Extensibility Getting started
Oracle Banking Payments
Release 14.0.0.0.0
[Feb] [2018]

Contents

1 Preface

This document describes the concepts and helps reader to get started using Extensible framework of Oracle Banking Payments application, to develop additional functionalities.

1.1 Audience

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

- Modify the layouts of existing Oracle Banking Payments 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

1.2 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, "Oracle Banking Payments Extensibility"
- Chapter 5, "Oracle Banking Payments Extensible features"
- Chapter 6, "Extensible Development Life Cycle"

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 Oracle Banking Payments Extensibility

4.1 Business Areas

One of the primary goals of the Oracle Banking Payments 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 Oracle Banking Payments Extensibility approach

Oracle Banking Payments provides the following approach to address the extensible requirement.

Pattern	*Industry Approach*	*Oracle Banking Payments 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	Oracle Banking Payments call outs based on release type CUSTOM, CLUSTER Banking
User defined events	Ability to define new events/message types	notification and messaging architecture to define new XML message

		types
Protocol tweaking	Configuration of protocol without software change	Oracle Banking Payments ISO8583 protocol definition in XML file that can be modifiable.
User/Regional specific processing logic	Ability to extend the core processing logic	Oracle Banking Payments UDF extensions

4.3 Oracle Banking Payments Extensibility user roles

Oracle Banking Payments 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 Oracle Banking Payments 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 Oracle Banking Payments*	Yes	Yes	Yes	Yes
Bank Application User *Application users are the bank Oracle Banking Payments functional users*	Yes			
Bank IT User *Bank IT user could be system administrators and have technical skill to extend the Oracle Banking Payments*		Yes	Yes	Yes

5 Oracle Banking Payments Extensible features

This section describes the extensible features available in Oracle Banking Payments.

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.

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)

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

5.1.4 Style Sheet changes

Oracle Banking Payments 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

5.1.5 Language conversion of screens

Oracle Banking Payments 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.

5.3 Processing logic

5.3.1 Additional validation logic for a field or group of fields

Oracle Banking Payments 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 STDCIFCR function to add capital letter validation for the field "Customer name". This can be achieved as follows:

Edit the **STPKS_STDCIFCR_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_stdcifcr           IN stpks_stdcifcr_Main.ty_stdcifcr,
  p_Prev_stdcifcr      IN OUT stpks_stdcifcr_Main.ty_stdcifcr,
  p_Wrk_stdcifcr       IN OUT stpks_stdcifcr_Main.ty_stdcifcr,
  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_stdcifcr:= p_stdcifcr;

IF  p_wrk_stdcifcr.v_sttms_customer.CUSTOMER_NAME1 NOT IN
(upper(p_wrk_stdcifcr.v_sttms_customer.CUSTOMER_NAME1))
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_stdcifcr_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_stdcifcr.v_sttms_customer.CUSTOMER_NAME1

5.3.2 Modify defaulting logic for fields

Oracle Banking Payments 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 Enquire Screens of Contract extensibility

Oracle Banking Payments allows to query the online screen 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 PADOVIEW which belongs to PA module, package would be

PAPKS_PADOVIEW_CUSTOM

5.4 Reports

Oracle Banking Payments provides factory shipped BIP canned reports. User can extend the reports to suite the local requirements.

5.4.1 New BIP Reports

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

5.5 Interface

5.5.1 Switch Interface ISO8583 configuration

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

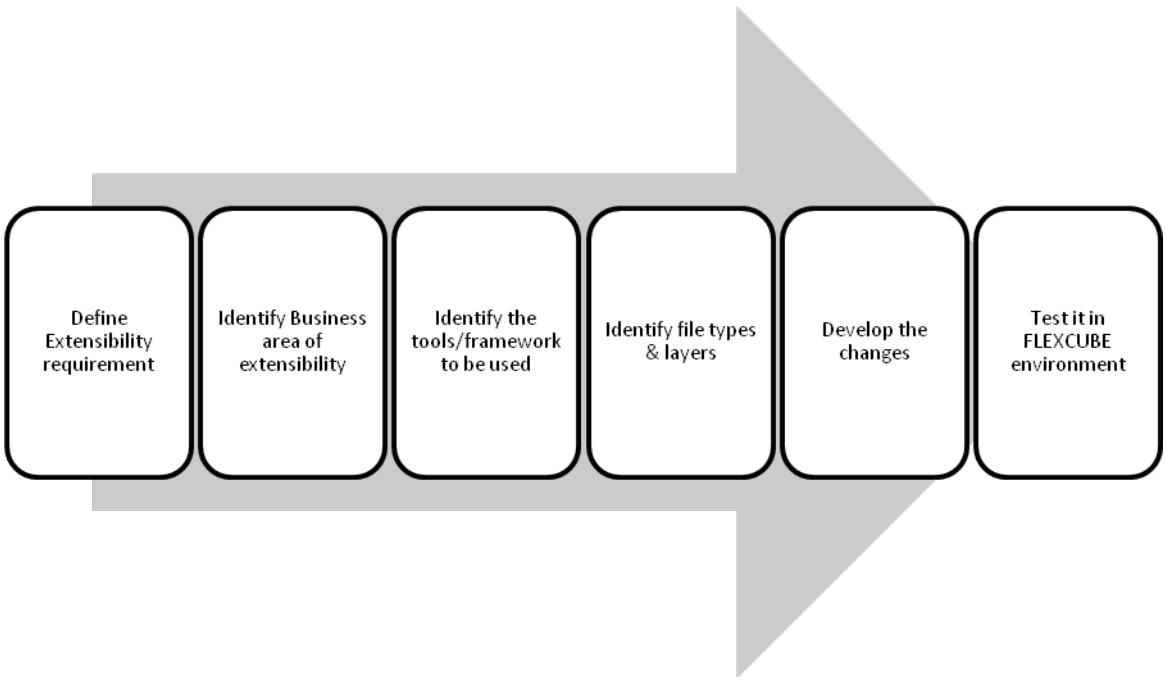
5.5.2 Configurable Generic Interface for upload/handoff

User can define following interfaces

- Incoming – to get data into Oracle Banking Payments
- Outgoing – to get data out of Oracle Banking Payments

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 Oracle Banking Payments 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)
- Accounting Interface
- Batch (New batch function during EOD time or intraday)
- Report (new report or modify existing report query)
- Interface (New incoming or outgoing)

6.3 Identify the tools/framework to be used

Area	Oracle Banking *Payments* *Tools/Framework*
Function ID	▪ RAD ▪ Style sheet editor
Processing logic	▪ PLSQL programming on RAD generated packages ▪ PLSQL programming on core packages
Reports	▪ BIP report 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 ORACLE Banking PAYMENTS Packages
Reports		RTF file	RAD generated Report packages
Interface	NA	NA	NA

6.5 Develop changes

User can develop the required changes using respective tools documents.

6.6 Test it in Oracle Banking Payments environment

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

Oracle Banking Payments Extensibility Getting Started
[Feb] [2018]
Version 14.0.0.0.0

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/