Oracle FLEXCUBE Investor Servicing®
Extensibility Getting started

Release 12.0

April 2012

Oracle Part Number E51528-01

Contents

1 Preface 3
1.1 Audience 3
1.2 Related documents 3
1.3 Conventions 4
2 Introduction 4
2.1 How to use this Guide 4
3 Extensibility Introduction 5
3.1 What is extensibility 5
3.2 Industry Pattern 5
3.3 Industry Approach 5
4 FLEXCUBE IS Extensibility 5
4.1 Business Areas 5
4.2 FLEXCUBE IS Extensibility approach 6
4.3 FLEXCUBE IS Extensibility user roles 6
5 FLEXCUBE IS Extensible features 8
5.1 Screen changes 8
5.1.1 New Screens 8
5.1.2 Screen Modifications 8
5.1.3 Amend field level attributes 9
5.1.4 Style Sheet changes 9
5.1.5 Language conversion of screens 9
5.2 Functional 9
5.2.1 User Defined fields at Maintenance 9
5.2.2 New SDE and rule for IC calculation 10
5.2.3 Configurable workflow of Branch Screens 10
5.3 Processing logic 10
5.3.1 Additional validation logic for a field or group of fields 10
5.3.2 Modify defaulting logic for fields 11
5.3.3 Online contract extensibility 12
5.3.4 Batch extensibility 12
5.4 Notifications 12
5.4.1 Event based notifications 12
5.5 Reports 13
5.5.1 New BIP Reports 13
5.5.2 New OBIEE based reports 13
5.6 User defined Advice tags 13
5.6.1 New tag in Advice message 13
5.7 Interface 13
5.7.1 Switch Interface ISO8583 configuration 13
5.7.2 Configurable Generic Interface for upload/handoff 13
5.7.3 Upload adapter framework 14
6 Extensibility development life cycle 14
6.1 Define Extensibility Requirement 14
6.2 Identify the Business area of extensibility 15
6.3 Identify the tools/framework to be used 15
6.4 Identify the file types & layers applicable 16
6.5 Develop changes 16
6.6 Test it in FLEXCUBE environment 16
7 Resources 16

1 Preface

This document describes the concepts and helps reader to get started using Extensible framework of FLEXCUBE IS Application, to develop additional functionalities.

1.1 Audience

The Extensibility getting started book is intended for FLEXCUBE Application Developers/Users who are authorized to perform the following tasks:

- Modify the layouts of existing FLEXCUBE Screens
- Modify the existing functionality by adding new fields/tabs/data blocks
- Extend the existing screen to have fields based on customer specific table/fields
- Add customer specific validations at extension hooks
- Add customer specific processing logics in batch processing
- Add customer specific notifications
- Add customer specific calculation elements
- Add customer specific reports

To Use this manual, you need conceptual and working knowledge of the below:

Proficiency	*Resources*
FLEXCUBE IS Development overview	FCIS-FD01-01-01-Development Overview Guide
RAD function ID development getting started	FCIS-FD02-01-01-RAD Getting Started

1.2 Related documents

For more information on RAD development and extensibility, refer the below documents:

- *FCIS-FD01-01-01-Development Overview Guide*
- *FCIS-FD02-02-01-RAD Function ID Development*
- *FCIS-FD02-03-01-RAD Web Service Development*
- *FCIS-FD02-04-01-RAD BIP Report Integration*
- *FCIS-FD02-05-01-RAD Notification Development*
- *FCIS-FD05-02-01-RAD-Reference*
- *FCIS-FD03-02-01-Extensibility Reference Guide*
- *FCIS-FD03-03-01-Extensibility By Example Volume 1*
- *FCIS-FD03-03-02-Extensibility By Example Volume 2*
- *FCIS-FD04-02-01-Generic Interface Configuration Guide*
- *FCIS-FD04-03-01-Upload Adapter Development Guide*

1.3 Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements (for example, menus and menu items, buttons, tabs, dialog controls), including options that you select.
italic	italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates language and syntax elements, directory and file names, URLs, text that appears on the screen, or text that you enter.

2 Introduction

2.1 How to use this Guide

The information in this guide includes:

- Chapter 2, " Introduction"
- Chapter 3, "Extensibility Introduction"
- Chapter 4, "FLEXCUBE IS Extensibility"
- Chapter 5, "FLEXCUBE Extensible features"
- Chapter 6, "Extensible Development Life Cycle"
- Chapter 7, "Resources"

3 Extensibility Introduction

3.1 What is extensibility

Extensibility is an ability of the software system to allow and accept the significant extension of its capabilities without major rewriting of code or changes in its basic architecture. Extensible systems provide technology, tools, languages that designed so that developers can expand or add to its capabilities.

3.2 Industry Pattern

Following are the industry pattern to address the extensibility in software architecture

- Frameworks
- Configuration files
- Extension using scripts
- User specific extension software packages
- Object based programming where inheritance is used for extensibility

3.3 Industry Approach

Industry approaches to extensibility typically includes following:

- Tools to allow to extend the functionality of base product
- Program hooks to allow developers to insert their program routines
- Ability define new business events to address change in process
- Ability to create regional specific software changes
- Ability to add/remove fields at business messages
- Ability to configure interface protocols without software change

4 FLEXCUBE IS Extensibility

4.1 Business Areas

One of the primary goals of the FLEXCUBE IS architecture is that system should be able to be extendable in required business specific areas. Following are such areas where extensibility is required:

Business Area	*Why extensibility required*
Screen changes	User may want to keep some screens simple to improve training & operational efficiency
Language of Screens	User may wish to provide screens other than the default language of software
Business / legal requirements	Certain processing/calculation logic may be applied specific to region/country judiciary.
Events & eco system participation	The software has to be part of bigger eco

	system by providing other integration/notification mechanism
User configurable messages /reports	Software should provide mechanism to extract required information. System to be open to provide the same
Ad hoc exchange of information between systems	System should provide mechanism to exchange information with ad-hoc systems over the period time.

4.2 FLEXCUBE IS Extensibility approach

FLEXCUBE IS provides the following approach to address the extensible requirement.

Pattern	*Industry Approach*	*FLEXCUBE Approach*
Framework	Tools and framework to extend the base product	RAD framework
Configuration files	XML files / Text files to configure	XML configuration files and Text based configuration files
Extension using Scripts	Scripts	Java Script based extensions to enable extension at user interface layer
User specific extension of software package	Program hooks to allow extension logic call outs	FLEXCUBE call outs based on release type CUSTOM, CLUSTER
User defined events	Ability to define new events/message types	FEXCUBE notification and messaging architecture to define new XML message types
Protocol tweaking	Configuration of protocol without software change	FLEXCUBE ISO8583 protocol definition in XML file that can be modifiable.
User/Regional specific processing logic	Ability to extend the core processing logic	FLEXCUBE UDF, SDE (IC and CL) extensions

4.3 FLEXCUBE IS Extensibility user roles

FLEXCUBE IS Extensibility development can be classified into 4 types based on the complexity and user competencies required:

- Application maintenance/definition of components
  User expected to login into FLEXCUBE application and use certain function IDs to define the new components. This is typically applicable to Bank business user who requires new functionality.

Example, user need to use function ID UDDMAINT to define new UDF field

- Configuration files
  User expected to modify some of the parameters in configuration files. This may require restart of application or relevant applications. Typically this is required for application administrators in bank.

  Example, user may need to modify the ISO8583 protocol definition

- Tools based development
  User expected to understand the given function ID working and required to extend the functionality by adding new data sources and fields. This is typically required by IT developer in bank.

  Example: User needs to change screen layout, to add new data blocks based on new tables added in database.

- Programming
  User expected to achieve granular control and validations using programming extensions. User expected to know the language used thoroughly in this context. This is typically required by advanced developers in bank.

Example, bank required to modify the defaulting and validation or modify the processing flow at specific call out points.

Developer role and extensible approach matrix

Given below matrix depicts the developer role and possible extensible approaches to apply:

Developer role	*Maintenance/Definition*	*Configuration*	*Tools*	*Programming*
Implementer *Implementer could be OFSS staff or customer / partner staff who implements FLEXCUBE*	Yes	Yes	Yes	Yes

IS				
Bank Application User	Yes			
Application users are the bank FLEXCUBE functional users				
Bank IT User		Yes	Yes	Yes
Bank IT user could be system administrators and have technical skill to extend the FLEXCUBE IS				

5 FLEXCUBE IS Extensible features

This section describes the extensible features available in FLEXCUBE IS

5.1 Screen changes

This section describes features that are specific to Function ID (screens) extensibility. RAD tool is used for function ID extensibility.

5.1.1 New Screens

RAD tool used to develop the new screens depending upon the bank requirement. The screens are based on existing or new tables added in database.

Refer following documents for more information on working with screens.

- *FCIS-FD02-01-01-RAD Getting Started*
- *FCIS-FD05-02-01-RAD-Reference*
- *FCIS-FD03-02-01-Extensibility Reference Guide*

5.1.2 Screen Modifications

Existing screens layouts can be modified using RAD tool to suite as follows:

- Hide fields that are not relevant to a given implementation
- Modify the placement of the fields (example moving from one tab to other tab)

- Add LOV to a given field
- Changing the data type
- Adding enumerations to a given field to restrict user inputs
- To increase the set fields (example adding the address line 5)

Refer *FCIS-FD03-03-01-Extensibility By Example Volume 1* for examples

5.1.3 Amend field level attributes

Existing file level attributes can be modified to add below:

- Defaulting some value to reduce user input/errors.
- Restring the maximum and minimum value
- Precision settings

Refer *FCIS-FD03-03-01-Extensibility By Example Volume 1* for examples.

5.1.4 Style Sheet changes

FLEXCUBE IS provides style editor to enable CSS changes to have following user specific UI elements design:

- Page template changes
- Dialog template changes
- Form elements look and feel
- Text fonts
- Tables look and feel
- Colors changes

Refer *FS_StyleDesigner_for_FC_IS_10.3.0.0.0.0.0.zip* for more information

5.1.5 Language conversion of screens

FLEXCUB screens can be extended to support languages other than English.

5.2 Functional

5.2.1 User Defined fields at Maintenance

UDF framework enables the bank user to add the new field without changing any table structure. This is used in maintenance function IDs where new field required by bank user.

Refer *FCIS-FD03-03-01-Extensibility By Example Volume 1* for examples

5.2.2 New SDE and rule for IC calculation

IC module SDE framework enables user to add the user specific system data element for which user can write the data fetch logic. This SDE can be used further in building the interest calculation logic.

Refer *FCIS-FD03-03-01-Extensibility By Example Volume 1* for examples

5.2.3 Configurable workflow of Branch Screens

Configurable stages available for FLEXCUBE IS Branch function IDs. Branch function ID can be identified the module type WB in menu static data. User can define the function ID and applicable stages.

Refer *Savings.zip* user manual section 3 for workflow definition features.

5.3 Processing logic

5.3.1 Additional validation logic for a field or group of fields

FLEXCUBE IS provides the extension call outs in database layer. These extension call outs are extensible package and pre-named procedures to be used for extensibility. The base product will call this call outs during runtime with required PLSQL data type as parameters.

Example:

User wanted extends STDCIF function to add capital letter validation for the field "card holder name". This can be achieved as follows:

Edit the **STPKS_STDCIF_CUSTOME.Fn_Pre_Default_Validate** as below

```
FUNCTION Fn_Pre_Default_And_Validate
   (p_Source           IN VARCHAR2,
   p_Source_Operation IN VARCHAR2,
   p_Function_Id      IN VARCHAR2,
   p_Action_Code      IN VARCHAR2,
   p_Child_Function   IN VARCHAR2,
   p_stdcif           IN stpks_stdcif_Main.ty_stdcif,
   p_Prev_stdcif      IN OUT stpks_stdcif_Main.ty_stdcif,
   p_Wrk_stdcif       IN OUT stpks_stdcif_Main.ty_stdcif,
   p_Err_Code         IN OUT VARCHAR2,
   p_Err_Params       IN OUT VARCHAR2)
RETURN BOOLEAN IS
BEGIN
```

```
   Dbg('In Fn_Pre_Default_And_Validate..');

--extensibility code start
 p_Wrk_stdcif:= p_stdcif;

IF  p_wrk_stdcif.v_sttms_customer.CARD_HOLDER_NAME NOT IN
(upper(p_wrk_stdcif.v_sttms_customer.CARD_HOLDER_NAME))
THEN
   p_err_code    := 'ST-OTHR-097';
   p_err_params  := NULL;
   Dbg('Out of validation code-Sarva');
   RETURN FALSE;
END IF;
--extensibility code ends

   Dbg('Returning Success From fn_pre_default_and_validate..');


RETURN TRUE;
EXCEPTION
WHEN OTHERS THEN
   Debug.Pr_Debug('**',
   'In When Others of stpks_stdcif_Custom.Fn_Pre_Default_And_Validate
..');
   Debug.Pr_Debug('**', SQLERRM);
   p_Err_Code   := 'ST-OTHR-001';
   p_Err_Params := NULL;
   RETURN FALSE;
END Fn_Pre_Default_And_Validate;
```

Note:
Open RAD XML for a given function ID using RAD tool to understand the data block and field name. This would give above complete path to access the field name. you can prefix "**p**_" to get function ID data type and "**v**_" to data block to get data block name.

Example: to know the card holder name element at runtime, use following template:

[function_id type].[data block name].[field name]
p_wrk_stdcif.v_sttms_customer.CARD_HOLDER_NAME

Refer *FCIS-FD03-03-01-Extensibility By Example Volume 1* for detailed steps involved in this exercise.

5.3.2 Modify defaulting logic for fields

FLEXCUBE IS call outs allows to change defaulting logic for elements using PLSQL data types.

Note:

Refer example given in *section 5.3.1* to know how to identify the element name

5.3.3 Online contract extensibility

FLEXCUBE IS allows to modify or enrich the oline processing logics at given call out functions.

Note: How to identify package name ?

Refer the RAD generated packages for CUSTOM and CLUSTER types to know the possible call outs available which has PLSQL data type parameters. To arrive at the package name using following template.

Template:
<*Module code*>*PKS*_<*Function ID*>_<*Release type*>

Example:
To get the CUSTOM release of function ID FTDTRONL which belongs to FT module, package would be

FTPKS_FTDTRONL_CUSTOM

5.3.4 Batch extensibility

Apart from RAD generated function ID based data base packages, FLEXCUBE IS allows to modify below core service packages. Note that these are core packages which don't have any function ID associated.

ACPKS package is extensible using following packages:

- ACPKS_CUSTOM
- ACPKS_CLUSTER

WRP_BATCH batch running package is extensible using following packages:

- WRP_BATCH_CLUSTER
- WRP_BATCH_CUSTOM

Refer the *FCIS-FD03-03-02-Extensibility By Example Volume 2* for example.

5.4 Notifications

5.4.1 Event based notifications

FLEXCUBE IS supports events based notification framework, where notification triggers can be developed as per user requirement. Once the event occurs, the framework pushes the required data to external systems.

Refer *FCIS-FD02-05-01-RAD Notification* Development

5.5 Reports

FLEXCUBE provides factory shipped BIP canned reports and OBIEE repositories. User can extend the reports or repositories to suite the local requirements.

Refer *FCIS-FD07-01-01-Report Getting started* for more information

5.5.1 New BIP Reports

User can develop the new report or modify the existing report to change report query , result columns or filter criteria.

Refer *FCIS-FD07-02-01-BIP Report Development Guide* for more information on BIP report development

5.5.2 New OBIEE based reports

User can develop the new OBIEE repositories or work with existing OBIEE repositories.

Refer *FCIS-FD07-03-01-OBIEE repository Development Guide* for more information on OBIEE repositories development.

5.6 User defined Advice tags

5.6.1 New tag in Advice message

FLEXCUBE provides rich set of advices with pre-defined tags for each message type. During run time, the contents are fetched and filled in advices. User can define new TAG and add code to fetch the logic.

5.7 Interface

5.7.1 Switch Interface ISO8583 configuration

- FLEXCUBE user can configure the version and protocol fields of ISO8583 based SWITCH interface gateway.
- User can define the mapping of ISO processing code and FLEXCUBE internal transaction code.

5.7.2 Configurable Generic Interface for upload/handoff

User can define following interfaces

- Incoming – to get data into FLEXCUBE

- Outgoing – to get data out of FLEXCUBE

Refer *FCIS-FD04-02-01-Generic Interface Configuration Guide* on how to define generic interface

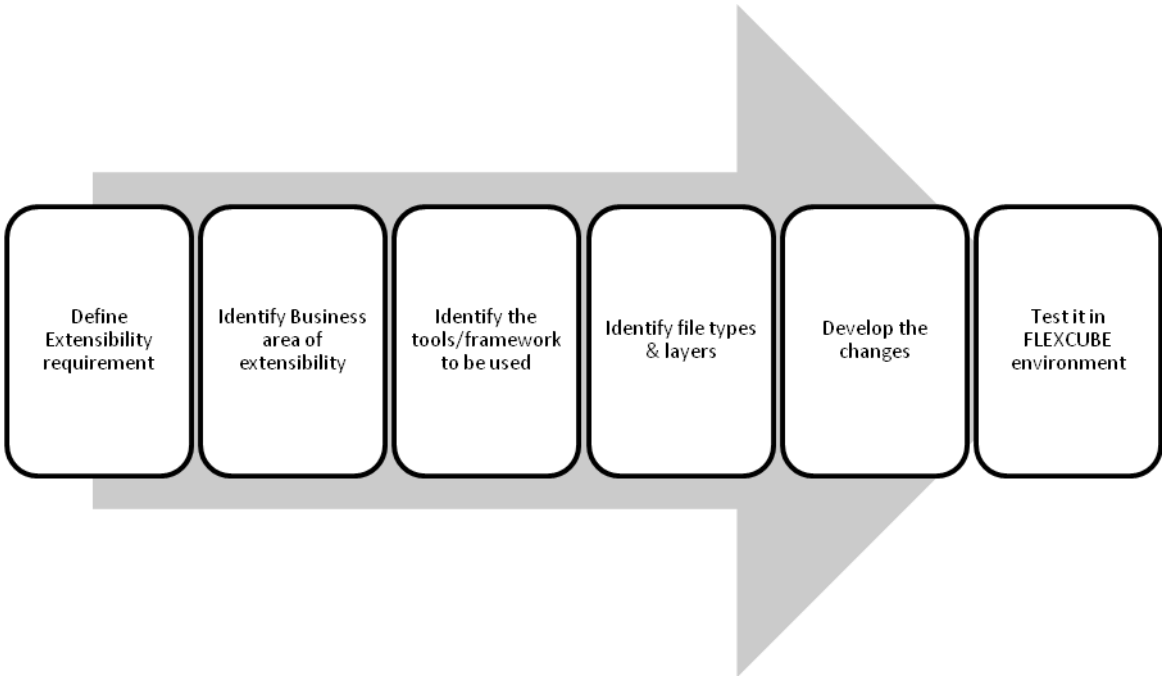
5.7.3 Upload adapter framework

FLEXCUBE IS provides factory shipped adapters (spread sheet based upload) for incoming Interface upload purpose. User can extend by developing new adapter using upload adapter framework.

Refer FCIS-FD04-03-01-Upload Adapter Development Guide

6 Extensibility development life cycle

Extensibility development involves following stages. These stages are explained in detail further down the line.



6.1 Define Extensibility Requirement

Extensibility Requirements need to be clearly defined and documented. This requirement should describe the module, function ID (if applicable) and intended functionality required. This requirement should have justification of why extensibility needed compared with base functionality. It also should cover other alternatives to achieve the functionality without extensibility.

6.2 Identify the Business area of extensibility

Depending upon the Requirement, user needs to identify the FLEXCUBE business area that requires extensibility development. This includes:

- Function ID (New, modify existing, add fields, hide fields)
- Processing logic (defaulting , enriching, validating)
- UDF (New UDF fields for identified function IDs)
- SDE (new SDE for calculation purpose)
- Accounting
- Batch (New batch function during EOD time or intraday)
- Notification (New event notification)
- Report (new report or modify existing report query)
- Interface (New incoming or outgoing)
- Adapter (Migration / data upload into FLEXCUBE)

6.3 Identify the tools/framework to be used

Area	*FLEXCUBE Tools/Framework*
Function ID	▪ RAD ▪ Style sheet editor
Processing logic	▪ PLSQL programming on RAD generated packages ▪ PLSQL programming on core packages
UDF	▪ UDDMAINT function ID ▪ PLSQL programming on -Hand coding of UDF logics
SDE	▪ SDE maintenance ▪ PLSQL programming on -Hand coding of SDE data fetch logic
Accounting	▪ PLSQL programming on - Accounting extensible packages
Batch	▪ PLSQL programming on -Batch extensible
Notification	▪ RAD – Notification trigger

	development
Reports	▪ BIP report development ▪ OBIEE based reports development
Interface	▪ Generic Interface framework

6.4 Identify the file types & layers applicable

The below table described the layer and file types developed for each extensibility business areas that involves software modification.

Area	*Client Layer*	*Application Layer*	*Database Layer*
Function ID	Java script files	UIXML	RAD generated CUSTOM/CLUSTER packages
Processing logic			RAD generated CUSTOM/CLUSTER packages Core FLEXCUBE Packages
UDF			UDF specific procedures and packages
SDE			IC and CL packages
Accounting			Accounting packages
Batch			Batch processing package
Notification			RAD generated Notification Triggers
Reports		RTF file XDO file	RAD generated Report packages
Interface	NA	NA	NA

6.5 Develop changes

User can develop the required changes using respective tools documents.

Refer section 5 for development documentation help on each area of extensibility.

6.6 Test it in FLEXCUBE environment

User need to copy the developed files to target environment and can test the developed functionality. Refer the FLEXCUBE IS installation manuals on how to deploy the changes.

7 Resources

Refer the below resources to gain further working knowledge with extensibility

To Do	*Resources*
RAD Getting started	*FCIS-FD02-01-01-RAD Getting Started*
RAD complete reference guide	*FCIS-FD05-02-01-RAD-Reference*
RAD screen development step by step procedure	*FCIS-FD02-02-01-RAD Function ID Development*
RAD web service development	*FCIS-FD02-03-01-RAD Web Service Development*
BIP report integration with RAD screen	*FCIS-FD02-04-01-RAD BIP Report Integration*
Outbound Notification trigger development	*FCIS-FD02-05-01-RAD Notification Development*
Extensibility Reference guide	*FCIS-FD03-02-01-Extensibility Reference Guide*
Extensibility use case development examples	*FCIS-FD03-03-01-Extensibility By Example*
Branch work flow definition	*Savings.zip (WB user manual)*
Style sheet editor	*FS_StyleDesigner_for_FC_IS_10.3.0.0.0.0.0.zip*
Generic Interface configuration Guide	*FCIS-FD04-02-01-Generic Interface Configuration Guide*
FLEXCUBE Upload adapter development	*FCIS-FD04-03-01-Upload Adapter Development Guide*

FCIS-FD03-01-01-Extensibility Getting started
April 2012
12.0

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com/ financial_services/