## **Oracle® Banking Enterprise Originations**

UI Extensibility Guide Release 2.12.0.0.0 F42013-01

May 2021



Oracle Banking Enterprise Originations UI Extensibility Guide, Release 2.12.0.0.0

F42013-01

Copyright © 2017, 2021, Oracle and/or its affiliates.

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

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

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.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. 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 or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

# Contents

Preface	14
Audience	14
Documentation Accessibility	14
Related Documents	14
Conventions	14
1 About This Guide	17
2 Objective and Scope	19
2.1 Overview	19
2.2 Objective and Scope	19
2.2.1 Extensibility Objective	19
2.3 Complementary Artefacts	19
2.4 Out of Scope	20
3 Overview of Use Cases	21
3.1 Extensibility Use Cases	21
3.1.1 ADF Screen Customization Using UI Extensions	21
3.1.2 ADF Screen Customization Using MDS	22
3.1.3 Print Receipt Functionality	
4 ADF Screen Customizations Using UI Extensions	
4.1 UI Extension Interface	27
4.2 Default UI Extension	
4.3 UI Extension Executor	29
4.4 Extension Configuration	32
4.5 Customization Examples	33
4.5.1 Replacing skin	

4.5.2 Changing the logo in the branding bar	. 35
4.5.3 Modifying fonts	35
4.5.4 Modifying images	.36
4.5.5 Graphics	.36
4.5.6 Adding a simple field to a product screen	. 36
4.5.7 Adding a complex field popup to a product screen (popup, table, tree, region, tf)	37
4.5.8 Removing an existing field from a product screen	.37
4.5.9 Making certain product optional product fields mandatory or optional	.37
4.5.10 Adding a new column to an existing product grid	.37
4.5.11 Hiding columns from an existing product grid	.38
4.5.12 Graying out certain columns from an existing product grid	39
4.5.13 Modifying properties of product table (rows or tablesummary)	39
4.5.14 Adding a new section to an existing product screen	.39
4.5.15 Hiding a section from a product screen	40
4.5.16 Adding a new tab to an existing product screen made of tabs	.40
4.5.17 Hiding a tab from a product screen made of multiple tabs	41
4.5.18 Adding new buttons or links	.41
4.5.19 Overriding / Customizing the product behaviour on certain actions like button clicks or tab-outs	.42
4.5.20 Overriding the product validation pattern	42
4.5.21 Overriding the product lengths (min/max)	42
4.5.22 Disable / Enable certain product fields	. 42
4.5.23 Change certain product fields to read-only either on load or based on ce tain conditions	er- . 42
4.5.24 Change label of existing product fields	.42
4.5.25 DC validation	43

4.5.26 LOV Extension– LOV Delegate Pattern	43
4.6 Using the JSFF Utils	44
4.6.1 How to Use JSFF Utils	
4.6.2 Sample JSFF Utils Code Snippet	44
5 ADF Screen Customizations Using MDS	47
5.1 Seeded Customization Concepts	47
5.2 Customization Layer	48
5.3 Customization Class	48
5.4 Enabling Application for Seeded Customization	50
5.5 Customization Project	53
5.6 Customization Role and Context	
5.7 Customization Layer Use Cases	
5.7.1 Adding a UI Table Component to the Screen	56
5.7.2 Approvals Framework	70
5.7.3 Override the product managedBean	111
6 Human Task Screen Extension	113
6.1 Introduction	113
6.2 Custom CSS Skin	113
6.2.1 Create New ADF Skin	
6.2.2 Apply New Skin	
7 Receipt Printing	
7.1 Prerequisite	
7.1.1 Identify Node Element for Attributes in Print Receipt Template	
7.1.2 Receipt Format Template (.rtf)	119
7.2 Configuration	
7.2.1 Parameter Configuration in the BROPConfig.properties	

7.2.2 Configuration in the ReceiptPrintReports.properties	121
7.3 Implementation	121
7.3.1 Default Nodes	122
7.4 Special Scenarios	
8 Extensibility Usage – OBP Localization Pack	
8.1 Localization Implementation Architectural Change	126
8.2 Customizing UI Layer	127
8.2.1 JDeveloper and Project Customization	127
8.2.2 Generic Project Creation	133
8.2.3 MAR Creation	133
8.3 Source Maintenance and Build	141
8.3.1 Source Check-ins to SVN	141
8.3.2 .mar files Generated during Build	142
8.3.3 adf-config.xml	
8.4 Packaging and Deployment of Localization Pack	142
9 Deployment Guideline	145
9.1 Customized Project Jars	145
9.2 Database Objects	145
9.3 Extensibility Deployment	

# **List of Figures**

Figure 3–1 ADF Screen Extensions	21
Figure 3–2 ADF Screen Customization	22
Figure 3–3 Print Receipt Functionality	23
Figure 4–1 UI Extension Pre Hook and Post Hook Taskflow	26
Figure 4–2 Save Method in IntegrableTaskflowHelper	27
Figure 4–3 Example of UI Extension	28
Figure 4–4 Example of Default UI Extension	29
Figure 4–5 UI Extension Executor Class Taskflow	30
Figure 4–6 Example of UI Extension Executor Class	31
Figure 4–7 Example of UI Extension Executor Class	32
Figure 4–8 Replacing skin	34
Figure 4–9 Replacing skin	34
Figure 4–10 Example: Replacing skin	35
Figure 4–11 Replacing the logo	35
Figure 4–12 Example: To modify images	36
Figure 4–13 Example: To add a simple field to a product screen	37
Figure 4–14 Example: To remove an existing field from a region	37
Figure 4–15 Example: To add a new column to an existing prouct grid	38
Figure 4–16 Example: To hide columns from an existing product grid	39
Figure 4–17 Example: To modify the properties of product table	39
Figure 4–18 Example: To add a new section to an existing product screen	40
Figure 4–19 Example: To add a new tab to existing product screen made of tabs .	41
Figure 4–20 Example: To hide a tab from a product screen made of multiple tabs .	41
Figure 4–21 Example: To add new buttons or links	42

Figure 4–22 Example: To override the product validation pattern	42
Figure 4–23 LOV Extension– LOV Delegate Pattern	43
Figure 4–24 Sample Code Snippet	44
Figure 4–25 Example of JSFF Utils	44
Figure 5–1 Customization Application View	47
Figure 5–2 CustomizationLayerValues.xml	48
Figure 5–3 Customization Class	49
Figure 5–4 Implementation for the abstract methods of CustomizationClass	50
Figure 5–5 Enable Seeded Customizations	51
Figure 5–6 Adding com.ofss.fc.demo.ui.OptionCC.jar	52
Figure 5–7 Adding com.ofss.fc.demo.ui.OptionCC.OptionCC	52
Figure 5–8 Adf-config.xml	53
Figure 5–9 Customization Developer	54
Figure 5–10 Selecting Always Prompt for Role Selection on Start Up	55
Figure 5–11 View Customization Context	56
Figure 5–12 Adding a UI Table Component - Party Search screen	57
Figure 5–13 Adding a UI Table Component - Related Party screen	57
Figure 5–14 Creating Binding Bean Class	59
Figure 5–15 Create Event Consumer Class	60
Figure 5–16 Creating Managed Bean	60
Figure 5–17 Create Data Control	61
Figure 5–18 Adding View Object Binding to Page Definition - Add Tree Binding	62
Figure 5–19 Adding View Object Binding to Page Definition - Update Root Data Source	63
Figure 5–20 Page Data Binding Definition - Insert Item	64
Figure 5–21 Page Data Binding Definition - Create Action Binding	65
Figure 5–22 Edit Event Map	66

Figure 5–23 Event Map Editor	67
Figure 5–24 Add UI Components to Screen	68
Figure 5–25 Application Navigator	69
Figure 5–26 Party Search	70
Figure 5–27 Contact Point Screen	71
Figure 5–28 Create Table	72
Figure 5–29 Create Java Project	72
Figure 5–30 Create Domain Objects	73
Figure 5–31 Create Interface	73
Figure 5–32 Create Class	74
Figure 5–33 Set OBP Plugin Preferences	74
Figure 5–34 Set OBP Plugin Preferences	75
Figure 5–35 Set OBP Pugin Prefernces	76
Figure 5–36 Create Application Service	77
Figure 5–37 Application Service Classes Generated	77
Figure 5–38 Modify Data Transfer Object (DTO)	
Figure 5–39 Generate Service and Facade Layer Sources	79
Figure 5–40 Modify ContactExpiryApplicationServiceSpi.java	80
Figure 5–41 Modify ContactExpiryApplicationServiceSpi.java	81
Figure 5–42 Modify ContactExpiryApplicationServiceSpi.java	82
Figure 5–43 Java Packages	82
Figure 5–44 Export Java Project as JAR	83
Figure 5–45 Create ContactExpiry.hbm.xml	84
Figure 5–46 Configure hostapplicationlayer.properties	84
Figure 5–47 Configure ProxyFacadeConfig.properties	85
Figure 5–48 Configure JSONServiceMap.properties	

Figure 5–49 Create Model Project	86
Figure 5–50 Create Model Project - Configure Java Settings	87
Figure 5–51 Create Application Module	88
Figure 5–52 Set Package and Name of Application Module	89
Figure 5–53 Summary of Application Module Created	90
Figure 5–54 Create View Object	91
Figure 5–55 View Attribute	92
Figure 5–56 Application Module	93
Figure 5–57 Create View Object - Summary	94
Figure 5–58 Create View Controller Project	95
Figure 5–59 Name your Project	96
Figure 5–60 Libraries and Classpath	97
Figure 5–61 Dependencies	97
Figure 5–62 Create Maintenance State Action Interface	98
Figure 5–63 Create Update State Action Class	99
Figure 5–64 Create Update State Action Class	100
Figure 5–65 DemoContactPoint.java displays the View Objects	101
Figure 5–66 DemoCreateContactPoint / DemoUpdateContactPoint	102
Figure 5–67 Create Contact Expiry DTO	102
Figure 5–68 Value Change Event Handler for the Expiry Date UI Component	103
Figure 5–69 Value Change Event Handlers for Existing UI Components	103
Figure 5–70 Method to fetch Screen Data using Contact Expiry Proxy Service .	104
Figure 5–71 Create Managed Bean	104
Figure 5–72 Create Event Customer Class	105
Figure 5–73 Create Data Control	106
Figure 5–74 Generated contactPoint.jsff.xml	107

Figure 5–75 Add an attributeValues binding	
Figure 5–76 Create Attribute Binding	
Figure 5–77 Add a methodAction binding	
Figure 5–78 Create Action Binding	109
Figure 5–79 Select the Event Consumer Method	110
Figure 5–80 Generated contactPoint.jsff.xml	
Figure 5–81 PI041 - Contact Point Screen	111
Figure 6–1 Sample trinidad-skins.xml	
Figure 6–2 Package Structure	114
Figure 6–3 Sample Data	114
Figure 6–4 Sample Implementation	115
Figure 7–1 Input Property Files	
Figure 7–2 Build Path of Utility	118
Figure 7–3 Utility Execution	119
Figure 7–4 Excel Generation	
Figure 7–5 Receipt Format Template	
Figure 7–6 Receipt Print Reports	121
Figure 7–7 Sample of Print Receipt	
Figure 8–1 Perfection Capture Screen	
Figure 8–2 Localization Implementation Architectural Change	
Figure 8–3 Package Structure	127
Figure 8–4 Customization of the JDeveloper	128
Figure 8–5 Customization of the JDeveloper	128
Figure 8–6 Configure Design Time Customization layer	129
Figure 8–7 Enabling Seeded Customization	130
Figure 8–8 Library and Class Path	131

Figure 8–9 MDS Configuration	.132
Figure 8–10 MDS Configuration	.133
Figure 8–11 MAR Creation	. 134
Figure 8–12 MAR Creation - Application Properties	.135
Figure 8–13 MAR Creation - Create Deployment Profile	.136
Figure 8–14 MAR Creation - MAR File Selection	.137
Figure 8–15 MAR Creation - Enter Details	. 138
Figure 8–16 MAR Creation - ADF Library Customization	. 139
Figure 8–17 MAR Creation - Edit File	.140
Figure 8–18 MAR Creation - Application Assembly	.141
Figure 8–19 Package Deployment	.143
Figure 9–1 Extensibility Deployment	.146

# **List of Tables**

Table 8–1 Path Structure	.14	11	
		г ,	1

# Preface

This guide explains customization and extension of Oracle Banking Enterprise Originations.

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

## Audience

This guide is intended for the users of Oracle Banking Enterprise Originations.

# **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.

#### Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

# **Related Documents**

For more information, see the following documentation:

- For installation and configuration information, see the Oracle Banking Enterprise Originations Localization Installation Guide Silent Installation guide.
- For a comprehensive overview of security, see the Oracle Banking Enterprise Originations Security Guide.
- For the complete list of licensed products and the third-party licenses included with the license, see the Oracle Banking Enterprise Originations Licensing Guide.
- For information related to setting up a bank or a branch, and other operational and administrative functions, see the Oracle Banking Enterprise Originations Administrator Guide.
- For information on the functionality and features, see the respective Oracle Banking Enterprise Originations Functional Overview documents.
- For recommendations of secure usage of extensible components, see the Oracle Banking Enterprise Originations Secure Development Guide.

# Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

# **1 About This Guide**

This guide is applicable for the following products:

- Oracle Banking Platform
- Oracle Banking Enterprise Originations
- Oracle Banking Enterprise Default Management
- Oracle Banking Loans Servicing
- Oracle Banking Deposits and Lines of Credit Servicing

References to Oracle Banking Platform or OBP in this guide apply to all the above mentioned products.

# 2 Objective and Scope

This chapter defines the objective and scope of this document.

## 2.1 Overview

Oracle Banking Platform (OBP) is designed to help banks respond strategically to today's business challenges, while also transforming their business models and processes to reduce operating costs and improve productivity across both front and back offices. It is a one-stop solution for a bank that seeks to leverage Oracle Fusion experience for its core banking operations, across its retail and corporate offerings.

OBP provides a unified yet scalable IT solution for a bank to manage its data and end-to-end business operations with an enriched user experience. It comprises pre-integrated enterprise applications leveraging and relying on the underlying Oracle Technology Stack to help reduce in-house integration and testing efforts.

# 2.2 Objective and Scope

Most product development can be accomplished through highly flexible system parameters and business rules. Further competitive differentiation can be achieved through IT configuration and extension support. In OBP, additional business logic required for certain services is not always a part of the core product functionality but could be a client requirement. For these purposes, extension points and customization support have been provided in the application code which can be implemented by the bank and / or by partners, wherein the existing business logic can be added with or overridden by customized business logic. This way the time consuming activity of custom coding to enable region specific, site specific or bank specific customizations can be minimized.

## 2.2.1 Extensibility Objective

The broad guiding principles with respect to providing extensibility in OBP are summarized below:

- Strategic intent for enabling customers and partners to extend the application.
- Internal development uses the same principles for client specific customizations.
- Localization packs
- Extensions by Oracle Consultants, Oracle Partners, Banks or Bank Partners.
- Extensions through the addition of new functionality or modification of existing functionality.
- Planned focus on this area of the application. Hence, separate budgets specifically for this.
- Standards based OBP leverages standard tools and technology
- Leverage large development pool for standards based technology.
- Developer tool sets provided as part of JDeveloper and Eclipse for productivity.

## 2.3 Complementary Artefacts

The document is a developer's extensibility guide and does not intend to work as a replacement of the functional or technical specification, which would be the primary resource covering the following:

- OBP Zen training course
- OBP installation and configuration
- OBP parameterization as part of implementation
- Functional solution and product user guide

References to plugin indicate the eclipse based OBP development plugin for relevant version of OBP being extended. The plugin is not a product GA artefact and is a means to assist development. Hence, the same is not covered under product support.

# 2.4 Out of Scope

The scope of extensibility does not intend to suggest that OBP is forward compatible.

# **3 Overview of Use Cases**

The use cases that are covered in this document shall enable the developer in applying the discipline of extensibility to OBP. While the overall support for customizations is complete in most respects, the same is not a replacement for implementing a disciplined, thoughtful and well-designed approach towards implementing extensions and customizations to the product.

# 3.1 Extensibility Use Cases

This section gives an overview of the extensibility topics and customization use cases to be covered in this document. Each of these topics is detailed in the further sections.

### 3.1.1 ADF Screen Customization Using UI Extensions

In OBP, additional business logic or UI component changes might be required for certain ADF screen. This additional logic is not part of the core product functionality, but could be a client requirement. For this purpose, hooks have been provided in the application code wherein additional business logic can be added with custom business logic.

#### Figure 3–1 ADF Screen Extensions



#### Note

Screen changes can be implemented using the UI extensions or ADF Screen Customization. It is recommended to use the UI extensions where possible as migration path to higher release of OBP is easier.

#### **UI Extension:**

This hook resides in the ADF taskflow. This hook is present before as well as after the actual UI event execution. The additional business logic has to implement the interface I<taskflow\_name>UIExt and extend and override the default implementation Void<taskflow\_name>UIExt provided for the taskflow. Multiple implementations can be defined for a particular taskflow. The UI extensions executor invokes all the implementations defined for the particular taskflow both before and after the actual UI event execution.

## 3.1.2 ADF Screen Customization Using MDS

OBP application may need to be customized for certain additional requirements. However, since these additional requirements differ from client to client, and the base application functionality remains the same, the code to handle the additional requirements is kept separate from the code of the base application. For this purpose, Seeded Customizations (built using Oracle Meta-data Services framework) can be used to customize an application.

When designing seeded customizations for an application, one or more customization layers need to be specified. A customization layer is used to hold a set of customizations. A customization layer supports one or more customization layer value which specifies the set of customizations to apply at runtime.





### 3.1.3 Print Receipt Functionality

OBP has many transaction screens in different modules where it is desired to print the receipt with different details about the transaction. This functionality provides the print receipt button on the top right corner of the screen which gets enabled on the completion of the transaction and can be used for printing of receipt of the transaction details.





# 4 ADF Screen Customizations Using UI Extensions

This chapter describes how additional business logic can be added prior to (pre hook) and / or post the execution (post hook) of a particular UI event business logic on the UI side. Extension prior to a UI event execution may be required for the purposes of additional input validation, input manipulation, custom logging, and so on. A few examples in which the UI extensions in the form of pre and post hook are required are mentioned below.

A UI extension in the form of a pre hook can be important in the following scenarios:

- Additional input validations
- Execution of business logic, which necessarily has to happen before going ahead with normal event execution
- Request manipulation prior to making host call

A UI extension in the form of a post hook can be important in the following scenarios:

- Output response manipulation
- Custom UI components rendering, changing to read only



Figure 4–1 UI Extension Pre Hook and Post Hook Taskflow

The pre hook is provided after the invocation of *Ulevent* call inside the Abstract Taskflow Handler. The extension method is provided with the ADF event and the Taskflow Handler Instance as parameters. The handler instance may be required in such cases where the VO attributes or the UI components need to be accessed as a part of the customization.

The post hook is provided after the event business logic. Similar parameters are provided in the post extension. Hooks are provided in handler and assembler methods, for taskflows using the Integrable Taskflow framework. Hooks are provided in backing bean methods for all other taskflows.



Figure 4–2 Save Method in IntegrableTaskflowHelper

For taskflows implementing the ADF Integrable Taskflow Framework, the pre and post hooks are provided for the common Integrable taskflow helper methods. Refer to the above sequence diagram for the Save method in IntegrableTaskflowHelper.

The following sections detail the important concepts which should be understood for extending in this UI layer.

## 4.1 UI Extension Interface

The OBP ADF Taskflow Generator generates an interface for the extensions of a particular taskflow. The interface name is of the form I<Taskflow\_Name>UIExt. This interface has a pair of pre and post method definitions for each public method present in the Abstract Taskflow Handler and the Integrable Taskflow Helper. The signatures of these methods are:

```
public void pre<Method_Name>(<Method_Parameters>) throws
FatalException;
public void post<Method_Name>(<Method_Parameters>) throws
FatalException;
```

A single method is provided for Integrable Assembler. The signature as below:

```
public void assembler<Method_Name>(<Method_Parameters>) throws
FatalException;
```

A UI extension class has to implement this interface. The pre method of the extension is executed before the actual method and the post method of the extension is executed after the method.

The return type for certain methods are boolean (for example, public boolean preValidateData).

Figure 4–3 Example of UI Extension



# 4.2 Default UI Extension

The OBP plug-in generates a default extension for a particular taskflow in the form of the class **Void<Taskflow\_Name>UIExt**. This class implements the aforementioned UI extension interface without any business logic, that is, the implemented methods are empty.

The default extension is a useful and convenient mechanism to implement the pre and / or post extension hooks for specific methods of a taskflow. Instead of implementing the entire interface, one should extend the default extension class and override only the required methods with the additional business logic. Product developers DO NOT implement any logic, including product extension logic, inside the default extension classes. This is because the classes are auto-generated, reserved for product use, and get overwritten as a part of a bulk generation process.

Figure 4–4 Example of Default UI Extension



## 4.3 UI Extension Executor

The OBP plug-in for Eclipse generates a UI extension executor interface and an implementation for the executor interface. The naming convention for the generated executor classes which enable "extension chaining" is shown below:

```
Interface : I<Taskflow_Name>UIExtExecutor
Implementation : < Taskflow Name >UIExtExecutor
```

The UI extension executor class, on load, creates an instance each of all the extensions defined in the UI extensions configuration. If no extensions are defined for a particular service, the executor creates an instance of the default extension for the taskflow. The executor also has a pair of pre and post methods for each method. These methods in turn call the corresponding methods of all the extension classes defined for the taskflow.

Figure 4–5 UI Extension Executor Class Taskflow



Figure 4–6 Example of UI Extension Executor Class

🔘 Java EE - Task	Flows/com.ofss.fc.ui.taskflows.agent/src/com/ofss/fc/ui/taskflows/agent/agentdetails/view/ext/IAgentDetailsUIExtExecutor.java - Eclipse		×					
Eile Edit Source Refactor Navigate Search Project Run Window Help								
	Quick Access 😰 😰 Java EE 🏪 SVN Repository Exploring 🖆 Team Synchronizing 🚸 Plug-in Development 🚸 Debu	g 🥳 \	Neb					
📜 🚺 IAgent	DetailsUIExtExecutor.java 🛛	• #						
1 pa	ackage com.ofss.fc.ui.taskflows.agent.agentdetails.view.ext;	*	- -					
2			<u> </u>					
3⊛ import oracle.jbo.View0bject;[]								
289 /	28@ /**							
29	*							
30	* ExtensionFactory hook for Managed Bean AgentDetails. Extension Factories should implement the							
31	ABENTUETAIISUIEXTEXECUTON		행					
33 '		E	2					
34 pt	ublic interface IAgentDetailsUIExtExecutor extends IIntegrableTaskFlowUIExtExecutor <object, agentdetailstaskflowcontext=""> {</object,>							
35	/**		-					
37	* This is the extension point for Managed Bean AgentDetailsAssembler The parameters related to this method are same		2					
38	* as it is in the public method along with the instance of Managed Bean The javadoc for the original method and the							
39	* params can be seen from the See Also link.		<b>U</b>					
40	* * * @see AgentDetailsAssember#AnEetshAgentDetails		Ð					
41	isce vBeilingeranzwiseimei wolli errikaeirineranza		1					
43	public void assemblerCreateAgentDetailsDTOForFetchAgentDetails(AgentDetailsDTO agentDetailsDTO, ViewObject agentDetailsVO, Vie	eı	f0					
44	1		*					
450	/*** * This is the extension point for Managed Bean AgentDetailsAssembler The parameters related to this method are same		<u> </u>					
47	* as it is in the public method along with the instance of Managed Bean The javadoc for the original method and the							
48	* params can be seen from the See Also link.							
49	* *							
50	*/  (see AgencoelaiisAssember#onrectaiis							
52	public void assemblerPopulateVOForFetchAgentDetails(AgentDetailsResponse agentDetailsResponse, ViewObject agentDetailsVO, View	W						
53								
540	/** * This is the extension point for Managed Rean AgentDatails The parameters related to this method are same as it is							
56	* in the public method along with the instance of Managed Bean The parameters related to this method and the params							
57	* can be seen from the See Also link.							
58								
59	<pre>~ @see AgentDetalls#intorCommandLinkUnClick */</pre>							
61	public void preInforCommandLinkOnClick(IIntegrableTaskFlowHandler <object, agentdetailstaskflowcontext=""> iIntegrableTaskFlowHandler<object, agentdetailstaskflowcontext=""> iIntegrableTaskFlowHandler</object,> iIntegrableTaskFlowHandler iIntegrableTaskFlowHandler iIntegrableTaskFlowHandler iIntegrableTaskFlowContext> iIntegrableTaskFlo	ni						
62								
630	630 /**							
65	b4 ints is the extension point for Managed Bean AgentDetails The parameters related to this method are same as it is 55 in the public method along with the instance of Managed Bean The jayador for the original method and the parameters							
66	66 * can be seen from the See Also link.							
· · · · ·	* III							
	Writeble Smootlegent 1.1							
	vviluable Smart III							
		_	-					

Figure 4–7 Example of UI Extension Executor Class



## 4.4 Extension Configuration

The extension classes that implement the extension interface are mapped to the taskflow with the help of seed data in **FLX\_FW\_CONFIG\_ALL\_B**.

Following is a sample implementation.

Single Extension Class

```
insert into
FLX_FW_CONFIG_ALL_B(CATEGORY_ID, PROP_ID, PROP_VALUE, PROP_
COMMENTS, OBJECT_VERSION_NUMBER, CREATED_BY, CREATION_DATE, LAST_
UPDATED_BY, LAST_UPDATED_DATE, OBJECT_STATUS_FLAG, FACTORY_SHIPPED_
FLAG)
values
('UIExtensions','com.ofss.fc.ui.taskflows.account.accountholderpre
ferencesetup.view.backing.AccountHolderPreferenceSetup','com.ofss.
fc.lz.au.ui.taskflows.account.accountholderpreferencesetup.view.ex
t.RegionalAccountHolderPreferenceSetupUIExt','',1,'ofssuser',SYSDA
TE,'ofssuser',SYSDATE,'A','y');
```

#### Multiple Extension Classes

insert into

```
FLX FW CONFIG ALL B(CATEGORY ID, PROP ID, PROP VALUE, PROP
COMMENTS, OBJECT VERSION NUMBER, CREATED BY, CREATION DATE, LAST
UPDATED BY,LAST UPDATED DATE,OBJECT STATUS FLAG,FACTORY SHIPPED
FLAG)
values
('UIExtensions','com.ofss.fc.ui.taskflows.account.accountholderpre
ferencesetup.view.backing.AccountHolderPreferenceSetup', 'com.ofss.
fc.lz.au.ui.taskflows.account.accountholderpreferencesetup.view.ex
t.RegionalAccountHolderPreferenceSetupUIExt','',1,'ofssuser',SYSDA
TE, 'ofssuser', SYSDATE, 'A', 'y');
insert into
FLX FW CONFIG ALL B(CATEGORY ID, PROP ID, PROP VALUE, PROP
COMMENTS, OBJECT VERSION NUMBER, CREATED BY, CREATION DATE, LAST
UPDATED BY,LAST UPDATED DATE,OBJECT STATUS FLAG,FACTORY SHIPPED
FLAG)
values
('UIExtensions','com.ofss.fc.ui.taskflows.account.accountholderpre
ferencesetup.view.backing.AccountHolderPreferenceSetup', 'com.ofss.
fc.lz.au.ui.taskflows.account.accountholderpreferencesetup.view.ex
t.RegionalAccountHolderPreferenceSetupUIExtForUseCase1','',1,'ofss
user',SYSDATE,'ofssuser',SYSDATE,'A','y');
insert into
FLX FW CONFIG ALL B(CATEGORY ID, PROP ID, PROP VALUE, PROP
COMMENTS, OBJECT VERSION NUMBER, CREATED BY, CREATION DATE, LAST
UPDATED BY,LAST UPDATED DATE,OBJECT STATUS FLAG,FACTORY SHIPPED
FLAG)
values
('UIExtensions','com.ofss.fc.ui.taskflows.account.accountholderpre
ferencesetup.view.backing.AccountHolderPreferenceSetup', 'com.ofss.
fc.lz.au.ui.taskflows.account.accountholderpreferencesetup.view.ex
t.RegionalAccountHolderPreferenceSetupUIExtForUseCase2','',1,'ofss
user',SYSDATE,'ofssuser',SYSDATE,'A','y');
```

It is possible to configure multiple implementations of pre or post extensions for a taskflow in this layer. This is achieved with the help of the extension executor. It has the capability to loop through a set of extension implementations, which conform to the extension interface supported by the taskflow.

## 4.5 Customization Examples

Following are some examples of customization.

### 4.5.1 Replacing skin

Colours are maintained as a variable in the css lib files of the respective modules. Skin can be replaced to change the colours.

Replace skin: inside preCustomBranding()

@Override

```
public void preCustomBranding(Main main) {
```

/\*setting skin \*/

FacesContext fc = FacesContext.getCurrentInstance();

ELContext elc = fc.getELContext();

String skinId = "skyros";

ExpressionFactory exprFact = fc.getApplication().getExpressionFactory();

```
ValueExpression ve = exprFact.createValueExpression(elc, "#{sessionScope.skinFamily}", Object.class);
```

ve.setValue(elc, skinId);

/\* setting fonts \*/

main.setFontPath("/css/lato.css");

/\* set this flag to false so as to execute pre hook only once when main is loaded \*/

ELHandler.set("#{pageFlowScope.isCustomBranding}","false");

super.preCustomBranding(main);

```
}
```

#### Figure 4–8 Replacing skin

```
@Override
public void preCustomBranding(Main main) {
    /*setting skin */
    FacesContext fc = FacesContext.getCurrentInstance();
    ELContext elc = fc.getELContext();
    String skinId = "skyros";
    ExpressionFactory exprFact = fc.getApplication().getExpressionFactory();
    ValueExpression ve = exprFact.createValueExpression(elc, "#{sessionScope.skinFamily}", Object.class);
    ve.setValue(elc, skinId);
    /* setting fonts */
    main.setFontPath("/css/lato.css");
    /* set this flag to false so as to execute pre hook only once when main is loaded */
    ELHandler.set("#{pageFlowScope.isCustomBranding}","false");
    super.preCustomBranding(main);
}
```

#### Figure 4–9 Replacing skin

```
public Main() {
    if (voidMainUIExt == null || mainUIExtExecutor == null) {
        voidMainUIExt == new VoidMainUIExt();
        mainUIExtExecutor = new MainUIExtExecutor();
        extension = (IMainUIExtExecutor) UIExtensionFactory.getUIExtensionExecutor(Main.class.getName());
    }
    populateTargetUnitSOC();
    /* set the value of flag isCustomBranding as false when overridden in customization */
    if (ELHandler.get("#{pageFlowScope.isCustomBranding}") == null || Boolean.valueOf(ELHandler.get("#{pageFlowScope.isCustomBranding}").toString())) {
        customBranding();
    }
}
```

Figure 4–10 Example: Replacing skin

🖉 (IDBPR-74681) Residenti x 🔨 (IDBPR-73680) TOL- PIOL: x 1 (D. Cross Reference: #326, C x 1 😇 Oracle Banking Platform: x 2 🐑 Collablet Subversion Fix: x 2 💟 List of Environments x 1 🛛 🕹 - 🗗 X	
🗧 🔿 🕐 🗛 Not secure   https://10.180.85.202.2002/com.ofs.f.cuiview/laces/main.jspx	-
👯 Apps 🗅 OBP DexOps - Your 🗧 {O. Grok. {O. Cross Reference: /? 🔄 (O. new grok 🧧 local 📡 jiraa 🏆 CJIRA 🖸 Primavera - Timeshe: 🧧 85.202 👼 T11 🔛 envior 🖷 cam 💽 App form 🗃 Tag Guide	,p
Protect Date: 1-4:4.2.2.1.6   Protect Date: 1-4:4.2.2.1.6  Protect Date: 1	0
€	9
	-
	1
	<u>a</u> .
	02
	۰.
	4-7
*	a a
	2
	w
	く口々
	6:59 PM
	Tuesday 31-Oct-17
	5

### 4.5.2 Changing the logo in the branding bar

Given the multi-brand nature, the ability is provided to display appropriate brand in OBP. For example, Westpac, St George, Bank SA & Bank of Melbourne. Logos are given in the jspx/jsff files in the current code 'Oracle' logo is maintained in "main.jspx" file. To replace a logo, refer to the following screen shot.

#### Figure 4–11 Replacing the logo

```
617
618
         @Override
619 🗆
        public void postViewBeforePhaseListener(Main main, PhaseEvent phaseEvent) {
620
621
            main.getLogoImage().setSource("/images/common/images/risks.png");
             super.postViewBeforePhaseListener(main, phaseEvent);
622
        }
623
624
625
        @Override
        public void postViewPoll(Main main, PollEvent pollEvent) {
626 🗉
```

### 4.5.3 Modifying fonts

Font-family is maintained as a variable and inherited the variable in the mixins which are used to style the various ADF components. Hence if changed the variable's value font will change.

Variable for href to be maintained in backing bean and this variable will be overridden in customization.

Example:

Main.jspx has a placeholder:

href="\${pageContext.request.contextPath}\${Main.fontPath}"

Main.java holds the path of variable

private String fontPath = "/css/roboto.css";

### 4.5.4 Modifying images

Images are maintained in the jsff as well as in css files.

Many ADF components provide provisions to give icons for different states of the components.

Example: set the icon and hover icon attribute

#### Figure 4–12 Example: To modify images

#### Self-Service Login Preferences

View 💌	Detach 🤇	2 <b>X</b>						
*	Target Unit	ColumnNew	ColumnNew					
No data to display.								
۰ III ا								

this.getButtonAdd().setIcon("/images/common/search/search\_16\_ena.png");

this.getButtonAdd().setHoverIcon("/images/common/search/search\_16\_ena.png");

Also wherever component's image can be replaced using css by applying it to the particular selector if component exposes the selector. Same as graphics.

### 4.5.5 Graphics

Graphics include buttons, warnings, and so on.

Button styles are maintained in the respective css files. (Handled through Section 4.5.1 Replacing skin)

Warning text is maintained in the resource bundle (". properties") files: Replace the properties file in config/resources/taskflows/module

SamplePath : config/resources/taskflows/BankPolicyDefinition\_en.properties

### 4.5.6 Adding a simple field to a product screen

Example: Adding input text to a panel form layout

/\* create component and set relevant properties \*/

RichInputText ui = new RichInputText();

ui.setId("rit1");

ui.setLabel("Input text");

ui.setValue("Hello");

ui.setContentStyle("font-weight:bold;color:red");

/\*add the newly created component to existing form \*/
this.getPfl1().getChildren().add(ui);

#### Figure 4–13 Example: To add a simple field to a product screen

⊿ Bank Parameters New				
* Bank Code	٩	Input text Hello		
Bank Name	BankName	圆 Button		
Bank Group Code	٩	Bank Short Name		

# 4.5.7 Adding a complex field popup to a product screen (popup, table, tree, region, tf)

/\*Handle via making mds changes\*/

### 4.5.8 Removing an existing field from a product screen

Example: Hiding an LOV from panel form layout

this.getBankCodeLOV().setVisible(false);

### Figure 4–14 Example: To remove an existing field from a region

Bank Parameters New					
	* Bank Code		Input text	Hello	
	Bank Name	BankName		Button	
Ba	ink Group Code	Q	Bank Short Name		

# 4.5.9 Making certain product optional product fields mandatory or optional

Example: Setting bank name to required

this.getBankName().setRequired(true);

### 4.5.10 Adding a new column to an existing product grid

Example: Adding a new column to a table in CS26

```
/* create new column component */
```

```
RichColumn ui1 = new RichColumn();
```

```
ui1.setHeaderText("ColumnNew");
```

```
ui1.setId("col3");
```

```
ui1.setAlign("center");
```

```
ui1.setRowHeader("unstyled");
```

/\* get the table from bindings where column needs to be added \*/

getT1().getChildren().add(ui1);

AdfFacesContext.getCurrentInstance().addPartialTarget(getT1());

```
/* set the value in column 3 as required on any event */
```

```
RichInputText ui11 = new RichInputText();
```

```
ui11.setId("rit1");
```

```
ui11.setLabel("Input text");
```

```
ui11.setValue("Hello");
```

```
ui11.setContentStyle("font-weight:bold;color:red");
```

```
ui11.setReadOnly(false);
```

```
getT1().getChildren().get(3).getId();
```

```
getT1().getRowIndex();
```

```
getT1().getChildren().get(3).getChildren().add(ui11);
```

AdfFacesContext.getCurrentInstance().addPartialTarget(getT1());

### Figure 4–15 Example: To add a new column to an existing prouct grid

### Self-Service Login Preferences

View 💌	Detach	۹ <b>X</b>	
* Ta	rget Unit	* Self Service Login Strategy	ColumnNew
Q			Hello
•			۰

# 4.5.11 Hiding columns from an existing product grid

Example: Hiding an existing column from a table in CS26

```
/* get the corresponding column and set its rendered property to false */
```

this.getT1().getChildren().get(1).setRendered(false);

### Figure 4–16 Example: To hide columns from an existing product grid

Self-Service Login Preferences			
View 🔻 🗑 Detach 🔍 🗙			
* Target Unit		ColumnNew	
٩		Hello	

## 4.5.12 Graying out certain columns from an existing product grid

/\* disabling the component that was set inside the column \*/

this. ui11.setDisabled(true);

# 4.5.13 Modifying properties of product table (rows or tablesummary)

this.getT1().setEmptyText("NewText");

in case where properties are picked up via RB the file itself can be replaced in customization

### Figure 4–17 Example: To modify the properties of product table

```
✓ Self-Service Login Preferences

      View
      Image: Detach
      Q
      X

      * Target Unit
      * Self Service
Login Strategy
      ColumnNew

      NewText
      Image: Detack
      Image: Detack

      Image: Detack
      Image: Detack
      Image: Detack
```

## 4.5.14 Adding a new section to an existing product screen

/\* create a new panel form layout \*/

RichPanelFormLayout pfl111 = new RichPanelFormLayout();

pfl111.setId("pfl111");

pfl111.setMaxColumns(2);

pfl111.setRows(1);

pfl111.setFieldWidth("60%");

pfl111.setLabelWidth("40%");

```
getPb1().getChildren().add(pfl111);
```

AdfFacesContext.getCurrentInstance().addPartialTarget(getPb1());

/\* create components to be added to that section \*/

RichInputText ui = new RichInputText();

ui.setId("rit1");

ui.setLabel("Input text");

ui.setValue("Hello");

ui.setContentStyle("font-weight:bold;color:red");

getPfl1().getChildren().add(ui);

AdfFacesContext.getCurrentInstance().addPartialTarget(getPfl1());

RichCommandButton ui2 = new RichCommandButton();

ui2.setId("ch1");

ui2.setText("Button");

ui2.setInlineStyle("font-weight:bold;");

ui2.setIcon("/images/common/search/search\_16\_ena.png");

ui2.setIcon("/images/common/print/printreciept\_16\_ena.png");

/\*add new components to the new section \*/

getPb1().getChildren().get(2).getChildren().add(ui2); getPb1().getChildren().get(2).getChildren().add(ui);

### Figure 4–18 Example: To add a new section to an existing product screen

⊿ Bank Parameters New				
* Bank Code		Bank Name	BankName	
Bank Group Code	Q	Bank Short Name		
🗐 Button		Input text	Hello	
⊿ Address Details				

### 4.5.15 Hiding a section from a product screen

/\* Hiding all components inside the panel form layout \*/

this.getPfl1().setVisible(false);

### 4.5.16 Adding a new tab to an existing product screen made of tabs

/\* create a new tab and add its relevant properties \*/

RichCommandNavigationItem ui2 = new RichCommandNavigationItem(); ui2.setId("newTab"); ui2.setSelected(false); ui2.setText("newTab");

/\* add it to the navigation pane \*/

this.getNp1().getChildren().add(ui2);

### Figure 4–19 Example: To add a new tab to existing product screen made of tabs

Back to Summary		
Facilities 🕞		Overdue DueToday Due In Future
Review Renew Exceptions newTab		
		Overdue
No items to display	<< 1 2 3 >	Show 10 💌

# 4.5.17 Hiding a tab from a product screen made of multiple tabs

this.getNp1().getChildren().get(1).setRendered(false);

Figure 4–20 Example: To hide a tab from a product screen made of multiple tabs

Back to Summary		
Facilities 🖓		Overdue DueToday Due In Future
Review Exceptions newTab		
		★ Overdue ▼
No items to display	<< 1 2 3 >	Show 10 💌

## 4.5.18 Adding new buttons or links

This approach will not work for "Approvals" and "UI level security"

```
/* Create a new command button and set its relevant properties*/
RichCommandButton ui2 = new RichCommandButton();
ui2.setId("ch1");
ui2.setText("Button");
ui2.setInlineStyle("font-weight:bold;");
ui2.setIcon("/images/common/search/search_16_ena.png");
```

/\* add it to the relevant panel component \*/

this.getPfl1().getChildren().add(ui2);

AdfFacesContext.getCurrentInstance().addPartialTarget(getPfl1());

Figure 4–21 Example: To add new buttons or links

Bank Parameters New				
* Bank Code		Input text	Hello	
Bank Name	BankName		Button	
Bank Group Code	Q	Bank Short Name		

# 4.5.19 Overriding / Customizing the product behaviour on certain actions like button clicks or tab-outs

/\* need to create a new link programmatically and link the action listener method to it \*/

## 4.5.20 Overriding the product validation pattern

this.getPolicyName().setPattern("[a-zA-Z]\*");

### Figure 4–22 Example: To override the product validation pattern

5-Jan-2016	Serror: The format is incorrect.	9 ¼ ¼ ✓ X 0 ✓ OFSSUser ✓	E
CS26 ×	Enter a value that matches this patient. [a*2*2]		
Bank Policy	Any Character In a ZA-Z Zero or More Times	+ Print Client-Id 🖶 Print 🗸 OK Z Clear 🗙 C	ancel
A Bank Policy Definition	Enter between 1 and 10 characters		
	Policy Name		
* Policy Code	* Policy Name	abcABC11	
* Policy Description			
* Bank Code 10	Bank Name	Emerald Bank Global	
			18

### 4.5.21 Overriding the product lengths (min/max)

this.getPolicyName().setMaxLength("10");

### 4.5.22 Disable / Enable certain product fields

this.getBankName().setDisabled(true);

# 4.5.23 Change certain product fields to read-only either on load or based on certain conditions

this.getBankName().setReadOnly(true);

### 4.5.24 Change label of existing product fields

this.getBankName().setValue("BankName");

# 4.5.25 DC validation

The text for error message comes from "CommonValidationMessages\_en.properties" and this file can be replaced in customization. However the values for Min and Max length inside the message can be overridden.

# 4.5.26 LOV Extension– LOV Delegate Pattern

- Consulting use case:
  - Display the list of accounts of the logged in user from a third party system.
- Implementation:
  - Re-use "LOVDelegate" framework
  - Override the existing implementation in HostQueries.xml with a <service> tag while the existing product implementation is present conditionally.
  - <Service> tag in-turns points to a new LOVDelegate class which implements the ILOVDelegate interface.
  - The entire custom implementation to fetch external records will be present in the LOVDelegate class.
  - Conditionally invoke the consulting implementation or product implementation based on the requirements.
- Key Benefits:
  - Easy to plug-in with minimal changes. Host layer only impacted with no impact to the presentation layer.
  - Query can be overridden in a very sophisticated way with the use of <Service> tag.
  - Plug-in-play and can be easily turn off if required.
- Visual representation is given below:





#### Figure 4–24 Sample Code Snippet



# 4.6 Using the JSFF Utils

### 4.6.1 How to Use JSFF Utils

Following is the example to use JSFFUtils:

```
JSFUtils.insertPanelHeader
("rphRomanianDetails","Details",parentId,uiComponent);
JSFUtils.insertRichPanelFormLayoutEnd
("rpfRomanian1","60%","40%",2,1,"rphRomanianDetails",uiComponent);
```

#### Figure 4–25 Example of JSFF Utils

Other Identification	n Details	
	* Type Romanian ID	* ID Fetch
Details		
	Name Test	Romanian Number 2229
Birth Date		
	Birth Date	

### 4.6.2 Sample JSFF Utils Code Snippet

```
/**
*This method adds af:PanelFormLayout ADF component at the end of
the given parent.
* @param id sets id attribute on the af:PanelFormLayout.Type
String.
* @param fieldWidth sets fieldWidth attribute on the
af:PanelFormLayout.Type String.
* @param labelWidth sets labelWidth attribute on the
af:PanelFormLayout.Type String.
```

```
* @param maxColumns sets maxColumns attribute on the
af:PanelFormLayout.Type integer.
* @param row sets row attribute on the af:PanelFormLayout.Type
integer.
* Oparam parentId is the id of the immediate parent component where
af:PanelFormLayout need to be appended.Type String
* @param superParent is the component where parentId is placed.Type
UIComponent
*/
public static void insertRichPanelFormLayoutEnd(String id,String
fieldWidth, String labelWidth, int maxColumns, int row, String
parentId, UIComponent superParent) {
UIComponent uiComponentPGL = superParent.findComponent(parentId);
RichPanelFormLayout richPanelFormLayout = new RichPanelFormLayout
();
richPanelFormLayout.setId(id);
richPanelFormLayout.setFieldWidth(fieldWidth);
richPanelFormLayout.setLabelWidth(labelWidth);
richPanelFormLayout.setMaxColumns(maxColumns);
richPanelFormLayout.setRows(row);
uiComponentPGL.getChildren().add(richPanelFormLayout);
*This method adds af:panelHeader ADF component at the end of the
given parent.
* @param id sets id attribute on the af:panelHeader.Type String.
* @param text sets text attribute on the af:panelHeader.Type
String.
* Oparam parentId is the id of the immediate parent component where
af:panelHeader need to be appended.Type String
* @param superParent is the component where parentId is placed.Type
UIComponent
*/
public static void insertPanelHeader (String id, String text, String
parentId,UIComponent superParent) {
UIComponent uiComponentParent = superParent.findComponent
(parentId);
RichPanelHeader richPanelHeader = new RichPanelHeader();
richPanelHeader.setId(id);
richPanelHeader.setText(text);
uiComponentParent.getChildren().add(richPanelHeader);
}
```

# **5 ADF Screen Customizations Using MDS**

OBP provides the extensibility to an application for customizing certain additional requirements of a client. However, since these additional requirements differ from client to client, and the base application functionality remains the same, the code to handle the additional requirements should be kept separate from the code of the base application. For this purpose, **Seeded Customizations** (built on the Oracle Metadata Services framework) can be used to customize an application.

### Note

It is recommended to use ADF screen extensions for UI changes instead of mds where ever possible as it is easier to upgrade to new version of product.

# **5.1 Seeded Customization Concepts**

When designing seeded customizations for an application, one or more customization layers need to be specified. A customization layer is used to hold a set of customizations. A customization layer supports one or more customization layer value which specifies which set of customizations to apply at runtime.

Custom Application View can be represented as follows:





Oracle JDeveloper 11g includes a special role for designing customizations for each customization layer and layer value called the Customization Developer Role.

The following section explains the details about the Oracle JDeveloper customization mode as well as customizing and extending of the ADF application artifact. The detailed documentation for customizing and extending ADF Application Artifacts is also available at the Oracle website:

http://docs.oracle.com/cd/E25178\_01/fusionapps.1111/e16691/ext\_busobjedit.htm

# 5.2 Customization Layer

To customize an application, you must specify the customization layers and their values in the CustomizationLayerValues.xml file, so that they are recognized by JDeveloper.

For example, you can create a customization layer with the name **option** and values **demo** and another bank name.

To create the customization layer, follow these steps:

 From the main menu, choose the File -> Open option. Locate and open the file CustomizationLayerValues.xml which is found in the <JDEVELOPER\_HOME>/jdeveloper/jdev directory. In the XML editor, add the entry for a new customization layer and values as shown in the following image.

Figure 5–2 CustomizationLayerValues.xml



2. Save and close the file.

# **5.3 Customization Class**

Before customizing an application, a customization class needs to be created. This class represents the interface that the *Oracle Metadata Services* framework uses to identify the customization layer that should be applied to the application's base metadata.

To create a customization class, follow these steps:

- 1. From the main menu, choose File -> New.
- 2. Create a generic project and give a name (com.ofss.fc.demo.ui.OptionCC) to the project.
- Go to Project Properties for this project and add the required MDS libraries in the classpath of the project.

Figure 5–3 Customization Class

<u>E</u> ile <u>E</u> dit <u>Y</u> iew <u>Application</u> Refa <u>c</u> tor <u>S</u> earch <u>M</u> avigate <u>B</u> uild <u>R</u> un Versi <u>o</u> ning <u>T</u> ools <u>W</u> indow <u>H</u> elp	
[22日日]19(11) 21  21  2-0- 2- 2  2  2  2  2  2  2  2  2  2  2  2	( <b>#</b>
🕼 🥝 Application Navigator X	(Þ.
	~
Projects     Solution	
■ ⊕ ⊇ com ofss fc demo ui core [10.180.22.95] ⊕ Project Source Paths ⊕ ⊕ com ofss fc demo ui core [10.180.22.95] ⊕ Broject Source Paths Use Custom Settings Custom ige Settings	
Comors & Comors	
B R com ofss fc demo ul view party [10.180.22.95] B Ant [ava 5c version: [ava 5c ve	
H = "#22 con.offss.tc.ul.vew [10.180.22.55]	
B Dependencies Classpart entries:	
B Com Deployment Deployment Add IDDPlicetopy	
B B Linnees Extension	
a a autoration and a second and	
B - S WebHelp - Java Et Application     G - S WebHelp - Java Et Application <u>Edit.</u>	
B S Page Flows Share As	
error.html 215051 — Run/Debug/Profile	
login.html 227634	
🕀 🔂 Application Sources	
Gimeb Content	
B	
😑 🍙 com.ofss.fc	
20 option 0 option	
DoptionCC.java	
Help OK Cancel	J.
39	-
Source Design History <	>

4. Create the customization class in this project. The customization class **must** extend the *oracle.mds.cust.CustomizationClass* abstract class.

Following are the abstract methods of the CustomizationClass:

- getCacheHint() This method will return the information about whether the customization layer is applicable to all users, a set of users, a specific HTTP request or a single user.
- getName() This method will return the name of the customization layer.
- getValue() This method will return the customization layer value at runtime.

The screenshot below depicts an implementation for the methods:



Figure 5–4 Implementation for the abstract methods of CustomizationClass

- 5. Build this class and deploy the project as a JAR file (com.ofss.fc.demo.ui.OptionCC.jar). This JAR file should only contain the customization class.
- 6. Place this JAR file in the location <JDEVELOPER\_HOME>/jdeveloper/jdev/lib/patches so that the customization class is available in the classpath of JDeveloper.

# 5.4 Enabling Application for Seeded Customization

Seeded customization of an application is the process of taking a generalized application and making modifications to suit the needs of a particular group. The generalized application first needs to be enabled for seeded customization before any customizations can be done on the application.

To enable seeded customization for the application, follow these steps:

- 1. Go to the Project Properties of the application's project.
- 2. In the ADF Views section, check the Enable Seeded Customizations option.

1 1 PApplication Navigator X		
ທຸ 👝 View	S Project Proper	rcies - /home/rshandha/work/jdev_workspace/view/com.orss.rc.ui.view/com
		ADF View
<pre>Projects Project</pre>	5]     ⊕ Project Source Pat       80.22.95]     ⊕ ADF Model       22.95]     ⊕ ADF Model       1.22.95]     ⊕ Ant       ⊕ Deployment     ⊕ Deployment       - EJB Module     - Extension       - Java EE Application     - Java EE Application       - JSP Tay Libraries     - JSP Tay Libraries       - JSP Tay Libraries     - Sequere Bundle       - Run/Debug/Profile     - Technology Scope	hs Use <u>Qustom Settings</u> © <u>Use Project Settings</u> When ADF Faces is present. JSF HTML widgets will not show when dropping datacontrols. □ Include JSF HTML Widgets Configure customization options for ADF Faces. Note that write access to WEB-INF/web_will is required. ♥ Enable User Customizations ○ <u>Across Sessions using MDS</u> ♥ Enable <u>Seeded Customizations</u> Configure default skin family for this project. <u>Default Skin Family</u> : <u>#(sessionScope.skinFamily)</u> ▼
	Help	OK Cancel
	Overvi	ew Source History < >

Figure 5–5 Enable Seeded Customizations

3. In the Libraries and Classpath section, add the previously deployed com.ofss.fc.demo.ui.OptionCC.jar which contains the customization class.

Figure 5–6 Adding com.ofss.fc.demo.ui.OptionCC.jar



4. In the Application Resources tab, open the adf-config.xml present in the Descriptors/ADF META-INF folder. In the list of Customization Classes, remove all the entries and add the com.ofss.fc.demo.ui.OptionCC.OptionCC class to this list.

Figure 5–7 Adding com.ofss.fc.demo.ui.OptionCC.OptionCC

Application Navigator X	📓 DemoContactPoint.java 🗴 📓 OptionCC.java 🗴 📓 OptionCC.java 🗴 🞯 adf-config.xml 🗴 💣 contactPoint.jsff 🗴 👹	•••
🔁 View 💌 🐨 🕶		? ^
Projects     Com. offs.f. demo. ul. core [10.180.22.95]     Com. offs.f. demo. ul. core [10.180.22.95]     Com. offs.f. demo. ul. optionCC [10.180.22.95]     Com. offs.f. demo. optionCC [10.180.22.95]     Com. optionCC [10.180.22.95]     Com. optionCC [10.180.22.95]     Com. optionCC [10.180.22.95]     C	Business Components          MDS Configuration         Centroller         View         Customization Configuration: Match Path = "/"         Use the following editor for cases where the customization configuration map to the global camesoace (CO)         Customization Classes         configure Design Time Customization Layer Values	
	Overview Source   History S	- P

#### Figure 5–8 Adf-config.xml



# 5.5 Customization Project

After creating the Customization Layer and the Customization Class and enabling the application for Seeded Customizations, the next step is to create a project which will hold the customizations for the application.

To create the customization project, follow these steps:

- 1. From the main menu, choose **File -> New**. Create a new Web Project with the following technologies:
  - ADF Business Components
  - Java
  - JSF
  - JSP and Servlets
- 2. Go to the **Project Properties** of the project and in the classpath of the project, add the following jars:
  - Customization class JAR (com.ofss.fc.demo.ui.OptionCC.jar)
  - The project JAR which contains the screen / component to be customized. For example, if you
    want to customize the Party -> Contact Information -> Contact Point screen, the related project
    JAR is com.ofss.fc.ui.view.party.jar.
  - All the dependent JARS / libraries for the project JAR.
  - Enable this project for Seeded Customizations.

# **5.6 Customization Role and Context**

Oracle JDeveloper 11g includes a specific role called Customization Developer Role that is used for editing seeded customizations.

To edit customizations to an application, you will need to switch JDeveloper to that role, follow these steps:

1. In **Tools > Preferences > Roles**, select the Customization Developer Role.

Figure 5–9 Customization Developer

8 Preferences	
( Search	Roles
⊕ Diagrams Extensions	Bole: Default Role
- External Editor File Types	Enables all technologies.      Or Customization Developer
Global Ignore List Http Analyzer	Configures the product for customizing metadata.
JavaScript Editor	Database Edition Includes only features for core database development.
Mouseover Popups	Java EE Edition Includes only features for core Java EE development.
Run Shortcut Keys	Java Edition     Includes only features for core lava development.
⊕ Tasks ⊕ TopLink	
UML Usage Reporting	
Web Browser and Prox	
XML Schemas	✓ Always prompt for role selection on startup
Help	OK Cancel

2. Select the "Always prompt for role selection on start up" option.

Figure 5–10 Selecting Always Prompt for Role Selection on Start Up

8 Select Role					
Select the role that matches your requirements. You can also change roles using the Roles page in preferences.					
<u>R</u> ole:					
O Default Role Enables all technologies.					
Oustomization Developer					
Configures the product for customizing metadata.					
<ul> <li>Database Edition</li> <li>Includes only features for core database development.</li> </ul>					
Java EE Edition Includes only features for core Java EE development.					
Java Edition Includes only features for core Java development.					
Always prompt for role selection on startup					
OK Cancel					

- 3. On restarting JDeveloper, you will be prompted for role selection. Select *Customization Developer Role.*
- 4. Once Oracle JDeveloper 11g has restarted, ensure that the application to be customized is selected in the Application Navigator and have a look around the integrated development environment. You will notice a few changes from the Default Role. The first change you might notice is that files (such as Java classes), that are not customizable, are now read only. The Customization Developer Role can only be used for editing seeded customizations. Anything that is not related to seeded customizations will be disabled. The second major difference you might notice is the MDS Customization Context window that is displayed.
- 5. Check the *Edit with following Customization Context* option. You will see a list of customization layer name and customization layer values which were defined in the *CustomizationLayerValues.xml* file.
- 6. Select the Customization Context for which, the customizations you edit should be applicable.

### Figure 5–11 View Customization Context

View - Customization Context ×								
🔿 View without Customizat	View without Customizations							
<ul> <li>Edit with following Custo</li> </ul>	mization Context							
Tip layer	Name		Value					
0	option		demo (demo)	<b>_</b>				
			demo (demo)					
		2	Ubank (Ubank)					
Customization Context : option/demo								
Override global laver value	•							
System of Store and Store and Store	*							

All the customizations which are done to the application are now stored for the selected Customization Context.

# 5.7 Customization Layer Use Cases

### 5.7.1 Adding a UI Table Component to the Screen

This second example of customization, explains adding a table *UI Component*, which displays data to a screen.

**Use Case Description**: The Advanced Search screen is used to display the related accounts and their details for a party. The *Party -> On-Boarding -> Related Party* screen displays the related parties for a party. This section explains adding the table UI component used for displaying the related parties on the *Related Party* screen to the *Advanced Search* screen and populate data in this table on search and selection of a party.

P1040 P1030										
	Party ID 00	0005296				Full Name				
	First name					Last Name				
	Short Name					Email ID				
								S	earch Re	ese
Party Search Re	esults								Ck	ear
View - R Det	ach				2					
arty ID	Name	Туре	Number of Roles	Date of Birth or	arty Class	Email ID				
00005296	Daniel Johnson	Individu	2	05-Dec-1980	thers	dipika.patpaik@oraci	e.com			
Account Details				Account Specific	Details					
Account Details	s ccount Number Accou	unt Type		Count Specific	Details er 0000000000	007510	Account Title	Daniel Corp		
Account Details	s ccount Number Accou	unt Type		Count Specific Account Numb Account Openin Da	Details or 0000000000 g 15-Jan-2016 e	207510	Account Title Account Currency	Daniel Corp AUD		
Account Details	s ccount Number Accou	unt Type		Account Specific Account Numb Account Openin Da Party	Details er 0000000000 g 15-Jan-2016 e 000005297	207510	Account Title Account Currency Party Name	Daniel Corp AUD Daniel Corp		
Account Details erial Number At	s ccount Number Accou 0000000000751LON	unt Type		Account Specific Account Numb Account Openin Da Party Off	Details er 0000000000 g 15-Jan-2016 e D 00005297 er LOF003 NAB LOAN - LOF00	007510 TAILORED HOME	Account Title Account Currency Party Name Branch	Daniel Corp AUD Daniel Corp 082991 U Bank	Operations	8
Account Details rial Number A OC	s ccount Number Accos 0000000000751LON	unt Type		Account Specific Account Numb Account Openin Da Party Off Facility Cor	Details r 0000000000 g 15-Jan-2016 b D 000005297 r LOF003 NAB LOAN - LOF00 le FC201601500	207510 TAILORED HOME 33 318764	Account Title Account Currency Party Name Branch Facility Name	Daniel Corp AUD Daniel Corp 082991 U Bank Home Loan	Operations	8
Account Details	s ccount Number Accor 0000000000751LON	unt Type		Account Specific Account Numb Account Openii Da Party Om Facility Cor Total Disburss Amou	Details er 0000000000 ig 15-Jan-2016 e D 000005297 r LOF003 NAB LOAN - LOF00 FC201601500 d \$200,000.00 %	007510 TAILORED HOME 13 118764	Account Title Account Currency Party Name Branch Facility Name Last Disbursement Date	Daniel Corp AUD Daniel Corp 082991 U Bank Home Loan 31-Jan-2016	Operations	8
Account Details	s ccount Number Accou 0000000000751LON	unt Type		Account Specifi Account Numb Account Openi Da Party i Orr Facility Cor Total Disburs- Armou Date Of Maturi	Details r 000000000 g 15-Jan-2016 b D 00005297 r LOF00 NAB LOAN - LOF00 le FC201601500 d \$200,000.00 t t 15-Jan-2017	207510 TAILORED HOME 13 218764	Account Title Account Currency Party Name Branch Facility Name Last Disbursement Date Accrual Status	Daniel Corp AUD Daniel Corp 082991 U Bank Home Loan 31-Jan-2016 Normal	Operations	

Figure 5–12 Adding a UI Table Component - Party Search screen

Figure 5–13 Adding a UI Table Component - Related Party screen

			_								
PI	140	PI030									
Rel	ated Pa	arty									
Ģ	<u>R</u> ead	Crea	te 🥖 Update							🛷 ା <u>ଧ</u> 🎸 Clear	🗙 Exit 🚔 Print
	Primary	Party	Information								
	Party D	etails									
				Party ID	000005296		Date of	of Birth	05-Dec-1980		
	NO	)		Full Name	Daniel Johnson		(	Gender	Undisclosed	NO	
	INA	CE.		Home Branch	082991-U Bank 0	Operations BR		Roles	Customer	10.0.0	CE
	IMA	JE		Party Class	Others				<ul> <li>Director</li> </ul>	IMA	GE
	AVAILA			Party Type	IND		Onboardin	ig Date	15-Jan-2016	AVAILA	
	Address	e Data	le.								
	Addres	s Deta	lis .								
8	Relation	Detai	ls								
1	View 🗸	🛃 Det	tach								
Se	rial No.	_	Party Id	Related Party ID	Relationship Type	Direct Relation Name	Inverse Relation S Name	hare Col	llateral Share Exposure	Share Income	
1			000005295	000005296	Business	Authorized Signal	Authorized Signat				
II-									2		
									PG.		
C	olumns Hi	dden :	1								

To create the customization as mentioned in this use case, start JDeveloper in the *Default Role* and follow these steps:

### Step 1 Create Customization Project

- 1. As mentioned in the section **Customization Project**, create a project (*com.ofss.fc.demo.ui.view.party*) to hold the customization.
- 2. Add the required libraries and JARS along with JAR which contains the above screen (com.ofss.fc.ui.view.party.jar).
- 3. Enable the project for seeded customizations.

### Step 2 Create Binding Bean Class

You will need to create a class which will contain the binding for the *UI Components* which will be added to the screen during customization. Create the class with the following features:

- Private members for the UI Components and public accessors for the same.
- Private member for the backing bean of the screen (*PartySearchMaintenance*) which is initialized in the constructor of this class.
- Private member for the parent UI Component of the newly added UI components and public accessors which returns the corresponding component of the backing bean.





### Step 3 Create Event Consumer Class

You will need to create a class which contains the business logic for populating the table UI component with the related parties' data. The search and selection of a party in the *Advanced Search* screen raises an event. By binding this event consumer class to the party's selection event, the business logic for populating the related party's data will be executed automatically on selection of a party by the user.

The original event consumer class bound to this event contains the business logic for populating the accounts data. Since your event consumer class would be over-riding the original binding, you will need to incorporate the original business logic for populating the accounts data in your event consumer class.

#### Figure 5–15 Create Event Consumer Class



#### Step 4 Create Managed Bean

You will need to register the binding bean class as a managed bean. Open the project's adfc-config.xml which is present in the WEB-INF folder. In the Managed Beans tab, add the binding bean class as a managed bean with request scope as follows:

Figure 5–16	Creating	Managed	Bean
-------------	----------	---------	------

adfc-config.xml ×						38	
General Description	S Managed Beans						×
Activities Control Flows Managed Beans	Name *         Class *         Search Maintenance         Search Maintenance						
Metadata Resources	Managed Properties: Demo	PartySearchM	aintenance			4	X
	Name * 🛋		Class	Value			1.

### Step 5 Create Data Control

For the event consumer class's method to be exposed as an event handler, you will need to create a *data control* for this class.

- 1. In the Application Navigator, right-click the event consumer Java file and create data control.
- 2. On creation of data control, an XML file is generated for the class and a *DataControls.dcx* file is generated containing the information about the data controls present in the project. You will be able to see the event consumer data control in the *Data Controls* tab.



Figure 5–17 Create Data Control

- 3. Restart JDeveloper in the Customization Developer Role to edit the customizations.
- 4. Ensure that the appropriate Customization Context is selected.

### Step 6 Add View Object Binding to Page Definition

You will need to add the view object binding to the page definition of the screen. To open the page definition of the screen, follow these steps:

- 1. In the Application Navigator, open the Navigator Display Options for Projects tab and check the Show Libraries option.
- 2. In the navigator tree, locate the JAR that contains the screen (com.ofss.fc.ui.view.party.jar).
- 3. Inside this JAR, locate and open the page definition XML (*com.ofss.fc.ui.view.party.partySearch.pageDefn.PartySearchMaintenancePageDef.xml*)

4. After opening the page definition XML, add a tree binding for the view object (RelatedPartiesAndDetailsTableVO1) as follows:

Figure 5-1	8 Addina V	/iew Obiect	Bindina to	Page De	finition - Add	Tree Bindina

🛃 Party Search Main	tenancePageD	ef.xml ×				◄
Page Data Bindin	g Definition					^
This shows the Ora Data Binding Regi	icle ADF data b stry: <u>/com/ofs</u>	bindings defined for y	your page. Select a bin taBindings.cpx	ding to see its rela	ationship to the u	und
Bindings and Exe	cutables C	ontextual Events	Parameters			- 1
🗆 Model						-1
Bindings		+/%	Executables		+ / ×	
🔚 ha 🔚 ha 🔛 Ad	indleTaskCode indleCustTask indleAccountTa countDetailsV	eChangeEvent CodeEvent askCodeEvent iewObj1	variables taskFlow	- PartySearchTask etailsViewObj1ltera - accountDetailsTa	Flowl ator askFlowl	
	Incost Iton					
~	msert iten	"				
Sele	<u>c</u> t the categor	y of components fro	m which you would like	to find an item:		
Gen	eric Bindings			•		
Sele	ct the <u>i</u> tem to	be created:				
	graph			^		
	ist					
	IstOfValues					
	navigationlist					
	able			-		
	ree					
	reetable			M2 🚔		
Des	cription:					
Tre hier rela	e binding for rarchical list o ationships as	the control. Tree b of attributes derive specified by the b	pinding lets users vie ed from master-deta usiness services in v	wa il our Model		
	<u>H</u> elp		ОК	Cancel		

PartySearchMaintenancePageDef.xml X	
Page Data Binding Definition	^
This shows the Oracle ADF data bindings defined for your page. Select a binding to see its	relationship to the und
Data Binding Danistan, (see (sfee (fe/sit) inv/set (Data Bindings and	
Data binding Registry: //om/orss/rc/ul/view/party/DataBindings.cbx	
Bindings and Executables Contextual Events Parameters	
🗆 Model	
	0 2 9 2
Bindings P X Executables	• / *
handleTaskCodeEvent i variables i kaskFlow - PartySearch	TaskFlow1
handleAccountTaskCodeEvent 🛛 🙀 AccountDetailsViewObj	Literator
AccountDetailsViewObj1	ilsTaskFlow1
😣 Create Tree Binding	
display in the tree. To add additional tree level rules for child collections, select the	-
parent tree level rule andclick the Add icon. If no child collections are available for the selected node, the Add icon is disabled.	
Root Data Source: Root Data Control.RelatedParties  Add	
Tree Level Pules: PartyAppModuleDataControl PelatedPartiesApdDetailsTableV01	
com ofes for ui model party related party vo Related Parties And Details Table V	
com, orss.rc.u. model, party, relaced party, vo. Kelaced Parte Skilubetans rablev	
Learning Contract and Contract	
	×
Available Attributes: Display Attributes:	
UdfKey DeleteCheck	
DirectRelationshipName	n Context
InverseRelationshipName	ue
PartyType	mo (demo). 💌
ReadOnlyFlag	
Target Data Source	emo
<u>n</u> eip OK Cancel	

Figure 5–19 Adding View Object Binding to Page Definition - Update Root Data Source

5. In Root Data Source, locate the view object which is present in the *PartyAppModuleDataControl*. Select the required display attributes and click **OK**.

### Step 7 Add Method Action Binding to the Page Definition

You will need to add the method action binding for the event consumer data control to the page definition of the screen.

1. After opening the page definition XML, add the method action binding for the *DemoPartySearchConsumer* data control to the page definition as follows:

Figure 5–20 Page Data Binding Definition - Insert Item

PartySearchMaintenancePageDef.xml ×	•
Page Data Binding Definition	~
This shows the Oracle ADF data bindings defined for your page. Data Binding Registry: <a href="mailto://com/ofss/fc/ui/view/party/DataBindings">/com/ofss/fc/ui/view/party/DataBindings</a>	Select a binding to see its relationship to the und
Bindings and Executables Contextual Events Parameters	8
🖃 Model	
Bindings 🕂 🖊 🗶	Executables 🛛 🕂 🦉
<ul> <li>handleTaskCodeChangeEvent</li> <li>handleCustTaskCodeEvent</li> <li>handleAccountTaskCodeEvent</li> <li>AccountDetailsViewObj1</li> <li>RelatedPartiesAndDetailsTableVO1</li> </ul>	variables         taskFlow - PartySearchTaskFlow1         AccountDetailsViewObjlterator         taskFlow - accountDetailsTaskFlow1         RelatedPartiesAndDetailsTableVOlterator
😣 Insert Item	
Sele <u>c</u> t the category of components from which you woul	d like to find an item:
Generic Bindings	-
Select the item to be created:	
📅 graph 🛃 list 🛃 listOfValues	^
🖭 methodAction	
avigationlist 😽	
treetable	
Description:	¥
Method binding for the control	2
method binding for the control.	~
Help	OK Cancel

2. Browse and locate the data control and click **OK**.

ge Data Binding	Definition				
s shows the Oracle ata Binding Registr Bindings and Execu ⊒ <b>Model</b>	e ADF data bindings de y: <u>/com/ofss/fc/ui/viev</u> utables Contextual E	efined for your p w/party/DataBine Events Param	oage. Select a binding dings.cpx leters	to see its relat	ionship to the u
Bindings		- / 🗙	Executables		
i hand hand hand Acco Relat	lleTaskCodeChangeEve lleCustTaskCodeEvent lleAccountTaskCodeEve untDetailsViewObj1 tedPartiesAndDetailsTa	ent ent ableVO1	↓ variables ↓ taskFlow E Account ↓ taskFlow E RelatedP	s - PartySearchT DetailsViewObj1l - accountDetail artiesAndDetail	askFlow1 terator sTaskFlow1 sTableVO1iterat
Create Acti Select a data collect	on Binding tion and the action you the selected collection	u want your con 1.	trol to initiate. The co	ntrol initiates th	e action on
ata <u>C</u> ollection:		3			
DDBankerAc	countConfigurationEve	ntProducer			^
DDCloseAcco	ountEventProducer				
🕀 🖂 DeliverDocur	mentEventProducer				
DeliverDocur     DemoPartyS	mentEventProducer earchConsumer				
DeliverDocur     DemoPartyS     DepositBasic	mentEventProducer <mark>earchConsumer</mark> cDetailsEventProducer				
DeliverDocur     DemoPartyS     DepositBasid     DirectRolePa      DecumentCo	mentEventProducer earchConsumer cDetailsEventProducer intyDetailsConsumer				
BeliverDocur     DemoPartyS     DepositBasic     DirectRolePa     DocumentCa	mentEventProducer earchConsumer cDetailsEventProducer ntyDetailsConsumer ategoryEventConsumer	r			·
BeliverDocur     DemoPartyS     DepositBasic     DirectRolePa     DocumentCa     DocumentCa	mentEventProducer earchConsumer cDetailsEventProducer nrtyDetailsConsumer ategoryEventConsumer	r		•	<u>N</u> ew
BeliverDocur     DemoPartyS     DepositBasic     DirectRolePa     DocumentCa     DocumentCa	mentEventProducer earchConsumer cDetailsEventProducer intyDetailsConsumer ategoryEventConsumer	r		-	New
DeliverDocur     DemoPartyS     DepositBasic     DirectRolePa     DocumentCa     Select an Iterator:     Operation:	mentEventProducer earchConsumer cDetailsEventProducer ntyDetailsConsumer ategoryEventConsumer	r JeAndPartyRelat	ionshipEvent(Object) <b>1</b>	•	New
DeliverDocur     DemoPartyS     DepositBasia     DirectRolePa     DocumentCa     DocumentCa     Select an Iterator:     Operation:	earchConsumer cDetailsEventProducer intyDetailsConsumer ategoryEventConsumer andleAccountTaskCod	r JeAndPartyRelat in page defintic	ionshipEvent(Object) <b>T</b>	•	<u>N</u> ew
DeliverDocur     DemoPartyS     DepositBasi     DirectRolePa     DocumentCa     Select an Iterator:     Operation:     Parameters :	mentEventProducer earchConsumer cDetailsEventProducer intyDetailsConsumer ategoryEventConsumer handleAccountTaskCod	r JeAndPartyRelat in page defintio	ionshipEvent(Object) <b>v</b>	•	New
DeliverDocur     DemoPartyS     DepositBasic     DirectRolePa     DocumentCa     DocumentCa     Select an Iterator:     Operation:     Parameters :     Name	mentEventProducer earchConsumer cDetailsEventProducer intyDetailsConsumer ategoryEventConsumer mandleAccountTaskCod	leAndPartyRelat in page definito	ionshipEvent(Object) <b>v</b>	Option	<u>N</u> ew
DeliverDocur     DemoPartyS     DepositBasic     DirectRolePa     DocumentCa     DocumentCa     Select an Iterator:     Operation:     Parameters :     Name     object	mentEventProducer earchConsumer cDetailsEventProducer intyDetailsConsumer ategoryEventConsumer andleAccountTaskCod Apply to all iterators Type java.lang.Object	r JeAndPartyRelat in page defintic Value	ionshipEvent(Object) <b>v</b>	Option	New
DeliverDocur     DemoPartyS     DepositBasi     DirectRolePa     DocumentCa     DocumentCa     DocumentCa     DepositBasi     DocumentCa     DocumentCa	mentEventProducer earchConsumer cDetailsEventProducer intyDetailsConsumer stegoryEventConsumer mandleAccountTaskCod Apply to all iterators Type java.lang.Object	r JeAndPartyRelat in page defintio Value	ionshipEvent(Object) <b>v</b>	Option	<u>N</u> ew
DeliverDocur     DemoPartyS     DepositBasi     DirectRolePa     DocumentCa     DocumentCa     Select an Iterator:     Operation:     Parameters :     Name     object	mentEventProducer earchConsumer cDetailsEventProducer intyDetailsConsumer ategoryEventConsumer andleAccountTaskCod Apply to all iterators Type java.lang.Object	r leAndPartyRelat in page defintio Value	ionshipEvent(Object) <b>v</b>	Option	<u>N</u> ew
DeliverDocur     DemoPartyS     DepositBasi     DirectRolePa     DocumentCa     Select an Iterator:     Operation:     Parameters :     Name     object	mentEventProducer earchConsumer cDetailsEventProducer intyDetailsConsumer ategoryEventConsumer mandleAccountTaskCod Apply to all iterators Type java.lang.Object	in page definition	ionshipEvent(Object) <b>v</b>	Option	<u>N</u> ew
DeliverDocur     DemoPartyS     DepositBasic     DirectRolePa     DocumentCa     DocumentCa     DepositBasic     DocumentCa     Document	mentEventProducer earchConsumer cDetailsEventProducer intyDetailsConsumer ategoryEventConsumer mandleAccountTaskCod Apply to all iterators Type java.lang.Object	IeAndPartyRelat	ionshipEvent(Object) <b>v</b>	Option	New

Figure 5–21 Page Data Binding Definition - Create Action Binding

### Step 8 Edit Event Map

You will need to map the *Event Producer* for the party selection event to the **Event Consumer** defined by you in the page definition.

- 1. In the Application Navigator, select the page definition XML file.
- 2. In the *Structure panel* of JDeveloper, right-click the page definition XML and select *Edit Event Map*.

### Figure 5–22 Edit Event Map



- 3. In the Event Map Editor panel, edit the mapping for the required event.
- 4. Select the newly added Event Consumer's method.

#### Figure 5–23 Event Map Editor

Event Map Editor	r			
Add, Update and Delete	event entries			
Events Map				🕂 🖊 🔶
Producer PartySearchTaskFlow1.P PartySearchTaskFlow1.P PartySearchTaskFlow1.P	Event Na arty handleTa arty handleCu arty handleAd	me askCodeChang ustTaskCodeE countTaskCod	Consumer handleTaskCode handleCustTask handleAccountTa	ChangeEvent CodeEvent askCodeEvent
😣 Modify EventMa	o Entry		_	
Select an appropriate P <u>P</u> roducer:	roducer, Event	Name and Cons	umer.	AccountEvent
Event Name:	handleAccour	+TaskCodeEvent		
<u>C</u> onsumer:	PageDef.hand	leAccountTaskC	odeAndPartyRelat	ionshipEvent 🔻
Consumer Params			<b>+</b> ×	
Param Name	Pa	ram Value		
payLoad	#	(payLoad}		
				\$
		(	ОК	Cancel

### Step 9 Add UI Components to Screen

After making the required changes to page definition of the screen, you will need to add the UI components to the screen JSFF. After opening the JSFF for the screen

(com.ofss.fc.ui.view.party.partySearch.PartySearchMaintenance.jsff), follow these steps:

- 1. Drag and drop the *Panel Box*, *Panel Collection* and *Table* components onto the screen.
- 2. Set the required columns for the *Table* component.
- 3. Drag and drop the *Output Text* or *Check Box* components as required inside the columns.
- 4. For each component, set the required attributes using the *Property Inspector* panel of JDeveloper.
- 5. Add the binding for required components to the binding bean members.
- 6. Add the view object binding to the *Table* component.
- 7. Save changes made to the JSFF.

Figure 5–24 Add UI Components to Screen

PartySea	archMaintenance.jsff 🗵 👻
💏 - Find	4 (c)
	<pre><af:activecommandtoolbarbutton <="" binding="#{PartySearchMaintenance.cbl}" id="cbl" pre="" text="#{rbPartySearchMaintenance.LBL_Exit}"></af:activecommandtoolbarbutton></pre>
	/af:panelGroupLayout
	<af:region binding="#{PartySearchMaintenance.r1}" id="r1" value="#{bindings.PartySearchTaskFlov1.regionModel}"></af:region>
	<af:panelbox id="olpb1" t<="" td="" text="#{rbRelatedParty.LBL_RELATION_DETAILS_PANEL}" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:panelbox>
8	<af:panelcollection 10="olpcl" <="" adf="" binding="#{DemoPartySearchMaintenanceHelper&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;8&lt;/td&gt;&lt;td&gt;&lt;af:table xmins:af=" faces="" http:="" rich"="" td="" value="#jbindings.RelatedPartiesAndDetailsTableVOL.collection" xmins.oracle.com="" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:panelcollection>
8	<ar:column headerlext="#ifbreiatedParty.SexLaL_NUME&lt;/td" sortable="faise" xmins:at="http://xmins.oracle.com/ad/Taces/rich"></ar:column>
	<arbourput 10="01015" text="" value="#trow.seriathumbery=" xmthstat="http://xmths.oracte.com/adi/taces/rich"></arbourput>
-	
	<pre>satecodemi Andre at http://white.org/is costing costing acting acti</pre>
8	<af:column adf="" faces="" headertext="#frbRelatedParty.LBL RELATED&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;af:outputText xmlns:af=" http:="" id="olot4" rich"="" sortable="false" value="#[row.RelatedPartyId}" xmlns.oracle.com="" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:column>
8	<af:column adf="" faces="" headertext="#{rbRelatedParty.LBL_RELATIQ&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;af:outputText xmlns:af=" http:="" id="olot3" rich"="" sortable="false" value="#{row.RelationshipType}" xmlns.oracle.com="" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:column>
8	<af:column =<="" adf="" faces="" headertext="#{rbRelatedParty.LBL_DIRECT_&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;af:outputText xmlns:af=" http:="" id="olot6" rich"="" sortable="false" td="" value="#[row.DirectRelationshipName]" xmlns.oracle.com="" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:column>
	<arccount faces="" fich="" nederrext="#truetatedparty.bd_inverse&lt;/td" nttp:="" out="" sortable="faces" wturs.oracte.com="" xturstate=""></arccount>
	<pre></pre>
8	<pre><af:column ;<="" headertext="Share Collateral" id="olc6" pre="" sortable="false" value:af="http://value.oracle.com/adf/faces/rich"></af:column></pre>
	<af:selectbooleancheckbox adf="" faces="" headertext="Share Exposure" http:="" id="olc3" label="#(rbRelatedParty.Lbl sh&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/af;colum&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;8&lt;/td&gt;&lt;td&gt;&lt;af:column xmlns:af=" rich"="" sortable="false" text=" " xmlns.oracle.com="" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:selectbooleancheckbox>
	<af:selectbooleancheckbox adf="" faces="" headertext="Share Income" http:="" id="olc4" label="#{rbRelatedParty.Lbl_sh&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/af:colum&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;8&lt;/td&gt;&lt;td&gt;&lt;af:column xmlns:af=" rich"="" sortable="false" text=" " xmlns.oracle.com="" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:selectbooleancheckbox>
	<af:selectbooleancheckbox #{partysparthmaintenance_on]3}"="" binding="sfartySarchAsintenance.pcl" id="on]3" idinsetula="heink+ 340v" label="#{rbRelatedParty.Lbl_sh&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/articolum&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/article/actions/acti&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/artipanetCorrection&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;pre&gt;vaf.pametowar vaf.pametowar vaf.pametowar interimetation=" layout="serval1" pre="" sol"="" solitterposition="400" style"<="" style"lass="AEStretzbuil &lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;8&lt;/td&gt;&lt;td&gt;&lt;pre&gt;cafinanelSolitter id=" text=" " visible="false" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:selectbooleancheckbox>
E.	:facet name="first"
isp:root ->	af panelgrouplayout#pg[] +> af:panelbox#olpb1 +>
Design Sou	rce Bindings Preview (
200	

After saving all these changes, you will notice that JDeveloper has created a customization XML for each of the customized entities in the *ADF Library Customizations Sources* folder packaged as per the corresponding base document's package and customization context (*Customization Layer Name & Customization Layer Value*). These XML's store the difference between the base and customized entity. In our customization, you can see the following generated XML's:

- PartySearchMaintenancePageDef.xml for the page definition customizations.
- DataBindings.cpx.xml for the data binding (view object binding) customizations.
- PartySearchMaintenance.jsff.xml for the UI customization to the screen JSFF.





### Step 10 Deploy Customization Project

After finishing the customization changes, exit the *Customization Developer Role* and start JDeveloper in *Default Role*. Deploy the customization project as an ADF Library JAR (*com.ofss.fc.demo.ui.view.party.jar*).

- 1. Go to the **Project Properties** of the main application project and in the *Libraries* and *Classpath*, add the following JARS:
  - Customization Project JAR (com.ofss.fc.demo.ui.view.party.jar)
  - Customization Class JAR (com.ofss.fc.demo.ui.OptionCC.jar)
  - All dependency libraries and JARS for the project.
- 2. Start the application and navigate to the Advanced Search screen.
- 3. Search for a party ID and select a party from the Party Search Results table.
- 4. On selection of a party, the *Relation Details* panel containing the related party's data is displayed.

### Figure 5–26 Party Search

	P1040													
Search Indiv	idual													
	Party	ID 00000	5296						Full Nam	e				
	First nar	ne							Last Nam	e				
	Short Nat	ne							Email I	D				
													Search	Rese
Party Search	Results													Clear
View - 🛃	Detach													
Party ID	Name		Туре	Number of Ro	les	Date of Birth or Incorporation	Party Cla	lass	Email ID					
000005296	Daniel J	ohnson	Individu	2		05-Dec-1980	Others		dipika.patnaik@ora	acle.com				
B Relation Det	ails				_									
View - 🔄 Detach														
						Direct Belation	Inverse	a Delation						
Serial No.	Party Id	Related	Party ID	Relationship	Тур	Direct Relation Name	Inverse Name	e Relation	Share Collateral	Share Exposure	Share	Income		
Serial No.	Party Id 000005295	Related	Party ID 296	Relationship Business	Тур	Direct Relation Name Authorized Signa	Inverse Name t Authoriz	e Relation ized Signat	Share Collateral	Share Exposure	Share	Income		
Serial No.	Party Id 000005295	Related	Party ID 296	Relationship Business	Тур	Direct Relation Name Authorized Signa	Inverse Name t Authoria	e Relation ized Signat	Share Collateral	Share Exposure	Share	Income		
Serial No.	Party Id 000005295	Related	Party ID 296	Relationship Business	Тур	Direct Relation Name Authorized Signa	Inverse Name t Authoriz	e Relation	Share Collateral	Share Exposure	Share	Income		
Serial No. 1	Party Id 000005295	Related	Party ID 296	Relationship Business	Typ	Direct Relation Name Authorized Signa	Inverse Name t Authoriz	e Relation ized Signat	Share Collateral	Share Exposure	Share	Income		
Serial No. 1 Account Det Serial Number	Party Id 000005295 ails Account Number	Related 000005	Party ID 296 Type	Relationship Business	Type	Direct Relation Name Authorized Signa Account Specifi Account Num	Inverse Name t Authoriz	e Relation ized Signat	Share Collateral	Share Exposure	Share	Income Daniel Corp		
Serial No. 1 Account Det Serial Number	Party Id 000005295 alls Account Number 000000000007	Related 0000053	Party ID 296 Type	Relationship Business	Тур	Direct Relation     Name     Authorized Signa     Authorized Signa     Account Specif     Account Qper     Account Oper	Inverse Name t Authoriz ic Details ber 0000 ing 15-3	e Relation ized Signat Is 0000000000 Jan-2016	Share Collateral	Share Exposure	Share Title D	Daniel Corp		
Serial No. 1 Account Det Serial Number	Party Id 000005295	Account 51LON	Party ID 296	Relationship Business	Typ	C Direct Relation Name Authorized Signs Authorized Signs Account Specifi Account Open C C C C C C C C C C C C C C C C C C C	Inverse Name t Authoriz ic Details ber 0000 ing 15-J ate	e Relation ized Signat Is 0000000000 Jan-2016 0005297	Share Collateral	Share Exposure Account Account Party No	Share Title D ency A	Daniel Corp AUD Daniel Corp		
Serial No. 1 Account Det Serial Number	Party Id 000005295 alls Account Number 000000000007	Related 0000053	Party ID 296 Type	Relationship Business	Typ	Direct Relation     Name     Authorized Signa     Account Specifi     Account Nam     Account Nam     Account Ope     Party     O	Inverse Name t Authonia ic Details ber 0000 ing 15-3 ate ID 0000 ffer LORI	e Relation ized Signat is 0000000000 Jan-2016 0005297 F003 NAB 1 NN - LOF00	Share Collateral O7510 TAILORED HOME 3	Share Exposure Account Account Party Na Bra	Title D ency A ame D onch 0	Daniel Corp AUD Daniel Corp 282991 U 8	lank Oper	ations BR
Serial No. Account Det Serial Number I	Party Id 000005295 alls Account Number 00000000007	Related 000005	Party ID 296 Type	Relationship Business	Typ	Direct Relation Name Authorized Signal Account Specific Account Nam Account Oper C Party O Facility C	Inverse Name t Authoriti t Authoriti t Authoriti ing 15-3 ate 1D 0000 ffer LOFF LOAN ade FC20	e Relation ized Signat bootootootoot Jan-2016 boot5297 F003 NAB 1 NN - LOF002 2016015002	Share Collateral 07510 TAILORED HOME 3 18764	Share Exposure Account Account Account Braty N Bra Facility N	Title D ncy A ame D nch 0	Daniel Corp NUD Daniel Corp 182991 U B Home Loan	ank Oper	ations BR
Serial No. 1 Account Det Serial Number	Party Id 000005295 alls Account Number 00000000007	Related 000005	Party ID 296 Type	Relationship Business	Typ	Direct Relation Name Authorized Signal Account Specific Account Num Account Num Account Num Count Oper C Directility C Total Disbur	Inverse Name t Authorit t Authorit Authorit t Authorit t Authorit t Authorit t Authorit	e Relation Ized Signat 0000000000 Jan-2016 0005297 F003 NAB 1 NN - LOF00 201601500 00,000.00	Share Collateral 07510 TAILLORED HOME 3 18764	Share Exposure Account Account Account Curre Party N Bra Facility N Last Disbursen	Title D mcy A ame D mch 0 ame H nent 3 Date	Daniel Corp NUD Daniel Corp 182991 U B Home Loan 81-Jan-2016	lank Oper	ations BR
Serial No.	Party Id 000005295 alls Account Number 00000000007	Related 000005	Party ID 296 Type	Relationship Business	Typ	Direct Relation Name Authorized Signal Account Specifi Account Nam Account Oper C Party O Facility C Total Disbur Date Of Matu	Inverse Name t Authoriti t Authoriti t Authoriti t Authoriti t C Detailin ber 0000 ing 15-3 ate 1D 0000 ffer LORI LOAN ode FC20 sed \$200 unt rity 15-3	e Relation ized Signat 5 0000000000 Jan-2016 0005297 F003 NAB 1 NN - LOF003 201601500 00,000.00 -Jan-2017	Share Collateral 07510 TAILORED HOME 3 18764	Share Exposure Account Account Account Curre Party N Bra Facility N Last Disburser Accrual St	Title D ancy A ame D anch 0 ame H hent 3 Date atus N	Income Daniel Corp AUD Daniel Corp 82991 U 8 4ome Loan 81-Jan-2010 iormal	ank Oper	ations BR
Serial No. 1 Account Det Serial Number 1	Party Id 000005295	Related 000005	Party ID 296 Type	Relationship Business	Typ	Direct Relation     Name     Authorized Signa     Authorized Signa     Account Specifi     Account Num     Account Num     Account Num     C     Partin     O     Facility C     Total Disbur     Ame     Date Of Matu     Approved Ame	Inverse Name t Authonia ic Details ber 0000 ing 15-3 ate ID 0000 ffer LOFf LOAT ode FC20 out rity 15-3 unt \$200	e Relation ized Signat b000000000 Jan-2016 b005297 F003 NAB 1 NN - LOF003 201601500 00,000.00 -Jan-2017 b0,000.00	Share Collateral 07510 TAILORED HOME 3 18764	Share Exposure Account Account Account Curre Party Ni Bra Facility Ni Last Disburser C Accrual St Next Instalin Am	Title D ancy A ame D anch 0 ame H anent 3 Date atus N hent \$ ount	Daniel Corp AUD Daniel Corp AUD Daniel Corp 82991 U E 40me Loan 81-Jan-2014 iormal 60.00	ianik Oper	ations BR

## 5.7.2 Approvals Framework

It is recommended to use ADF screen extensions for UI changes instead of mds in this scenario as it is easier to upgrade to new version of product however the mds approach is described below.

This third example of customization explains adding a Date Component to an existing screen to capture date input from the input. This input is saved in the database.

**Use Case Description:** The Party  $\rightarrow$  Contact Information  $\rightarrow$  Contact Point screen is used to store the various contact point details for a party. In the Contact Point Details tab, the user can select a Contact Point Type and a Contact Preference Type and provide details for the same. User will be adding a field Expiry Date as a date component to this tab. User will be adding a table to the database to save the user input for this field and services for this screen will be added or modified.

#### Figure 5–27 Contact Point Screen

PI <u>0</u> 41	
Contact Point	
🛄 Read 💠 Create 🥖 Upda	te 🔗 Ok 🥔 Clear 🕱 Exit 🚊 Print
Party Details	
NO IMAGE AVAILABLE	Party ID 000005295     Date of Incorporation  mpany Name Daniel trustee Party Class FOREIGN PUBLIC BODY Party Type LEG     Onboarding Date 15-Jan-2016
Contact Point Details	
<ul> <li>Contact Point Type</li> </ul>	Mobile Contact Preference Type Home
Seasonal Start Date Allowed Purposes	Seasonal End Date Communication Alert
Preferred Contact	Preferred Contact
Marketing Consent	Marketing Consent
Marketing Consent Start Date	Marketing Consent End Date
Telephone Details	
Country Code	Area Code
Number	32577789 Extension
Service Provider	VOIP Code
Timing Preferences	
DND Shart	
Weekdays	Weekdays
From	То
Weekends	U Weekends
From	То
Hide Modification History	
Created By ofssuser	On 24-Aug-2012 12:00:00 AM Approved V & 2 OF ×
Approved By ofssuser	On 24-Aug-2012 12:00:00 AM Active

To create the customization as mentioned in this use case, follow these steps:

### Step 1 Host Application Changes

Since in this use case you need to save the input data in the database of the application, you need to do certain modifications on the host application before creating the customizations on the client application. Following are the changes that need to be done to the host application.

### Step 2 Create Table in Application Database

To save the input data for the Expiry Date field, create a table in the application database. The table will also need to have the Key columns for this field and the columns needed to store information about the record. Create appropriate primary and foreign keys for the table as well.

### Figure 5–28 Create Table

0	REATE TABLE "FLX_PI_CONTACT_	EXPIRY*		
ľ	"PARTY_ID"	VARCHAR2(40 BYTE) NOT NULL ENABLE,	Key Colu	mns
	"CONTACT_POINT_TYPE" "EXPIRY_DATE"	VARCHARZ(3 BYTE) NOT NULL ENABLE, VARCHAR2(4 BYTE) NOT NULL ENABLE, DATE,	Expiry	Date Field
	"CREATED BY" "CREATION DATE" "LAST UPDATED BY"	VARCHAR2(254 BYTE) NOT NULL ENABLE, TIMESTAMP (6) NOT NULL ENABLE, VARCHAR2(254 BYTE) NOT NULL ENABLE	Reco	ord Information Columns
	"LAST_UPDATE_DATE" "OBJECT_VERSION_NUMBER"	TIMESTAMP (6) NOT NULL ENABLE, NUMBER(9,0) NOT NULL ENABLE,		
	CONSTRAINT "ELY PL CONTACT	CHAR(I BYTE) NUT NULL ENABLE,	ONTACT POINT	TYPE" "CONTACT PREE TYPE") ENABLE
)	CONSTRAINT "FLX_PI_CONTACT_	EXPIRY_FK1" FOREIGN KEY ("PARTY_ID") I	REFERENCES FL	X_PI_PARTIES_B" ("PARTY_ID") ENABLE

After creating the table, you need to create the domain object and service layers. To create these entities, follow these steps.

### Step 3 Create Java Project

To contain the domain object and service layer classes, create a Java Project in eclipse. Give a title to the project (com.ofss.fc.demo.party.contactexpiry) and add the required projects to the classpath of the project.

Properties for com.o	fss.fc.demo.party.contactexpiry	
type filter text 🛛	Java Build Path	(⊃v ⊂)v <b>v</b>
<ul> <li>Resource Builders</li> </ul>	Source Projects Libraries Order and Export	
File Transfer	Required projects on the build path:	
Formatter	com.ofss.fc.app.service.json	Add
Java Build Path	Com.ofss.fc.app.service.vo	
Java Code Style	com.ofss.fc.app.xface	Edit
Java Compiler	Com.ofss.fc.appcore	Remove
Java Editor	Com.ofss.fc.appcore.dto	Remove
Javadoc Location	Com.ofss.fc.appx.client.proxy	
Preview Settings	com.ofss.fc.appx.json.client	
Project Facets	com.ofss.fc.appx.service.json	
Project Natures	com.ofss.fc.appx.service.vo	
Project References	Com.ofss.fc.appx.spi	
Run/Debug Settings	Com.ofss.fc.common	
Task Repository	com.ofss.fc.datatype	
Task Tags	com.ofss.fc.enumeration	
Validation	com.ofss.fc.framework.domain	
WikiText	Com.ofss.fc.framework.dto	
	Com.ofss.fc.infra	
	com.ofss.fc.module.party	
2		
	Carro	
	Cance	ON

Figure 5–29 Create Java Project

### Step 4 Create Domain Objects

You need to create the domain objects for the newly added table. As per the structure and package conventions of OBP, create the domain objects as follows:
Create class (com.ofss.fc.demo.domain.party.entity.contact.ContactExpiryKey) for the key columns
of the table. This class must extend the com.ofss.fc.framework.domain.AbstractDomainObject
abstract class. Add the properties, getters and setters for the key columns of the table in this class.
Implement the abstract methods of the superclass.

Figure 5–30 Create Domain Objects



2. Create interface (com.ofss.fc.demo.domain.party.entity.contact.IContactExpiry) for the domain object class with getters and setters abstract methods for the Key domain object and the field Expiry Date. This interface must extend the interface com.ofss.fc.framework.domain.AbstractDomainObject.

Figure 5–31 Create Interface



 Create class (com.ofss.fc.demo.domain.party.entity.contact.ContactExpiry) for the domain object. This class must implement the previously created interface and extend com.ofss.fc.framework.domain.AbstractDomainObject abstract class. Add the properties, getters and setters for Key object and Expiry Date field. Implement the abstract methods of the superclass.

### Figure 5–32 Create Class



After creating the domain objects, build the project. You need to use the Flex cube development eclipse plug-in to generate the service layers.

# Step 5 Set OBP Plugin Preferences

Before using the plug-in for generating service layer classes, you will need to set the required preferences for the plug-in. In eclipse, go to Windows  $\rightarrow$  Preferences  $\rightarrow$  OBP Development and the set the preferences as follows.

Preferences				
type filter text	Service Publisher			¢• ⇔• •
<ul> <li>▶ General</li> <li>▶ Ant</li> </ul>	Output project location	Client		<u>C</u> hange
Aptana Studio	Service URL	http://localhost:8080		
AspectJ Compiler	Path of Dynamic Property file:	/home/rshanbha/Work/eclipse workspace	s/ngp/config	Browse
Data Management				
<ul> <li>FLEXCUBE Development</li> </ul>	Path of User Home	/home/rshanbha/Work/eclipse_workspace	s/ngp/config	Browse
GEFU				
JUnit				
. Procedure Wrapper			N	
Reverse Engineering			13	
RMI				
Service Deployer				
Service Publisher				
WorkSpace Path				
XML/JSON Facade				
▶ Install/Undate				
' ► Java			Restore Defaults	Apply
?			Cancel	ОК
1	201 H			

8 Preferences		
type filter text 🛛 🗷	WorkSpace Path	⇔• ⇒• •
<ul> <li>General</li> <li>Ant</li> <li>Aptana Studio AspectJ Compiler</li> <li>Data Management</li> <li>FLEXCUBE Development GEFU GEFX JUnit</li> <li>Procedure Wrapper Reverse Engineering RMI Service Deployer Service Publisher</li> <li>WorkSpace Path XML/JSON Facade</li> <li>Help</li> <li>Install/Update</li> </ul>	Path of the JavaProjects:       /home/rshanbha/Work/eclipse_workspaces/ngp	<u>B</u> rowse
▶ Java	Restore <u>D</u> efaults	Apply
0	Cancel	ок



8 Preferences			
type filter text	XML/JSON Facade		¢+ ⇒+ +
► General ► Ant	Path of the Facade Library folder in your system:	/home/rshanbha/Work/eclipse_workspaces/generator	Browse
Aptana Studio	Sample XML Output Path:	/home/rshanbha/Work/eclipse_workspaces/generator/xml	Browse
AspectJ Compiler     Data Management	Facade XSD Output Path:	/home/rshanbha/Work/eclipse_workspaces/generator/xsd	Browse
FLEXCUBE Development CEEU	List of XSD included		
GEFX			Remove
JUnit Procedure Wrapper			Up
Reverse Engineering			Down
Service Deployer			
Service Publisher WorkSpace Path			
XML/JSON Facade		A	
Install/Update		<b>N</b> 0	
<ul> <li>Java</li> <li>Java EE</li> </ul>			
Java Persistence		Restore <u>D</u> efaults	Apply
(?)		Cancel	ОК

## Step 6 Create Application Service

You need to generate the application service layer classes using the OBP development plugin. Follow these steps:

- 1. Open the domain object class (ContactExpiry).
- 2. On the getter method of the Key object, add a javadoc comment @PK.
- 3. Right click on the editor window and from context menu that opens, choose OBP Development → Generate Application Service.
- 4. In the dialog that opens, select the Java project for generated classes. You can use the project previously created by you.

## Figure 5–36 Create Application Service

😣 🖲 💿 Application Ser	rvice Generator
Choose APP Project and APP project location	package name com.ofss.fc.demo.party.contactexpir Change
Enter the App package:	com.ofss.fc.demo.app.party.contact
Generate	

5. Click on Generate. Application Service classes is generated in the project.

The Java source might contain some compilation errors due to syntax. Fix these errors and build the project. The following classes should have been generated in the project.

Figure 5–37 Application Service Classes Generated

#* com.ofss.fc.demo.party.contactexpiry [master]
▼ 🚰 src
com.ofss.fc.demo.app.party.assembler.contact
ID ContactExpiryAssembler.java
com.ofss.fc.demo.app.party.dto.contact
ID ContactExpiryDTO.java
ContactExpiryInquiryResponse.java
ID ContactExpiryKeyDTO.java
🔻 🔠 com.ofss.fc.demo.app.party.service.contact
ID ContactExpiryApplicationService.java
IContactExpiryApplicationService.java
🔻 🔠 com.ofss.fc.demo.app.party.service.contact.ext
ID ContactExpiryApplicationServiceExtExecutor.java
IContactExpiryApplicationServiceExt.java
IContactExpiryApplicationServiceExtExecutor.java
VoidContactExpiryApplicationServiceExt.java
🔻 🔠 com.ofss.fc.demo.domain.party.entity.contact
ContactExpiry.java
ContactExpiryFactory.java
ContactExpiryKey.java
IContactExpiry.java
com.ofss.fc.demo.domain.party.entity.contact.repository
ContactExpiryRepository.java
🔻 🔠 com.ofss.fc.demo.domain.party.service.contact
ContactExpiryService.java

# Step 7 Generate Service and Facade Layer Sources

Before generating the service and facade layer sources, you need to modify the Data Transfer Object (DTO). When a service call is made from the client application for a transaction related to Contact Point, the Contact Expiry transaction for the newly added Expiry Date field should be done in addition to the Contact Point transaction. Hence, the DTO for this transaction should also contain the DTO for the Contact Point transaction.

To modify the Data Transfer Object:

- 1. Open the ContactExpiryDTO class.
- 2. Delete the member ContactExpiryKey member and add ContactPoint member.
- 3. Re-factor references of the deleted member with the added member.

Figure 5–38 Modify Data Transfer Object (DTO)



To generate the service and facade layer sources:

- 1. Open the application service class (ContactExpiryApplicationService).
- 2. Right click on the editor window and from the context menu that opens, choose OBP Development → Generate Service and Facade Layer Sources.
- 3. In the dialog box that opens, select the Java project for the generated classes. You can use the project previously created by you. Un-check the Overwrite Existing Files option.

Figure 5–39 Generate Service and Facade Layer Sources

Senerate Sources
Generate Layer Sources
Choose the wsdl output location
<ul> <li>Com.ofss.fc.demo.module.loan [branches/DEMO/middleware/projects/module/code</li> <li>Com.ofss.fc.demo.module.party [branches/DEMO/middleware/projects/module/code</li> <li>Com.ofss.fc.demo.module.pc [branches/DEMO/middleware/projects/module/come</li> <li>Com.ofss.fc.demo.module.pm [branches/DEMO/middleware/projects/module/cone</li> <li>Com.ofss.fc.demo.party.contactexpiry [master]</li> <li>Com.ofss.fc.fact [trunk/core/middleware/projects/module/com.ofss.fc.fact]</li> <li>Com.ofss.fc.fact [trunk/core/middleware/projects/module/com.ofss.fc.fact]</li> <li>Com.ofss.fc.framework.batch [trunk/core/middleware/projects/framework/com.ofss.fc.framework.domain [trunk/core/middleware/projects/framework/com.ofss.fc.framework/com.ofss.fc.framework/com.ofss.fc.framework/com.ofss.fc.framework.domain [trunk/core/middleware/projects/framework/com.ofss.fc.framework/com.ofss.fc.framework.domain [trunk/core/middleware/projects/framework/com.ofss.fc</li></ul>
Cancel Finish

4. Click Finish.

Service and facade layer sources is generated in the project.

- 5. Certain classes might be generated twice. Delete the newly created copy of the classes and keep the original.
- 6. Certain compilation errors might be present in the generated classes due to erroneous syntax. Fix these compilation errors.
- 7. You will need to include a corresponding call to the Contact Point Application Service in the add, update and fetch transactions of the Contact Expiry Application Service.
- 8. Open ContactExpiryApplicationServiceSpi and modify the code as shown below.





Figure 5–41 Modify ContactExpiryApplicationServiceSpi.java







9. The project should contain the Java packages as shown below:





## Step 8 Export Project as a JAR

You need to export the Java project containing the domain object, application service and facade layer source as a JAR.

To export java project as JAR:

- 1. Right click on the project and choose Export.
- 2. Choose JAR File in the export options.
- 3. Provide an export path and name (com.ofss.fc.demo.party.contactexpiry.jar) for the JAR file.
- 4. Click Finish.

Figure 5-44 Export Java Project as JAR

8 JAR Export
JAR File Specification
Define which resources should be exported into the JAR.
Select the resources to export: Select the resources to export: Select the resources to export. Select the resources to export generated class files and resources Export all output folders for checked projects Export Java gource files and resources Export refactorings for checked projects. Select refactorings
JAR file: ce/UISetup/fcr.host.domain/WEB-INF/lib/com.ofss.fc.demo.party.contactexpiry.jar 👻 Browse
Options: ✓ Compress the contents of the JAR file □ Add directory entries □ Qverwrite existing files without warning
? < Back Next > Cancel Finish

### Step 9 Create Hibernate Mapping

You need to create a hibernate mapping to map the database table to the domain object.

Follow these steps:

- 1. Create ContactExpiry.hbm.xml file in the orm/hibernate/hbm folder of the config project of the host application.
- 2. Add the entry for this XML in the orm/hibernate/cfg/party-mapping.cfg.xml hibernate configuration XML.
- 3. Add the mapping in ContactExpiry.hbm.xml as shown below.

Figure 5–45 Create ContactExpiry.hbm.xml



## Step 10 Configure Host Application Project

You need to configure the Contact Expiry Application Service and Facade Layer in the host application.

To configure, follow these steps:

- 1. Configure APPX layer as the service layer for Contact Expiry service.
- 2. Open properties/hostapplicationlayer.properties present in the configuration project and add an entry as shown below.





3. Configure APPX layer proxy as the proxy for Contact Expiry service.

4. Open properties/ProxyFacadeConfig.properties present in the configuration project and add an entry as shown below.



lostapplicationlayer.properties	ProxyFacadeConfig.properties 🛛 📄 JSONServiceMap.properties
184 PartyOnBoardingApplicationS 185 IdentificationApplicationSe 186 PartyRelationshipApplicatio	<pre>erviceProxy=com.ofss.fc.app.party.service.core.service.client.proxy.Pa rviceProxy=com.ofss.fc.appx.party.service.identity.service.client.prox nServiceProxy=com.ofss.fc.app.party.service.relation.service.client.pr</pre>
187 ContactPreferenceApplicatio	<pre>nServiceProxy=com.ofss.fc.app.party.service.contact.service.client.pro</pre>
188 ContactPointApplicationServ	iceProxy=com.ofss.fc.appx.party.service.contact.service.client.proxy.C
189 ContactExpiryApplicationSer	<pre>viceProxy=com.ofss.fc.demo.appx.party.service.contact.service.client.</pre>
190 GroupApplicationServiceProx	y=com.ofss.fc.app.party.service.global.service.client.proxy.GroupAppli
191 PartyDemographicsApplicatio	<pre>nServiceProxy=com.ofss.fc.appx.party.service.core.service.client.proxy</pre>
192 DemoPartyDemographicsApplic	ationServiceProxy=com.ofss.fc.demo.appx.party.service.theftProtection.

- 5. Configure the JSON and Facade layer mapping for Contact Expiry service.
- 6. Open properties/JSONServiceMap.properties present in the configuration project and add the two entries as shown below.

### Figure 5–48 Configure JSONServiceMap.properties

hostapplicationlayer.properties	ProxyFacadeConfig.properties	D JSONServiceMap.properties
1086 PartyAddressApplicationServ	<pre>viceSpiV0FacadeService = com.ofs;</pre>	s.fc.appx.party.service.core.vo.service.json.Part
1087 PartyNameApplicationService	SpiV0FacadeService = com.ofss.fr	c.appx.party.service.identity.vo.service.json.Par
1088 IdentificationApplicationSer	erviceSpiV0FacadeService = com.or	fss.fc.appx.party.service.identity.vo.service.jsc
1089 ContactPointApplicationServ	viceSpiV0FacadeService=com.ofss	fc_appy_party_service_contact_vo_service_json_Cor
1090 ContactExpiryApplicationServ	rviceSpiV0FacadeService=com.ofss	.fc.demo.appx.party.service.contact.vo.service.s
1091 PartyAccountRelationshipApp	<pre>blicationServiceSpiV0FacadeServic</pre>	<pre>ce = com.ofss.fc.appx.party.service.relation.accc</pre>
1092 CommentApplicationServiceSp	biV0FacadeService = com.ofss.fc.	appx.party.service.core.vo.service.json.CommentAp
1093 BlacklistReportApplicationS	ServiceSpiV0FacadeService = com.on	ofss.fc.appx.party.service.role.customer.vo.servi
1094 PartyRelationshipApplicationService	onServiceSpiV0FacadeService = com.ofs	m.ofss.fc.appx.party.service.relation.vo.service.json.
1095 RelationshipApplicationService	viceSpiV0FacadeService = com.ofs	fc.appx.party.service.core.vo.service.json.F
1097 KYCHistoryApplicationService	ServiceSpiV0FacadeService = com.ofs.	fc.appx.party.service.core.vo.service.json.KYCHis

## Step 11 Deploy Project

After performing all the above mentioned changes, deploy the project as follows:

- 1. Add this project (com.ofss.fc.demo.party.contactexpiry) to the classpath of the branch application project.
- Open the launch configuration of the Tomcat Server. Add this project to the classpath of the server as well.
- 3. Deploy the branch application project on the server and start it.
- 4. Client Application Changes.

After creating database table to hold the input data and after creating the related domain objects and service and facade layers, you can customize the user interface. The customizations to the application have to be done on the client application. To customize the UI, follow these steps.

### Step 12 Create Model Project

You need to create a model project to hold the required view objects and application module.

To create the model project, follow these steps:

1. In the client application, create a new project of the type ADF Model Project.

Figure 5–49 Create Model Project

This list is filtered according to the	current project's selected technolo	ogies,
🎁 Search		
<u>C</u> ategories:	Items:	Show All Description
⊟General	Generic Project	2
Applications Ant Connections	ADF Model Project     Creates a project that defin     application using ADF Busin	nes a data model for an ADF web ness Components.
-Deployment Profiles	D ADF Swing Project	
Diagrams Java Projects XML Web Tier	ADF ViewController Project	
	EJB Project	
	🛅 Java Application Project	
Applet	📴 Java Project	
JSF	Project from Existing Source	ce
JSP Servlets	Project from WAR File	
All Items	Repiect Template	

2. Give the project a title (com.ofss.fc.demo.ui.model.party) and set the default package as the same.

😣 Create ADF Mo	del Project - Step 2 of 2		
Configure Java set	tings	010101010101010101040494949494	
Project Name	Your new project starts with a defa directory.	ault package, a source root directo	ry, and an output
V Project Java Setti	Default Pac <u>k</u> age:		
	com.ofss.fc.demo.ui.model.party		
	Java Source Path:		
	/rshanbha/Work/jdev_workspace/Vi	iew/com.ofss.fc.demo.ui.model.pa	d Bro <u>w</u> se
	Output Directory:		
	anbha/Work/jdev_workspace/View/c	om.ofss.fc.demo.ui.model.party/c	la Brow <u>s</u> e
<>			
Help		< <u>B</u> ack <u>N</u> ext > <u>F</u> ini	sh Cancel

Figure 5–50 Create Model Project - Configure Java Settings

3. Click on Finish to create the project.

## Step 13 Create Application Module

You need to create an application module to contain the information of all the view objects that you need to create. To create an application module, follow these steps:

- 1. Right click on the model project and select New.
- 2. Choose Application Module from the dialog box that opens.

Figure 5–51 Create Application Module

New Gallery     All Technologies     Current Project T     This list is filtered according to the c     Search	echnologies aurrent project's <u>selected technologies</u> .	
<u>C</u> ategories:	Items:	Show All Descriptions
General	Business Components from Table	is Alleria and A
Connections     Connections     Deployment Descriptors     Deployment Profiles     Diagrams     Java     Projects     Business Tier     ADF Business Components     Data Controls	Application Module Launches the Create Application Module wizard, which allows you to create an application module. Use application modules to assemble and organize view objects, to handle transactions, and to provide business service methods. To enable this option, you must select a project in the Application Navigator. Before you can finish creating the new application module, you will be prompted to select (or create) a database connection.	
Security	🔁 Association	
All Items	📓 Business Components Deployment Profiles	
	ତ Business Components Diagram	
	💼 Default Data Model Components	
	-	M
Help		OK Cancel

- 3. Set the package of the application module to the default package (com.ofss.fc.demo.ui.model.party).
- 4. Provide a name to the application module (DemoPartyAppModule).

Figure 5–52 Set Package and Name of Application Module

	8	Create Applicat	ion Module -	Step 1 of 5	
N	la	me		01	2
	ę	Name	Application mod of an applicatio	dules are for assembling, packaging, and deploying the view objects and busines: n.	s services
1	ų	Data Model	Package:	com.ofss.fc.demo.ui.model.party	Browse
	ģ.	Application Modules	Na <u>m</u> e:	DemoPartyAppModule	
	Ý.	Java	Display Name:	Demo Party App Module	Q
	Ċ	Summary	Extends:		Browse
			Property Set	<none></none>	
					,
<		, >			
[	_	Help		< Back Next > Finish	Cancel

- 5. Click on Next and let the rest of the options be set to the default options.
- 6. You will see a summary screen for the application module. Click on Finish to create the application module.

Figure 5–53 Summary of Application Module Created

😣 Create Applicat	ion Module - Finish
Summary	
V Name V Data Model V Application Modules V Iava Summary	You have finished the Create Application Module Wizard.         You have selected the following options:         Application Module         > Name         > Data Model         > View Object Instances:         > None         > View Link Instances:         > None         > Application Modules         > None         > Generate AppModule Class: False
< <u> </u>	When you click Finish, the application module will be created.           < Back         Next >         Einish         Cancel

### Step 14 Create View Object

You need to create a view object for the newly added Expiry Date field. This view object is used on the screen to display the value of the field as well as to take the input for the field.

To create the view object, follow these steps:

- 1. Right click on the Java package com.ofss.fc.demo.ui.model.party and select New View Object.
- 2. In the dialog box that opens, provide a name (ContactExpiryVO) for the view object.
- 3. Provide a package (com.ofss.fc.demo.ui.model.party.contactexpiry) for the view object.
- 4. For the Data Source Type option, select Rows populated programmatically, not based on a query.
- 5. Click on Next.
- 6. In the Attributes dialog, create a new attribute for Expiry Date field.
- 7. Provide a name (ExpiryDate) and type (Date) for the attribute.
- 8. For the Updatable option, select Always.

Figure	5-54	Create	View	Object
--------	------	--------	------	--------

6	Create View Ol	bject - Step 1 (	of 9	
N	ame		01	
8	Name	View objects ar given applicatio	e for joining, filtering, projecting, and sorting your business data for the specific in task.	needs of a
,	Entity Objects	Package:	com. ofs s. fc. demo. ui. model. party. contactexpiry	Browse
	Attributes	Na <u>m</u> e:	ContactExpirtVO	
2	Attribute Settings	Display Name:	Contact Expirt Vo	
4	Query	Extends:		Browse
4	Bind Variables	Descento Cat		
4	Java	Property Set:	<none></none>	1
9	Application Module	Select the data	source type you want to use as the basis for this view object.	
G	Summary	O Updatable a	ccess through entity objects	
		O Read-only as	ccess through SQL query	
		Rows popula	ited programmatically, not based on a query	
		Rows popula	ited at design time (Static List)	
	Help		< Back Next > Einish	Cancel

Figure 5–55 View Attribute

🔋 Create View Obje	ct - Step 2 of 6				
Attributes	😣 View Att	ribute			
A Name	<u>N</u> ame:	ExpiryDate		]	Updatable
Attributes	Type: Property Set	Oate <none></none>	-	Browse	
2 U Java	Value Type: Value:	Literal      Expression		<u>E</u> dit	Always
Summary	☐ Mapped f ♥ Selected Disgrimir Default V Passivat	to <u>C</u> olumn or SQL in Query nator: View Entity alue: e	☐ Key Attribute ✔ Queryable ☐ Effective Date		O Neger
	Alia <u>s</u> : [	VIEW_ATTR	Type: DATE		
	Help			ок	Cancel
Help	Alias:		< Back Next	> <u>E</u> ir	ish Cancel

9. Click Next. On the Application Module dialog, browse for the previously created DemoPartyAppModule.

Figure 5–56 Application Module

😣 Create View Ob	ject - St	ep 5 of 6	
Application Module	e	01010101010101010101010101010101010101	
Attributes	Select the application	e checkbox to add an instance of this view object to an application module. If the specifie on module does not exist, it will be created. ation <u>M</u> odule	rd
🤪 <u>lava</u>	Package:	com. ofss. fc. demo. ui. model. party	Browse
Application Modul	Nam <u>e</u> :	DemoPartyAppModule	Browse
↓ <u>Summarv</u>			
Help		< <u>B</u> ack <u>Next</u> > <u>Finish</u>	Cancel

- 10. For all other dialogs, keep the default options.
- 11. Click Next till you reach the summary screen as shown below.
- 12. Click on Finish to create the view object.

Figure 5–57 Create View Object - Summary

8 Create View Ob	ject - Finish
Summary	
Name Attributes Attribute Settings Java Application Module Summary	You have finished the Create View Object Wizard.         You have selected the following options:         Yiew Object         Name         Attributes         You Generate View Object Class: False         Senerate View Row Class: False         DemoPartyAppModule
Help	< <u>B</u> ack <u>Next</u> > <u>Finish</u> Cancel

## Step 15 Create View Controller Project

You need to create a view controller project to contain the UI elements. This project will also hold the customizations to the application.

To create the view controller project, follow these steps:

- 1. In the client application, create new project of the type ADF View Controller Project.
- 2. Give the project a title (com.ofss.fc.demo.ui.view.party) and set the defaults package to the same.

Figure 5–58 Create View Controller Project

8 New Gallery					
All Technologies Current Project Technologies					
This list is filtered according to the cu	irrent project's selected technologies.				
🤲 Search					
<u>C</u> ategories:	Items: Show All Desc	riptions			
General	Generic Project	^			
Applications Ant	ADF Model Project				
Connections	D ADF Swing Project				
Deployment Profiles Diagrams Java	ADF ViewController Project Creates a project that defines view and controller components for an ADF web application using ADF Faces and ADF Task Flows.				
Projects	EJB Project				
-Web Tier	Java Application Project				
Applet	🛅 Java Project				
JSF	Project from Existing Source				
JSP Servlets	D Project from WAR File				
All Items	Project Template	~			
Help	ОК	Cancel			

3. Click on Finish to finish creating the project.

Figure 5–59 Name your Project

😣 Create ADF Vie	wController Project - Step 1 of 2
Name your project	
Project Name	Project Name:       com.ofss.fc.demo.ui.view.party         Directory:       /_workspace/View/com.ofss.fc.demo.ui.view.party         Browse
	Project Technologies       Generated Components       Associated Libraries         Available:       Selected:         ADF Business Components       ADF Faces         ADF Desktop Integration       ADF Page Flow         ADF Mobile Browser       ADF Swing         Ant       Image: Selected:         Database (Offline)       Image: Selected:         EJB       JavaBeans         Image: Selected:       Image: Selected:         Image: Selected: Selected:       Image: Selected:         Image: Selected:
Help	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish Cancel

- 4. Right click on the project and go to Project Properties. In the Libraries and Classpath tab, add the following:
- 5. The Jar containing the screen to be customized (com.ofss.fc.ui.view.party.jar).
- 6. The Jar containing the domain objects and services for Contact Expiry (com.ofss.fc.demo.party.contactexpiry.jar) as created in host application project.
- 7. All the required dependent Jars for the above Jars.
- 8. The Jar containing the customization class (com.ofss.fc.demo.ui.OptionCC.jar).
- 9. In the Dependencies tab, browse for and add the previously created adf model project (com.ofss.fc.demo.ui.model.party).
- 10. In the ADF View tab, check the Enable Seeded Customizations option to enable this project for customizations.

😣 Project Properties - /	home/rshanbha/Work/jdev_workspace/View/com.ofss.fc.dem	o.ui.view.party/com.ofss.fc
( Search	Libraries and Classpath	
<ul> <li>⊕ Project Source Paths</li> <li>⊕ ADF Model</li> </ul>	<ul> <li>○ Use <u>C</u>ustom Settings</li> <li>○ <u>U</u>se Project Settings</li> </ul>	Customize Settings
ADF View     B- Ant     Business Components     Generites	Iava SE Version: 1.6.0_29 (Default)	Cha <u>n</u> ge
Compiler     Dependencies     Deployment	Classpath Entries: Export Description	Add Library
EJB Module Extension ⊕ Javadoc	ADF DVT Faces Databinding MDS Runtime      CUSTOM_LIB      TEMPLATE_LIB	Add JAR/Directory
<ul> <li>Java EE Application</li> <li>JSP Tag Libraries</li> <li>JSP Visual Editor</li> </ul>		cies jars <u>E</u> dit
Libraries and Classpath     Resource Bundle     Run/Debug/Profile	ASATEWEB     ARTY_UB     party sc	
- Technology Scope	ADDRESS_INSR CON_ELB     Com.ofss.fc.ui.customtags.jar     Com.ofss.fc.ui.components.jar     DEMO_OPTION_CC     Customization     ADF Common Web Runtime     ADF Faces Databinding Runtime	class jar . host domain jar
Help		OK Cancel

Figure 5–60 Libraries and Classpath

Figure 5–61 Dependencies

8 Project Properties - ,	/home/rshanbha/Work/jdev_workspace/View/com	ofss.fc.demo.ui.view.party/com.ofss.fc.
🏟 Search 💦	Dependencies	
Project Source Paths     ADF Model	<ul> <li>○ Use <u>C</u>ustom Settings</li> <li>● <u>Use</u> Project Settings</li> </ul>	Customize Settings
ADF View	Dependent Projects and Archives:	1 ×
Business Components     Compiler     Dependencies     Deployment     EjB Module     Extension     Java EE Application     JSP Tag Libraries     JSP Visual Editor     Libraries and Classpath     Resource Bundle     Run/Debug/Profile     Technology Scope	🖻 com.ofss.fc.demo.ui.model.party.jpr	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Help		OK Cancel

11. Save the changes by clicking OK and rebuild the project.

# Step 16 Create Maintenance State Action Interface

Create an interface containing the method definition for a maintenance action. This interface is implemented by the required maintenance state actions classes for the screen to be customized. The state action method will take the instance of the backing bean as a parameter.

### Figure 5–62 Create Maintenance State Action Interface

B IDemoContactPoint.java ×			
💏 - Find	- V 🖓 🖄 🦧 I 🛤 🛟 🕾 🗔 I 🗖 I 🐘 🕪 🖿 I 🚟 🔛		
<pre>package com.ofss.fc.demo.ui.view.party.contactPoint.backing;</pre>			
🗸 🗉 public interface IDemoContactPoint ┨			
<pre>4     public boolean performStateAction(DemoContactPoint demoContactPoint);</pre>			
6			

## Step 17 Create State Action Class

You need to create a class which will contain the business logic for the create transaction for this screen. This class should have following features:

- Implements the previously created state action interface.
- Creates the Contact Point DTO from the users input.
- Creates an instance of the Contact Point service proxy.
- Calls the add method of the service passing the DTO.

# Step 18 Create Update State Action Class

You will need to create a class which will contain the business logic for the update transaction for this screen. This class should have following features:

- Implements the previously created state action interface.
- Creates the Contact Point DTO from the users input.
- Creates an instance of the Contact Point service proxy.
- Calls the update method of the service passing the DTO.

Figure 5–63 Create Update State Action Class

🕹 De	mo	ocreateContactPoint.java 🗴
-66	Fin	a 🕹 🚱 🖄 🍻 🐙 i 🖓 🖓 🖓 🖓 🖓 🖬 i 🗍 🕪 🦬 🖬 🛯
Ŷ		public class DemoCreateContactPoint implements IDemoContactPoint {
22 23 24		<pre>private final Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(DemoCreateContactPoint.class.getName());</pre>
26 27	8	<pre>public boolean performStateAction(DemoContactPoint demoContactPoint) {     // Create the DTO from the screen and call proxy.     boolean status = false;</pre>
29 30 31		<pre>SessionContext context = SessionContextFactory.getSessionContextFactory().getSessionContextInstance(); context.setServiceCode(Constants.SERVICE_CODE); TransactionStatus transactionStatus = null;</pre>
33 34 35		ContactExpiryDTO contactExpDTO = demoContactPoint.createContactExpDTO(); IContactExpiryApplicationServiceProxy contactExpProxy = null;
36 37 38		<pre>try {     contactExpProxy = (IContactExpiryApplicationServiceProxy) ProxyFactory.getInstance().getProxy(     DemoContactPoint.CONTACT_EXPLAY_PROXY); </pre>
39 40 41		<pre>xt (logger.lsLoggable(Leve(.FIME)) {     logger.log(Level.FIME, "Calling addContactExpiry service"); }</pre>
42 43		<pre>transactionStatus = contactExpProxy.addContactExpiry(SessionContextFactory.getSessionContextFactory()</pre>
45 46		<pre>if (transactionStatus != null &amp;&amp; (transactionStatus.getErrorCode().equals("0"))) {     MessageHandler.addMessage(transactionStatus);</pre>
47 48 49		} } catch (FatalException e) { MessageHandler.addMessage(e);
50 51 52		<pre>logger.log(Level.SEVERE, MultiEntityLogger.getUniqueInstance().formatMessage(</pre>
53 54		<pre>MessageHandler.addMessage(e); logger.log(Level.SEVERE, MultiEntityLogger.getUniqueInstance().formatMessage(</pre>
56 57		<pre>&gt; catch (Throwable e) {     MessageHandler.addErrorMessage("Internal error occured. Please contact system administrator");</pre>
58 59		} return status;

Figure 5–64 Create Update State Action Class



## Step 19 Create Backing Bean

You need to create a backing bean class for the screen to be customized. This class should have the following features:

- Should implement the interface ICoreMaintenance.
- Private members for the to be added UI Components in customization and public accessors for the same.
- Private member for the backing bean of the original backing bean of the screen (ContactPoint) which is initialized in the constructor of this class.
- Private member for the parent UI Component of the newly added UI components and public accessors which returns the corresponding component of the backing bean.
- Private member for the newly added view object (ContactExpiryVO) and the current view objects present on the screen.

Figure 5–65 DemoContactPoint.java displays the View Objects

DemoContactPoint.java ×					
(🚓 • Find 🕹 🚱 🤮 🍻 🐗 🛛 🖓 🗐 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓					
45 ⊟ public (	lass DemoContactPoint implements ICoreMaintenance {				
46 47 priv 48 priv 49	<pre>static final String WD_CONTACT_EXP = "ContactExpiryVOlIterator"; ate static final String EXPIRY_DATE = "ExpiryDate";</pre>				
50 prot	<pre>ected static final String CONTACT_EXPIRY_PROXY = "ContactExpiryApplicationServiceProxy";</pre>				
51 52 priv 53 priv 54 priv	rate UIXGroup <b>formData;</b> rate ContactPoint <b>contactPoint;</b> rate ViewObject <b>contactPoint¥0 =</b> IteratorHandler. <i>getViewObject</i> (Constants. <b>PAGE_DEF</b> , Constants. <b>V0_CONTACT_POINT</b> );				
56 Cont 57 priv 58 priv 59 priv	actPointBusinessRules <b>contactPointBR = new</b> ContactPointBusinessRules(); rate RichPanelLabelAndMessage <b>plan18</b> ; rate DateComponent <b>expiryDate</b> ; rate ViewObject <b>contactExpVO =</b> IteratorHandler.getViewObject(Constants.PAGE_DEF, VO_CONTACT_EXP);				
61 priv 62 63	ate transient Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(DemoContactPoint.class.getName());				
64 ₪ publ 65 66 [ 67 }	<pre>ic DemoContactPoint() { super(); contactPoint = (ContactPoint)ELHandler.get("#{ContactPoint}");</pre>				
69 E publ 70 71 } 72	<pre>ic void setFormData(UIXGroup formData) { this.formData = formData;</pre>				
73 E pub 74 75 76 }	<pre>ic UIXGroup getFormData() { this.formData = contactPoint.getFormData(); return formData;</pre>				

- clear() method which handles the user action Clear.
- save() method which handles the maintenance state actions Create and Update.
- Depending on the current state action, the save() method should instantiate either DemoCreateContactPoint or DemoUpdateContactPoint and perform the corresponding state action methods.





A public method to create the Contact Expiry DTO from the user's input on the screen.

Figure 5–67 Create Contact Expiry DTO



A value change event handler for the Expiry Date UI Component.

public void onExpiryDateChange(ValueChangeEvent valueChangeEvent) { if (logger.isLoggable(Level.FINE)) { logger.log(Level.FINE, MultiEntityLogger.getUniqueInstance().formatMessage("Entering onExpiryDateChange method.")); Date processdate · new com.ofss.fc.datatype.Date(((oracle.jbo.domain.Date)ELHandler.get(\*\*{pageFlowScope.defaultValues.postingDate}\*) if (valueChangeEvent.getNevValue() != null) { Date expDate = new Date(((oracle.jbo.domain.Date)valueChangeEvent.getNewValue()).dateValue()); date if (!expDate.isBefore(processdate)) { MessageHandler.addErrorMessage(getExpiryDate().getClientId(), "Expiry date should not be less than the current date", nulli contactExpV0.getCurrentRow().setAttribute(EXPIRY\_DATE, null); this.getExpiryDate().reset(); AdfFacesContext.getCurrentInstance().addPartialTarget(expiryDate); } else if (valueChangeEvent.getNevValue() == null) {
 MessageHandler.addErrorMessage(getExpiryDate().getClientId(), "Select Expiry Date", null); } }

Figure 5–68 Value Change Event Handler for the Expiry Date UI Component

 Value change event handlers for the existing UI Components on change of which the screen data is to be fetched.

Figure 5–69 Value Change Event Handlers for Existing UI Components



Method containing the business logic to fetch screen data using Contact Expiry proxy service.





## Step 20 Create Managed Bean

You need to register the DemoContactPoint backing bean as a managed bean with a backing bean scope.

- 1. Open the project's adfc-config.xml which is present in the WEB-INF folder.
- In the Managed Beans tab, add the binding bean class as a managed bean with backing bean scope as follows:

DemoContactPoint.java × Cadfc-config.xml ×							
							28   🥲
General Description Activities	So Managed Beans				<b>+ X</b>		
Control Flows	Name * 🔺	lame *  Class * Scope *					
Managed Beans DemoContactPoint com.ofss.fc.demo.ui.view.party.contactPoint.backing.DemoContactPoint backing				backingBean			
Metadata Resources	⊡ Managed Properties: DemoContactPoint					<b>+</b> ×	
	Name * 🛋		Class Value			11.	

Figure 5–71 Create Managed Bean

### Step 21 Create Event Consumer Class

You need to create an event consumer class to consume the Party Id Change event. When the user inputs a party id on the screen, the business logic in this event consumer class will be executed automatically.





# Step 22 Create Data Control

For the event consumer class's method to be exposed as an event handler, you need to create a data control for this class.

1. In the Application Navigator, right click on the event consumer Java file and create data control.

On creation of data control, an XML file is generated for the class and a DataControls.dcx file is generated containing the information about the data controls present in the project. You will be able to see the event consumer data control in the Data Controls tab.

Figure 5–73 Create Data Control



2. You should now restart JDeveloper in the Customization Developer Role to edit the customizations. Ensure that the appropriate Customization Context is selected.

# Step 23 Add UI Components to Screen

Browse and locate the JSFF for the screen to be customized

(com.ofss.fc.ui.view.party.contactPoint.contactPoint.jsff) inside the JAR (com.ofss.fc.ui.view.party.jar). Open the JSFF and do the required changes as follows:

- 1. Drag and drop the Panel Label & Message and Date UI components at the required position on the screen.
- 2. For each component, set the required attributes using the Property Inspector panel of JDeveloper.
- 3. Modify the containing Panel's width and number of columns attributes as required.
- 4. For each component, add the binding to the DemoContactPoint backing bean's corresponding members.
- 5. Add the value change event binding for the Expiry Date UI component to the backing bean's corresponding method.
- 6. Change the value change event binding for the existing UI component on change of which the screen data is fetched.
- 7. Change the backing bean attribute of the screen to the previously created DemoContactPoint backing bean.
- 8. Save the changes. You will notice that JDeveloper has created a customization XML in the ADF Library Customizations folder to save the differences between the base JSFF and the customized JSFF. The generated contactPoint.jsff.xml should look similar to as shown below.

Figure 5–74 Generated contactPoint.jsff.xml

e co	contactPoint.jsff.xml 🗵				
<b>#</b>	And 🕹 🌚				
1	<mds:customization version="11.1.1.61.92" xmlns:mds="http://xmlns.oracle.com/mds"></mds:customization>				
2	<pre><mds:insert after="smcAllowedpurpose" parent="formData" xmlns:af="http://xmlnsioracle.com/adf/faces/rich" xmlns:fc="/com/ofss/fc/ui/components"></mds:insert></pre>				
3	<ar binding="#(DemoContactPoint.plant8)" id="plant8" label="Expiry Date" sparellabelandressage="" xmlns:at="http://xmlns.oracle.com/add/taces/rich"></ar>				
4	<pre><tc:date date="" pinding="#(democnate/form.transpryvate)&lt;br" tablet="party" to="expryvate" xeturs:rt="/com/orss/rt/u//components">wsbue_fetbridges_Exprising_Exprising_interface interface interface and formation interface and interface interf</tc:date></pre>				
6	<pre></pre>				
7					
8	<pre><mds:modify backingbeanclass")"="" element="ptl(xmlns(f=http://java.sun.com/jsf/core))/f:attribute(ename="></mds:modify></pre>				
9	<mds:attribute name="value" value="com.ofss.fc.demo.ui.viev.party.contactPoint.backing.DemoContactPoint"></mds:attribute>				
10					
11	<pre><mds:modify element="pill3"></mds:modify></pre>				
12	<pre>substattribute name= maxCounts _ value= 2 /&gt; undstattribute name= maxCounts _ value= 2 /&gt; </pre>				
14	- and statistic name is bolight value=2007/				
15					
16	<pre><mds:modify element="socContactPointType"></mds:modify></pre>				
17	<pre><mds:attribute name="valueChangeListener" value="#[DemoContactPoint.onContactPointTypeChange]"></mds:attribute></pre>				
18					
19	<mds:modify element="socContactPref"></mds:modify>				
20	<pre>stattibute name="value.nange_istener" value="#juemocontactProint.oncontactProtecence.nange;"/&gt; c.mds.andifus.</pre>				
22	<pre></pre> //bis/shouldy/ //dis/statistation>				
23					

## Step 24 Add View Object Binding to Page Definition

You need to add the view object binding for the previously created ContactExpiryVO view object to the page definition of the screen to be customized.

- Browse and locate the page definition for the screen to be customized (com.ofss.fc.ui.view.party.contactPoint.pageDef.ContactPointPageDef.xml) and open it.
- 2. Add an attributeValues binding as shown below.

Figure 5–75 Add an attributeValues binding

😣 Insert Item		
Sele <u>c</u> t the category of components from which you would like to find an ite	em:	
Generic Bindings 🗸		
Select the item to be created:		
🛃 action	_	
attributeValues		
🔚 button		
🗇 eventBinding		
🔚 graph		
🛃 list		
🛃 listOfValues		
methodAction	~	
Description:		
Supports binding to one (or more) attributes.	^	
	~	
Help	OK Cancel	

- 3. For Data Source option, locate the previously created ContactExpiryVO view object present in the DemoPartyAppModule.
- 4. For Attribute option, choose the ExpiryDate attribute present in the view object.

Figure 5–76 Create Attribute Binding

8 Create Attribute Binding				
<u>D</u> ata Source:	TemoPartyAppModuleDataControl.ContactExpiry 🔻 🕂			
A <u>t</u> tribute: <u>H</u> elp	ExpiryDate  OK Cancel			

## Step 25 Add Method Action Binding to Page Definition

You need to add the method action binding for the previously created DemoPartyIdEventChangeConsumer event consumer class to the page definition of the screen to be customized.

1. Add a methodAction binding as shown below.

### Figure 5–77 Add a methodAction binding

😣 Insert Item		
Sele <u>c</u> t the category of components from which you would like to find an item:		
Generic Bindings	•	
Select the item to be created:		
raph -	^	
list		
🛃 listOfValues		
🔤 methodAction		
🛃 navigationlist	-	
table		
Stree		
Un treetable	M	
Description:		
Method binding for the control.	^	
	¥ .	
Help	OK Cancel	

2. For the Data Collection option, locate the previously created DemoPartyIdChangeEventConsumer data control.
Figure 5–78 Create Action Binding

😣 Create Ac	tion Binding			
Select a data colle the selected colle	ection and the action you wa ction.	nt your control to initiate. The	control initiates the action on the data objects of	
Data Collection:				
DemoParty     DemoParty     DemoParty     DepositBa     DirectRolel     DirectRolel     DirectRolel     Documenti     Documenti     Documenti     Documenti	AppModuleDataControl VIdChangeEventConsumer sicDetailsEventProducer PartyDetailsConsumer CategoryEventConsumer DetailsEventProducer ExecutionEventProducer		G	
Select an Iterator	:		• <u>N</u> ew	1
Operation:	partyldChangeEvent(Object			
	Apply to all iterators in p	age defintion		
Parameters :				
Name	Type	Value	Option	
object	java.lang.Object		· · · · · ·	
Help			OK Cancel	

### Step 26 Edit Event Map of Page Definition

You need to map the Event Producer for the party id change event to the previously created Event Consumer.

- 1. In the Structure panel of JDeveloper, right click on the page definition and select Edit Event Map.
- 2. In the Event Map Editor dialog that opens, edit the mapping for the party id change event. Select the previously created Event Consumer's method.

Figure 5–79 Select the Event Consumer Method

roducer:	contactPointPageDef.partvDetailsTaskFlowDefn1.partvDetailsPageDef.ra	iseEvent
vent Name:	partyldOnChangeEvent	
onsumer:	contactPointPageDef.party/dChangeEvent1	
Consumer Params	+ ×	
Param Name	Param Value	
payLoad	#{payLoad}	

3. Save the changes. You will notice that JDeveloper has created a customization XML in the ADF Library Customizations folder to save the differences between the base JSFF and the customized JSFF. The generated contactPoint.jsff.xml should look similar to as shown below.

Figure 5–80 Generated contactPoint.jsff.xml



### Step 27 Deploy Customization Project

After finishing the customization changes, exit the Customization Developer Role and start JDeveloper in Default Role. Deploy the view controller project as an ADF Library Jar (com.ofss.fc.demo.ui.view.party.jar).

Go to Project Properties of the main application project and in the Libraries and Classpath, add the following:

- 1. View controller project Jar (com.ofss.fc.demo.ui.view.party.jar)
- 2. Host domain Jar (com.ofss.fc.demo.party.contactexpiry.jar)
- 3. Customization Class Jar (com.ofss.fc.demo.ui.OptionCC.jar)
- 4. All dependency libraries and Jars for the project.
- Start the application and navigate to Party → Contact Information → Contact Point screen. Input a
  party id on the screen and perform the read, create and update actions on Contact Point. You need to
  input data and fetch value for the newly added Expiry Date field.

Figure 5-81 Pl041 - Contact Point Screen

PI041			
Contact Point			
🛄 Bead 📲 Cr <u>e</u> ate 🖉 <b>Update</b>			🗸 Ok 🖉 Clear 📓 Cancel 🚔 Print
B Party Details			
NO IMAGE available	* Party ID 00005295 Home Branch 083991-J Bank Operations BR Company Name Daniel Trustee Party Class FOREIGN PUBLIC BODY Party Type LEG	Date of Incorporation Roles - Customer - Trustee Onboarding Date 15-Jan-2016	NO IMAGE AVAILABLE
Address Details			
Contact Point Details			
Contact Point Type	Mobile	Contact Preference Type Home	
Seasonal Start Date		Peasonal End Date	
<ul> <li>Allowed Purposes</li> </ul>	Scommunication Alert	Expiry Date 31-Aug-2012	16
Preferred Contact	Preferred Contact		
Marketing Consent	Marketing Consent		
Marketing Consent Start Date		Marketing Consent End Date	
Telephone Details			
Country Code		Area Code	
Number	32577789	Extension	
Service Provider		VOIP Code	
Timing Preferences			
DND	DND		
DND Start		DND End	
Weekdays	Uwekdays		
From		То	
Weekends	Weekends		
From		То	

### 5.7.3 Override the product managedBean

Screen customizations could be used to handle a product code which does not serve the necessary functionality and needs to be re-written.

# **6 Human Task Screen Extension**

# 6.1 Introduction

Human task screen uses internally ADF taskflows except the primary worklist actions. Human screen extensions work similar to ADF screen extensions, where the backing bean and controller classes of human task have the provision for extension, similar to the UI extension as explained in Section 4.1 UI Extension Interface, Section 4.2 Default UI Extension, Section 4.3 UI Extension Executor, and Section 4.4 Extension Configuration. From the perspective of customization, the recommendation is to do customization on the taskflows used internally. For CSS customization, see the below the details.

# 6.2 Custom CSS Skin

If ADF skin is already available and you just want to change the skin family reference in "trinidad-config.xml", directly apply the Section 6.2.2 Apply New Skin . If you want to create a new ADF skin and merge with the existing skin families to "trinidad-skins.xml", start from Section 6.2.1 Create New ADF Skin .

# 6.2.1 Create New ADF Skin

To create a new ADF skin and package it as library to refer to new skin families files, perform the following steps:

 Create ADF skin project and skin families as given in https://docs.oracle.com/cd/E16162\_ 01/user.1112/e17456/adfsg\_project.htm#ADFSG490:

Figure 6–1 Sample trinidad-skins.xml

```
<?xml version="1.0" encoding='\windows-1252"?>
<skins xmlns="http://myfaces.apache.org/trinidad/skin">
</skins xmlns="http://myfaces.apache.org/trinidad/skin">
</skins xmlns="http://myfaces.apache.org/trinidad/skin">
</skins xmlns="http://myfaces.apache.org/trinidad.desktop</skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skins/skinskins/skins/skins/skins/skins/skins/skins/skins/skins/skins
```

 Package the above project as given in https://docs.oracle.com/cd/E16162\_ 01/user.1112/e17456/adfsg\_apply.htm#ADFSG404.

After deployment, the package structure should be as shown in the following screenshot:

#### Figure 6–2 Package Structure

```
ADFLibraryJARRootDirectory
+---META-INF
        MANIFEST.MF
    T
        oracle.adf.common.services.ResourceService.sva
    L
н
        trinidad-skins.xml
I
L
    I
    +---adf
н
        \---skins
    ı
L
            \---skin1
                 \---images
                     \---af_column
    Т
L
                              sort des selected.png
    \---skins
н
        \---skin1
I
                skin1.css
L
+---resources
        skinBundle.properties
н
L
\---WEB-INF
        faces-config.xml
```

- 3. Deploy or add the deployed jar to obp ui library references.
- 4. Reference implementation can be provided on request.

### 6.2.2 Apply New Skin

 Option #1: The skin id can be configured with the help of property 'default.humantask.skinfamily' in seed data available as part of FLX\_FW\_CONFIG\_ALL\_B.

Figure 6–3 Sample Data

```
INSERT
INTO flx_fw_config_all_b
(
    PROP_ID, CATEGORY_ID, PROP_VALUE, FACTORY_SHIPPED_FLAG, PROP_COMMENTS,
    SUUMARY_TEXT, CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATED_DATE,
    OBJECT_STATUS_FLAG, OBJECT_VERSION_NUMBER
)
VALUES
(
    'default.humantask.skinfamily', 'UiConfig', 'NEWOBPALTASKIN', 'Y',
    NULL, 'NEWOBPALTASKIN UiConfig', 'ofssuser', to_timestamp('25-JUL-17 02.13.06.000000000 PM','DD-MON-RR HH.MI.SSXFF AM'),
    'ofssuser', to_timestamp('25-JUL-17 02.13.06.000000000 PM','DD-MON-RR HH.MI.SSXFF AM'), 'A', 1
);
```

• **Option #2:** Over and above the above the option 1, the extension can be implemented for "postCustomBranding" operation of CommonTaskFormPagePhaseListenerImpl [extension class: com.ofss.fc.workflow.ui.common.ext.ICommonTaskFormPagePhaseListenerImplUIExt]

#### Figure 6–4 Sample Implementation

```
FacesContext fc = FacesContext.getCurrentInstance();
ELContext elc = fc.getELContext();
String skinId = "NEWOBPALTASKIN";
ExpressionFactory exprFact = fc.getApplication().getExpressionFactory();
ValueExpression ve = exprFact.createValueExpression(elc, SS_SKIN_FAMILY, Object.class);
ve.setValue(elc, skinId);
ELHandler.set(IS_CUSTOM_BRANDING, "false");
```

# **7 Receipt Printing**

OBP has many transaction screens in different modules where it is desired to print the receipt with different details about the transaction. This functionality provides the print receipt button on the top right corner of the screen which gets enabled on the completion of the transaction and can be used for printing of receipt of the transaction details.

For example, if the customer is funding his term deposit account, the print receipt option will print the receipt with the details like Payin Amount, Deposit Term etc at the end of the transaction. The steps to configure this option in the OBP application are given in the following section.

# 7.1 Prerequisite

Following are the prerequisites for receipt printing.

# 7.1.1 Identify Node Element for Attributes in Print Receipt Template

The list of all the elements that are present in the particular task code screen and need to be displayed in the printed receipt can be identified with the help of the VO object utility. This utility helps in identifying all the node elements which are available on the screen and can be used in the print receipt template. This utility VOObjectUtility can be used to generate the data required for the functionality to work.

Once the utility is imported in the workspace, the input properties file needs to be updated with the location of module's UI, location of task flow directory, location of config directory and the output directory where you want the output of the utility.



Figure 7–1 Input Property Files

In the build path of the utility, three libraries (commons-io, xalan and xmlparserv2) need to be added as they are required for execution of the utility.

### Figure 7–2 Build Path of Utility

type filter text 🛛 🗷	Java Build	l Path			↓ ↓ ↓ .
Resource Builders	@Source	₿Projects	Libraries	%Order and Expo	ort
Java Build Path	JARs and	class folders	on the build p	ath:	
Java Code Style	CUST	OM_LIB/con	nmons-io-1.4.	jar - /home/vishal/	Add JARs
Java Compiler Java Editor	<ul> <li>CUST</li> <li>CUST</li> </ul>	rom_LIB/tld/ rom_LIB/tld/	/xalan.jar - /ho /xmlparserv2.	ome/vishal/eclipse jar - /home/vishal/	Add External JARs
Javadoc Location	🕨 🛋 JRE S	System Libra	ry [JavaSE-1.6	]	Add Variable
PMD			Add Library		
Project References			Add Class Folder		
Run/Debug Settings Server					Add External Class Folder.
Subversion					Edit
Task Tags					Remove
Validation WikiText					Migrate JAR File

Then the main method of the VOAttributesFinder.java class in the utility is executed.

#### Figure 7–3 Utility Execution



On the execution of the utility, the Excel file is generated. The task codes can be filtered in the Excel file for viewing different RTF node value of different attributes available on the particular screen.

Figure 7–4	Excel	Generation	
------------	-------	------------	--

2	А	В	С	D	E
L	Task Code 📝	View Object 🔹	Attribute Name 🛛 💽	Attribute Type 🛛 💌	RTF Node 💌
15	5 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	accountNo	java.lang.String	FundTermDepositVO_accountNo?
16	5 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	principalBalance	java.math.BigDecimal	FundTermDepositVO_principalBalance?
17	7 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	payinAmount	java.math.BigDecimal	FundTermDepositVO_payinAmount?
18	3 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	transactionRefNo	java.lang.String	FundTermDepositVO_transactionRefNo?
19	TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	userRefNo	java.lang.String	FundTermDepositVO_userRefNo?
20	TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	acctCCY	java.lang.String	FundTermDepositVO_acctCCY?
21	L TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	productCode	java.lang.String	FundTermDepositVO_productCode?
22	2 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	partyld	java.lang.String	FundTermDepositVO_partyId?
23	3 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	branchid	java.lang.String	FundTermDepositVO_branchId?
24	TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	primaryReason	java.lang.String	FundTermDepositVO_primaryReason?
25	5 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	secondaryReason	java.lang.String	FundTermDepositVO_secondaryReason?
26	5 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	narrative	java.lang.String	FundTermDepositVO_narrative?
	ြ Task Code	ک View Object Path (Screen/taskflow)	ل ل VO Attribute Name	ے ل Attribute Type	Reference in RTF template

### 7.1.2 Receipt Format Template (.rtf)

This template is used for defining the format of the output receipt. Different data elements which need to be shown in the output receipt are mentioned in this RTF report format template. The node will be taken from the above generated Excel file from 'RTF Node' column for specifying the output value in the final output RTF.

The sample rtf template is shown below:

Figure 7–5 Receipt Format Template

9	Home	- Ŭ = Insert	Page Layout	t References	Document in D: Mailings	:\SVN\extensib Review \	iility∖OBP-Extensibility_ View	Guide.docx [Com	patibility Mod	e] - Microsoft	Word			-	• :	×
Paste	Cut Copy	y nat Painter	Times New Roo	man v 10 v abe x <sub>2</sub> x <sup>2</sup> A Font	× A* A* ↔ a* ♥ × <u>A</u> ×		*示*) 律 律 (2↓ (*   ■ (目* (2) * Ⅲ Paragraph	AaBbCcI	AaBbCcI 11 No Spaci	AaBbC Heading 1	AaBbCo Heading 2	AaBbC	Change Styles *	A Find ▼ a a Replace Select ▼ Editing		
L			1 - 1 - 1 -	X	1 * 1 * 3 * 1 * 4		6 . 1 . 7 . 1 . 8 . 1 .	9 • 1 • 10 • 1 • 1	1 - 1 - 12 - 1 -	13 · 1 · 14 · 1	· 15 · I · 16 ·	1 - 17 👌 :	8 · + · 19			23
2 · · · · · · · · · · · · · · · · · · ·				Bank Name Branch Addr Posting Date Transaction I Event Name Account Nur Deposit No: Account Title Account Cur Payin Amour Value Date: Net Interest I Maturity Datu Deposit Terr TDPayint<br Interest Payo Narrative:	ess & time: Ref No: : mber: : : rency: : : : : : : : : : : : : : : : : : :	onths?>mc	BankShortN<br BranchName<br PostingDate<br Transaction#<br Fund Term Dep AccountDeta<br PayinDeposi<br AccountDeta<br AccountDeta<br AccountDeta<br AccountDeta<br AccountDeta<br AccountDeta<br AccountDeta<br AccountDeta<br AccountDeta<br TDPayinDeta<br TDPayinDeta<br TDPayinDeta</td <th>ame?&gt; a?&gt; ?&gt; RefNo?&gt; iosit iilsVO_accou itNo?&gt; ailsVO_accou ailsVO_accou iilsVO_netRat a?&gt; ailsVO_netRat a?&gt; ailsVO_netRat a?&gt; ailsVO_netRat a?&gt; ailsVO_netRat a?&gt;</th> <td>ntNo?&gt; ntTitle?&gt; ncy?&gt; te?&gt; % ?&gt; years : ays?&gt; days youtFreq?&gt; ive?&gt;</td> <td>s</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>	ame?> a?> ?> RefNo?> iosit iilsVO_accou itNo?> ailsVO_accou ailsVO_accou iilsVO_netRat a?> ailsVO_netRat a?> ailsVO_netRat a?> ailsVO_netRat a?> ailsVO_netRat a?>	ntNo?> ntTitle?> ncy?> te?> % ?> years : ays?> days youtFreq?> ive?>	s						
Page:	1 of 1 W	/ords: 62	English (India)										2	Θ	Į	ŧ

# 7.2 Configuration

This section describes the configuration details.

# 7.2.1 Parameter Configuration in the BROPConfig.properties

Following configuration parameters are required to be set in the BROPConfig.properties file.

- receipt.print.copy: Set to 'S' (default) if Single receipt is required. Else, set to 'M' for multiple receipts. The receipt will be stored in current posting date 'month/date' folder structure.
- receipt.base.in.location: Location for the RTF templates, which is configured as 'config\receipt\basein\template\' structure on the UI server. (For RTF development purpose this location will also have the XML generated while processing receipt.)
- receipt.base.out.location: Location for generated receipt, which is configured as 'config\receipt\baseout\' structure on the UI server.
- ui.service.url : UI URL http://IP:port format.

### 7.2.2 Configuration in the ReceiptPrintReports.properties

This file contains the key value pair of the Task Code of the screen and the corresponding template names, comma separated if more than 1 template is referred by screen.

### TaskCode=RTF Filename

Where TaskCode: task code of screen for which receipt print will be enabled and RTF Filename: name of the RTF template which will be used for the creation of the output with the same filename.

For example, TD002=FundTermDeposit

#### Figure 7–6 Receipt Print Reports

PrintVo	ucherReports.properties
1	BRM06=BranchBatchStatusInquiry
2	CASA001=CashDeposit_Ubank,CashDeposit_NAB

# 7.3 Implementation

The implementation for the print receipt functionality is explained in the following steps:

- 1. Once the screen is opened, Template checks '*ReceiptPrintReports.properties*' file if the Task code of the opened screen is present in the property file. The 'Receipt Print' button will be rendered in a disabled state.
- 2. On successful completion of transaction (successful Ok click), Receipt Print button gets enabled.
- 3. On click of Receipt Print button, all the VO's on current screen are fetched and created as a XML with data (for RTF development reference, this XML is not deleted at the moment but on environments these will be deleted). The RTF and XML merge up to create and open the receipt in the pdf format.
- 4. Receipt will be stored with the file name as <Logged in userId\_TemplateName>

The sample output receipt in the PDF form is shown below:

Figure 7–7 Sample of Print Receipt



### 7.3.1 Default Nodes

As per the functional specification requirement, some default nodes are already added in the generated XML. The list of those nodes are as follows:

- BankCode
- BankShortName
- BranchName
- PostingDate
- UserName
- BankAddress
- BranchAddress
- LocalDateTimeText

# 7.4 Special Scenarios

There are some cases, where some of the attributes are not available in the VOs of the screen and the value needs to be picked from the response of the transaction. There are also some data values which need to be

formatted first and then published in the PDF.

These values can be added to the pageFlowScope Map variable 'receiptPrintOtherDetailsMap'.

The population of those values needs to be done in the Backing Bean, after getting the response of the transaction in the following manner:

```
MessageHandler.addMessage(payinResponse.getStatus());
receiptDetails.put("TransactionRefNo", payinResponse.getStatus
().getInternalReferenceNumber());
SimpleDateFormat receiptTimeFormat = new SimpleDateFormat("hh:mm:ss
a");
SimpleDateFormat receiptDateFormat = new SimpleDateFormat("dd-MMM-
yyyy");
HashMap<String, String> receiptDetails = new HashMap<String, String>
();
Date date=new Date(getSessionContext().getLocalDateTimeText());
receiptDetails.put("PostingTime", receiptTimeFormat.format
(date.getSQLTimestamp()));
if(payinResponse!=null && payinResponse.getValueDate()!=null) {
receiptDetails.put("ValueDate", receiptDateFormat.format
(payinResponse.getValueDate().getSqlDate());
}
ELHandler.set("#{pageFlowScope.receiptPrintOtherDetailsMap}",
receiptDetails);
```

Internally, the functionality adds all the details in map variable, other than VO's data. While receipt printing, template checks the Map variable and if not null, it gets all the key-value from the map and show them in XML which is used later on for generation of receipt.

# 8 Extensibility Usage – OBP Localization Pack

OBP shall be releasing localization pack which ensures an optimized implementation period by adapting the product to different regions by implementing common region specific features pre-built and shipped. Every bank in different regions have different tax laws, different financial policies and so on. The policies in US will be different from those in Australia.

The localization packs leverage OBP extensibility to incorporate regional features and requirements by implementing different extension hooks for host and / or different JDeveloper customization functionalities for UI layer. This section presents a use case from OBP localization pack as implemented using the extensibility guidelines as a sample which can be referred to and followed as a guideline. Customization developers can implement bank's specific requirements on similar lines.

For example, in LCM022 'Perfection Capture' screen, the details section is shown with the additional fields which are defined for a particular location.

A	ccount 🔻	Back Offi	ce 🔻	CASA 🔻	Channel 🔻	Collection -	LCM 🕶	Loan 🔻	Origination 👻	Party 🔻	Payment And	d Collection 👻	Securi	‰☆	🗙 - F F	ast 🔤	
	🔒 LC	M <u>0</u> 22															
	Perfecti	on Capti	ure														
											📖 <u>R</u> e	ad + Cr <u>e</u> ate	🗎 Prin	t 🗸 o	k 🤣 Cle	ar 🛛 C	ancel
F																	
																	*
	Perfect Cr	harge															
	Collater	ral Perfect	ion De	etails													
	Cha	arge Regis	tration	Number						* (	Charge Status	Proposed					
			Execu	tion Date		120				Title Docu	iments Status			•			
	*	Charge Re	egistra	tion Date		E.			Charg	ge Registra	ation Required	Yes					
		* Da	ate of S	Stamping		20				Stam	ping Required	Yes					
		* Regis	stration	n Amount	\$0.00		Al	JD									
1		* Sta	mping	g Amount	\$0.00		Al	JD									=
		End	Date	Changed		120				Pr	ocessed Date		120		1		
			Amen	ded Date		E.					Amended By						
			Remo	ved Date		20					Removed By						
				Token						Secure	ed Party Group						
		C	hange	Number							Origination						
	Ea	rlier Regie	tration	Number							mananuonai						
	La	mer Kegis	addon	Number	L										OK	)	
	View F	looumont	Dotail													J	
	. view L	Jocument	Details	5													
	View -	De 🛃	etach														
	Index Typ	be	Index	value													

### Figure 8–1 Perfection Capture Screen

# 8.1 Localization Implementation Architectural Change

Architecturally, the following points are considered:

- Localization package will be over and above the product.
- Customization packages will be over the Localization and the Product.
- Any changes done for Localization should ensure that future product changes as well as customization changes will work seamlessly without any impact.

The additional fields which get identified and developed as part of localization requirements are in its own project, package, configuration, constant files and tables.

For example, the typical flow of the above mentioned perfection attributes added as part of localization requirement is shown below:



#### Figure 8–2 Localization Implementation Architectural Change

The Package structure for the implementation is shown below:

### Figure 8–3 Package Structure



# 8.2 Customizing UI Layer

This section explains the customization of UI layer.

### 8.2.1 JDeveloper and Project Customization

For the customization of the UI layer, JDeveloper needs to be configured in the customizable mode as explained in the ADF Screen Customization Sections.

The example for the customization of the JDeveloper is described below:

CustomizationLayerValues.xml





Figure 8–5 Customization of the JDeveloper



#### adf-config.xml

If the changes are not reflecting, adf-config.xml needs to be opened from the application resources and *Configure Design Time Customization layer values* highlighted in the below image needs to be clicked. It will create a CustomizationLayerValues.xml inside MDS DT folder in application resources. All the content from

*<JDEVELOPER\_HOME>/jdeveloper/jdev/CustomizationLayerValues.xml* needs to be copied to this CustomizationLayerValues.xml. This is to ensure that the changes are reflected at the application level.

Figure 8–6 Configure Design Time Customization layer

Source JDeveloper 11g Release 1 - View.jws : con	n.ofss.fc.lz.au.ui.view.lcm.jpr	- • • × *
<u>File Edit View Application Refactor Sea</u>	rch <u>N</u> avigate <u>B</u> uild <u>R</u> un Versi <u>o</u> ning <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🖹 🗁 🖩 🗿 1 🤊 (* 1 X 🗎 🛍 1 🔾 • C	) -   💩 -   💩 🕍 🗮 -   🕨 - 🕸 -   🛕	💏 🕶 Search
Application × 🕲 BPM Pro × 🖓 × 📮	CustomizationLayerValues.xml × @adf-config.xml × @CustomizationLayerValues.xml ×	Component Palette ×
🍓 View 👻 🕞 🔹	0	All Pages
🗢 Projects 💽 🖏 🏹 + 🚟 +	Business Components	ee (
🕀 🛅 com.ofss.fc.lz.au.ui.model.lcm	MDS Configuration (1) Note that additional configuration can be edited manually in the source.	#cdata-section (XML)
com.ofss.fc.lz.au.ui.OptionCC	Controller 🛛 Customization Configuration: Match Path = "/"	#comment (XML)
com.ofss.fc.ui.core	View Use the following editor for cases where the customization configuration map to the global namespace ("/").	#processing-instruction (XML)
🖶 🛅 com.ofss.fc.ui.template	Contraction of course	A #text (XML)
com.ofss.fc.ui.view	oracle.adf.share.confo.UserCC	🚸 adf-adfm-config (http://xmlns.oracle
H. Com.otss.tc.ul.view.lcm	com.ofss.fc.lz.au.ui.optioncc.OptionCC	adf-controller-config (http://xmlns.o
Application Resources		BC4JConfig (http://xmlns.oracle.con
Connections		cache-config (http://xmlns.oracle.cc
Descriptors		cust-config (http://xmlns.oracle.com
application.xml	Configure Design Time Oustomization Laver Values	event-notification (http://xmlns.orac
cwallet.sso		file-type-config (http://xmlns.oracle)
jazn-data.xml		mds-component-config (http://xmlns
jps-config.xml		mds-config (http://xmlns.oracle.com
weblogic-application.xml		metadata-store (http://xmins.oracle
adf-config.xml		metadata-store-usage (http://xmins
connections.xml		namespace (nttp://xmins.oracle.com
services		See Decsistence-contro (http://xmins.ora)
B MDS DT		<pre></pre>
CustomizationLayerValues.xml		Property Inspector ×
		🔣   📌 📴   🥒 🍈 Find
🕨 Data Controls 🛛 🖓 🍸 🗌		
Recently Opened Files		
ScustomizationLayerValues.xml - Structure ×		
📌 🗟		
Harnings (1)		
This file defines all valid customization layer value	Overview   Source   History   <	
Erw cust-layers	EMessages - Log X Simulations X Documentation X View - Customization Context X	
⊞ Cust-layer	Loaded global customization layer values from D:\R2BOr, O View without Customizations	
Customization layers that are only meant for	Loaded customization layer values for View from D:\R2B( ) Edit with following Customization Context	
	WARNING: Project "com.ofss.fc.lz.au.ui.view.lcm.jpr" AI Tplayer Name Value	-
1		

Figure 8–7 Enabling Seeded Customization



### Libraries and Classpath

In the "*Libraries and Classpath*" section, the previously deployed *com.ofss.fc.lz.au.ui.OptionCC.jar* containing the customization class then needs to be added.

😓 Oracle JDeveloper 11g Release 1 - View.jws : com.ofss.fc.ui.view.jpr \_ 0 %^ <u>F</u>ile <u>E</u>dit <u>V</u>iew <u>Application</u> Refa<u>c</u>tor <u>S</u>earch <u>N</u>avigate <u>B</u>uild <u>R</u>un Versi<u>o</u>ning <u>T</u>ools <u>W</u>indow <u>H</u>elp 🕐 🔁 🗃 🖓 🔍 I 🗶 🛄 😋 Project Properties - D:/R280rade\LZNview\com.ofssfc.ui.view\com.ofssfc.ui.view.jpr **8**-× Application × 🗟 Database Navig... ette 👸 Seard Libraries and Classpath 🗄 View Projects O Use Custom Settings Com.ofss.fc.lz.au.ui.view.lcm Use Project Settings ADF Model
 ADF View
 ADF View
 Ant
 B- Ant
 B- Business Components
 Dependencies Java SE Ve 1.6.0\_29 (Default) Change.. Classpath Entries: Export Description P ADF Swing Runtime P ADF Designitime API P ADF Designitime API P ADF Designitime API P ADF Designitime API P ADF Nodel Runtime P A Classpath Entries: Add Library... Deployment EJB Module Add JAR/Directory.. - Extension - Javadoc - Java EE Application - JSP Tag Libraries Remove View - JSP Visual Editor - Libraries and Classpath - Resource Bundle Share As... Move Up Run/Debug/Profile Technology Scope Move Do<u>w</u>n Com.ofss.fc.lz.au.ui.Op ADF Management Pages
 Oracle ADF DataTag
 DataTag
 BC4) HTML
 BC41 Struts Runtime
 Commons Digester 1.8
 Struts Runtime
 BI Data Control Runtime
 EJB SOO Client
 Facelets Runtime  $\mathbf{\overline{\mathbf{x}}}$ Application Resources Data Controls Recently Opened Files Structure × WStack × Hat ADF St

Figure 8–8 Library and Class Path

### adf-config.xml

In the *Application Resources* tab, the *adf-config.xml* present in the *Descriptors/ADF META-INF* folder needs to be opened. In the list of *Customization Classes*, all the entries should not be removed and the *com.ofss.fc.lz.au.ui.OptionCC.OptionCC* class to this list needs to be added.

#### Figure 8–9 MDS Configuration

ᇰ Oracle JDeveloper 11g Release 1 - View.jws : com.ofss.fc.ui.vie	w.jpr : D:\R2BOracle\LZN\view	v\.adf\META-INF\adf-config.xml		- I <u>- ×</u> ·
<u>File Edit View Application Refactor Search Naviga</u>	ite <u>B</u> uild <u>R</u> un Versi <u>o</u> ni	ng <u>T</u> ools <u>W</u> indow <u>H</u> elp		
• 🔒 : - O - O : 🏥 🗮 🗶 : 🔊 - O - : 🏤 •	📥 🕍 🚢 🛥 •   🕨 •	🅸 - 🔲> 🗐 🗐 JI 🕼 II- II- 👔 I 🛆 I ᇘ 🦳 🧶 🏉 💬		💏 • Search
Physication × Database Navig × 🕲 × 🎼 × 🗉	🕺 adf-config.xml 🗴			Component Palette ×
🔁 View 🔻 🖼 🔻			? ^	All Pages
🗢 Projects 💽 🖏 🏹 + 🚟 + 🗌	Business Components			<b>69</b> (
com.ofss.fc.lz.au.ui.view.lcm	MDS Configuration	<ol> <li>Note that additional configuration can be edited manually in the source.</li> </ol>		() #cdata-section (XMI.)
com.ofss.fc.ui.view	Controller	🖃 Customization Configuration: Match Path = "/" 🛛 💠 💥		#comment (XML)
	View	Use the following editor for cases where the customization configuration map to the global		#processing-instruction (XML)
		namespace ("/").		A #text (XML)
		Customization Classes		🕸 adf-adfm-config (http://xmlns.oracle.com/adfm
		orade.adf.share.config.UserCC		adf-controller-config (http://xmlns.oracle.com/z
		com.ofss.fc.lz.au.ui.optioncc.OptionCC		BC43Config (http://xmlns.oracle.com/bc4j/confi
		4		Cache-config (http://xmlns.oracle.com/mds/con
		2		cust-config (http://xmins.oracle.com/mds/config
				fie-type-config (http://xmins.oracle.com/mde/c
		Configure Design Time Customization Layer Values		me type comp (http://xmins.oracle.com
				mds-config (http://xmlns.oracle.com/mds/confic
✓ Application Resources				metadata-store (http://xmins.orade.com/mds/c
				metadata-store-usage (http://xmlns.oracle.com
Descriptors				namespace (http://xmlns.oracle.com/mds/config
application yml				persistence-config (http://xmlns.oracle.com/md
cwallet.sso				🔇 type-config (http://xmlns.oracle.com/mds/confi
				untyped-dependency-config (http://xmlns.orac
jps-config.xml				
ADE META-INE				
adf-config.xml				
connections.xml				
services				
			~	
Data Controls	Overview Source History <		>	
Recently Opened Files	ELog × ●≣Breakpoints ×	∰Smart Data ×  13]Data ×  13]Watches ×  ∞EADF Data ×  ≪ ×   → ×   → ×	_	
<	(#	Act	tions •	
Structure × WStack × ADF Structure × K. ×	at <u>weblogic</u>	.servlet.internal.WebAppServletContext\$ServletInvocationAction.wra	pRu^	
	at weblogic	.servlet.internal.WebAppServletContext\$ServletInvocationAction.run	(We	

Jdeveloper is then restarted and the entry needs to be checked for *com.ofss.fc.lz.au.ui.OptionCC*. If the jar entry is not reflecting, then source needs to be clicked and the entry as highlighted and shown in the below image needs to be manually added.

#### Figure 8–10 MDS Configuration



### 8.2.2 Generic Project Creation

After creating the Customization Layer, Customization Class and enabling the application for Seeded Customizations, the next step is to create a project which will hold the customizations for the application. Generic project is then created with the following technologies:

- ADF Business Components
- Java
- JSF
- JSP and Servlets

Following jars must then be added to the Project Properties and in the classpath:

- Customization class JAR (com.ofss.fc.lz.au.ui.OptionCC.jar)
- The project JAR which contains the screen / component to be customized. For example, if you want to customize the Collateral Perfection Capture screen, the related project JAR is com.ofss.fc.ui.view.lcm.jar.
- All the dependent JARS / libraries for the project needs to added.
- Finally newly created project (example: 'com.ofss.fc.lz.au.view.lcm') needs to be enabled for Seeded Customizations.

### 8.2.3 MAR Creation

After implementing customizations on objects from an ADF library, the customization metadata is stored by default in a subdirectory of the project called *libraryCustomizations*. Although ADF library customizations at

the project level is created and merged together during packaging to be available at the application level at runtime. Essentially, ADF libraries are JARs that are added at the project level, which map to library customizations being created at the project level. However, although projects map to web applications at runtime, the MAR (which contains the library customizations) is at the EAR level, so the library customizations are seen from all web applications.

Therefore, an ADF library artifact are customized in only one place in an application for a given customization context (customization layer and layer value). Customizing the same library content in different projects for the same customization context would result in duplication in MAR packaging. To avoid duplicates that would cause packaging to fail, customizations are implemented for a given library in only one project in your application.

### Step 1

Select the Application Properties.



#### Figure 8–11 MAR Creation

#### Step 2

Import com.ofss.fc.lz.au.ui.view.lcm project into application. Click Application Menu and select Application Properties.



Figure 8–12 MAR Creation - Application Properties

Select Deployment and click New.

Grade JDeveloper 11g Release 1 - Viewjws : comofss.fclz.au.ui.view.kmjpr	
File Edit View Application Refactor Search Navigate Build Run Versioning Tools Window Help	
■ 22 目前:22 - 22 - 12 - 12 - 12 - 12 - 12 - 12 -	( 💏 • Search )
Application X @ Database N X @ X @ X . eraPerfectionCapture.jsff X @ CollateraPerfectionCapture?ageDef.sml.xml X @ CollateraPerfectionCapture.jsff.xml X	Component Palette X
📓 Vew 🔹 🖥 🛛 🦓 🖓 🕅	M. •
Projects     Q R 7 • S •       Ords:custonization version="11.1.1.60.13"	Broutt O
Com.ofss.fclz.au.ui.model.cm     Xalns:ads="http://xalns.oracle.coa/ads">     Xalns:ads="http://xalns.oracle.coa/ads"     XAlns:Ads="http://xalns.oracle.coa/ads"	⊉cdata-section
e-G com.ofss.fc.lz.au.ul.OptonCC	₽ #comment
B Connofss.fc.ul.view Berch Deployment	#processing-instruction
Application Create Deployment Profile	A ≠text
- Custom	
Click OK to create your new deployment profile and immediately open it to see its configuration.     Reposito	
Resource Archive Type:	
B-Run Prekrie WS Pale EAR File	
MAR Fie Delete	
SOA Bunde	
gestripion: Creates a profile for deploying a metadata MAR file.	
Help OX Carrel	
Paride whether to micrate the following security objects	Property Inspector X
V Lisers and groups	Si 🖉 📓 🖉 (👸 Find 🛛 🕹 😭 🔇
<u>Hep</u> OK Cance	
> Application Resources	
> Data Controls	

Figure 8–13 MAR Creation - Create Deployment Profile

Select the MAR File option.

Stracle JDeveloper 11g Release 1 - View.jws : com.	fss.fc.ui.view.jpr	_ 0 _ X
<u>File Edit View Application Refactor Searce</u>	h <u>N</u> avigate <u>B</u> uild <u>R</u> un Versi <u>o</u> ning <u>T</u> ools <u>Wi</u> ndow <u>H</u> elp	
🔋 🗁 🖩 🎒 1 🤊 (* 1 🗶 🖩 🛍 1 O - O	- 1 🎄 - 1 🎄 📸 😹 - 1 🕨 - 🕸 - 📕 - + 💷 🏭 JI 💷 🕪 II 🟦 1 🛦 1 🕽 🖉 🏈 🏈 🕞	( Search
Application X 🗟 Database Navig X 🔞 X 🕼	x _	Component Palette X
🔁 View		
Projects     Q     Q     V	§•1	
- Com.ofss.fc.lz.au.ui.OptionCC	Application Departure - D/D280 restabil 7M view/View inc	
B-C com.ofss.fc.ui.view	Application Properties - Drinzboracie (Lzin (new (view, jws	
B C Application Sources	🛞 Search Deployment	
Resources	Applicat 🔍 Create Deployment Profile	
B com	- Customi	
🕀 🧰 css	Deploym Click OK to create your new deployment profile and immediately open it to see its configuration.	
B images	Resource Edt	
B WebHelp	B-Run MAR File Vew	
🛞 🧰 WEB-INF	WS Pold Name:	
Page Flows     About to inff	27MAR	
Default.jsff	Description:	
error.html	Creates a profile for deploying a metadata MAR file.	
login.html	e la companya de la c	
main.ispx		
	Hep OK Cancel	
		No Components
	Decide whether to migrate the following security objects.	
	✓ Users and Groups	
	OX Cancel	

Figure 8–14 MAR Creation - MAR File Selection

Select MAR from Archive Type and give a name ending with MAR and click **Ok**.



S Oracle JDeveloper 11g Release 1 - View.jws : com.ofss.fcui.view.jpr	
<u>File Edit View Application Refactor Search Navigate Build Run Versioning Iools Window H</u> elp	
월 28 월 1월 1월 1월 1월 2 월 2 8 8 1 월 28 월 4 9 1 월 2 <b>章 • 월 • 월 1</b> 8 월 1 월 1 월 1 월 1 월 1 월 1 월 2 8 월 2 8 년	( 🍓 • Search
<mark>ØAppication ×</mark> C Database Navig ×  ⊕ ×  ⊕ × .	Component Palette X
Projects         Projects	Ro Component Palette X

Select the ADF Library Customization for com.ofss.fc.lz.au.ui.view.lcm.

Oracle JDeveloper 11g Release 1 - View.jws : com.ofss.fcui.view.jpr	a long forward in a regulation from the stand for a	
File Edit View Application Refactor Search Navigate Build Run Versigning	g Iools Window Help 🌢 - 📕 - 🔶 🏭 📲 🚚 🕼 🕪 🛯 🏦 💧 😭 🍘 🏀 🏉 💽	( 🍓 • Search
Application ×       C Database Navig ×       S ×       S         Wew       ✓       S ×       S         Projects       S ⊗ ⊽ * S ×       S       S         S ⊚       On official state Antipology       S ×       S         S ⊚       S ∞       S ⊗ ⊽ * S ×       S         S ⊚       S ∞       S ⊗ ⊽ * S ×       S         S ⊚       S ∞       S ⊗ ⊽ * S ×       S         S ⊚       S ⊗ ∞       S ⊗ ∞       S ⊗ ∞	(Barley)	Component Palette X
Content      Content	ADF Library Customization Sources for com.ofss.fc.lz.au.ui.view.kcm         Ele Group Name:       ADF Library Customization Sources for com.ofss.fc.lz.au.ui.view.lcm         Target Directory in Archive:	zations Add Remove Move Up
HHR Root Dir for com.ofss.fc.uk.view     Directories     Directories     Directories     More large / Customization Sources for com.ofss.fc.uk.view     Directories     Mercen     Delete     Heb		Moye Down

Figure 8–16 MAR Creation - ADF Library Customization

Select the project for which Library Customization will be included in MAR (com.ofss.fc.lz.au.ui.view.lcm) and click **OK.** 

### Step 8

Select View (EAR File) and click Edit.

Figure 8–17 MAR Creation - Edit File



Select Application Assembly and check the created MAR (IznMAR) and click ok on defaults.



Figure 8–18 MAR Creation - Application Assembly

# 8.3 Source Maintenance and Build

This section describes the source maintenance and build details.

### 8.3.1 Source Check-ins to SVN

Along with UI and middleware source maintenance, there is a set of metadata files required to be packaged in the deployable packages in order for customization. When performing any changes to a product screen in "customization mode" the corresponding <screen filename>.xml gets generated. In case of taskflows, the metadata file is <page definition filename>.xml. The path structure is provided in the below table.

<i>For</i> For page definit ion	Path Structure
File name (with path)	adfmsrc/com/ofss/fc/ui/view/lcm/collaterals/collateralPerfectionCapture/ pageDefn/CollateralPerfectionCapturePageDef.xml
Meta- data file name	com\ofss\fc\ui\view\lcm\collaterals\collateralPerfectionCapture\ pageDefn\mdssys\cust\option\LZ\CollateralPerfectionCapturePageDef.xml.xml

Та

(with path)	
For Scree ns	
File name (with path)	com/ofss/fc/ui/view/lcm/collaterals/collateralPerfectionCapture/form/CollateralPerfectionCaptur e.jsff
Meta- data file name (with path)	com\ofss\fc\ui\view\lcm\collaterals\collateralPerfectionCapture\form\mdssys\cust\option\LZ\Col lateralPerfectionCapture.jsff.xml

These meta-data sources are checked into the METADATA folder in the product SVN under the localization path. During deployment, the EAR implementing these customizations must include these above mentioned sources in a .mar file.

### 8.3.2 .mar files Generated during Build

The localization specific build will include a last step, which is creation of .mar (metadata archive) file from the files checked-in the METADATA folder. This step will create separate .mar files, based on the modules which these represent. These MAR files are then packaged inside the deployable application EAR (com.ofss.fc.ui.view.ear).

Typical mar files generated during build will follow the naming convention com.ofss.fc.lz.au.ui.view.<module>.mar. Example, com.ofss.fc.lz.au.ui.view.pc.mar

# 8.3.3 adf-config.xml

adf-config.xml stores design time configuration information. The cust-config section (under mds-config) in the file contains a reference to the customization class. As part of the build activity, this file needs to be placed in the path com.ofss.fc.ui.view.ear@/adf/META-INF/. Also the customization class should be available in the classpath during deployment.

# 8.4 Packaging and Deployment of Localization Pack

In the OBP application, different projects will be shipped in the form of library jars which can be customized and the new localization-specific application libraries can be created. In the application, the assembly has been specifically modularized to take care of multiple localizations by prevention of mix-up of jars. The naming convention for the jars can be defined for different clients differently.

The new customized jars for hosts and UI needs to be packed with the original jars in the EAR files which will be deployed on the server. Let's say, we are creating the extension hooks of 'obp.host.app.domain' jar, then the separate jars can be defined as 'lz.au.obp.host.app.domain' and 'lz.us.obp.host.app.domain' for Australia and US respectively.

The similar structure can also be maintained for the other applications across UI and SOA channels. 'lz.au.obp.ui.domain' can be defined for the customized jar of the project 'obp.ui.domain'. The new customized jars for hosts and UI are packed below with the original jars in the EAR files which will be deployed on the servers.




# **9 Deployment Guideline**

This chapter explains the deployment guidelines.

## 9.1 Customized Project Jars

The customized extension projects are to be bundled in the different extensibility jars which are required to be added in the extensibility.

### 9.2 Database Objects

User has to update the corresponding seed data for the implementation of different extensibility features.

## 9.3 Extensibility Deployment

The new customized extensibility jars will be added in the extensibility libraries as ext.obp.host.domain for the host middleware layer, ext.obp.ui.domain for UI or presentation layer and ext.obp.soa.domain for the SOA layer. These extensibility application libraries will be packaged and shipped as the separate library folders along with the original library folders so that the extensibility feature can be added.

The OBP deployed applications shall reference these libraries so that customization jars included into these get automatically referenced in the corresponding EAR and WAR files.

#### Figure 9–1 Extensibility Deployment

