED, FLAGPAGINATION, ADTNL_PARAMS, FLGORCH, ISONLYAUTHVALIDATE, ISGENERICTEMPLATE, HASEXTENDEDRESPONSE, ALERTREQUIRED, VALID_IDREQUEST, TOKEN1, TOKEN2, TOKEN3, TYPE_REQUEST)
values
('RROAT02', 'C', 'N', '01', 'OAT', 'Y', 'DHTML', 'genericscreentemplate.xsl', 'genericpaymentinit.xsl', 'RR', '2', '2', 'genericscreentemplate.xsl', 'eos.xsl', 'Y', 'genericpaymentprepare.xsl', '', 'RROAT01', 'Y', 'N', null, '', 'C', 'Y', 'Y', 'Y', 'N', 'RROAT01,RROAT99,RRDDT01,RRDDT02,RRDDT03', '', '', '', 'T');

insert into mstchannelats (IDREQUEST, TYPPLUGIN, AUDITREQUIRED, IDCHANNEL, IDTXN, REQUIRESLOGIN, CONTENTSTYLE, NAMRESOURCE, IDSERVICE, IDAPP, FLAGPREPROCESS, FLAGPOSTPROCESS, NAMEOTRESOURCE, NAMEOSRESOURCE, ISONLYVALIDATEREQUEST, IDSERVICE_EOT, IDSERVICE_EOS, IDREQUEST_EOT, AUTHREQUIRED, TXNPWDREQUIRED, FLAGPAGINATION, ADTNL_PARAMS, FLGORCH, ISONLYAUTHVALIDATE, ISGENERICTEMPLATE, HASEXTENDEDRESPONSE, ALERTREQUIRED, VALID_IDREQUEST, TOKEN1, TOKEN2, TOKEN3, TYPE_REQUEST)
values
('RROAT03', 'C', 'Y', '01', 'OAT', 'Y', 'DHTML', 'genericscreentemplate.xsl', 'genericpaymentinit.xsl', 'RR', '6', '', 'genericscreentemplate.xsl', 'eos.xsl', 'N', 'genericpaymentprepare.xsl', '', 'RROAT01', 'Y', 'N', null, '', 'C', 'N', 'Y', 'Y', 'N', 'RROAT02,RRDTD03', '', '', '', 'T');
```

4.3.6 TESTING THE SERVICE INTERFACE

A utility 'TestService' is provided with the product for the purpose of unit testing a new / modified service. This utility accepts the service request XML and returns the service response XML, for scrutiny. If there are any issues in this request-response process, they can be identified and rectified at this stage easily, even if the corresponding screens have not been designed yet.

4.3.6.1 PRE-REQUISITES:

- The application server hosting Oracle FLEXCUBE Direct Banking must be running
- The 'A1' data source must be correctly defined in your application server.
  For the details of data source setting, please refer to Oracle FLEXCUBE Direct Banking installation guide.

- The 'FNDI.A1.java.naming.factory.initial' and 'FNDI.A1.java.naming.provider.url' properties in **fcat.properties** file must be correctly defined, as per the settings applicable to your environment

4.3.6.2 PROCESS

- Create a new text file (location, file name and extension does not matter). This file will be used as an input for the service testing utility
- Write a valid service request XML in this file. You can either write it manually, if you have knowledge of FCDB's XML structure, or follow the steps below to 'wire-tap' a service request from the running application:
  - Open the logger.properties file located in application's 'home' folder (<FCDB base directory>/system/home)
  - Go to the log4j.category.com.iflex.fcat.channels.plugins.FLEXCUBEConnectHostPlugin property
  - Set the logging level to 'DEBUG' and provide appropriate location for log file
  - Restart the application server
  - The log file will be named FLEXCUBEConnectHostPlugin.log will be generated at location specified above. FCDB application will log all service requests and service responses in this file.
  - Pick up the required service request string and copy it to the plain-text file you had created
- Save the plain-text file
- Run TestService tool with the following arguments, in sequence and separated by a space
  - Base directory location of Oracle FLEXCUBE Direct Banking application
  - Complete path of the file containing service request XML
    Note that the file path must not contain any spaces
  - Text-encoding (UTF-8, if you don't know what to use)
- If you are using a demo certificate (in Weblogic environment), you are additionally required to specify the following JVM argument:
  - -Dweblogic.security.TrustKeyStore=DemoTrust

4.3.6.3 RESULT

The service response XML will be displayed on screen. You can copy this to a text-file for analysis

4.4 INVOKE EXISTING JAVA ENDPOINT SERVICE AND INTERFACE

Each 'Service', within the Oracle FLEXCUBE Direct Banking Services Architecture, defines an interface or a method within an existing interface that provides the business definitions for that function. The business definitions always define and exchange of DTOS (Request DTO and Response DTO) that require to be implemented by the implementation component or the endpoint

Oracle FLEXCUBE Direct Banking Service Catalogue gives a list of all such Business Services with their signatures.

A Sample Invocation of a Existing Business Service is as follows

```
public InformationResponseDTO getInformation(
      TransactionContext         p_context
,     InformationResponseDTO     p_request
) {

//------------ account number list initialization -----------------

          l_actresp = new GetAccountsResponseDTO();
          l_actresp    = (GetAccountsResponseDTO)ServiceManager.invokeService(
                                  S_GET_ACCOUNT_LIST
                              ,   p_request
                              ,   null
                              ,   p_context
                              ,   true
                              );
          if(!ServiceUtils.isSuccess(l_actresp)) {
            l_resp.result = l_actresp.result;
            return l_resp;
          }
          l_resp.custAcctDetails = l_actresp.custAccounts;
}
```

The above example shows how to invoke the out of the box Account Service.

A Sample Invocation of a Host Adaptor from the Business Service is as follows

```
public InformationResponseDTO getInformation(
      TransactionContext        p_context
,     InformationResponseDTO    p_request
) {

//------------ account number list initialization -----------------

          l_actresp = new GetAccountsResponseDTO();
          l_actresp    = (GetAccountsResponseDTO)ServiceManager.invokeService(
                                    "AccountsServiceInterface.AccountsService.getCasaAccounts"
                               ,    p_request
                               ,    null
                               ,    p_context
                               ,    true
                               );
          if(!ServiceUtils.isSuccess(l_actresp)) {
            l_resp.result = l_actresp.result;
            return l_resp;
          }
          l_resp.custAcctDetails = l_actresp.custAccounts;
```

```
        l_hreq          = HostAdapterHelper.buildHostRequest (
                                    p_request
                               ,    p_context
                               );

            l_hresp           = HostAdapterManager.processRequest (
                                    l_hreq
                               );
            l_resp      = (NullResponseDTO) l_hresp.response;
}
```

The above example shows how to invoke the out of the Host Manager .processRequest.

4.5 LINK THE FRONT SCREENS TO JAVA BUSINESS SERVICE

Each 'Service', within the Oracle FLEXCUBE Direct Banking Services Architecture can be linked to the front end screens via a GUI components called SERVICE XSL

4.5.1 SERVICE XSL

This "SERVICE XSL" is configured as part of the Channel ATS entries mentioned below. This links the request which is submitted from the client side to be linked to Request DTO.

A sample Request XSL showing the following

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="default" version="1.0">
<xsl:output doctype-public="-//W3C//DTD HTML 4.01 Transitional//EN" method="xml" indent="yes" />
<xsl:template match="/">
<ServiceRequest>
<serviceName>LoanOriginationInterface.LoanOriginationService.initiateLoanOrigin</serviceName>
<request>
<LoanOriginationRequestDTO>
	<loanProduct><xsl:value-of select="//faml/request/fldselloancat"/></loanProduct>
	<loanProductDesc></loanProductDesc>
	<loanPurpose><xsl:value-of select="//faml/request/fldloanpurpose"/></loanPurpose>
	<loanAmount><xsl:value-of select="//faml/request/fldloanamt"/></loanAmount>
	<loanCurrency><xsl:value-of select="//faml/request/fldloancurr"/></loanCurrency>
	<loanTenor><xsl:value-of select="//faml/request/fldtenor"/></loanTenor>
	<custDetails>
		<CustomerDetailDTO>
			<fullName><xsl:value-of select="substring-after(//faml/request/fldcustname,'~')"/></fullName>
			<idCustomer><xsl:value-of select="substring-before(//faml/request/fldcustname,'~')"/></idCustomer>
			<sex><xsl:value-of select="//faml/request/fldgender"/></sex>
			<maritalStatusType><xsl:value-of select="//faml/request/fldmaritalstat"/></maritalStatusType>
			<nationality><xsl:value-of select="//faml/request/fldnationality"/></nationality>
			<birthdate><xsl:value-of select="//faml/request/flddob"/></birthdate>
			<email><xsl:value-of select="//faml/request/fldemail"/></email>
		</CustomerDetailDTO>
	</custDetails>
	<mobileno><xsl:value-of select="//faml/request/fldmobno"/></mobileno>
	<income><xsl:value-of select="//faml/request/fldincamt"/></income>
```

```
        <presentAddr>
                <AddressDTO>
                        <line1><xsl:value-of select="//faml/request/fldaddr1"/></line1>
                        <line2><xsl:value-of select="//faml/request/fldaddr2"/></line2>
                        <line3><xsl:value-of select="//faml/request/fldaddr3"/></line3>
                </AddressDTO>
        </presentAddr>
        <empName><xsl:value-of select="//faml/request/fldempname"/></empName>
        <empDesgination><xsl:value-of select="//faml/request/flddesignation"/></empDesgination>
</LoanOriginationRequestDTO>
</request>
</ServiceRequest>
</xsl:template>
</xsl:stylesheet>
```

4.5.2 SAMPLE DATABASE ENTRIES

These are the sample database entries which would need to be applied on the ADMIN schema

4.5.2.1 TABLE: MSTCHANNELATS

```
insert into mstchannelats
(IDREQUEST, TYPPLUGIN, AUDITREQUIRED, IDCHANNEL, IDTXN, REQUIRESLOGIN, CONTENTSTYLE, NAMRESOURCE, IDSERVICE, IDAPP, FLAGPREPROCESS, FLAGPOSTPROCESS, NAMEOTRESOURCE, NAMEOSRESOURCE, ISONLYVALIDATEREQUEST, IDSERVICE_EOT, IDSERVICE_EOS, IDREQUEST_EOT, AUTHREQUIRED, TXNPWDREQUIRED, FLAGPAGINATION, ADTNL_PARAMS, FLGORCH, ISONLYAUTHVALIDATE, ISGENERICTEMPLATE, HASEXTENDEDRESPONSE, ALERTREQUIRED, VALID_IDREQUEST, TOKEN1, TOKEN2, TOKEN3, TYPE_REQUEST)
values
('RRLOT01', 'C', 'N', '01', 'LOT', 'Y', 'DHTML', 'genericscreentemplate.xsl', 'prepareloanorigination.xsl', 'RR', '', '', 'eot.xsl', 'eos.xsl', 'N', '', '', '', 'N', 'N', null, '', 'C', 'N', 'N', 'Y', 'N', '', '', '', '', 'T');

insert into mstchannelats
(IDREQUEST, TYPPLUGIN, AUDITREQUIRED, IDCHANNEL, IDTXN, REQUIRESLOGIN, CONTENTSTYLE, NAMRESOURCE, IDSERVICE, IDAPP, FLAGPREPROCESS, FLAGPOSTPROCESS, NAMEOTRESOURCE, NAMEOSRESOURCE, ISONLYVALIDATEREQUEST, IDSERVICE_EOT, IDSERVICE_EOS, IDREQUEST_EOT, AUTHREQUIRED, TXNPWDREQUIRED, FLAGPAGINATION, ADTNL_PARAMS, FLGORCH, ISONLYAUTHVALIDATE, ISGENERICTEMPLATE, HASEXTENDEDRESPONSE, ALERTREQUIRED, VALID_IDREQUEST, TOKEN1, TOKEN2, TOKEN3, TYPE_REQUEST)
```

```
values
('RRLOT02', 'C', 'N', '01', 'LOT', 'Y', 'DHTML', 'genericscreentemplate.xsl', 'initiateloanorigination.xsl', 'RR', '', '', 'genericscreentemplate.xsl',
'eos.xsl', 'N', 'prepareloanorigination.xsl', '', 'RRLOT01', 'N', 'N', null, '', 'C', 'N', 'N', 'Y', 'N', '', '', '', '', 'T');
```

4.5.2.2 CONFIGURING UI PAGINATION

```
insert into mstchannelats (IDREQUEST, TYPPLUGIN, AUDITREQUIRED, IDCHANNEL, IDTXN, REQUIRESLOGIN, CONTENTSTYLE, NAMRESOURCE,
IDSERVICE, IDAPP, FLAGPREPROCESS, FLAGPOSTPROCESS, NAMEOTRESOURCE, NAMEOSRESOURCE, ISONLYVALIDATEREQUEST,
IDSERVICE_EOT, IDSERVICE_EOS, IDREQUEST_EOT, AUTHREQUIRED, TXNPWDREQUIRED, FLAGPAGINATION, ADTNL_PARAMS, FLGORCH,
ISONLYAUTHVALIDATE, ISGENERICTEMPLATE, HASEXTENDEDRESPONSE, ALERTREQUIRED, VALID_IDREQUEST, TOKEN1, TOKEN2, TOKEN3,
TYPE_REQUEST)
values
('RRBTG04', 'C', 'N', '01', 'BTG', 'Y', 'HTML', 'genericbeneficiarymaintenancepreparеinitial.xsl', 'beneficiarydetailsservice.xsl', 'RR', '', '2',
'genericbeneficiarymaintenancepreparеinitial.xsl', 'eos.xsl', 'N', 'beneficiarymaintenancepreparеinitialservice.xsl', '', 'RRBTG01', 'N', 'N', 1, '', 'C',
'N', 'N', 'Y', 'N', '', '', '', '', 'T');
```

4.5.2.3 CONFIGURING CHANNEL REQUEST STORAGE

```
insert into mstchannelats (IDREQUEST, TYPPLUGIN, AUDITREQUIRED, IDCHANNEL, IDTXN, REQUIRESLOGIN, CONTENTSTYLE, NAMRESOURCE,
IDSERVICE, IDAPP, FLAGPREPROCESS, FLAGPOSTPROCESS, NAMEOTRESOURCE, NAMEOSRESOURCE, ISONLYVALIDATEREQUEST,
IDSERVICE_EOT, IDSERVICE_EOS, IDREQUEST_EOT, AUTHREQUIRED, TXNPWDREQUIRED, FLAGPAGINATION, ADTNL_PARAMS, FLGORCH,
ISONLYAUTHVALIDATE, ISGENERICTEMPLATE, HASEXTENDEDRESPONSE, ALERTREQUIRED, VALID_IDREQUEST, TOKEN1, TOKEN2, TOKEN3,
TYPE_REQUEST)
values
('RROAT01', 'C', 'N', '01', 'OAT', 'Y', 'DHTML', 'genericscreentemplate.xsl', 'genericpaymentprepare.xsl', 'RR', '', '', 'eot.xsl', 'eos.xsl', 'N', '', '', '', 'N',
'N', null, '', 'C', 'N', 'Y', 'Y', 'N', '', '', '', '', 'T');

insert into mstchannelats (IDREQUEST, TYPPLUGIN, AUDITREQUIRED, IDCHANNEL, IDTXN, REQUIRESLOGIN, CONTENTSTYLE, NAMRESOURCE,
IDSERVICE, IDAPP, FLAGPREPROCESS, FLAGPOSTPROCESS, NAMEOTRESOURCE, NAMEOSRESOURCE, ISONLYVALIDATEREQUEST,
IDSERVICE_EOT, IDSERVICE_EOS, IDREQUEST_EOT, AUTHREQUIRED, TXNPWDREQUIRED, FLAGPAGINATION, ADTNL_PARAMS, FLGORCH,
ISONLYAUTHVALIDATE, ISGENERICTEMPLATE, HASEXTENDEDRESPONSE, ALERTREQUIRED, VALID_IDREQUEST, TOKEN1, TOKEN2, TOKEN3,
```

```
TYPE_REQUEST)
values
('RROAT02', 'C', 'N', '01', 'OAT', 'Y', 'DHTML', 'genericscreentemplate.xsl', 'genericpaymentinit.xsl', 'RR', '2', '2', 'genericscreentemplate.xsl',
'eos.xsl', 'Y', 'genericpaymentprepare.xsl', '', 'RROAT01', 'Y', 'N', null, '', 'C', 'Y', 'Y', 'Y', 'N', 'RROAT01,RROAT99,RRDDT01,RRDDT02,RRDDT03', '', '',
'', 'T');
```

4.6 LINK THE BUSINESS SERVICES TO HOST COMPONENTS

This is the component where the Response from the Business Service gets transformed as a XML over which a Host Interfacing GUI XSL gets applied to interfacing with the Core Banking system. Oracle FLEXCUBE Direct Banking support the following communication protocols

- JMS
- EJB
- WebService

It can use the following parameterization available in the Host Interfacing Layer

Term	Reference Document
FLEXML_MODE_EJB	This decides that the EJB mode gets called from the out of the box <<FlexmlHostAdapter>> implemented for transactions traversing towards the host.
FLEXML_MODE_JMS	This decides that the JMS mode gets called from the out of the box <<FlexmlHostAdapter>> implemented for transactions traversing towards the host
FLEXML_MODE_JWS	This decides that the Webservices mode gets called from the out of the box <<FlexmlHostAdapter>> implemented for transactions traversing towards the host
FLEXML.REQUSET.XSL	This decides that the which Request XSL to apply and gets called from the out of the box <<FlexmlHostAdapter>> implemented for transforming FCDB Request DTO XML to an xml which the Host understands.
FLEXML.RESPONSE.XSL	This decides that the which Response XSL to apply on the response received from host and gets called from the out of the box <<FlexmlHostAdapter>> implemented for transforming Host XML in FCDB Response DTO XML

RESPONSE.CLASS.NAME	This decides that the which Response Class Name to cast the response coming <<FlexmlHostAdapter>>
FLEXML.EODCHECK	This decides whether a particular transaction needs EOD check or no. The default values are ❖ Y: check EOD; if EOD in process, put the request for retry post EOD ❖ N: Do not check EOD and initiate; passthrough ❖ T: Check EOD: If EOD in process, terminate request with exception
FLEXML.RETRY.FLAG	This decides whether a particular transaction needs Retry or no. The default values are Y/N. If Retry is Enabled then the transaction is tanked and later retried via a Java Timer.

5 MODIFY BUSINESS FUNCTIONALITY

Oracle FLEXCUBE Direct Banking product gives the facility modify new business functionality by piggy banking on the core framework components mentioned above.

Typically when a existing business functionality needs to be modified or extended, the following would be the key element by which they can be classified

1. Modify new Front Screen and Client Side Javascript
2. Extend new Java Endpoint Service and Interface or add UDF fields to Existing DTO
3. Modify Any Host Interfacing GUI Component and Create New Host Adaptors

5.1 MODIFY FRONT END SCREEN

Oracle FLEXCUBE Direct Banking provides GUI rendering tool (refer section above) for creating and modifying LEAP Data Structure for Internet and mLEAP Data Structures for mobile apps and browsers.

There are also functionalities which do not follow the LEAP methodology of creating screens. We will see how to manage both of them in the coming sections

5.1.1 MODIFY USING OPEN DEVELOPMENT WORKBENCH

The developer would need to do the following

- Identify from the MSTCHANNEL ATS which screen is being modified
- Ensure that the Style ID is “DHTML” to ensure that it is a LEAP structure
- The name of the XML would correspond to the IDREQUEST eg. RROAT01.xml
- Open RROAT01.xml in Development Workbench and modify the layout as well as javascript as needed

5.1.2 MODIFY USING XSL

The developer would need to do the following

- Identify from the MSTCHANNEL ATS which screen is being modified
- Ensure that the Style ID is not “DHTML” to ensure that it is a a plain XML/XSL transformation
- The name of the XSL would be available in IDRESOURCE column
- Modify the XSL appropriately and ensure while deploying you override that XSL by maintaining the same in the ENTITY FOLDER. Eg if the original XSL was in the gui/ECU/01/eng, one would put the same in gui/ECU/01/eng

A sample UI XSL showing the following

```
 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="default" version="1.0"
xmlns:java="http://xml.apache.org/xslt/java" exclude-result-prefixes="java">
<xsl:import href="common.xsl" />

<xsl:include  href="dateformatter.xsl"/><xsl:output  doctype-public="-//W3C//DTD  HTML  4.01  Transitional//EN"
method="html" encoding="UTF-8" indent="yes" />
<xsl:template match="/">
<xsl:apply-imports />
<html>

<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta HTTP-EQUIV="no-cache" />
<link HREF="css/default.css" TYPE="text/css" REL="STYLESHEET"/>
<script type="text/JavaScript" src="jsdir/Calendar.js" language="JavaScript"></script>
<script language="JavaScript" type="text/JavaScript" src='jsdir/common.js'></script>
<script language="JavaScript" type="text/JavaScript" >
//-------------------------------------------------------------------------
var fldCustData = '<xsl:value-of select="/faml/request/fldcustlist"/>'.split('~');
//-------------------------------------------------------------------------
function bulkFileViewLink (p_filerefno) {

       document.frmmain.fldbtid.value = '<xsl:value-of select="/faml/request/fldfileuploadtype"/>' ;
       document.frmmain.fldfilerefno.value = p_filerefno;
       document.frmmain.fldallcustomerflag.value = "Y";
       document.frmmain.target                 =      "";
       SendTxnRequest('02','BVF');
       return false;
}
//-------------------------------------------------------------------------
</script>
</head>

<body class="workarea" onload="displayStatusMessage ();">

<xsl:variable name="idlang">
       <xsl:value-of select="//faml/response/sessioninfo/@idlang"/>
</xsl:variable>

<form action="internet" method="post" name="frmmain">

<table id="mainbox" class="mainbox" cellpadding="0" cellspacing="0" border="0"><tr><td>
```

```
	<div class="toppanel"><ul><li></li></ul></div>
	<div class="middlepanel"> <!--mainbox middlepanel start-->
	<table cellspacing="0" cellpadding="0" border="0" id="maintable" class="maintable">
		<tr>
			<td valign="top">
				<div id="pageheadingpanel">
					<div id="pageheading" nowrap="true">%%K_FILE_UPLOAD%%</div>
					<div id="pageheadingdate"><xsl:call-template
name="formatted_date"/></div>
				</div>
			</td>
		</tr>
	<tr>
	<td height="100%" valign="top">
		<div class="y_scroll"  id="contentarea">  <!--y_scroll start-->
			<div class="contentarea"><!--contentarea  start-->
			<span id="box" class="box"> <!--rounded curve/border  start-->
					<div class="toppanel"><ul><li></li></ul></div>
					<div  class="middlepanel">  <!--contentarea    box  middlepanel
start-->
			<table border="0" cellspacing="1" cellpadding="1" class="infotable">
				<tr>
					<td class="labeltext col1">%%K_BUFILEREFNO%%:</td>
					<td   class="col80"><a  href="javascript:void(0)"  title="{btidsearch}"
onclick="return              bulkFileViewLink('{/faml/response/bulkfileuploadpreparesponsedto/filerefno}');"
class="NavBold"><xsl:value-of select="/faml/response/bulkfileuploadpreparesponsedto/filerefno"/></a></td>
				</tr>
			</table>
		</div>  <!--contentarea  box middlepanel end-->
					<div class="bottompanel"><ul><li></li></ul></div>
					</span>  <!--rounded curve/border  end-->
			<span id="box" class="box"> <!--rounded curve/border  start-->
					<div class="toppanel"><ul><li></li></ul></div>
					<div  class="middlepanel">  <!--contentarea    box  middlepanel
start-->
			<table border="0" cellspacing="1" cellpadding="1" class="infotable">
				<tr>
					<td class="labeltext col1">%%K_FILE_UPLOAD_TYPE%%:</td>
					<td                                    class="col80"><xsl:value-of
select="/faml/request/fldfileuploadtypetext"/></td>
				</tr>
				<tr>
```

```
                                        <td class="labeltext">%%K_ENCODINGTYPEFORFILEUPLOAD%%:</td>
                                        <td><xsl:value-of select="/faml/request/fldencodingtypetext"/></td>
                                  </tr>
                                  <tr>
                                        <td class="labeltext">%%K_UPLOADFILE%%:</td>
                                        <td><xsl:value-of select="/faml/request/flduploadfilename"/></td>
                                  </tr>
            </table>
            </div>  <!--contentarea  box middlepanel end-->
                                              <div class="bottompanel"><ul><li></li></ul></div>
                                              </span>  <!--rounded curve/border  end-->
      <div class="buttonarea">
            <ul>
                  <li><input    alt="    %%K_OK%%    "    name="fldok"    onClick="SendRequest('01','BFU');"
value="%%K_OK%%" class="buttons" size="16" type="button" maxlength="16" /></li>
            </ul>
      </div>
                  </div>
                  </div>
            </td>
      </tr>
</table>
      </div><!--mainbox middlepanel end-->
      <div class="bottompanel"><ul><li></li></ul></div>
      </td>
      </tr>
</table>

<input type="hidden" name="fldRequestId"    value=""/>
<input type="hidden" name="fldSessionId"    value="{//faml/request/fldSessionId}"/>
<input type="hidden" name="fldServiceType"     value="{//faml/request/fldServiceType}"/>
<input type="hidden" name="fldDataId"       value="{//faml/request/fldDataId}"/>
<input type="hidden" name="fldSectionId"    value="{//faml/request/fldSectionId}"/>
<input type="hidden" name="fldfilerefno"    value=""/>
<input type="hidden" name="fldallcustomerflag"    value=""/>
<input type="hidden" name="fldbtid"    value=""/>
</form>

</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

5.2 EXTENDING EXISTING JAVA ENDPOINT SERVICE

For a existing Business 'Service', within the Oracle FLEXCUBE Direct Banking Services Architecture, one can extend this functionality by either of the following ways

5.2.1 CREATE A NEW BUSINESS SERVICE CALLING EXISITNG BUSINESS

The Business Definitions are represented as Java Interfaces. The business definitions method signatures that provide the required business definitions for data exchange. The steps would be as follows

1) Identify the MSTCHANNELATS for which functionality needs to be changed
2) Create a New Service XSL by keeping the existing SERVICE XSL as a base
3) Modify the Existing MSTCHANNELATS entry ID_SERVICE entry to point to this new SERVICE XSL created above.
4) Create a NEW DTO and ensure that the existing Business Service Request / Response DTO are variable inside your new DTO.
5) A new validation entry would also need to be made.

A Sample Invocation of a Existing Business Service is as follows (assuming that the existing business service was called using "getInformation".

```
public NewInformationResponseDTO getInformation(
        TransactionContext              p_context
,       NewInformationRequestDTO  p_request
) {

//------------ account number list initialization -----------------

          l_newInfo = new NewInformationRequestDTO ();
          l_oldresp    = (InformationResponseDTO)ServiceManager.invokeService(
                                  "InformationServiceInterface.InformationService.getInformation"

                                ,     p_request
                                ,     null
                                ,     p_context
                                ,     true
                                );
          if(!ServiceUtils.isSuccess(l_actresp)) {
            l_resp.result = l_actresp.result;
            return l_resp;
          }
          l_newInfo.businessInfoDTO = l_oldresp;
}
```

Please note that 2 new DTO NewInformationResponsDTO and New
The above example shows how to invoke the out of the box Account Service.

5.2.2 DATA TRANSFER OBJECTS

All Data Transfer Objects or DTOs are the data carriers within and external to the system. All DTO objects have UDF DTO which can be used by the developers to add to functionalities. The Request DTO as well as Response DTO have UDF DTO capability are available to extend

One could add to the UDF DTO with a new UDF field name and field value by adding the same through the SERVICE XSL without modifying the existing business service.
A sample Request XSL showing the following

```
<ServiceRequest>
<serviceName>GenericPaymentServiceInterface.GenericPaymentService.initiateGenericPaymentDetails</serviceName>

<serviceAction><xsl:value-of select="/faml/request/fldnextaction"/></serviceAction>
<referenceNo><xsl:value-of select="/faml/request/fldreferenceno"/></referenceNo>
<stateBit><xsl:value-of select="/faml/request/fldstatebit"/></stateBit>

<request>
	<GenericPaymentRequestDTO>
		<srcAccount>
			<AccountNoInputDTO>
				<nbrAccount><xsl:value-of select="//faml/request/fldsrcacctno"/></nbrAccount>
				<refIdEntity><xsl:value-of select="//faml/request/fldsrccodcountry"/></refIdEntity>
				<idCustomer><xsl:value-of select="//faml/request/fldsrccustid"/></idCustomer>
				<ccyDesc><xsl:value-of select="//faml/request/fldsrcaccntcurr"/></ccyDesc>
				<codBranch><xsl:value-of select="//faml/request/fldsrcbankcode"/></codBranch>
				<namBranch><xsl:value-of select="//faml/request/fldsrcnambranch"/></namBranch>
				<codSwift><xsl:value-of select="//faml/request/fldsrcswiftid"/></codSwift>
			</AccountNoInputDTO>
		</srcAccount>
		<destAccount>
			<AccountNoInputDTO>
				<nbrAccount><xsl:value-of select="//faml/request/flddestacctno"/></nbrAccount>
				<refIdEntity><xsl:value-of select="//faml/request/fldsrccodcountry"/></refIdEntity>
				<idCustomer><xsl:value-of select="//faml/request/flddestcustid"/></idCustomer>
				<ccyDesc><xsl:value-of select="//faml/request/flddestaccntcurr"/></ccyDesc>
				<codBranch><xsl:value-of select="//faml/request/flddestbankcode"/></codBranch>
				<namBranch><xsl:value-of select="//faml/request/flddestnambranch"/></namBranch>
				<codSwift><xsl:value-of select="//faml/request/flddestswiftid"/></codSwift>
			</AccountNoInputDTO>
		</destAccount>
		<txnAmount><xsl:value-of select="//faml/request/fldtxnamount"/></txnAmount>
		<txnCurrency><xsl:value-of select="//faml/request/fldtxncurrency"/></txnCurrency>
		<idUserRefNo><xsl:value-of select="/faml/request/flduserrefno"/></idUserRefNo>
		<isOnlineRateReq><xsl:value-of select="/faml/request/fldisonlineratereq"/></isOnlineRateReq>
		<isNextStatusAuth><xsl:value-of select="/faml/request/fldnextstatusauth"/></isNextStatusAuth>
		<xsl:if test="/faml/request/fldudf">
		<udfFields>
			<xsl:for-each select="/faml/request/fldudf">
				<UDFDTO>
```

```
                                    <xsl:variable    name="udfname"><xsl:value-of    select="concat    ('/faml/request/',
.)"/></xsl:variable>
                                    <udfName><xsl:value-of select="."/></udfName>
                                    <udfValue><xsl:value-of select="dyn:evaluate($udfname)"/></udfValue>
                                    </UDFDTO>
                        </xsl:for-each>
                </udfFields>
                </xsl:if>


        </GenericPaymentRequestDTO>
</request>
</ServiceRequest>
```

5.2.3 USING PRE SERVICE HELPERS

Oracle FLEXCUBE Direct Banking provides capability to call a pre-service helper which is configured at the SERVICE level.

This helper should implement interface `com.iflex.fcat.services. ServiceHelper` and provided concrete implementation for below mentioned callback methods viz.

- processRequest – called before the service is called
- processResponse – called after the service is called

5.2.3.1 SAMPLE DATABASE ENTRIES

These are the sample database entries which would need to be applied on the `ADMIN` schema

5.2.3.1.1 TABLE: MSTSERVICES

```
update mstservices set custom_helper= '<<Custom_helper class full qualified name>>' where namservice =
'<<Interface.service.method>>'
```

5.2.4 USING IN SERVICE LIFE CYCLE HANDLERS

Oracle FLEXCUBE Direct Banking provides capability to extended processing behavior of certain milestones in user's lifecycle.

The handler should implement the interface `com.iflex.fcat.services.apps.UserLifeCycleExtendedHandler` and provided concrete implementation for below mentioned callback methods.

The handler can be optionally configured in table MSTENTITYUSERTYPES.USERLIFECYCLEHANDLER for each user type.

A default handler `com.iflex.fcat.services.apps.handlers.DefaultUserLifeCycleAlertHandler` provides out-of-box implementation of the above interface. This handler should be explicitly configured in the table if the extension is needed.

Extension available	Implementation in DefaultUserLifeCycleAlertHandler
Create User	Generate alert
Create Channel User	Generate alert containing channel user id
Modify User	Generate alert
Modify Channel User	Generate alert containing modified channel user
Create User – Login Password generation	Generate alert containing generated password
Create User – Transaction Pin generation	Generate alert containing generated pin
Change Login password	Generate alert containing changed password
Change Transaction pin	Generate alert containing changed pin
Reset Login password	Generate alert containing generated password
Reset Transaction pin	Generate alert containing generated pin
Change activation code	Generate alert containing activation code

5.2.5 USING IN EXTENDED HANDLERS

Oracle FLEXCUBE Direct Banking provides capability to extended processing behavior of responses given by the Business Service.

The handler should implement the interface `com.iflex.fcat.services.ExtResponseHandler` and provided concrete implementation for below mentioned callback methods.

- processExtendedResponseDetails : which provide capability to change the Response DTO based on business needs.

A default handler `com.iflex.fcat.services.apps.handlers.ExtResponsHandler` provides out-of-box implementation of the above interface. This handler should be explicitly configured in the table if the extension is needed.

Extensions available	Implementation in ExtResponsHandler
LoginExtnResponseHandler	This is to be added when accounts need to be populated during Login.
NullExtResponseHandler	This is to be added against a service when no responses is needed
CompleteExtResponseHandler	This is to be added against a service when needs extended responses. The extended responses include ➢ Entity Time Zone ➢ UI Download Definition ➢ Screen Hints ➢ Local Currency ➢ Business Calendar Dates ➢ Dependant Transaction

5.3 LINK THE BUSINESS SERVICES TO HOST COMPONENTS

The developer and implementor can modify the following components inside the Host Interfacing Layer using the following ways

- Modify the Host Adaptor Class
- Modify the Host Interfacing XSL which transforms the DTO into a XML the core banking understands
- Modify the Communication protocols as mentioned in the Host Interfacing Section above.

6 MODIFY WEB ARCHIVE

Oracle FLEXCUBE Direct Banking product comes with a web archive (.war file) per entity. The web archive contains images, Java script files, cascaded style sheets, static and dynamic web pages etc. Various configuration files are also available under the WEB-INF folder of web archive.

The web archive is optimized so as to improve content delivery. These optimizations include

1. Minification of Java script files
2. Minification of cascaded style sheets
3. Data URI conversion of cascaded style sheets
4. Compression of static and dynamic data using GZIP

Please refer to following documents for doing any modifications in the web archive contents

Document Name	Description
Oracle_FLEXCUBE_Direct_Banking_UI_Content_Delivery_Optimization_Guide	This document provides details on the process of modifying contents of web archive. It also provides details on the UI content optimization utility
Oracle_FLEXCUBE_Direct_Banking_User_Interface_Guide	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 them.

7 PLUGINS / EXTENSIONS AVAILABLE

Oracle FLEXCUBE Direct Banking provides various extensions which can be tweaked to customize products based on banks specific requirement.

Layers	Available Extensions
Presentation Layer	➢ Configurable Widgets as part of DAY Zero. ➢ Configurable Menu Structures ➢ Configurable Layout Structures. ➢ Out of box Theming capability. ➢ Open Development Workbench Available to create and modify GUI screens
Channel Layer	➢ Custom Audit Handler ➢ Separation of GUI and Channel Layer ➢ Configurable communication protocol to call Business Layer ➢ Inbuilt security feature to store Request and Response. ➢ Wizard capability.
Business Layer	➢ WebServices available for all function in FCDB. ➢ New Services Easily Configured. ➢ Extending Existing WebServices. ➢ User Defined Fields available for DTO (Request and Response) ➢ Facility to call o Pre Service – Ability to call helper for every service o In-Service – Default LifeCycle Handler can be called for in service calls. o Post Service – Custom Extended Handler after every service. ➢ Configurable Validation Engine. ➢ Configurable Modules can be linked Transaction Specific logic.
Host Layer	➢ Ability to communicate with disparate Communication protocols.

8 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,Indigo)**
Browsers	- Please refer the version specific Release Note
Database	- Oracle 11g
Database Tools	- Oracle 11g Client
JDBC Driver	- Oracle JDBC Thin Driver for 10g - 11g
Operating System	Windows XP, Windows 7 Professional Workstations, Oracle Enterprise Linux

9 APPENDIX: ENTITY MAINTENANCE

Entity can be use to represent a business units as banks, segments, partners. Oracle FLEXCUBE Direct Banking supports multiple entity scenario setups as following:

Oracle FLEXCUBE Direct Banking provides an out of the box utility to clone a entity.

9.1 ADDING MULTIPLE ENTITIES IN SAME SINGLE SETUP

1) Register the new entity in table MSTENTITY.
2) Map allowed user type for this entity. If required a new user type can also be defined. The user type for the new entity are registered in table MSTENTITYUSERTYPES.
3) Map allowed channels for the user types on the new entity in table MSTENTITYUSERCHANNEL.
4) Map allowed languages to the user type on the new entity in table MSTENTITYUSERTYPELANG.
5) Define transactional access, map transactions for the new entity, user type(s) & channel in table MSTUSERTYPETXN.
6) Add allowed preference option for the new entity and user types in table MSTENTITYUSERTYPEPREF.
7) Add service version mapping for the new entity, user type in table MSTSERVICESMAP
8) Add optionally user type information in tables MSTENTITYACCOUNTACCESS, MSTACCTTYPES, MSTENTITYCUSTTYPES, MSTENTITYUSERTYPEPREF.
9) In application server, deploy web-application to serve the requests. The web.xml of the application should be modified to change the parameter "FCAT.INTERNETSERVLET.DAEMON.NAME" to entity identifier.
10) A new presentation tier property file should be created named as < FCAT.INTERNETSERVLET.DAEMON.NAME>.xml

Application server restart will be required post above steps for new user type change to take effect.

10 APPENDIX: ADDING A NEW USER TYPE

Entity can be use to represent a business units as banks, segments, partners. Oracle FLEXCUBE Direct Banking supports multiple entity scenario setups as following:

Oracle FLEXCUBE Direct Banking provides an out of the box utility to clone a user type.

10.1 ADDING MULTIPLE USERTYPES IN SAME ENTITY

To add new user type to the application please follow below mentioned steps

1) Add the new user type information to table MSTUSERTYPES.
2) Map the new user type for relevant entity in table MSTENTITYUSERTYPES.
3) Map allowed channels to the new user type in table MSTENTITYUSERCHANNEL
4) Map allowed languages to the user type in table MSTENTITYUSERTYPELANG
5) Define transactional access, map transactions for the entity, new user type & channel in table MSTUSERTYPETXN.
6) Add allowed preference option for the new user type in table MSTENTITYUSERTYPEPREF
7) Update access of base user types on the new user type being added in table MSTENTITYUSERTYPES
8) Add service version mapping for the user type in table MSTSERVICESMAP
9) Add optionally user type information in tables MSTENTITYACCOUNTACCESS, MSTACCTTYPES, MSTENTITYCUSTTYPES, MSTENTITYUSERTYPEPREF.
10) Add branding information for the new user type.

Application server restart will be required post above steps for new user type change to take effect.

11 APPENDIX: ADDING A NEW LANGUAGE

To add new language to the application please follow below mentioned steps

1) Details of the new language should be added to table MSTLANG.
2) Details of the new language should be added to table MSTENTITYUSERTYPELANG.
3) Application messages specific to the new language should be added to table APPLICATIONMESSAGE.
4) UI Labels and application data for the new language should be added to table APPLDATA.
5) Add CSS in war for the new language. The CSS name includes language code.

Application server restart will be required post above steps for new language change to take effect.

12 APPENDIX: LIST OF DEVICES / CHANNELS

<table>
<tr><th>Device ID</th><th>Device Description</th></tr>
<tr><td>01</td><td>Internet

This channel is used for all Business User Types like Retail User, Corporate User, Corporate Administration etc. Also, as an exception, this is also used as a channel for the Helpdesk Users operating on behalf of the Business Users.

This channel also supports specific user agents viz

<table>
<tr><th>User Agent</th><th>Device</th></tr>
<tr><td>IPAD / Android Tablet</td><td></td></tr>
</table>
</td></tr>
<tr><td>11</td><td>Intranet

This channel is used for all Administration User type typically Internal User within the bank.</td></tr>
<tr><td>41</td><td>SMS Banking

This channel identifies all transactions originating from the SMS Banking channel. This is currently applicable only to Business User Types like Retail and Corporate Users.</td></tr>
<tr><td>42</td><td>Browser based Mobile Banking

This channel identifies all transactions originating from the Brower based Banking channel. This is currently applicable only to Business User Types like Retail and Corporate Users.This channel also supports specific user agents viz
<table>
<tr><th>User Agent</th><th>Device</th></tr>
<tr><td>Iphone Browser</td><td></td></tr>
<tr><td>Non Javascript Browser</td><td></td></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td>43</td>
<td>Application based Mobile Banking

This channel identifies all transactions originating from the Brower based Banking channel. This is currently applicable only to Business User Types like Retail and Corporate Users.This channel also supports specific user agents viz

<table>
<tr><th>User Agent</th><th>Device</th></tr>
<tr><td>Iphone / Android Phone</td><td></td></tr>
<tr><td>IPAD / Android Tablet</td><td></td></tr>
<tr><td>J2Me / Blackberry</td><td></td></tr>
<tr><td></td><td></td></tr>
</table>
</td>
</tr>
</table>

13 APPENDIX: LIST OF CONTENT GENERATORS

Sr. No.	Style ID	Class Name	Description
1	ACQ	com.iflex.fcat.gui.content.ApachePOIACQExcelContentGenerator	This class provides the content generation capabilities based on XSL Transformation to generate content output in Excel format. Apache POI library is used to generate the Excel output. This content generator is specific to the 'Account Overview' transaction as it uses CMN/acq_download_template.xsl for transformation of data.
2	ACSV	com.iflex.fcat.gui.content.ApachePOIExcelContentGenerator	This class provides the content generation capabilties based on XSL Transformation to generate content output in Excel format. Apache POI library is used to generate the Excel output.
3	AHTML	com.iflex.fcat.gui.content.DownloadContentGenerator	This class provides the content generation capabilities based on XSL Transformation. The generated content is in HTML format and sent as an attachment in the HTTP response and hence can be downloaded by the browser. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file

			extension for the attachment
4	ATTCH	com.iflex.fcat.gui.content.FileAttachmentContentGenerator	This content generator is used to set file attachment in the response.
5	AUDIT	com.iflex.fcat.gui.content.AuditViewContentGenerator	This content generator is specific for viewing the audited data. It is applied on the audited data XML and uses XSL transformation to generate the exact response that was displayed to user when the transaction was performed by user. For mobile application channel, the output of this XSL transformation is either XML or JSON data. Hence is it again converted to HTML using another XSL transformation.
6	CSV	com.iflex.fcat.gui.content.DownloadDataContentGenerator	This content generator is used to download the data using uidownload framework. It uses a common XSL download_template.xsl for transformation to generate the content. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment

7	DHTML	com.iflex.fcat.gui.content.DynamicHTMLContentGenerator	This content generator is used for all transactions that are developed using the Channel Workbench tool. It generates HTML output using XSL transformation of the channel response. This content generator caters to Internet as well as Mobile Browser channels. The data is HTML entity encoded before transformation. It is expected to apply the common XSL (genericscreentemplate.xsl) irrespective of transaction. It performs following steps: 1. adds the Javascript code available in <IDREQUEST>.js file into this XSL at runtime. The generated XSL template is cached for better performance 2. appends the screen layout metadata available in <IDREQUEST>.xml file into channel response and applies the XSL template created i step 1 on the resultant XML The XSL file is looked in folders as specified in the refPathUI configuration. If this configuration is not available, then the XSL file is first looked at currnet user type level, if not found it is looked at the base user type level.

8	DPDF	com.iflex.fcat.gui.content.DownloadDataContentGenerator	This content generator is used to download the data using UIDownload framework. It uses a common XSL download_template.xsl for transformation to generate the content. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment
9	EMAIL	com.iflex.fcat.gui.content.EmailContentGenerator	This content generator is used to email the generated output to user's email ID.
10	GPDF	com.iflex.fcat.gui.content.GenericPDFContentGenerator	This class provides the content generation capabilities based on XSL Transformation. This is a generic class which uses XSL transformation to generate the HTML of the screen for which the PDF needs to be generated. Once the HTML output is fetched in output stream, this output stream is being used by FOProcessor to generate the PDF file.

11	HTML	com.iflex.fcat.gui.content.HTMLContentGenerator	This content generator is used to generate HTML output using XSL transformation of the channel response. This content generator can set the Content-Type header of HTTP response dynamically, based on the list of possible values passed. It matches the possible values one by one with 'Accepts' header received in HTTP request. This content generator can be used for generating HTML or XHTML data. The data is HTML entity encoded before transformation. The XSL file is looked in folders as specified in the refPathUI configuration. If this configuration is not available, then the XSL file is first looked at current user type level, if not found it is looked at the base user type level.
12	HTMLD	com.iflex.fcat.gui.content.DownloadDataContentGenerator	This content generator is used to download the data using UIDownload framework. It uses a common XSL download_template.xsl for transformation to generate the content. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment

13	MDHTML	com.iflex.fcat.gui.content.DynamicHTMLContentGenerator	This content generator is used for all transactions that are developed using the Channel Workbench tool. It generates HTML output using XSL transformation of the channel response. This content generator caters to Internet as well as Mobile Browser channels. The data is HTML entity encoded before transformation. It is expected to apply the common XSL (genericscreentemplate.xsl) irrespective of transaction. It performs following steps: 1. adds the Javascript code available in <IDREQUEST>.js file into this XSL at runtime. The generated XSL template is cached for better performance 2. appends the screen layout metadata available in <IDREQUEST>.xml file into channel response and applies the XSL template created i step 1 on the resultant XML The XSL file is looked in folders as specified in the refPathUI configuration. If this configuration is not available, then the XSL file is first looked at current user type level, if not found it is looked at the base user type level.

14	MJSON	com.iflex.fcat.gui.content.MLEAPXSLTContentGenerator	This content generator is used for all 'Mobile Application' transactions that are developed using the Channel Workbench tool. It generates JSON output using XSL transformation of the channel response. It is expected to apply the common XSL (genericjsontemplate.xsl) irrespective of transaction. It appends the screen layout metadata available in <IDREQUEST>.xml file into channel response and applies the XSL template on the resultant XML. User-Agent specific screen metadata is given priority over generic screen metadata.
15	MPXML	com.iflex.fcat.gui.content.MLEAPXSLTContentGenerator	This content generator is used for all 'Mobile Application' transactions that are developed using the Channel Workbench tool. It generates JSON output using XSL transformation of the channel response. It is expected to apply the common XSL (genericjsontemplate.xsl) irrespective of transaction. It appends the screen layout metadata available in <IDREQUEST>.xml file into channel response and applies the XSL template on the resultant XML. User-Agent specific screen metadata is given priority over generic screen metadata.

16	OFX	com.iflex.fcat.gui.content.DownloadContentGenerator	This class provides the content generation capabilities based on XSL Transformation. The generated content is in OFX format and sent as an attachment in the HTTP response and hence can be downloaded by the browser. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment
17	PDF	com.iflex.fcat.gui.content.DownloadDataContentGenerator	This content generator is used to download the data using UIDownload framework. It uses a common XSL download_template.xsl for transformation to generate the content. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment
18	PDFV	com.iflex.fcat.gui.content.DownloadDataContentGenerator	This content generator is used to download the data using UIDownload framework. It uses a common XSL download_template.xsl for transformation to generate the content. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file

			extension for the attachment
19	PLHTML	com.iflex.fcat.gui.content.PreLoginHTMLContentGenerator	This content generator is specific to the Pre-Login functionality. It extends the DynamicHTMLContentGenerator.
20	PPDF	com.iflex.fcat.gui.content.OraclePDFContentGenerator	This class provides the content generation capabilities based on XSL Transformation. The generated content is in PDF format and sent as an attachment in the HTTP response and hence can be downloaded by the browser. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data.
21	PRNT	com.iflex.fcat.gui.content.OraclePDFContentGenerator	This class provides the content generation capabilities based on XSL Transformation. The generated content is in PDF format and sent as an attachment in the HTTP response and hence can be downloaded by the browser. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data.

22	PXML	com.iflex.fcat.gui.content.XMLXSLTContentGenerator	This class provides the content generation capabilities based on XSL Transformation. This is the default mechanism of generating content for the FCDB Presentation Layer. The XML data returned by the appropriate business processing components undergoes XSLT to generate the required output. The different outputs from this content generator are HTML, WML, CHTML, PDF, TXT and CSV. The implementation provides the capability to define and configure the appropriate content type for the output.
23	QIF	com.iflex.fcat.gui.content.DownloadContentGenerator	This class provides the content generation capabilities based on XSL Transformation. The generated content is in QIF format and sent as an attachment in the HTTP response and hence can be downloaded by the browser. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment

24	RHTML	com.iflex.fcat.gui.content.ReportContentGenerator	This content generator is used to download the report data. The channel response is supposed to have report data in Base64 encoded format. The data is sent in Zipped format. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment
25	RPDF	com.iflex.fcat.gui.content.ReportContentGenerator	This content generator is used to download the report data. The channel response is supposed to have report data in Base64 encoded format. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment
26	RXLS	com.iflex.fcat.gui.content.ReportContentGenerator	This content generator is used to download the report data. The channel response is supposed to have report data in Base64 encoded format. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment

27	RZIP	com.iflex.fcat.gui.content.ReportContentGenerator	This content generator is used to download the report data. The channel response is supposed to have report data in Base64 encoded format. The data is sent in Zipped format. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment
28	TCSV	com.iflex.fcat.gui.content.DownloadContentGenerator	This class provides the content generation capabilities based on XSL Transformation. The generated content is in CSV format and sent as an attachment in the HTTP response and hence can be downloaded by the browser. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment

29	TEXT	com.iflex.fcat.gui.content.XMLXSLTContentGenerator	This class provides the content generation capabilities based on XSL Transformation. This is the default mechanism of generating content for the FCDB Presentation Layer. The XML data returned by the appropriate business processing components undergoes XSLT to generate the required output. The different outputs from this content generator are HTML, WML, CHTML, PDF, TXT and CSV. The implementation provides the capability to define and configure the appropriate content type for the output.
30	TIMG	com.iflex.fcat.gui.content.TuringImageContentGenerator	This content generator is used for Captcha image generation
31	TXT	com.iflex.fcat.gui.content.DownloadContentGenerator	This class provides the content generation capabilities based on XSL Transformation. The generated content is in plain text format and sent as an attachment in the HTTP response and hence can be downloaded by the browser. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment

32	TXTD	com.iflex.fcat.gui.content.XMLDataContentGenerator	This content generator is used to convert the XML data into Text output. It uses XSL transformation for the same. It parses the channel response for xmldatadownloadresponsedto/data and applies XSLT on this data element only. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment
33	VPDF	com.iflex.fcat.gui.content.FODownloadDataContentGenerator	This class provides the content generation capabilities based on XSL Transformation. The generated content is in PDF format and sent as an attachment in the HTTP response and hence can be downloaded by the browser. The XSL file is first looked at user type leven. If not found, it is looked at base user type level. If not found there also, then looked at common folder. The flag ATTACHMENT.FLAG decides if the response data is to be sent as an attachment or as inline data. The parameter ATTACHMENT.EXT provides file extension for the attachment

34	XHTML	com.iflex.fcat.gui.content.HTMLContentGenerator	This content generator is used to generate HTML output using XSL transformation of the channel response. This content generator can set the Content-Type header of HTTP response dynamically, based on the list of possible values passed. It matches the possible values one by one with 'Accepts' header received in HTTP request. This content generator can be used for generating HTML or XHTML data. The data is HTML entity encoded before transformation. The XSL file is looked in folders as specified in the refPathUI configuration. If this configuration is not available, then the XSL file is first looked at currnet user type level, if not found it is looked at the base user type level.
35	XML	com.iflex.fcat.gui.content.XMLContentGenerator	This content generator is to be used when the response is expected in XML format.

14 APPENDIX: REFERENCES

The following material is useful for any documentation

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 them.
Oracle_FLEXCUBE_Direct_Banking_Security_Guide	Oracle FLEXCUBE Direct Banking Security Best Practices Guide

Oracle_FLEXCUBE_Direct_Banking_Program_Specs.zip	This is the API documentation (Program specifications) of the product. Please extract this zip and open index.html

15 APPENDIX: PAYMENTS DESIGN

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

ⓘ Refer the **Oracle FLEXCUBE Direct Banking Generic Payments Design** document for further details on configurations of Generic Payments Design.