Oracle® Banking Platform

Extensibility Guide Release 2.3.0.0.0 E56276-01

July 2014



Oracle Banking Platform Extensibility Guide, Release 2.3.0.0.0

E56276-01

Copyright © 2011, 2014, Oracle and/or its affiliates. All rights reserved.

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.

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

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

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

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

Contents

Preface	xix
Audience	xix
Documentation Accessibility	xix
Related Documents	xix
Conventions	хх

1 Objective and Scope

1.1	Overview	1-1
1.2	Objective and Scope	1-1
1.2.1	Extensibility Objective	1-1
1.2.2	Document Scope	1-2
1.3	Complementary Documentation	1-3
1.4	Out of Scope	1-3

2 Overview of Use Cases

2.1	Extensibility Use Cases	2-1
2.1.1	Extending Service Execution	2-1
2.1.2	OBP Application Adapters	2-2
2.1.3	User Defined Fields	2-3
2.1.4	ADF Screen Customization	2-4
2.1.5	SOA Customization	2-5
2.1.6	Batch Framework Extension	2-6
2.1.7	Uploaded File Processing	2-6
2.1.8	Alert Extension	2-7
2.1.9	New Reports Creation	2-8
2.1.10	Security Customization	2-9
2.1.11	Loan Schedule Computation Algorithm	2-11
2.1.12	Print Receipt Functionality	2-11
2.1.13	Facts and Business Rules	2-12
2.1.14	Composite Application Service	2-12
2.1.15	ID Generation	2-13
2.1.16	OCH Integration	2-13

3 Extending Service Executions

3.1 Service Extension – Extending the "app" Layer	3-1
---	-----

3.1.1	Application Service Extension Interface	3-3
3.1.2	Default Application Service Extension	3-3
3.1.3	Application Service Extension Executor	3-4
3.1.4	Extension Configuration	3-6
3.2	Extended Application Service Extension – Extending the "appx" Layer	3-7
3.2.1	Extended Application Service Extension Interface	3-8
3.2.2	Default Implementation of Appx Extension	3-9
3.2.3	Configuration	3-10
3.2.4	Extended Application Service Extension Executor	3-11
3.3	End-to-End Example of an Extension	3-13

4 **OBP Proxy Extension**

5 **OBP Application Adapters**

5.1	Adapter Implementation Architecture	5-1
5.1.1	Package Diagram	5-1
5.1.2	Adapter Mechanism Class Diagram	5-3
5.1.3	Adapter Mechanism Sequence Diagram	5-3
5.2	Examples of Adapter Implementation	5-4
5.2.1	Example 1 – EventProcessingAdapter	5-4
5.2.2	Example 2 – DispatchAdapter	5-5
5.3	Customizing Existing Adapters	5-6
5.3.1	Custom Adapter Example 1 – DispatchAdapter	5-6
5.3.2	Custom Adapter Example 2 – PartyKYCCheckAdapter	5-7

6 User Defined Fields

6.1	Enabling UDF for a Particular Screen	6-1
6.1.1	UDF Metadata	6-1
6.1.2	Seed Data for the Task Codes	6-3
6.1.3	Screen Changes for Incorporating UDF	6-3
6.1.4	Linking of UDF to a Screen (Taskflow Code)	6-4
6.2	Control Flow for UDF	6-4
6.2.1	Initial Screen Load	6-4
6.2.2	Extracting UDF Values on Submission	6-4
6.2.3	Handling the Fetch of UDF Values	6-7
6.2.4	UDF Enabling Special Cases	6-9
6.2.5	Tips for Trouble Shooting	6-10
6.3	Limitations and Special Cases	6-11

7 ADF Screen Customizations

7.1	Seeded Customization Concepts	7-1
7.2	Customization Layer	7-2
7.3	Customization Class	7-2
7.4	Enabling Application for Seeded Customization	7-4
7.5	Customization Project	7-7
7.6	Customization Role and Context	7-7

7.7	Customization Examples	7-10
7.7.1	Adding a Validator to Input Text Component	7-10
7.7.2	Adding a UI Table Component to the Screen	7-17
7.7.3	Adding a Date Component to a Screen	7-30
7.7.4	Removing existing UI components from a screen	7-64

8 SOA Customizations

8-1
8-2
8-3
8-4
8-4
8-19

9 Batch Framework Extensions

9.1	Typical Business Day in OBP	9-1
9.2	Overview of Categories	9-2
9.2.1	Beginning of Day (BOD)	9-2
9.2.2	Cut-off	9-2
9.2.3	End of Day (EOD)	9-2
9.2.4	Internal EOD	9-3
9.2.5	Statement Generation	9-3
9.2.6	Customer Communication	9-3
9.3	Batch Framework Architecture	9-3
9.3.1	Static View	9-3
9.3.2	Dynamic View	9-4
9.4	Batch Framework Components	9-6
9.4.1	Category Components	9-6
9.4.2	Shell Components	9-7
9.4.3	Stream Components	9-8
9.4.4	Database Components	9-10
9.5	Batch Configuration	9-11
9.5.1	Creation of New Category	9-11
9.5.2	Creation of Bean Based Shell	9-13
9.5.3	Creation of Procedure Based Shell	9-17
9.5.4	Population of Other Parameters	9-18
9.6	Batch Execution	9-20

10 Uploaded File Data Processing

10.1	Configuration	10-2
10.1.1	Database Tables and Setup	10-2
10.1.2	File Handlers	10-6
10.1.3	Record Handlers for Both Header and Details	10-7
10.1.4	DTO and Keys Classes for Both Header and Details	10-8
10.1.5	XFF File Definition XML	10-10
10.2	Processing	10-12

10.2.1	API Calls in the Handlers	10-13
10.2.2	Processing Adapter	10-14
10.3	Outcome	10-15
10.4	Failure/Exception Handling	10-16

11 Alerts Extension

11.1	Transaction as an Activity	11-1
11.1.1	Activity Record	11-1
11.1.2	Attaching Events to Activity	11-2
11.1.3	Event Record	11-2
11.1.4	Activity Event Mapping Record	11-3
11.1.5	Activity Log DTO	11-4
11.1.6	Alert Metadata Generation	11-4
11.1.7	Alert Message Template Maintenance	11-7
11.1.8	Alert Maintenance	11-8
11.2	Alert Subscription	11-9
11.2.1	Transaction API Changes	11-10
11.3	Alert Processing Steps	11-12
11.4	Alert Dispatch Mechanism	11-15
11.5	Adding New Alerts	11-18
11.5.1	New Alert Example	11-19
11.5.2	Testing New Alert	11-20

12 Creating New Reports

12.1	Data Objects for the Report	12-1
12.2	Catalog Folder	12-4
12.3	Data Source	12-5
12.4	Data Model	12-5
12.5	XML View of Report	12-9
12.6	Layout of the Report	12-10
12.7	View Report in BIP	12-11
12.8	OBP Batch Report Configuration - Define the Batch Reports	12-12
12.9	OBP Batch Report Configuration - Define the Batch Report Shell	12-12
12.10	OBP Batch Report Configuration - Define the Batch Report Shell Dependencies	12-12
12.11	OBP Batch Report Configuration	12-13
12.11.1	Batch Report Generation for a Branch Group Code	12-13
12.11.2	2 Batch Report Generation Status	12-14
12.11.3	Batch Report Generation Path	12-14
12.12	OBP Adhoc Report Configuration	12-15
12.12.1	Define the Adhoc Reports	12-15
12.12.2	2 Define the Adhoc Report Parameters	12-15
12.12.3	B Define the Adhoc Reports to be listed in Screen	12-16
12.12.4	Adding Screen Tab for Report Module	12-16
12.13	Adhoc Report Generation – Screen 7775	12-17
12.14	Adhoc Report Viewing – Screen 7779	12-18

13 Security Customizations

13.1 (OPSS Access Policies – Adding Attributes	13-3
13.1.1	Steps	13-3
13.2 (DAAM Fraud Assertions – Adding Attributes	13-5
13.2.1	Steps	13-6
13.3 1	Matrix Based Approvals – Adding Attributes	13-7
13.4	Security Validators	13-7
13.4.1	Customer Validators	13-8
13.4.2	Account Validators	13-8
13.4.3	Business Unit Validators	13-9
13.5 (Customizing User Search	13-9
13.5.1	Steps	13-9
13.6 (Customizing One-Time-Password (OTP) Processing Logic	13-10
13.6.1	Steps	13-10
13.7	Customizing Role Evaluation	13-10
13.7.1	Steps	13-10
13.8 (Customizing Limits Exclusions	13-10
13.8.1	Steps	13-11
13.9 (Customizing Business Rules	13-11
13.9.1	Steps to Update the Business Rules by Browser	13-11
13.9.2	Steps to Update the Business Rules in JDeveloper	13-20
Loan S	Schedule Computation Algorithm	

14.1	Adding a New Algorithm	14-1
14.2	Consuming Third Party Schedules	14-4

15 Receipt Printing

14

15.1	Prerequisite	15-1
15.1.1	Identify Node Element for Attributes in Print Receipt Template	15-1
15.1.2	Receipt Format Template (.rtf)	15-3
15.2	Configuration	15-4
15.2.1	Parameter Configuration in the BROPConfig.properties	15-4
15.2.2	Configuration in the ReceiptPrintReports.properties	15-5
15.3	Implementation	15-5
15.3.1	Default Nodes	15-6
15.4	Special Scenarios	15-6

16 Facts and Rules Configuration

16.1	Facts	16-1
16.1.1	Type of Facts	16-1
16.1.2	Facts Vocabulary	16-2
16.1.3	Generation of Facts using Eclipse Plug-in	16-3
16.2	Business Rules	16-22
16.2.1	Rules Engine	16-22
16.2.2	Rules Creation by Guided Rule Editor	16-23

16.2.3	Rules Creation By Decision Table	16-24
16.2.4	Rules Storage	16-25
16.2.5	Rules Deployment	16-25
16.2.6	Rules Versioning	16-25
16.3	Rules Configuration in Modules	16-26
16.3.1	Generic Rules Configuration	16-26
16.4	Rules Migration	16-29
16.4.1	Rules Configured for Modules	16-29

17 Composite Application Service

17.1	Composite Application Service Architecture	17-2
17.2	Multiple APIs in Single Module	17-2

18 ID Generation

18.1	Database Setup	18-2
18.1.1	Database Configuration	18-3
18.2	Automated ID Generation	18-3
18.3	Custom ID Generation	18-6

19 Extensibility of Domain Objects - Dictionary Pattern

19.1	Customized Domain Object Attribute Placeholders	19-2
19.2	Customized Domain Object DTO Interceptor in UI Layer	19-3
19.2.1	Interceptor Hook to Persist Customized Domain Object Attributes	19-3
19.2.2	Interceptor Hook to Fetch Customized Domain Object Attributes	19-4
19.3	Dictionary Data Transfer from UI to Host	19-5
19.3.1	Customized Domain Object DTO Transfer from UI to Host	19-5
19.3.2	Customized Domain Object DTO transfer from Host to UI	19-9
19.4	Translating Dictionary Data into Custom Domain Object	19-13
19.4.1	Instantiation and Persistence of Custom Domain Objects	19-13
19.4.2	Fetching of Customized Domain Objects	19-14
19.5	Customized Domain Object ORM Configuration	19-15
19.5.1	Case 1 - Non-Inheritance based mapping	19-15
19.5.2	Case 2 - Mapped as a Hibernate Subclass	19-18
19.5.3	Case 3 - Mapped as a Hibernate Union-Subclass or Joined-Subclass	19-20
19.5.4	Case 4 - Mapped as a Hibernate Component	19-23
19.6	Extensibility using Dictionary in Origination Application	19-23
19.6.1	ICustomDataHandler's as DictionaryArray Interceptor	19-23
19.6.2	Create Customized Abstract Domain Object Class	19-24
19.6.3	Create Customized Abstract Domain Object Hibernate Mapping File	19-25
19.6.4	Create Customized Abstract Domain Object Attribute Columns	19-26
19.7	Extensibility using Attributes of Various Supported Datatypes	19-26

20 Deployment Guideline

20.1	Customized Project Jars	20-1
20.2	Database Objects	20-1
20.3	Extensibility Deployment	20-1

21 Extensibility Usage – OBP Localization Pack

21.1	Localization Implementation Architectural Change	21-2
21.2	Customizing UI Layer	21-4
21.2.1	JDeveloper and Project Customization	21-4
21.2.2	Generic Project Creation	21-9
21.2.3	MAR Creation	21-9
21.3	Source Maintenance and Build	21-17
21.3.1	Source Check-ins to SVN	21-17
21.3.2	.mar files Generated during Build	21-18
21.3.3	adf-config.xml	21-18
21.4	Packaging and Deployment of Localization Pack	21-18

22 OCH Integration

22.1	Integration Adapter Interface	22-2
22.2	Abstract Integration Adapter Class	22-2
22.3	Sample Integration Adapter	22-3
22.4	Integration Abstract Assembler	22-4
22.5	Sample Assembler	22-5

A Appendix

List of Figures

	Extending Service Execution	2-2
2–2	OBP Application Adapters	2-3
2–3	Configure User Defined Fields	2-3
2–4	ADF Screen Customization	2-5
2–5	SOA Customization	2-5
2–6	Batch Framework Extension	2-6
2–7	Upload File Processing	2-7
2–8	Alerts Extension	2-8
2–9	Creating New Reports	2-9
2–10	Security Customization	2-10
2–11	Loan Schedule Computation Algorithm	2-11
2–12	Print Receipt Functionality	2-11
2–13	Facts and Business Rules	2-12
2–14	Composite Application Service	2-13
2–15	ID Generation	2-13
2–16	OCH Integration	2-14
3–1	Standard Set of Framework Method Calls	3-2
3–2	Extension Hook for DocumentTypeApplicationService	3-3
3–3	Default Application Service Extension	3-4
3–4	Application Service Extension Executor	3-5
3–5	ExtensionFactory Hook for DocumentTypeApplicationService	3-5
3-6	Eactory Implementation of Extension Hook for DocumentType ApplicationService	3-6
3-7	Extended Application Service Extension	3-7
3-8	Extended Application Service Extension - Post and Pre Hook	3-8
3-9	Extension Hook for DocumentTypeApplicationServiceSpi	3-9
3-10	Default Implementation of Appy Extension	3-10
3_11	Extended Application Service Extension Executor	3-11
3_12	Extension Factory Hook for Document Type Application Service Spi	3-12
3-12	Eactory Implementation of Extension Hook for DocumentType ApplicationServiceSpi	0-12
0 10	3-13	•••••
3–14	Maintenance of Document Types	3-14
3_15	DocumentTypeApplicationServiceSpiExt - Appy Laver	0 1
3_16	DOCUMENTITY per application of a recopilizat - rappy Layer	3-15
	DocType Application Service SpiExt - Appy Laver	3-15
3_17	DocTypeApplicationServiceSpiExt - Appx Layer	3-15 3-16 3-17
3–17 3–18	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer	3-15 3-16 3-17 3-18
3–17 3–18 5–1	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram	3-15 3-16 3-17 3-18
3–17 3–18 5–1	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram	3-15 3-16 3-17 3-18 5-2 5-3
3–17 3–18 5–1 5–2 5–3	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram	3-15 3-16 3-17 3-18 5-2 5-3 5-4
3–17 3–18 5–1 5–2 5–3 5–4	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8
3–17 3–18 5–1 5–2 5–3 5–4	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Dofault Implementation of IParty/KYCCheck Adapter Interface	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8 5-8
3–17 3–18 5–1 5–2 5–3 5–4 5–5	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface VCC Adapter Factory with Mocking Support	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8 5-8 5-8
3–17 3–18 5–1 5–2 5–3 5–4 5–5 5–6 6	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8 5-8 5-8 5-9 6 2
3–17 3–18 5–1 5–2 5–3 5–4 5–5 5–6 6–1	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support UDF Metadata	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8 5-8 5-8 5-8 5-9 6-2
3–17 3–18 5–1 5–2 5–3 5–4 5–5 5–6 6–1 6–2	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support UDF Metadata Data Stored into the FCRThreadAttribute	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8 5-8 5-8 5-8 5-9 6-2 6-3 6-3
3-17 3-18 5-1 5-2 5-3 5-4 5-5 5-6 6-1 6-2 6-3 6-4	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support UDF Metadata Data Stored into the FCRThreadAttribute LinkedUDFDTO	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8 5-8 5-8 5-9 6-2 6-3 6-5
3-17 3-18 5-1 5-2 5-3 5-4 5-5 5-6 6-1 6-2 6-3 6-4 c	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support UDF Metadata Data Stored into the FCRThreadAttribute LinkedUDFDTO Extracting UDF DTO using instance of the LinkedUDFsHelper	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8 5-8 5-9 6-5 6-5 6-5
3-17 3-18 5-1 5-2 5-3 5-4 5-5 5-6 6-1 6-2 6-3 6-4 6-5 6-5	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support UDF Metadata Data Stored into the FCRThreadAttribute LinkedUDFDTO Extracting UDF DTO using instance of the LinkedUDFsHelper UIConfig.properties	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8 5-9 6-2 6-5 7 6-5 7 7 7 7 7 7 7 7
3-17 3-18 5-1 5-2 5-3 5-4 5-5 5-6 6-1 6-2 6-3 6-4 6-5 6-6 7	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support UDF Metadata Data Stored into the FCRThreadAttribute LinkedUDFDTO Extracting UDF DTO using instance of the LinkedUDFsHelper UIConfig.properties Package Level Interactions	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8 5-9 6-5 7-5
3-17 3-18 5-1 5-2 5-3 5-4 5-5 5-6 6-1 6-2 6-3 6-4 6-5 6-6 6-7 2	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support UDF Metadata Data Stored into the FCRThreadAttribute LinkedUDFDTO Extracting UDF DTO using instance of the LinkedUDFsHelper UIConfig.properties Package Level Interactions Sequence Diagram for UDF DTO	3-15 3-16 3-17 3-18 5-2 5-3 5-4 5-8 5-2 5-4 5-8 5-2 5-8 5-2 6-5 6-5 6-5 6-6 6-7 2
3-17 3-18 5-1 5-2 5-3 5-4 5-5 5-6 6-1 6-2 6-3 6-4 6-5 6-6 6-7 6-8 6-8	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface VDF Metadata Data Stored into the FCRThreadAttribute LinkedUDFDTO Extracting UDF DTO using instance of the LinkedUDFsHelper UIConfig.properties Package Level Interactions Sequence Diagram for UDF DTO Package Level Interactions	3 - 15 3 - 16 3 - 17 3 - 18 5 - 2 5 - 3 5 - 4 5 - 3 5 - 4 5 - 8 5 - 8 5 - 9 6 - 5 6 - 5 6 - 5 6 - 6 6 - 7 6 - 7 7 - 7
3-17 3-18 5-1 5-2 5-3 5-4 5-5 5-6 6-1 6-2 6-3 6-6 6-7 6-8 6-9	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support UDF Metadata Data Stored into the FCRThreadAttribute LinkedUDFDTO Extracting UDF DTO using instance of the LinkedUDFsHelper UIConfig.properties Package Level Interactions Sequence Diagram for UDF DTO Package Level Interactions Sequence Diagram	3 - 15 3 - 16 3 - 17 3 - 18 5 - 3 5 - 3 5 - 3 5 - 3 5 - 4 5 - 3 5 - 4 5 - 8 5 - 9 6 - 5 6 - 5 6 - 6 6 - 7 6 - 9 7 - 8 6 - 7 7 - 8 6 - 7 6 - 7 6 - 9 7 - 8 6 - 7 7 - 8 7 - 8
3-17 3-18 5-1 5-2 5-3 5-4 5-5 5-6 6-1 6-2 6-3 6-4 6-5 6-6 6-7 6-9 7-1	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support UDF Metadata Data Stored into the FCRThreadAttribute LinkedUDFDTO Extracting UDF DTO using instance of the LinkedUDFsHelper UIConfig.properties Package Level Interactions Sequence Diagram for UDF DTO Package Level Interactions Sequence Diagram for UDF DTO Package Level Interactions Sequence Diagram for UDF DTO Package Level Interactions Sequence Diagram Customization Application View	3 - 15 3 - 16 3 - 17 3 - 18 5 - 3 5 - 3 5 - 3 5 - 3 5 - 3 5 - 3 5 - 4 5 - 3 5 - 3 5 - 4 5 - 3 5 - 5 5 - 5 6 - 5 6 - 5 6 - 6 6 - 7 6 - 7 7 - 8 6 - 7 7 - 8 6 - 7 7 - 8 7 - 8 6 - 7 6 - 7 6 - 7 7 - 8 7 - 7 7 - 8 7 - 8 7 - 8 7 - 8 7 - 7 7 - 8 7 - 7 7 - 8 7 - 8 7 - 7 7 - 8 7 - 7 7 - 7 7 - 8 7 - 7 7 - 7
3-17 3-18 5-1 5-2 5-3 5-4 5-5 5-6 6-1 6-2 6-3 6-4 6-5 6-6 6-7 6-9 7-2 7-2	DocTypeApplicationServiceSpiExt - Appx Layer DocumentTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram	3 - 15 3 - 16 3 - 17 3 - 18 5 - 3 5 - 3 5 - 3 5 - 4 5 - 3 5 - 4 5 - 3 5 - 4 5 - 3 5 - 4 5 - 3 5 - 5 5 - 5 6 - 5 6 - 6 6 - 7 6 - 7 7 - 12 7
3-17 3-18 5-1 5-2 5-3 5-4 5-5 6-1 6-2 6-3 6-4 6-5 6-6 6-7 6-9 7-2 7-2 7-3	DocTypeApplicationServiceSpiExt - Appx Layer DocumenTypeApplicationServiceSpiExt - App Layer DocTypeApplicationServiceSpiExt - App Layer Package Diagram Adapter Mechanism Class Diagram Adapter Mechanism Sequence Diagram Party KYC Status Check Adapter Interface Default Implementation of IPartyKYCCheckAdapter Interface KYC Adapter Factory with Mocking Support UDF Metadata Data Stored into the FCRThreadAttribute LinkedUDFDTO Extracting UDF DTO using instance of the LinkedUDFsHelper UIConfig.properties Package Level Interactions Sequence Diagram for UDF DTO Package Level Interactions Sequence Diagram for UDF DTO Package Level Interactions Sequence Diagram for UDF DTO Package Level Interactions	3 - 15 3 - 16 3 - 17 3 - 18 5 - 3 5 - 3 5 - 3 5 - 3 5 - 4 5 - 8 5 - 2 5 - 3 5 - 5 5 - 5 5 - 5 5 - 5 5 - 5 6 - 5 6 - 5 6 - 6 6 - 7 7 - 2 7 - 2 7 - 2 7 - 3

7–5	Enable Seeded Customizations	7-5
7–6	Adding com.ofss.fc.demo.ui.OptionCC.jar	7-5
7–7	Adding com.ofss.fc.demo.ui.OptionCC.OptionCC	7-6
7–8	Adf-config.xml	7-6
7–9	Customization Developer	7-8
7–10	Selecting Always Prompt for Role Selection on Start Up	7-9
7–11	View Customization Context	7-10
7–12	Contact Point	7-11
7–13	DemoValidator.java	7-12
7–14	Managed Beans	7-12
7–15	Creating Managed Bean - Customization XML	7-13
7–16	Opening ISFF Screen - Show Libraries	7-13
7–17	Opening ISFF Screen - contactPoint.Isff	7-14
7–18	Bind Validator to Component - Validator Property	7-15
7–19	Bind Validator to Component - telNumberValidator	7-15
7-20	Bind Validator to Component - contactPoint isff xml	7-16
7_21	Contact Point screen	7-17
7_22	Adding a UI Table Component - Party Search screen	7-18
7_23	Adding a UI Table Component - Related Party screen	7-18
7-23	Creating Binding Boan Class	7-10
7 24	Create Event Consumer Class	7 20
7 20	Creating Managed Pean	7 20
7-20	Creating Manageu Dean	7-20
7 00	Adding View Object Dinding to Days Definition Add Tree Dinding	7 00
7-20	Adding View Object binding to Page Definition - Add Tree binding	7-22
7-29	Adding view Object Binding to Page Definition - Update Root Data Source	7-23
7-30	Page Data Binding Definition - Insert Item	7-24
7-31	Page Data Binding Definition - Create Action Binding	7-25
7-32	Edit Event Map	7-26
7-33	Event Map Editor	7-27
7-34	Add UI Components to Screen	7-28
7–35	Application Navigator	7-29
7–36	Party Search	7-30
7–37	Adding a Date Component	7-31
7–38	Create Table in Application Database	7-32
7–39	Create Java Project	7-32
7–40	Create Domain Objects	7-33
7–41	Create Interface	7-33
7–42	Create Class	7-34
7–43	Preferences - Service Publisher	7-34
7–44	Preferences - WorkSpacePath	7-35
7–45	Preferences - XML/JSON Facade	7-35
7–46	ApplicationService Generator	7-36
7–47	List of Classes Generated in the Project	7-36
7–48	ContactExpiryDTO. java file	7-37
7–49	Generate Service and Facade Layer Sources	7-38
7–50	ContactExpiryApplicationServiceSpi.java file before Modification	7-39
7–51	ContactExpiryApplicationServiceSpi.java file after Modification	7-39
7–52	Contact Expiry Application Service - Contact Point Transaction	7-40
7–53	Java Packages	7-40
7–54	Export Project as a Jar	7-41
7-55	Create Hybernate Mapping.	7-42
7-56	Adding an Entry in hostapplication layer properties file	7-42
7-57	Adding an entry in ProxyFacadeConfig properties file	7-43
7-58	Adding an entry in ISONServiceMap properties file	7-43
7_50	Create Model Project - ADF Model	7_4/
1 00	Create Model I Tojeet - ADI Model	, ++

7–60	Create Model Project - Click Finish	7-45
7–61	Create Application Module - ADF Business Components	7-46
7–62	Create Application Module - Set Package and Provide Name	7-46
7–63	Create Application Module - Summary	7-47
7–64	Create View Object - Provide Name	7-48
7–65	Create View Object - View Attribute	7-48
7–66	Create View Object - Application Module	7-49
7–67	Create View Object - Click Finish	7-49
7–68	Create View Controller Project - ADF View Controller Project	7-50
7–69	Create View Controller Project - Project Title	7-51
7–70	Create View Controller Project - Libraries and Classpath tab	7-52
7–71	Create View Controller Project - Dependencies Tab	7-52
7–72	Create an Interface	7-53
7–73	Create Update State Action Class	7-54
7–74	Create Update State Action Class - Service Exception	7-54
7–75	Create Backing Bean	7-55
7–76	Create Backing Bean - Save and Clear Method	7-56
7_77	Create Backing Bean - Contact Expiry DTO Method	7-56
7–78	Create Backing Bean - On Expiry Date Change	7-56
7_79	Create Backing Bean - Value Change Event Handler	7-57
7-80	Create Backing Bean - Contact Expire Proxy Service	7-57
7_81	Create Managed Bean - Register Demo Contact Point	7-58
7-82	Create Event Consumer Class	7-58
7-02	Create Data Control	7-50
7-03	Adding III to Scroops	7-60
7 95	Adding View Object Binding to Page Definition	7 60
7 96	Create Attribute Binding	7 61
7-00	Adding Mathad Action Binding	7 61
7 00	Adding Method Action Dinding Domo Darty Change Event Consumer	7 60
7-00	Edit Event Man of Dava Definition Edit Manning	7.60
7-09	Edit Event Map of Page Definition - Edit Mapping	7.62
7-90	Contact Deint concernment Deta Gold	7-03
7-91	Contact Point screen with Expiry Date field	7-04
7-92	Caracter A DE View Controller Project - Device Track a close	7-64
7-93	Create ADF view Controller Project - Project Technologies	7-05
7-94	Create View Controller Project - Libraries and Class Path	7-66
7-95	Modifications in the ActivityEventActionMaintenance.jsff.xml	7-67
7-96	Modified Alert Maintenance Screen	7-67
8-1	Add an entry for new Customization Layer	. 8-2
8-2	Create Customization Class	8-3
8-3	Enabling Application for Seeded Customization	. 8-4
8-4	Select SOA Project	. 8-5
8-5	Enter SOA Project Name	8-5
8–6	Configure SOA Settings	8-6
8–7	Create Mediator	. 8-7
8–8	Select Target Type	8-7
8–9	Request Transformation Map to create new mapper file	8-8
8–10	Mapping Input and Output string	8-8
8–11	Select Deployment Action	8-9
8–12	Deploy Configuration Settings	8-10
8–13	Select Deployment Server	8-10
8–14	Select Target SOA Server	8-11
8–15	Select SOA Domain	8-11
8–16	Test Web Service	8-12
8–17	Customization of SOA Application - Flow	8-13
8–18	Customization of SOA Application - Notify Customer	8-14

8–19	Add Partner Link Component	8-15
8–20	Add Invoke Component	8-16
8–21	Edit Copy Rules Variable	8-17
8–22	Add Assign Components - Reply	. 8-17
8–23	Design View of the BPEL Process	8-18
8–24	Test Customized Composite - Flow	8-19
8–25	Test Customized Composite - invokeEchoService	8-19
8–26	Select SOA Project	8-20
8-27	Create SOA Project Name	8-21
8-28	Configure SOA Settings	8-21
8-29	Configure BPEL Process Settings	8-22
8-30	Enter Human Task Name	8-22
8_31	Create Human Task - General Tab	8-23
8-32	Add Human Task Parameter	8-23
8_33	Create Human Task - Data Tab	8-23
0-00 8-34	Add Participant Type Dataile	8-24
0-04 8-25	Croate Human Task Assignment Tab	8-24
0-00	Cleate Human Task Parameters	0-24
0-30	Greate Human Task - Dalate Canditian	0.05
0-01	Create Human Task - Delete Condition	0-20
0-30	Create Human Task - Expression builder	0-20
0-39	Create Human Task - Copy Kules	0.07
8-40	Create Human Task - BPEL Process	. 8-27
8-41	Select Human Task Form	. 8-28
8-42	Select Human Task Form Deployment Action	8-29
8-43	Select Human Task Form - Weblogic Options	8-29
8-44	Add Customization Scope to SOA Application	8-30
8-45	Add Partner Link Component	8-31
8-46	Add Invoke Component	8-32
8-47	Add Receive Component using BPEL functions	. 8-33
8-48	Add Assign Component	. 8-34
8-49	Deploy and Test Customized SOA Composite - My Tasks Tab	. 8-35
8-50	Deploy and Test Customized SOA Composite - Flow	8-35
8–51	Deploy and Test Customized SOA Composite - Invoke Input	8-36
8–52	Deploy and Test Customized SOA Composite - Receive Output	8-36
9–1	Business Day in OBP	9-2
9–2	Batch Framework Architecture - Static View	9-4
9–3	Dynamic View Sequence Diagram	9-5
9–4	State Diagram of a Shell	9-6
9–5	Creation of New Category	. 9-13
9–6	Population of Other Parameters	. 9-18
9–7	Population of Other Parameters - General Tab	. 9-19
9–8	Population of Other Parameters - Connection Pool	. 9-19
9–9	Population of Other Parameters - Set IS_DB_RAC	. 9-20
9–10	Population of Other Parameters - Specify Data	9-20
9–11	Batch Execution	9-21
10–1	Uploaded Data File Processing Framework	. 10-2
10–2	File Handlers	. 10-7
10–3	Record Handlers for Both Header and Details	. 10-8
10–4	DTO and Keys Classes for Both Header and Details - HeaderRecDTOKey	. 10-9
10–5	DTO and Keys Classes for Both Header and Details - AbstractDTORec	10-10
10–6	XXF File Definition XML	10-12
10–7	API Calls in Adapters	10-14
10–8	Processing Adapter	10-15
11–1	Sample script for Activity Record	. 11-2
11–2	Sample script for Event Record	. 11-3

11–3	Activity Event Mapping Record	11-3
11–4	Activity Log DTO	11-4
11–5	Metadata Generation	11-5
11–6	Service Data Attribute Generation	11-6
11–7	Alert Message Template Maintenance	11-8
11–8	Alert Maintenance	11-9
11–9	Alert Subscription	11-10
11–10	Transaction API Changes - Service Call	11-10
11-11	Transaction API Changes - Conditional Evaluation	11-11
11–12	Transaction API Changes - persistActivityLog()	11-11
11–13	Transaction API Changes - Activity Log	11-11
11–14	Transaction API Changes - Register Activity	11-12
11-15	Alert Processing Steps	11-13
11-16	Event Processing Status Type	11-14
11_17	Batch Alerts	11-15
11_18	Alert Dispatch Mechanism	11-16
11_10	Alort Dispatch Mechanism - Dispatcher Factory	11_17
11-19	Alert Dispatch Mechanism - Dispatcher Pactory	11-17
12_1	Creating New Paparts	12_1
12-1	Clebal Temperature Table	12-1
12-2	Benert Record Type	12-2
12-3	Report Record Type	12-2
12-4	Report Table Type	12-3
12-5	Report DML Function	12-3
12-0	Report DDL Function	12-4
12-7	Catalog Folder	12-5
12-8	Data Source	12-5
12-9	Data Model	12-6
12-10	Data Set	12-7
12-11	Group Fields	12-7
12–12	XML Structure and Labels	12-8
12–13	XML Code	12-8
12–14	Add Input Parameters	12-9
12–15	XML View of Report	12-9
12–16	Layout of the Report - Create Layout	12-10
12–17	Layout of the Report - Batch Job Results	12-11
12–18	View Report in BIP	12-11
12–19	Batch Report Generation for a Branch Group Code	12-14
12–20	Batch Report Generation Path	12-15
12–21	Adhoc Report Generation - Report Request	12-17
12–22	Adhoc Report Generation - Report Generated	12-17
12–23	Advice Report	12-18
12–24	View Generated Adhoc Report	12-19
13–1	Security Customizations Interface	13-2
13–2	Security Use Case with Access Checks and Assertions	13-3
13–3	Add Attributes to Access Policy Rule	13-4
13–4	Attribute to Access Policy Rule - Authorization Management	13-4
13–5	Add or Modify Access Policy Rule	13-5
13–6	Add or Modify Fraud Rules in OAAM - Data Tab	13-6
13–7	Add or Modify Fraud Rules in OAAM - Conditions Tab	13-7
13–8	Log in to BPM Worklist Application screen	13-12
13–9	Task Configuration tab.	13-13
13–10	Stages of Approval	13-14
13–11	Select Test Condition	13-15
13–12	Select Values	13-16
13–13	Select Specific Task	13-17
	1	

13–14	Update Values	13-18
13–15	Save the Updated Rule	13-19
13–16	Commit the Changes	13-20
13–17	Expand Business Rules	13-21
13–18	Create New Stage	13-22
13–19	Add New Rule	13-23
13–20	Populate the New Rule	13-24
13–21	Deploy Project Jar	13-25
14–1	Add New Algorithm	14-1
14–2	Create New Installment	14-2
15–1	Input Property Files	15-2
15–2	Build Path of Utility	15-2
15–3	Utility Execution	15-3
15–4	Excel Generation	15-3
15-5	Receipt Format Template	15-4
15-6	Receipt Print Reports	15-5
15-7	Sample of Print Receipt	15-6
16_1	Select Window Preferences	16-4
16-2	Window Preferences - OBP Plugin Development	16-5
16_3	Enter the Preferences Fact values	16-6
16_4	Eact Properties - aggregateCodeFilePath	16-7
16_5	Fact Properties - sourceFilePath	16-7
16_6	Start Host Sourcer	16-8
16 7	Select Open Derengetive velue	16.0
16 9	Fact Explorer	16 10
10-0	Fact Explorer	16 11
16 10	Part vocabulary	16 10
10-10	Domain Category	10-12
10-11	Fact Groups	10-13
16 12	Facts	10-14
10-13	Jusiness Definition Tab	10-14
10-14		10-15
16-15	Enum Definition Tab	10-10
16-16	Aggregrate Definition Tab	16-17
16-17	Aggregate File Tab	16-18
16-18	Creating New Fact - Add	16-19
16-19	Creating New Fact - Fact Business Definition	16-20
16-20	Creating New Fact - Domain Group	16-21
16-21	Saving New Fact	16-21
16-22	Saving New Fact - Fact Added	16-22
16–23	Generic Rule Configuration	16-27
16–24	Rule Author - Decision Table	16-28
16–25	Rule Author - Expression Builder	16-29
17–1	Composite Application Service Architecture	17-2
18–1	Configuration of ID Generation Process	18-2
18–2	Automated ID Generation - Single Record View	18-5
18–3	Automated ID Generation - Generate Submission ID	18-5
18–4	Automated ID Generation - Submission ID Generation Service	18-6
18–5	Custom ID Generation - Custom ID Generator	18-7
18–6	Custom ID Generation - Custom ID Generation Constants	18-8
18–7	Custom ID Generation - Custom Pattern Based Generator	18-8
19–1	Extensibility of Domain Objects - Framework	19-2
19–2	Code Extract	19-3
19–3	Interceptor Hook to Persist Customized Domain Object	19-4
19–4	Interceptor Hook to Fetch Customized Domain Object	19-5
19–5	JSONClient constructs the JSON Object	19-6

19–6	SerializeDictionaryArray to include GenericName and Value attributes	19-7
19–7	Host Server JSONFacade extracts the attribute of JSON Object	19-8
19–8	AbstractJSONFacade's getDictionaryArray method	19-9
19–9	Host Server JSONFacade constructs the JSON Object	19-10
19–10	AbstractJSONFacade's serializeDictionaryArray to include Generic Name and Value	:
	attributes 19-11	
19–11	UI Server ISONClient extracts the DictionaryArray attribute	19-12
19–12	AbstractISONBindingStub's getDictionaryArray method	19-13
19–13	Instantiation of DataTransferObjects	19-15
19–14	Adding Discriminator Column Mapping in Existing HBM file	19-16
19–15	HBM File Mapping to Customized Domain Object	19-16
19–16	Adding New Java File to the Customized Domain Object	19-17
19–17	Adding Extra Columns along with the Discriminator Column	19-18
19–18	Adding a New HBM File Mapping to Customized Domain Object	19-19
19-19	Adding New Java File to Customized Domain Object	19-20
19-20	New HBM File Manning	19-21
19-21	Adding New Java File	19-22
19-22	Create a New Table CZ_NAB_LM_PROPOSED_FACILITY	19-22
19-23	CustomDataHandler's as Dictionary Array Intercentor	19-24
19-24	Create Customized Abstract Domain Object Class	19-25
19-25	Create Customized Abstract Domain Object Class	19-25
19-26	Create Customized Abstract Domain Object Attribute Columns	19-26
19-27	Customized Message Template Class	19-27
19-28	Domain Object Table	19-28
10_20	Hibernate File	10-28
19_30	II Init Toet Case	10-20
10_31	IUnit Adde Table Record	10-20
10_32	Dictionary Array Values	10-30
20_1	Extensibility Deployment	20-2
21_1	Perfection Capture Screen	21-2
21-1	Localization Implementation Architectural Change	21-2
21_3	Package Structure	21-3
21-0	Customization of the IDeveloper	21-0
21-4	Customization of the JDeveloper	21-4
21-5	Customization Context in Customization Developer Kole	21-5
21-0	Enabling Socied Customization	21-0
21-7	Library and Class Path	21-0
21-0	MDS Configuration	21-7
21-9	Manually Add antrias	21-0
21-10	MAR Creation	21-0
21-11	MAR Creation Application Properties	21-10
21-12	MAR Creation - Application Toperties	21-11
21-13	MAR Creation - Create Deployment 1 rome	21-12
21-14	MAR Creation - MAR The Selection	21-13
21-13	MAR Creation - DE Library Customization	21-14
21-10	MAR Creation - ADF Library Customization	21-10
21-17	MAR Creation - Europhication Accomply	21-10
21-10	MAR Creation - Application Assembly	21-17
21-19	rackage Deployment	21-19
22-1	Abstract Integration Adapter Class	22-2
22-2	Sample Integration Adapter	22-3 22 1
22-0 22-1	Integration Abstract Accomblar	22-4 22 F
22-4	Sampla Assamblar	22-3
22-0	Sample 1 Socialities	22-0

List of Tables

5–1	Components of Adapter Implementation	5-2
6–1	FLX_UD_SCREEN_BINDING	6-2
9–1	Database Server Components	9-10
9–2	FLX_BATCH_JOB_CATEGORY_MASTER	9-11
9–3	FLX_BATCH_JOB_GRP_CATEGORY	9-11
9–4	FLX_BATCH_JOB_CATEGORY_DEPEND	9-12
9–5	FLX_BATCH_JOB_SHELL_MASTER	9-13
9–6	FLX_BATCH_JOB_SHELL_DTLS	9-14
9–7	FLX_BATCH_JOB_SHELL_DEPEND	9-15
9–8	Driver Table	9-15
9–9	Actions Table	9-16
10–1	FLX_EXT_FILE_UPLOAD_MAST	10-3
10–2	Mandatory Fields in Record Tables	10-4
10–3	FLX_EXT_FILE_PARAMS	10-4
10–4	FLX_BATCH_JOB_SHELL_DTLS	10-5
10–5	XXF File Definition XML	10-11
10–6	Process Status	10-16
11–1	FLX_EP_ACT_B	11-2
11–2	FLX_EP_EVT_B	11-3
11–3	FLX_EP_ACT_EVT_B	11-3
11–4	Key Fields in FLX_MD_SERVICE_ATTR	11-6
16–1	Example of a Decision Table	16-24
16–2	Actions	16-25
16–3	Conditions	16-25
16–4	Rules Versioning	16-25
16–5	Details of Configured Rules in Modules	16-30
17–1	Java Classes	17-2
18–1	FLX_CS_ID_CONFIG_B	18-2
18–2	FLX_CS_ID_RANGE	18-3
18–3	FLX_CS_ID_USF	18-3
21–1	Path Structure	21-17

Preface

The Oracle Banking Platform Extensibility guide explains customization and extension of Oracle Banking Platform.

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

This guide is intended for the users of Oracle Banking Platform.

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 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 Installation Guide - Silent Installation
- For a comprehensive overview of security for Oracle Banking, see the Oracle Banking Security Guide
- For the complete list of Oracle Banking licensed products and the Third Party licenses included with the license, see the Oracle Banking Licensing Guide
- For information related to setting up a bank or a branch, and other operational and administrative functions, see the Oracle Banking Administrator's Guide

• For information on the functionality and features of the Oracle Banking product licenses, see the respective Oracle Banking Functional Overview documents

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.

Objective and Scope

This chapter defines the objective and scope of this document.

1.1 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 across 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.

1.2 Objective and Scope

While most product development can be accomplished using highly flexible system parameters and business rules, further competitive differentiation can be achieved through IT configuration and extension support. Time consuming, custom coding to enable region specific, site specific or bank specific customizations can be minimized by offering extension points and customization support which can be implemented by the bank and / or by partners.

1.2.1 Extensibility Objective

OBP when extended and customized by the Bank and / or Partners results in reduced dependence on Oracle. As a result of this, the Bank does not have to align plans with Oracle's release plans for getting certain customizations or product upgrades. The bank has the flexibility to choose and do the customizations themselves or have them done by partners.

One of the key considerations towards enabling extensibility in OBP has been to ensure that the developed software can respond to future growth. This has been achieved by disciplined software development leading to clearer dependencies, well-defined interfaces and abstractions with corresponding reduction in high cohesion and coupling. Hence, the extensions are kept separate from Core. Bank can take advantage of OBP Core solution upgrades as most extensions done for a previous release can be placed directly on top of the upgraded version. This reduces testing effort thereby reducing overall costs of planning and taking up an upgrade. This can also improve TTM significantly as the bank enjoys the advantage of getting universal features through upgrades. 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.

1.2.2 Document Scope

The scope of this document is to explain the customization and extension of OBP for the following use cases:

- Customizing OBP UI
- Adding an ADF screen side validation to an existing field
- Adding a new field or a table on the screen
- Removing fields from the UI
- Customizing OBP application services and implementing composite application services
- Adding pre-processing or post processing validations in the application services extension
- Altering the product behavior at customizations hooks provided as adapter calls in functional areas that are prone to change (for example, loan schedule generation) and in between modules that can be replaced (for example, alerts, content management)
- Adding new fields to the OBP domain model and including it on the corresponding screen.
- Adding a new report
- Adding a new batch program
- Customizing SOA based BPEL process with adding a partner link or a human task to an existing process.
- Adding new steps as a sub-process
- Adding or customizing facts and business rules in the application and configuring them for different modules
- Adding or customizing ID generation logic with options of automated, manual or custom generation
- Processing of the uploaded files data
- Printing of receipt once the transaction is over

- Defining the security related access and authorization policies
- Defining different security related rules, validator and processing logics
- Customizing different functionalities like user search, role evaluation and limit exclusion in the application related to security

This document is a useful tool for Oracle Consulting, bank IT and partners for customizing and extending the product.

1.3 Complementary Documentation

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

- **1.** OBP installation and configuration
- 2. OBP parameterization as part of implementation
- **3.** Functional solution and product user guide

1.4 Out of Scope

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

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.

2.1 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.

2.1.1 Extending Service Execution

In OBP, additional business logic might be required for certain services. This additional logic is not part of the core product functionality but could be a client requirement. For these purposes, hooks have been provided in the application code wherein additional business logic can be added or overridden with custom business logic.





Following are the two hooks provided:

Service Extensions

This hook resides in the app layer of the application service. This hook is present for, before as well after the actual service execution. The additional business logic has to implement the interface *I*<*service_name*>*ApplicationServiceExt* and extend and override the default implementation *Void*<*service_name*>*ApplicationServiceExt* provided for the service. Multiple implementations can be defined for a particular service. The service extensions executor invokes all the implementations defined for the particular service both before and after the actual service executes.

Service Provider Extension

This hook resides in the appx layer of the application service. This hook, too, is present for before as well after the actual service execution. The additional business logic has to implement the interface *I*<*service_ name*>*ApplicationServiceSpiExt* and extend and override the default implementation *Void*<*service_name*>*ApplicationServiceExt* provided for the service. Multiple implementations can be defined for a particular service. The service extensions executor invokes all the implementations defined for the particular service both before and after the actual service executes.

2.1.2 OBP Application Adapters

In OBP, adapters are used for helping two different modules or systems to communicate with each other. It helps in the consuming side adapters to any incompatibility of the invoked interface to work together. This is done to achieve cleaner build time separation of different functional product processor modules. Hence, when Loan Module needs to invoke services of Party Module or Demand Deposit module then an adapter class owned by the Loans module will be used to ensure that functions such as defaulting of values, mocking of an interface, and so on, are implemented in the adapter layer thereby relieving the core module functionality from getting corrupted.

Figure 2–2 OBP Application Adapters



2.1.3 User Defined Fields

There may be a requirement to capture additional data for certain objects/entities from the product screens. These additional attributes are not a part of the core product functionality but could be a client requirement.

Figure 2–3 Configure User Defined Fields



There are two ways in which additional data can be captured. These are:

User Defined Fields (UDF) Task-flow

The application provides a UDF task-flow which can be used for adding user defined fields on a screen. UDF are useful for capturing and displaying additional data. However, it is difficult to use this additional data in the business logic. Hence, UDF are ideal for capturing data and reporting purposes. When using this way for additional capture, simple changes on client side and minimal changes (or no changes) on host side are required.

Client: The UDF task-flow needs to be incorporated in the screen for which the additional fields need to be added. After adding the task-flow, you can add additional fields and specify various attributes for it like label for the field, mandatory field, and so on.

Host: The Appx layer needs to be enabled for the service. This layer contains the required call to the UDFApplicationService. However, once this layer has been enabled, you can add more fields without any need for modification on the host side.

Custom Entities:

Additional fields can be added to objects / entities from the very base level (ORM / POJO layer) to the front end (View layer) level. This way is more costly since it requires changes at all layers of the application. However, it has an advantage of the ability to use the additional data in the business logic of the application.

Client: The UI of the screen in which the additional data needs to be captured has to be modified for the additional fields. The view-service linkage also needs to be modified for transferring the additional data.

Host: On the host side, the ORM and POJO for the entity have to be modified to save the additional field's data. The service layer has to be modified for any business logic that is affected by the additional fields.

2.1.4 ADF Screen Customization

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.

Figure 2–4 ADF Screen Customization



2.1.5 SOA Customization

OBP Application provides the feature for customizing SOA composite applications based on the additional requirements which may vary from client to client. It includes implementing the partner link to an existing process or implementing human tasks or sub processes which can be hooked into an existing product process.





2.1.6 Batch Framework Extension

This extensibility feature is provided because most of the enterprise applications require the bulk processing of records to perform business operations in real-time environments. These business operations include complex processing of large volumes of information that is most efficiently processed with minimal or no user interaction. Such operations includes time-based events (For example, month-end calculations, notices or correspondence), periodic application of complex business rules processed repetitively across very large data sets (For example, rate adjustments).

All such scenarios form a part of batch processing for multiple records. Thus, Batch processing is used to process billions of records for enterprise applications. There are many categories in OBP Batch Processes like Beginning of Day (BOD), End of Day (EOD), and Statement Generation, and so on, for which the batch execution is performed.





2.1.7 Uploaded File Processing

File processing is an independent process and is done separately after file upload. Every upload provides a unique field for the uploaded file. The processing is then done for each upload as per the required functionality. The final status is provided at the end of the processing in the form of ProcessStatus.

An example can be salary credit processing. Once the employer account details (in header records) and the multiple employee account details (in detail records) are uploaded through the file upload, the salary credit processing can be done in which the employer account will be debited and the multiple accounts of the employees will be credited.

Figure 2–7 Upload File Processing



2.1.8 Alert Extension

OBP has to interface with various systems to transfer data which is generated during business activities that take place during teller operations or processing. The system requires a framework which can support on-line data transfer to interfacing systems.

This extension of event processing module of OBP provides a framework for identifying executing host services as activities and generating / raising events that are configured against the same. Generation of these events results in certain actions that can vary from dispatching data to subscribers (customers or external systems) to execution of additional logic. The action whereby data is dispatched to subscribers is termed as Alert. In OBP application, these Alerts can be customized and configured.





2.1.9 New Reports Creation

OBP application provides functionality for configuring multiple reports through integrated Oracle's Business Intelligence Publisher Enterprise. It is a standalone reporting and document output management solution that allows companies to lower the cost of ownership for separate reporting solutions. The developer can add and configure an Adhoc report to OBP using the BI Publisher.

The OBP application also has a batch framework using which a developer can easily add batch processes, also known as batch shells, to the application. The batch framework executes all the batch shells defined in the system as per their configuration. The results of these batch shell executions are stored in the database. In OBP, the user can create and customize the batch reports based on the requirements which can vary from client to client.

Figure 2–9 Creating New Reports



2.1.10 Security Customization

OBP application comprises of several modules which have to interface with various systems in an enterprise to transfer/share data. This data is generated during business activity that takes place during teller operations or processing. While managing the transactions that are within OBP's domain, it also needs to consider security and identity management and the uniform way in which these services need to be consumed by all applications in the enterprise. This is possible if these capabilities can be externalized from the application itself and are implemented within products that are specialized to handle such services.





OBP application therefore provides functionality where there is a provision for customizing the security attributes or functions. For example:

- Attributes participating in access policy rules
- Attributes participating in fraud assertion rules
- Attributes participating in matrix based approval checks
- Account validator
- Customer validator
- Business unit validator
- Adding validators
- Customizing user search
- Customizing 2FA 'Send OTP | Validate OTP' logic
- Customizing Role Evaluation
- Customizing Limit Exclusions

2.1.11 Loan Schedule Computation Algorithm

OBP application provides the extensibility by which the additional loan schedule computation algorithm can be customized based on client's requirement.





2.1.12 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.

Figure 2–12 Print Receipt Functionality



2.1.13 Facts and Business Rules

Fact (in an abstract way) is something which is a reality or which holds true at a given point of time. Business rules are made up of facts. Business Rules are defined for improving agility and for implementing business policy changes. This agility, meaning fast time to market, is realized by reducing the latency from approved business policy changes to production deployment to near zero time. In addition to agility improvements, Business Rules development also requires far fewer resources for implementing business policy changes. This means that Business Rules not only provides agility, it also provides the bonus reduced cost of development.





2.1.14 Composite Application Service

OBP provides the extensibility feature by which developer can write the composite service in which multiple service calls can be made as part of single call. Transactions in composite application service are made by composing the single transaction out of the multiple APIs transaction that gives the effect of single transaction.
Figure 2–14 Composite Application Service



2.1.15 ID Generation

OBP is shipped with the functionality of ID generation in three ways that is, Automatic, Manual and Custom. These three configurations can be defined by the banks as per their requirements and IDs can be generated accordingly.





2.1.16 OCH Integration

OBP provides various integration adapters and assemblers which are used to publish customer information to OCH. These adapters and assemblers can be customized for publishing details to OCH.

Customization developer can extend the existing integration adapters to fetch or gather more information about the customer and customize integration assembler to add new mappings.

Figure 2–16 OCH Integration



Extending Service Executions

This chapter describes how additional business logic can be added prior to execution (pre hook) and/or post the execution (post hook) of particular application service business logic on the host side. Extension prior to a service execution can be required for the purposes of additional input validation, input manipulation, custom logging and so on.

An application service 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 service execution.

An application service extension in the form of a post hook can be important in the following scenarios:

- Output response manipulation
- Third party system calls in the form of web service invocation, JMS interface and so on.
- Custom data logging for subsequent processing or reporting.

The OBP application provides two layers where the pre and post extension hooks for extending service execution can be implemented. These places are:

- Application Service referred to as the "app" layer extension.
- Extended Application Service referred to as the "appx" layer extension.

There are few differences in the extensions of the app and appx layer:

- In the appx layer extension, the validations can be added against the user defined fields which is not possible in case of the app layer.
- In the appx layer extension, the service response can be passed when the return type is not transaction status. This response can be validated or updated which is not available in case of app layer.
- In the appx layer, the approvals can be incorporated and can be validated in the appx layer extension which is not possible in app layer.

3.1 Service Extension – Extending the "app" Layer

The "app" layer is referred to as the application service layer. It denotes the business logic that executes as part of a service method exposed by OBP middleware host. Extension points provided as pre and post hooks are present in this layer at the same

points in the service. Every application service method has a standard set of framework method calls as shown in the sequence diagram below:

Figure 3–1 Standard Set of Framework Method Calls



The pre hook is provided after the invocation of createTransactionContext call inside the application service. At this step, the transaction context is set and the host application core framework is aware of the executing service with respect to the authenticated subject or the user who has posted the transaction, transaction inputs, financial dates, different determinant types applicable for the executing service, an initialized status and has the ability to track the same against a unique reference number. At this step, the database session is also initialized and accessible enabling the user to use the same in the pre hook for any database access which needs to be made.

The post hook is provided after the business logic corresponding to the application service invoked has executed and before the successful execution of the entire service is marked in the status object. This ensures that the status marking takes into consideration any execution failures of post hook prior to reporting the result to the calling source. Both, the pre and the post hooks accept the service input parameters as the inputs.

The following sections explain important concepts, which should be understood for extending in this service layer.

3.1.1 Application Service Extension Interface

The OBP plug-in for eclipse generates an interface for the extension of a particular service. The interface name is in the form I<Service_Name>ApplicationServiceExt. This interface has a pair of pre and post method definitions for each application service method of the present. 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 service extension class has to implement this interface. The pre method of the extension is executed before the actual service method and the post method of the extension is executed after the service method.

Figure 3–2 Extension Hook for DocumentTypeApplicationService

com.ofss.fc.module.content/sr	c/com/ofss/fc/app/content/service/ext/IDocume	entTypeApplicationServiceExt.java - Ecli	pse 🚺 🐧 🗤) 11:42 AM
		2 × 10 4 = 1 + 5 + 5	• *5 /5 • / ··
			• • • • • • • •
	Q Quick Acce	ss 📑 🔹 🐉 Java 🗟 SVN Reposito	ry Exploring 🗳 Team Synchronizing 🍄 Debug 🖪
🖉 🕖 DocumentTypeApplica	itionServic 🛛 💭 DocumentTypeApplicationServic	DocumentTypeApplicationServ	🕑 IDocumentTypeApplicationServ 🕱 📟 🖷
1 package com.ofs	s.fc.app.content.service.ext;		
2 3# import com ofse	fc ann content dto DocumentTyneDTO:		
8	. re. app. concent. aco. bocament rypebro,		
Ju 9⊖ /**			
10 *			
11 * Extension ho	ok for DocumentTypeApplicationService. The	default implementation for this inf	terface
12 * 15 the gener	molementing this interface	. Extensions should extend the void	DocumentTypeApptIcationServiceExt
14 *	inprementing this interfuce.		
15 * @see com.ofs	s.fc.app.content.service.VoidDocumentTypeA	pplicationServiceExt	
16 */			
17 public interfac	e IDocumentTypeApplicationServiceExt {		
18			
20 * This is	the extension point for DocumentTypeApplic	ationService.addDocumentType.	
21 * The Sess	ionContext object is not passed but the re-	st of the parameters are the same.	
22 * The java	doc for the original method and the params	can be seen from the See Also link.	
23 * @see con	.ofss.fc.app.content.service.DocumentTypeA	pplicationService#addDocumentType	
24 */	proiddDocumentType/com ofcs fc app contex	t SessionContext sessionContext Dec	mentTypeDT0 documentTypeDT0)
26 throws	FatalException:	C.SESSIONCONTEXT SESSIONCONTEXT, DOCT	mentrypebro documentrypebro/
27			
280 /**			
29 * This is	the extension point for DocumentTypeApplic	ationService.addDocumentType.	
30 * The Sess	ioncontext object is not passed but the re-	st of the parameters are the same.	
32 * @see cor	ofss.fc.app.content.service.DocumentTypeA	pplicationService#addDocumentType	
33 */	ionsonrenuppreoncenerser vicensoeamenerspon	pp cica ciono ci vice a a a a o cameri ci pe	
34⊖ public void	postAddDocumentType(com.ofss.fc.app.conte	xt.SessionContext sessionContext,Doo	cumentTypeDT0 documentTypeDT0)
35 throws	FatalException;		
30			
370 /** 38 * This is	the extension point for DocumentTypeApplic	ationService undateDocumentType	
39 * The Sess	ionContext object is not passed but the re	st of the parameters are the same.	
40 * The java	doc for the original method and the params	can be seen from the See Also link.	
41 * @see con	.ofss.fc.app.content.service.DocumentTypeA	pplicationService#updateDocumentType	2
42 */			
43 public void	FatalExcention:	text.Sessioncontext SessionContext,L	ocumentTypeDT0 documentTypeDT0)
45	ratation,		
460 /**			
47 * This is	the extension point for DocumentTypeApplic	ationService.updateDocumentType.	
48 * The Sess	ionContext object is not passed but the re-	st of the parameters are the same.	
49 * The java	doc for the original method and the params	can be seen from the See Also link.	
51 */	.orss.rc.app.content.service.DocumentTypeA	ppticationservice#updateDocumentType	
529 public voir	postUpdateDocumentType(com.ofss fc app co	ntext SessionContext sessionContext	DocumentTypeDT0 documentTypeDT0)

3.1.2 Default Application Service Extension

The OBP plug-in for eclipse generates a default extension for a particular service in the form of the class Void<Service_Name>ApplicationServiceExt. This class implements the aforementioned service extension interface without any business logic if 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 an application service. Instead of implementing the entire interface, one should extend the default extension class and override only 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 and reserved for product use and get overwritten as part of a bulk generation process.

Figure 3–3 Default Application Service Extension

```
com.ofss.fc.module.content/src/com/ofss/fc/app/content/service/ext/VoidDocumentTypeApplicationServiceExt.java - Eclipse
                                                                                                                                                                                                                🐱 🕇 🜒) 11:43 AM
File Edit Source Refactor Navigate Search Project Run Window Help
📸 र 🖂 🖓 🖕 🖓 🐨 🖓 र 🔕 र 🏶 🞯 र 🧶 🖉 🖉 🖉 🖉 🖉 🖉 🖉 🖉 🖉 🖉 🖉 🖉 🖉
                                                                                        🔍 Quick Access 📑 😰 🎝 Java 🖶 SVN Repository Exploring 🖆 Team Synchronizing 🌣 Debug 🗉
 🕫 🖉 DocumentTypeApplicati 🖉 DocumentTypeApplicati 😰 IDocumentTypeApplicat 🔹 IDocumentTypeApplicat 🔹 VoidDocumentTypeApplicat
              1 package com.ofss.fc.app.content.service.ext;
         扫
               3* import com.ofss.fc.app.content.dto.DocumentTypeDT0;
 Ju
               98/
             10 * ep>
11 * Extension hook for DocumentTypeApplicationService. The default for the extension points.
12 * Each application service method for DocumentTypeApplicationService has corresponding pre
13 * and post methods. This default implementation returns and does nothing.
14 * <u>Extenders</u> are encuraged to extend this class instead of implementing
15 * the interface as they would have to then implement all methods. This class
15 * the interface is a the extend the extensions.
                     * is provided for easing the writing of the extensions.
                  * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService
                 public class VoidDocumentTypeApplicationServiceExt implements IDocumentTypeApplicationServiceExt {
             220
                             This is the extension point for DocumentTypeApplicationService.addDocumentType
                         * The SesionContext object is not passed but the rest of the parameters are the same.
* The <u>sesionContext</u> object is not passed but the rest of the parameters are the same.
* The <u>javadoc</u> for the original method and the <u>parame</u> can be seen from the See Also lir
* @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#addDocumentType
             24
              25
26
27
                      public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDT0 documentTypeDT0)
          △ 280
             29
30
                                throws FatalException {
                              return;
             31
                       }
                      /**
* This is the extension point for DocumentTypeApplicationService.addDocumentType.
* The SessionContext object is not passed but the rest of the parameters are the same.
* The javadoc for the original method and the params can be seen from the See Also link.
* @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#addDocumentType

             33
34
35
             36
37
             390
40
41
42
                      public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
                           throws FatalException {
return;
             43
             440
                        /**

    This is the extension point for DocumentTypeApplicationService.updateDocumentType.
    The SessionContext object is not passed but the rest of the parameters are the same.
    The javadoc for the original method and the parama can be seen from the See Also link.
    @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#updateDocumentType

             45
46
47
48
                      public void preUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDT0 documentTypeDT0)
          △ 500
             51
52
                              throws FatalException {
  return;
```

3.1.3 Application Service Extension Executor

The OBP plug-in for eclipse generates a service extension executor interface and an implementation for the executor interface. The naming convention for the generated executor classes which enable 'extension chaining' is as shown below:

Interface : I<Application Service Qualifier>ApplicationServiceExtExecutor
Implementation : <Application Service Qualifier>ApplicationServiceExtExecutor

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



Figure 3–4 Application Service Extension Executor

Figure 3–5 ExtensionFactory Hook for DocumentTypeApplicationService







3.1.4 Extension Configuration

The extension classes that implement the extension interface are mapped to the application service class with the help of a property file mapping inside serviceextensions.properties. The mapping convention to be specified is a service's fully qualified class name to comma separated extensions' fully qualified class name in the following format:

```
<service_class_name>=<extension_class_name>,<extension_class_name>...
Example Mapping : config/properties/serviceextensions.properties
Single extension configuration
com.ofss.fc.app.content.service.DocumentTypeApplicationService=
com.ofss.fc.app.content.service.ext.DocumentTypeApplicationServiceExt
Multiple extension configuration
com.ofss.fc.app.content.service.DocumentTypeApplicationService=
com.ofss.fc.app.content.service.ext.in.DocumentTypeApplicationServiceExtension,
com.ofss.fc.app.content.service.ext.in.DocumentTypeApplicationServiceExtension
,
com.ofss.fc.app.content.service.ext.in.mum.DocumentTypeApplicationServiceExtension
,
com.ofss.fc.app.content.service.ext.in.blr.DocumentTypeApplicationServiceExtension
,
com.ofss.fc.app.content.service.ext.in.blr.DocumentTypeApplicationServiceExtension
,
com.ofss.fc.app.content.service.ext.in.blr.DocumentTypeApplicationServiceExtension
```

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

3.2 Extended Application Service Extension – Extending the "appx" Layer

The 'appx' layer is referred to as the extended application service layer. It represents the business logic that executes as part of a service method exposed by OBP middleware host with additional internal service calls to support extended features such as user defined fields (UDF). The appx layer also provides extension support, on top of and on the lines of the app layer. The implementation of extension support in this layer is similar to the implementation of extension support in the app layer.





The pre hook is provided before the invocation of actual application service call inside the extended application service layer. At this step, the extended host application core framework is aware of the executing service with respect to the authenticated subject or the user who has posted the transaction and an initialized status. At this step, the database session is also initialized and accessible enabling the user to use the same in the pre hook for any database access which might be required.

The post hook is provided after the primary application service which is extended in the appx layer along with the remaining internal service calls to support extended features like UDF complete execution and before the successful execution of the entire service is marked in the status object. This ensures that the status marking takes into consideration any execution failures of post hook prior to reporting the result to the calling source. Both, the pre and the post hooks accept the service input parameters including UDF data as their inputs. Additionally, if the response type of the object returned by the primary app layer application service is other TransactionStatus, the same is also accepted as input by the post hook.

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

Figure 3–8 Extended Application Service Extension - Post and Pre Hook

Ed	lit cou	rca Defactor Naviosta Search Droiert Duo Window Help	rjava - Ecupse		CO -1 -1-1	11.307
*	nc 500	The New York YO YO YO Y BY CHILD WINDOW HEP		•		
			and the second			
			Q Quick Access	💐 Java 🔒 SVN Rep	ository Exploring	9 91
10	Docum	nentTypeApplicationServiceSpiExt.java 🛛 DocTypeApplicationServiceSpiExt.java	DocumentTypeApplic	ationServiceSpi.java 🕮	-	
	2288# 289 299 299 299 299 299 299 299 299 299	<pre>public DocumentTypeInquiryResponseSet fetchDocumentTypes(com.ofss.fc DocumentTapsificationType documentClassificationType, FeeDetails throws FatalException { super_checkAccess{'com.ofss.fc.appx.content.service.DocumentTypes sessionContext, documentClassificationType, feeDetails, link com.ofss.fc.app.content.service.DocumentTypeApplicationService mu new com.ofss.fc.app.content.service.DocumentTypeApplicationService mu new com.ofss.fc.appx.content.service.ext.IDocumentTypeApplicationService</pre>	<pre>app-context.SessionConte .app.context.SessionConte .app.context.SessionConte .applicationServiceSpi.fet ddUDF070; manger = ervice(); //ccSpiExt helper = anServiceSpiExt*); pilcationServiceSpiExt*); sificationServiceSpiExt*); sificationType, feeDetail itClassificationType; tells; talls, taskCode); t>(); mtDT0Map); ssificationType, feeDetail</pre>	<pre>xt sessionContext, F0T0 linkedUDFDT0) chDocumentTypes", .fc.appx.ext.Servic s, linkedUDFDT0); ls, linkedUDFDT0);</pre>	eProviderExten	nsio
	314 315 316 317 318 329 321 322 323 324 325 326 327 328 329 338⊕	<pre>response); fillTransactionStatus(transactionStatus); response.setStatus(transactionStatus); } catch (FatalException e) { logper.log(Level.SEVERE, Interaction.fetchCurrentTask(), e); fillTransactionStatus(transactionStatus, e); } catch (Throwable e) { logper.log(Level.SEVERE, Interaction.fetchCurrentTask(), e); fillTransactionStatus(transactionStatus, e); } finally { Interaction.close(); } super.checkResponse(sessionContext, response); return response; } public DocumentTypeApplicationServiceSpi() {</pre>				
	331 332 333 334	<pre>extension = (IDocumentTypeApplicationServiceSpiExtExecutor) ServiceProvi .getServiceProviderExtensionExecutor(DocumentTypeApplicat) }</pre>	derExtensionFactory tionServiceSpi.class.getM	iame());		

The following concepts are important for extending in this service provider layer:

3.2.1 Extended Application Service Extension Interface

The OBP plug-in for eclipse generates an interface for the extension of a particular service. The interface name is in the form I<Service_Name>ApplicationServiceSpiExt. This interface has a pair of method definitions for each method of the present in the actual service. The signatures of these methods are:

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

An extended application service extension class has to implement this interface. The pre method of the extension is executed before the actual service method and the post method of the extension is executed after the service method.

Figure 3–9 Extension Hook for DocumentTypeApplicationServiceSpi



3.2.2 Default Implementation of Appx Extension

The OBP plug-in for eclipse generates a default service extension for a particular service in the form of the class *Void<Service_Name>ApplicationServiceSpiExt*. This class implements the aforementioned service provider extension interface without any business logic viz. 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 an application service. Instead of implementing the entire interface, one should extend the default extension class and override only 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 and reserved for product use and may get overwritten as part of a bulk generation process.

Figure 3–10 Default Implementation of Appx Extension

fss.	.fc.app	ox.spi/src/com/ofss/fc/appx/content/service/ext/VoidDocumentTypeApplicationServiceSpiExt.java - Eclipse 🛛 🗴 🗤 11:36 A
Edi	t Sou	ırce Refactor Navigate Search Project Run Window Help
· []		□ □ □ □ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
		🔍 Quick Access 📰 🖓 Java 🗟 SVN Repository Exploring 🛱 Team Synchronizing 🎋 Debug 🗉
D	Docur	mentTypeApplicationServiceExtExecutor.java 🛛 VoidDocumentTypeApplicationServiceSpiExt.java 🖾 🧧 🖉
	2⊕ C 4 p 5 6⊖ i 7 i 8 i 9 i	opyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
	10 1 11 1	<pre>mport com.orss.rt.enumeralion.content.oocumentclassification/ype; mport com.ofss.fc.infra.exception.FatalException;</pre>
	14⊕ 18 p	* VoidDocumentTypeApplicationServiceSpiExt[] ublic class VoidDocumentTypeApplicationServiceSpiExt implements IDocumentTypeApplicationServiceSpiExt {
A	200 21	<pre>public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentTypeDT0</pre>
	22 23 24 25	Void extension hook implemented with empty pre and post methods for the ApplicationServiceSpi. Each application service method for the ApplicationServiceSpi has corresponding pre and post methods. This default implementation does nothing. Usage guideline mandates extending this class instead of implementing the interface as they would have to then implement all methods. This class is provided for easing the
	26 27 28	writing of the extensions. An entry of extended class should be put in serviceextensions.properties. */ /*
	29 30 31 32	DO NOT ADD ANY CODE IN THESE METHODS AS IT WILL GET OVERWRITTEN WHEN BULK GENERATION OF CODE HAPPENS AND THE PERSON GENERATING DOES A BULK CHECK-IN !!! */ }
	33 340	public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentTypeDT
	35 36 37 38 39 40 41	/- Void extension hook implemented with empty <u>pre</u> and post methods for the ApplicationServiceSpi. Each application service method for the ApplicationServiceSpi has corresponding <u>pre</u> and post methods. This default implementation does nothing. Usage guideline mandates extending this class instead of implementing the interface as they would have to then implement all methods. This class is provided for easing the writing of the extensions. An entry of extended class should be put in serviceextensions.properties. */
	42 43 44 45	/* DO NOT ADD ANY CODE IN THESE METHODS AS IT WILL GET OVERWRITTEN WHEN BULK GENERATION OF CODE HAPPENS AND THE PERSON GENERATING DOES A BULK CHECK-IN !!! */
	46	}
	48⊕ 61	public void preUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentType
~	62⊕ 75	public void postUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentType
	76⊕	public void preFetchDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentTypeD
	89	nublic void postFetchDocumentType(com ofss fc app context SessionContext sessionContext, com ofss fc app context dto DocumentType

3.2.3 Configuration

The service provider extension class to the service class mapping is defined in a property file *ServiceProviderExtensions.properties* under "config/properties". Multiple extensions can be defined for a particular service provider with the help of the extension executor. The pre and post extensions are defined in the service layer.

The mapping is specified for a service provider extension interface's fully qualified class name to service provider extension class's fully qualified class name in the following format:

```
<service_provider_interface_name>=<service_provider_extension_class_
name>,<service_provider_extesion_class_name>
Example Mapping : config/properties/ServiceProviderExtensions.properties
Single extension configuration
com.ofss.fc.appx.content.service.ext.DocumentTypeApplicationServiceSpi=
com.ofss.fc.appx.content.service.ext.DocumentTypeApplicationServiceSpiExt
Multiple extension configuration
com.ofss.fc.appx.content.service.ext.in.DocumentTypeApplicationServiceExt,
com.ofss.fc.appx.content.service.ext.in.mum.DocumentTypeApplicationServiceExt,
com.ofss.fc.appx.content.service.ext.in.mum.ExtendedDocumentTypeApplicationServiceExt,
com.ofss.fc.appx.content.service.ext.in.blr.DocumentTypeApplicationServiceExt,
com.ofss.fc.appx.content.service.ext.in.blr.DocumentTypeApplicationServiceExt,
```

3.2.4 Extended Application Service Extension Executor

The OBP plug-in for eclipse generates a service provider extensions executor interface and an implementation class in the form of the following naming convention.

I<ApplicationServiceQualifier>ApplicationServiceSpiExtExecutor <ApplicationServiceQualifier>ApplicationServiceSpiExtExecutor The extended application service extension executor class, on load, creates an instance each of all the extensions defined in the service provider extensions configuration file. If no extensions are defined for a particular service provider, the executor creates an instance of the default extension for the appx service. The executor also has a pair of pre and post methods for each method of the actual appx service. These methods in turn delegate the call to the corresponding methods of all the extension classes configured inside the properties file for the service provider.





com.ofss.fc.appx.spi/src/com/ofss/fc/appx/content/service/ext/IDocumentTypeApplicationServiceSpiExtExecutor.java - Eclipse 🐱 🛊 🜒) 11:32 AN File Edit Source Refactor Navigate Search Project Run Window Help 🛅 Y 🔄 🖄 🖄 Y 🦓 Y 🕼 Y 🖓 Y 🚱 🚱 Y 🖉 🍠 🖉 🔳 🛐 🖞 Y 🖗 Y 💝 🖉 Y Q Ouick Access 🗄 📑 🐉 Java 🖶 SVN Repository Exploring 🌼 De 🚺 IDocumentTypeApplicat 🛿 🗖 🖷 DocumentTypeApplicati DocTypeApplicationSe DocumentTypeApplicati DocumentTypeApplicati 8 1 package com.ofss.fc.appx.content.service.ext; 開 3⊕ import com.ofss.fc.app.account.dto.transaction.FeeDetailsDTO; 100 /** 11 * 12 * ExtensionFactory hook for DocumentTypeApplicationServiceSpi. Extension Factories should implement 13 * the IDocumentTypeApplicationServiceSpiExtExecutor 14 * 15 */ 'F' 16 public interface IDocumentTypeApplicationServiceSpiExtExecutor { 189 * This is the extension point for DocumentTypeApplicationServiceSpi.preAddDocumentType. * The javadoc for the original method and the params can be seen from the See Also link. * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#preAddDocumentType 19 20 21 22 23© public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeDTO 24 25 26 27 28 29 30 31 throws FatalException; * This is the extension point for DocumentTypeApplicationServiceSpi.postAddDocumentType. * The javadoc for the original method and the params can be seen from the See Also link. * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#postAddDocumentType public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeDTO 32 33 throws FatalException; 34 35 36 37 38 /** * This is the extension point for DocumentTypeApplicationServiceSpi.preUpdateDocumentType. * The javadoc for the original method and the params can be seen from the See Also link. * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#preUpdateDocumentType */** 390 40 public void preUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeD throws FatalException; 41 420 /*: /** * This is the extension point for DocumentTypeApplicationServiceSpi.postUpdateDocumentType. * The <u>javadoc</u> for the original method and the <u>params</u> can be seen from the See Also link. * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#postUpdateDocumentType 43 44 45 46 470 public void postUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentType 48 throws FatalException; 49 500 /* This is the extension point for DocumentTypeApplicationServiceSpi.preFetchDocumentType. * The javadoc for the original method and the params can be seen from the See Also link. * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#preFetchDocumentType 51 52 53

Figure 3–12 ExtensionFactory Hook for DocumentTypeApplicationServiceSpi

com.o	fss. Edit	s.fc.appx.spi/src/com/ofss/fc/appx/content/service/ext/DocumentTypeApplicationServiceSpiExtExecutor.jav	ra - Eclipse 💌 📬 ♠)) 11:3	31 AM
E ²			\$\$ \$ \$ \$ ≥	
		Q Quick Access	🗄 🗈 🛃 Java 🗟 SVN Repository Exploring	🎋 De
8	D	DocumentTypeApplicationServic	🖸 DocumentTypeApplicationServic 🛿 🖻	8
間		1 package com.ofss.fc.appx.content.service.ext;		
		30 import com.otss.fc.app.account.dto.transaction.FeeDetailsDT0;		
		13© /** 14 *		
		15 * Factory Implementation of Extension hook for DocumentTypeApplicationServiceSpi. The def 6 * for the extension points.	ault	
		17 * The methods in this class when invoked, will internally call the pre/post		-
		19 * which implement the extension interface. 20 * (m)		
		<pre>20 * 21 * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi </pre>		
		<pre>22 */ 23 public class DocumentTypeApplicationServiceSpiExtExecutor implements IDocumentTypeApplicat</pre>	ionServiceSpiExtExecutor {	· · · ·
		<pre>24 25 private static IDocumentTypeApplicationServiceSpiExtExecutor uniqueInstance = new Docu</pre>	mentTypeApplicationServiceSpiExtExecutor()	;
	Sub.	26 private static String THIS COMPONENT NAME = DocumentTypeApplicationServiceSpiExtExecute 27	or.class.getName();	
		<pre>28 private List<idocumenttypeapplicationservicespiext> extensions = null; 29</idocumenttypeapplicationservicespiext></pre>		
		<pre>300 public DocumentTypeApplicationServiceSpiExtExecutor(){</pre>		
	Sa	<pre>31 extensions = 32 (List<idocumenttypeapplicationservicespiext>) ServiceProviderExtensionFactory</idocumenttypeapplicationservicespiext></pre>		
		<pre>33 .getServiceProviderExtensions(DocumentlypeApplicationServiceSpl.class.) 34 }</pre>	<pre>getName());</pre>	
	-	35 36⊖ public static IDocumentTypeApplicationServiceSpiExtExecutor getInstance() {		
		<pre>37 if (uniqueInstance == null) { 38 synchronized (DocumentTypeApplicationServiceSpiExtExecutor.class) { 39 } 39 } </pre>		
		<pre>39 if (uniqueInstance == null) { 40 uniqueInstance == null) { 51 uniqueInstance == null } 52 uniqueInstance == null } 53 uniqueInstance == null } 53 uniqueInstance == null } 54 uniqueInstance =</pre>) ·	
		41 }	<i>,</i>	
		42 }		
		44 return uniqueInstance; 45 }		
		46 48⊕ * This is the extension point for DocumentTypeApplicationServiceSpi.preAddDocumentTyp	e.[]	
		540 public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, context sessionContext)	om.ofss.fc.app.content.dto.DocumentTypeDT0	
		56 for (IDocumentTypeApplicationServiceSpiExt extension : extensions) { 57 extension predddocumentType(cessionContext_documentTypeDT0_feeDetails		
		58 }		
		00 1	-	
		<pre>62@ * This is the extension point for DocumentTypeApplicationServiceSpi.postAddDocumentTyp 68@ public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,</pre>	pe.[.] com.ofss.fc.app.content.dto.DocumentTypeDT	0

Figure 3–13 Factory Implementation of Extension Hook for DocumentTypeApplicationServiceSpi

3.3 End-to-End Example of an Extension

This section gives an end-to-end example of extensions written in the appx layer using the extended application service extensions as well as app layer application service extensions. The example shall implement this by extending the default implementation of the appx extension class

Void<ApplicationServiceQualifier>ApplicationServiceSpiExt class and app extension class *Void<ApplicationServiceQualifier>ApplicationServiceExt*.

For example, Back Office -> Content -> Document Type Definition screen of the application.

This screen is used for the maintenance of Document Types defined in the application.

Figure 3–14 Maintenance of Document Types

Oracle Bankin Eile Edit View History	g Platform - Mozilia Firefox y Bookmarks Tools Help			
E 127.0.0.1:7101/	com.ofss.fc.ui.view/faces/main.jspx?_afrLoop=236988579	198438_afrWindowMode=08_adf.ctrl-st	at 🗇 🕶 😋 🛃 🕶 Google	ء 🔄 🔍
Google 🗌 Local View	Server			
ORACLE Pesting	Date 25-Apr-2012 Local Currency AUD Last Login Date 13-3ul-201	2.04:12:40 PM Branch U Bank Operations BR	Sysiness On UBank Business Unit 💌	About
Account - Back Office -	CASA + Channel + Collection + LCM + Loan + Origination +	Party - Payment And Collection - Security	🔹 Service Req: 🛛 🏨 🌸 🖕	fest
Party Search	CNM91			Notes Save
Search Criteria	Document Type Definition			Font 📃
Party Id Search By	🗔 Bead 🌵 Create 🥒 Update 💥 Delete 🕞 Close		✔ Oğ 🥔 Clear 🔳 Cancel 🚍 Print	B/U 5,5'5
	Document Type Information			
Search Clear Advance Search	Document Type Id	 Document Type Description 		
	Document Type Inbound	Valued Document		
	Classification			
	(777)			
	Views Add Bare Delete Bare TR Delete		N	
			WE .	
	Tag Id			
	No data to display.			
				1
	3 User-Defined Fields			
	E Hide Modification History			1
	Created By ofssuser Approved By	On 13-Jul-2012 12:00:00 AM	Approved Active	
	-449-0100 UI	1971) 		
ENVNAME, BUILD DATE : BUIL	DOATE TNS Details : SERVICENAME = (DESCRIPTION = (ADDRESS =	(PROTOCOL = TCP)(HOST = DBHOST)(PORT -	- D8PORT)) (CONNECT_DATA - (SERVER - D	EDICATED) (SERVICE_NAME

The Create tab of the screen allows a user to create document types in the application. On click of Ok, and after successful validation of the input entered by the user, the screen extracts the values. It calls the DocumentTypeApplicationServiceSpi (in appx layer) and DocumentTypeApplicationService (in app layer) on the host application to save the document type in the system.

In this example, we have added multiple extensions to this service of the appx layer through the extension executor where the update of the description is done in one of the extension and check the length of name in another in the pre extension method.

com.ofss.fc.appx.spi.executor/src/com/ofss/fc/appx/content/service/ext/DocumentTypeApplicationServiceSpiExt.java - Eclipse	🚺 🚺 🕕)) 3:36 Pr
File Edit <u>S</u> ource Refac <u>t</u> or Navigate Search Project Run Window Help	
12 + + +	
Quick Access	N Repository Exploring 🎄 D
💭 DocumentTypeApplicationServiceSpiExt.java 🛱 🖸 DocTypeApplicationServiceSpiExt.java	
<pre>B DocumentTypeApplicationServiceSpiExtjava SI DocTypeApplicationServiceSpiExtjava</pre>	linkedUDFDTO)

Figure 3–15 DocumentTypeApplicationServiceSpiExt - Appx Layer





In this example, we have added multiple extensions to the service of the app layer through the extension executor and implemented a not null and size check on the document tags in pre hook of the app layer to validate that document tags are sent as input in the application service.

			** 🛐 🔹 🖩 🖩 🖓 * 🖗 * 🏠 🖒 * 🗢	
			Q Quick Access	lava 🗟 SVN Repository Exploring 🔅 De
DocTy	peApplicationServiceExt.java	DocumentTypeApplicationServic	eExt.java 🛛	- <i>a</i>
2⊕ Cop 4 pag	pyright (c) 2012, Oracle an ckage com.ofss.fc.app.conte	<pre>id/or its affiliates. All rights ent.service.ext;</pre>	reserved.	
5 6⊕ im	port java.util.ArrayList;			
17				
19 *	DocumentTypeApplicationSer	viceExt		
20 *	@version 1.0			
22 */ 23 pul	/ blic class DocumentTypeAppl	icationServiceExt extends VoidDo	<pre>cumentTypeApplicationServiceExt</pre>	
24		implement	<pre>s IDocumentTypeApplicationServiceExt{</pre>	
26	private static String THI	S COMPONENT NAME = DocumentType	<pre>pplicationServiceExt.class.getName();</pre>	
27 28	private transient Logger private List <validationer< td=""><td>logger = MultiEntityLogger.getU ror> errorList = new ArrayList<\</td><td>/alidationError>();</td><td>ME);</td></validationer<>	logger = MultiEntityLogger.getU ror> errorList = new ArrayList<\	/alidationError>();	ME);
29	private final static int	NAME_MAX_LENGTH = 20;		
310	public DocumentTypeApplic	ationServiceExt() {		
32	<pre>super(); }</pre>			
34	public void preAddDocumen	tType(com ofss fc app context Se	essionContext sessionContext DocumentTypeDT	0 documentTypeDT0)
36	throws Fa	talException {		o accamenci, pepio,
37	logger.log(Level.FINE	, "Pre add document type service	spi ext started.");	
39 40	if (documentTypeDT0!=	null && documentTypeDTO.getName)!=null && documentTypeDT0.getKeyDT0()!=n	ull) {
41	logger.log(Le	vel.WARNING, "Name exceeds the	prescribed length.");	
42	ValidationErr	<pre>or error = new ValidationError(' orConstants.INVALID LENGTH,</pre>	DocumentTypeDTO", "name", "null",	
44	"The	name exceeds the prescribed leng	jth.");	
45	}	i(error);		
47	} else {			
48	ValidationError e	WARNING, "Null Parameters"); error = new ValidationError("Docu	mentTypeDTO", "name", "null".	
50	CMErrorCo	instants.NULL_NAME,	· · · · · · · · · · · · · · · · · · ·	
51	"The name	d attribute value should not be	null");	
52	}	01);		
54	if (errorList != null	. && errorList.size() > 0) {		
55	throw new Validat	ionException(CMErrorConstants.N	<pre>JLL_DESCRIPTION, errorList);</pre>	
57	logger.log(Level.FINE	, "Pre add document type service	e spi ext ended.");	
20000				

Figure 3–17 DocumentTypeApplicationServiceSpiExt - App Layer

Figure 3–18 DocTypeApplicationServiceSpiExt - App Layer



OBP Proxy Extension

OBP Proxy Extension functionality is driven by a preference named "ProxyFacadeExtension" whose key-value properties are provided by a java class **com.ofss.fc.common.ProxyFacadeExtensionConfig**. This java class will have fully qualified name (replacing '.' With '_') of a proxy as a variable name and fully qualified name of a target proxy as a variable value.

For example,

```
public final String com_ofss_fc_xyz_ProductProxyFacade =
"com.ofss.fc.osb.xyz.ProductProxyFacade"; // notice usage of '_' in place of '.'
in variable name.
```

Sample Existing Code:

public TransactionStatus addReferenceObject(SessionContext sessionContext, ReferenceObjectDTO referenceObjectDTO) throws FatalException, ServiceException { if (logger.isLoggable(Level.FINE)) {

```
logger.log(Level.FINE, THIS_COMPONENT_NAME + " addReferenceObject()
Entry");
logger.log(Level.FINE, logAppServiceMessage(sessionContext));
logger.log(Level.FINE, logAppServiceMessage(referenceObjectDTO));
}
TransactionStatus returnObj = null;
try {
```

this.overrideProtocol("ReferenceObjectApplicationServiceProxy.addReferenceObject");

this.populateDictionaryData(referenceObjectDTO); if ("JSON".equals(protocol) && "APP".equals(hostApplicationLayer)) {

"com.ofss.fc.app.me.service.referencedata.ReferenceObjectApplicationService";

```
Object obj =
```

ReflectionHelper.getInstance().getClass(stringToCompleteClassName).newInstance();

```
returnObj = (TransactionStatus)
ReflectionHelper.getInstance().invokeMethod(obj, "addReferenceObject", args);
                } catch (Exception e) {
                    throw new ServiceException(SERVICE_NOT_AVAILABLE, e);
                }
            } else {
                logger.log(Level.SEVERE, THIS_COMPONENT_NAME, "No valid protocol
and hostApplicationLayer combination found");
                logger.log(Level.SEVERE, THIS_COMPONENT_NAME, SERVICE_NOT_
AVAILABLE);
            this.populateTransactionStatus(returnObj);
        } catch (IOException e) {
            logger.log(Level.SEVERE, THIS_COMPONENT_NAME, e);
            throw new ServiceException(SERVICE_NOT_AVAILABLE, e);
        }
        if (logger.isLoggable(Level.FINE)) {
            logger.log(Level.FINE, THIS_COMPONENT_NAME + " addReferenceObject()
Exit");
            logger.log(Level.FINE, logAppServiceMessage(returnObj));
        }
        return returnObj;
    }
```

Sample Existing Code will be changed to:

public TransactionStatus addReferenceObject(SessionContext sessionContext, ReferenceObjectDTO referenceObjectDTO) throws FatalException, ServiceException {

```
if (logger.isLoggable(Level.FINE)) {
                     logger.log(Level.FINE, THIS_COMPONENT_NAME + "
addReferenceObject() Entry");
                     logger.log(Level.FINE, logAppServiceMessage(sessionContext));
                     logger.log(Level.FINE,
logAppServiceMessage(referenceObjectDTO));
              }
              TransactionStatus returnObj = null;
              try {
                     if (isProxyExtended(this)) {
                           Serializable overriddenResponse =
invokeExtendedProxy(this, sessionContext, "addReferenceObject",
referenceObjectDTO);
                           if (overriddenResponse != null) {
                                  if (overriddenResponse instanceof
TransactionStatus) {
                                         return (TransactionStatus)
overriddenResponse;
                                  } else {
                                         logger.log(Level.SEVERE,
                                                    THIS_COMPONENT_NAME,
                                                     "Invalid response returned
from the overridden proxy. Response expected is an instance of
TransactionStatus.");
                                         throw new
ServiceException (BranchErrorConstants.FC_OVR_RESP_INV);
                                  }
                           } else {
                                  logger.log(Level.SEVERE,
                                             THIS_COMPONENT_NAME,
                                              "Null response returned from the
```

overridden proxy. Response expected is an instance of TransactionStatus."); throw new ServiceException(BranchErrorConstants.FC_OVR_RESP_NULL); } } else { this.populateDictionaryData(referenceObjectDTO); if ("JSON".equals(protocol) && "APP".equals(hostApplicationLayer)) { com.ofss.fc.app.me.service.referencedata.service.json.client.ReferenceObjectApplic ationServiceJSONClient jsonStub = new com.ofss.fc.app.me.service.referencedata.service.json.client.ReferenceObjectApplic ationServiceJSONClient(jsonServiceUrl); returnObj = jsonStub.addReferenceObject(sessionContext, referenceObjectDTO); } else if ("LOCAL".equals(protocol) && "APP".equals(hostApplicationLayer)) { trv { Object[] args = new Object[] { sessionContext, referenceObjectDTO }; String stringToCompleteClassName = "com.ofss.fc.app.me.service.referencedata.ReferenceObjectApplicationService"; Object obj = ReflectionHelper.getInstance().getClass(stringToCompleteClassName).newInstance(); returnObj = (TransactionStatus) ReflectionHelper.getInstance().invokeMethod(obj, "addReferenceObject", args); } catch (Exception e) { throw new ServiceException(SERVICE_NOT_ AVAILABLE, e); } } else { logger.log(Level.SEVERE, THIS_COMPONENT_NAME, "No valid protocol and hostApplicationLayer combination found"); logger.log(Level.SEVERE, THIS_COMPONENT_NAME, SERVICE_NOT_AVAILABLE); } this.populateTransactionStatus(returnObj); } } catch (Throwable e) { logger.log(Level.SEVERE, THIS_COMPONENT_NAME, e); throw new ServiceException(SERVICE_NOT_AVAILABLE, e); } if (logger.isLoggable(Level.FINE)) { logger.log(Level.FINE, THIS_COMPONENT_NAME + " addReferenceObject() Exit"); logger.log(Level.FINE, logAppServiceMessage(returnObj)); } return returnObj; }

OBP Application Adapters

An adapter, by definition, helps the interfacing or integrating components to adapt. In software it represents a coding discipline that helps two different modules or systems to communicate with each other and helps the consuming side to adapt to any incompatibility of the invoked interface to work together. Incompatibility could be in the form of input data elements which the consumer does not have and hence might require defaulting or the invoked interface might be a third party interface with a different message format requiring message translation. Such functions, which do not form part of the consumer functionality, can be implemented in the adapter layer.

In OBP, adapters are used for the above purposes as well as to achieve cleaner build time separation of different functional product processor modules. Hence, when Loan Module needs to invoke services of Party Module or Demand Deposit module then an adapter class owned by the Loans module will be used to ensure that functions such as defaulting of values, mocking of an interface, and so on, are implemented in the adapter layer thereby relieving the core module functionality from getting corrupted.

The design of the adapter layer is based on the Separated Interface design pattern and the access mechanism of the adapters by modules is implemented using an Abstract Factory design pattern. The adapter implementation is explained in the sections below as it exists in OBP.

5.1 Adapter Implementation Architecture

This section provides a detailed explanation of the adapter implementation architecture.

5.1.1 Package Diagram

The components of adapter implementation in OBP are structurally placed in separate projects to enable OBP to achieve build time independence between functional modules of the product. The way this is achieved is detailed in the table below and depicted with package diagram, class diagrams and an example usage mechanism.

Sr.	Project Name	Description	Example
1	com.ofss.fc.app.xface	DTO project. Holds all DTOs that are used in the module application services request and response DTOs.	
2	com.ofss.fc.app.adapte r.internal.interface	Package contains adapter interfaces for all modules and the abstract factory implementation (i.e. factory of adapter factories).	com.ofss.fc.app.adapter.ep.IEventProcess ingAdapter Abstract Factory com.ofss.fc.app.adapter.AdapterFactory
3	com.ofss.fc.app.adapte r.impl	This project has the implementation of adapter interfaces and corresponding adapter factories.	com.ofss.fc.app.adapter.ep. impl .EventPr ocessingAdapter com.ofss.fc.app.adapter.ep. impl .EventPr ocessingAdapterFactory

 Table 5–1
 Components of Adapter Implementation

Hence, if Loans module (that is, com.ofss.fc.module.loan) and Party module (that is, com.ofss.fc.module.party) are any two modules that need to communicate, the package dependency diagram is depicted below:





The dependencies among the packages are:

- Package com.ofss.fc.app.adapter.internal.interface only depends on DTO's.
- Any module package depends on the Adapter interfaces and DTO's to communicate with another module.
- Package com.ofss.fc.app.adapter.impl depends on all the packages.

In this manner, the loans module is developed into a functional module which is structurally modular and independent in terms of development and build from the party module and vice versa. Same is true for all modules developed in OBP.

5.1.2 Adapter Mechanism Class Diagram

An Application Service in calling module calls the getAdapterFactory() method of class AdapterFactoryConfigurator which returns an instance of an implementation of the abstract class AdapterFactory. The class of instance is decided by the string parameter passed to the method.

The getAdapter() method in the AdapterFactory returns an adapter instance. The class of instance is decided by the string parameter passed to the method.

The Application Service then uses this adapter instance to access any data from an application service within another module.

Figure 5–2 Adapter Mechanism Class Diagram



5.1.3 Adapter Mechanism Sequence Diagram

A method in an application service gets an instance of a desired adapter factory by calling getAdapterFactory() method of AdapterFactoryConfigurator class. The instance of the adapter factory obtained is used to call getAdapter() method to get an instance of the adapter. This adapter instance has all the methods to communicate to the service in another module.

Figure 5–3 Adapter Mechanism Sequence Diagram



5.2 Examples of Adapter Implementation

This section provides multiple adapter usage scenarios with code snippets. The section also has examples describing the steps for implementing custom adapters and their factory implementation. The same mechanism applies to all adapters which are provided by different modules in OBP. The adapter factory additionally supports mocking of the adapter. OBP depends on the DI feature function supported by Jmock to enable mocking of adapters.

The custom adapter, adapter factory and corresponding constants are depicted in code samples below:

5.2.1 Example 1 – EventProcessingAdapter

Code snippet to invoke a method *processActivityEvents()* in alerts module from a different module:

```
... Constants definition ...
public static final String EVENT_PROCESSING = "EVENT_PROCESSING";
public static final String MODULE_TO_ACTIVITY = "ModuleToActivityAdapter";
... Adapter usage ...
com.ofss.fc.app.adapter.IAdapterFactory adapterFactory =
AdapterFactoryConfigurator.getInstance().getAdapterFactory(ModuleConstant.EVENT_
PROCESSING);
IEventProcessingAdapter adapter = (IEventProcessingAdapter)
adapterFactory.getAdapter (EventProcessingAdapterConstant.MODULE_TO_ACTIVITY);
adapter.processActivityEvents();
```

The parameters passed in the **getAdapterFactory()** and **getAdapter()** methods are String constants denoting instance of which class has to be returned. These string values are maintained as constants. In the example given below, the string constant is created in a constants file (in this example, it the constants file is ModuleConstant).

```
public static final String EVENT_PROCESSING = "EVENT_PROCESSING";
```

An entry is made in AdapterFactories.properties corresponding to the string constant. This entry specifies the adapter factory class corresponding to the above constant which should be instantiated and returned. The adapter factory has the intelligence of all adapters along with adapter methods which are mocked as and where required.

EVENT_PROCESSING=com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapterFactory

While implementing the adapter factory, developers can choose to have a separate factory specific constants on the basis of which to manage multiple adapters from the same factory. Alternatively, developers can choose to create an adapter factory each for an adapter and its interface. The constants form the basis for instantiating and returning of respective adapters by the factory.

The respective adapter constant and corresponding usage in the **getAdapter** method of the adapter factory class is shown in a sample below.

```
... Adapter Factory Method ...
public IEventProcessingAdapter getAdapter(String adapter, NameValuePair[]
nameValues) {
    EventProcessingAdapter eventProcessingAdapter = null;
    If (adapter.equalsIgnoreCase(EventProcessingAdapterConstant.MODULE_TO_ACTIVITY)) {
    eventProcessingAdapter = new EventProcessingAdapter();
    }
    return eventProcessingAdapter;
}
```

The adapter implementation (that is, *EventProcessingAdapter*) can have implementation of the methods defined in the adapter interface it implements. This implementation is typically delegated calls to services of the module which is invoked by the consuming module. For example, the *EventProcessingAdapter* can implement the method *processActivityEvents()*.

```
public void processActivityEvents(ApplicationContext applicationContext,
HashMap<String, String> activityMap) throws FatalException {
EventProcessorApplicationService eventApplicationService =
new EventProcessorApplicationService();
eventApplicationService.processActivityEvents(AdapterContextHelper.fetchSessionCon
text(), key, activityDataId);
}
```

5.2.2 Example 2 – DispatchAdapter

Similar to the implementation of *EventProcessingAdapter*, an adapter implementation is provided by product for dispatch of an SMS alert. This adapter will always get customized during implementation depending on the SMS gateway used by the customer at their end.

The code snippet to invoke a method *dispatchSMS()* in alerts module by using the adapter interface is depicted below.

```
... Constants definition ...
public static final String EVENT_PROCESSING_DISPATCH = "EVENT_PROCESSING_
DISPATCH";
public static final String EP_TO_DISPATCH = "EpToDispatchAdapter";
... Adapter usage ...
com.ofss.fc.app.adapter.IAdapterFactory adapterFactory =
```

AdapterFactoryConfigurator.getInstance().getAdapterFactory(ModuleConstant. EVENT_ PROCESSING_DISPATCH);

```
adapter = (IDispatchAdapter) adapterFactory.getAdapter
(EventProcessingAdapterConstant.EP_TO_DISPATCH);
adapter.dispatchSMS();
```

An entry in *AdapterFactories.properties* corresponding to the *DispatchAdapterFactory* would look as below. This entry specifies the adapter factory class corresponding to the above constant which should be instantiated and returned.

 ${\tt EVENT_PROCESSING_DISPATCH=com.ofss.fc.app.adapter.impl.ep.DispatchAdapterFactory}$

The adapter *DispatchAdapter* is used in the alerts module to dispatch a message to an SMS destination endpoint. It has a method called *dispatchSMS(...)* and the default implementation is currently to write the SMS text generated as part of alert processing into a file called SMS.txt.

```
public boolean dispatchSMS(String recipientId, String dispatchMessage) throws
FatalException {
  return writeToFile(recipientId, dispatchMessage);
}
```

The customization developer can override this method by supplying a customized implementation of the adapter. Such custom implementation of the *dispatchSMS(...)* method invokes the APIs provided by the gateway client. A sample implementation which overrides the default implementation of *dispatchSMS* could look like the one below:

```
public boolean dispatchSMS(String recipientId, String dispatchMessage) throws
FatalException {
  NewGatewayAPI newGateway = new NewGatewayAPI();
  newGateway.sendMessage(recipientId,dispatchMessage);
  }
```

5.3 Customizing Existing Adapters

If an added functionality or replacement functionality is required for an existing adapter or existing method in an adapter, the customization developer has to develop a new adapter and corresponding adapter factory and override the method in a new custom adapter class. The custom adapter would have to override and implement the methods which need changes.

5.3.1 Custom Adapter Example 1 – DispatchAdapter

Depending on the client the SMS gateway they use and thus the corresponding interface to communicate with the gateway will differ. Also, OBP by default does not support interfacing with any SMS gateway. Hence, customization of DispatchAdapter is essential. The following steps can be followed for implementation of a custom *DispatchAdapter*.

Develop a *CustomDispatchAdapter* and *CustomDispatchAdapterFactory*. As a guideline, the custom adapter should extend the existing adapter and override the methods which need to be replaced with new functionality. Given below are examples of customizing the adapters which are detailed above.

The custom adapter, adapter factory and corresponding constant are depicted as a sample below:

```
... CustomDispatchAdapterFactory Method ...
public IDispatchAdapter getAdapter(String adapter, NameValuePair[] nameValues) {
    IDispatchAdapter adapter = null;
    If (adapter.equalsIgnoreCase(EventProcessingAdapterConstant.EP_TO_DISPATCH)) {
    adapter = new CustomDispatchAdapter();
    }
    return adapter;
}
```

The custom adapter implementation (that is, *CustomDispatchAdapter*) has the implementation of the methods defined in the adapter interface it implements. For example, the *CustomDispatchAdapter* would implement the method *dispatchSMS()* to reflect the desired functionality.

The entry in *AdapterFactories.properties* corresponding to the *DispatchAdapterFactory* can be modified to instantiate and return the *CustomDispatchAdapterFactory*. The same is shown below.

```
Original entry
EVENT_PROCESSING_DISPATCH=com.ofss.fc.app.adapter.impl.ep.DispatchAdapterFactory
Changed entry
EVENT_PROCESSING_
DISPATCH=com.ofss.fc.app.adapter.impl.ep.CustomDispatchAdapterFactory
```

This changed entry specifies the custom adapter factory class corresponding to the constant which is referred to in the product. The new entry shall ensure that the *AbstractFactory* instantiates and returns an instance of *CustomDispatchAdapterFactory* instead of the original *DispatchAdapterFactory* supplied with product.

5.3.2 Custom Adapter Example 2 – PartyKYCCheckAdapter

OBP ships an adapter implementation for KYC check of a party. The adapter implements to the interface shown below. The interface method *performOnlineKYCCheck* can be overridden by the customization developer while supplying the actual implementation of the desired functionality.

```
public interface IPartyKYCCheckAdapter {
  @External(name = "KYC", info = "Perform Online KYC Check")
  public abstract KYCHistoryDTO performOnlineKYCCheck(KYCHistoryDTO kycCheckDTO)
  throws FatalException;
}
```

This adapter is integrated in product and the default implementation of the KYC check returns a successful KYC check as shown below. This is depicted in the code snippets below.

}

Figure 5–4 Party KYC Status Check Adapter Interface

```
/*
Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
*/
package com.ofss.fc.app.adapter.party;
import com.ofss.fc.app.party.dto.core.KYCHistoryDTO;[]
/**
 * This interface represents the Party KYC status check adapter interface. Default implementation of <br>
 * this interface would return the KYCHistoryDTO with a KYC status indicating successful completion of<br>
 * the KYC for party.
*
 * @author OBPDev
 * @version 1.0
*/
public interface IPartyKYCCheckAdapter {
    @External(name = "KYC", info = "Perform Online KYC Check")
    public abstract KYCHistoryDTO performOnlineKYCCheck(KYCHistoryDTO kycCheckDTO) throws FatalException;
```

Figure 5–5 Default Implementation of IPartyKYCCheckAdapter Interface

```
* Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved...
package com.ofss.fc.app.adapter.impl.party;
import java.util.logging.Level;
 * Default implementation of IPartyKYCCheckAdapter interface. This would complement the adapter mocking<br>
 * done in the corresponding adapter factory
 * @author shravank
public class PartyKYCCheckAdapter implements IPartyKYCCheckAdapter {
   private static final String THIS_COMPONENT_NAME = PartyKYCCheckAdapter.class.getName();
private Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(THIS_COMPONENT_NAME);
private MultiEntityLogger formatter = MultiEntityLogger.getUniqueInstance();
     * This method would return the KYCHistoryDTO with a KYC status indicating successful completion of<br>
     * the KYC for party.
    @Override
    public KYCHistoryDT0 performOnlineKYCCheck(KYCHistoryDT0 kycCheckDT0) throws FatalException {
        if (logger.isLoggable(Level.FINE)) {
            logger.log(Level.FINE, formatter.formatMessage("Entered performOnlineKYCCheck."));
        kycCheckDT0.getAutomaticKYCDetails().setKycStatus(KYCStatus.CONFIRMED);
kycCheckDT0.getAutomaticKYCDetails().setKycProcessStage(KYCProcessStage.Complete);
kycCheckDT0.getAutomaticKYCDetails().setKycComments("KYC Staus maintained by Party");
        String bankCode = (String) FCRThreadAttribute.get(FCRThreadAttribute.USER_BANK);
Date postingDate = new CoreService().fetchBankDates(bankCode).getCurrentDate();
         kycCheckDT0.getAutomaticKYCDetails().setKycDate(postingDate);
        if (logger.isLoggable(Level.FINE)) {
            logger.log(Level.FINE, formatter.formatMessage("Exit performOnlineKYCCheck with KYCStatus:UNCONFIRMED and KYCProcessStage:Pending "));
        return kycCheckDT0:
   }
3
                         ... PartyKYCCheckAdapter performOnlineKYCCheck Method ...
                         public KYCHistoryDTO performOnlineKYCCheck(KYCHistoryDTO kycCheckDTO) throws
                         FatalException {
                         kycCheckDTO.getAutomaticKYCDetails().setKycStatus(KYCStatus.CONFIRMED);
                         kycCheckDTO.getAutomaticKYCDetails().setKycProcessStage(KYCProcessStage.Complete);
                         kycCheckDTO.getAutomaticKYCDetails().setKycComments("KYC Status maintained by
                         Party");
                         kycCheckDTO.getAutomaticKYCDetails().setKycDate(postingDate);
                         return kycCheckDTO;
                         }
                         In actual product implemented in production at the customer site, this is replaced with
```

```
an online KYC status check against a third-party system or the appropriate KYC
```

agency external system interface. Hence, this would always be a customization point during an implementation.

Depending on the client the KYC system uses, the corresponding interface to communicate will differ. Hence, customization of the party KYC status check adapter implementation is essential. The following steps would have to be followed for implementation of a custom *PartyKYCStatusCheckAdapter*.

The implementation of *getAdapter* method of KYC adapter factory with mocking support is given in the sample below for reference.

Figure 5–6 KYC Adapter Factory with Mocking Support

```
    This method returns instance of the KYC Adapter. If mocking is enabled, the method would return a mocked<br>
        instance of the adapter. Mocking helps in cases where the interface undergoes a change and the same has<br>
            to be handled with minor code changes at the adapter level.
        @return Object Instance of the adapter

public Object getAdapter(String adapter) {
    if (CommonAdapterConstants.PARTY_KYC_ADAPTER.equals(adapter)) {
        if ( !isMockEngbled) {
             return new PartyKYCCheckAdapter();
        } else {
            se ι
Mockery context = new Mockery();
final IPartyKYCCheckAdapter mockPartyKYCCheckAdapter = context.mock(IPartyKYCCheckAdapter.class);
            try {
                 context.checking(new Expectations() {
                        allowing(mockPartyKYCCheckAdapter).performOnlineKYCCheck(with(any(KYCHistoryDTO.class)));
final KYCHistoryDTO kycCheckDTO = new KYCHistoryDTO();
KYCDetailsDTO automaticKYCDetails = new KYCHetailsDTO();
automaticKYCDetails.setKycStatus(CMFIRMED);
automaticKYCDetails.setKycStatus(CMFIRMED);
automaticKYCDetails.setKycComments("KYC Staus mointained by Party");
String bankCode = (String) FCRThreadAttribute.get(FCRThreadAttribute.USER_BANK);
Date postingDate = new CoreService().fetchBankDates(bankCode).getCurrentDate();
automaticKYCDetails.setKycDetainaDate);
                         uate postinguate = new Coreservice().retchuankutes(bank
automaticKYCDetails.setKycDate(postingDate);
kycCheckDT0.setAutomaticKYCDetails(automaticKYCDetails);
will(returnValue(kycCheckDT0));
                    }
           };
};
catch (Exception e) {
throw new MockAdapterException(InfraErrorConstants.MOCK_METHOD_NOT_CONFGD, e, PartyKYCCheckAdapterFactory.class.getName());

            return mockPartyKYCCheckAdapter;
   } else {
        throw new ConfigurationInitializationException(InfraErrorConstants.ADAPTER_NOT_FOUND, PartyKYCCheckAdapterFactory.class.getName());
   }
                        ... Constants definition .
                        public static final String PARTY KYC ADAPTER FACTORY = "PARTY KYC ADAPTER
                        FACTORY";
                        public static final String PARTY_KYC_ADAPTER = "PartyKYCCheckAdapter";
                        ... PartyKYCStatusCheckAdapterFactory getAdapter Method ...
                        if (AdapterConstants.PARTY_KYC_ADAPTER.equals(adapter)) {
                        if ( !isMockEnabled) {
                        return new PartyKYCCheckAdapter();
                         else {
                        // 1. Creation of Mockery Object
                        Mockery context = new Mockery();
                        final IPartyKYCCheckAdapter mockPartyKYCCheckAdapter =
                        context.mock(IPartyKYCCheckAdapter.class);
                        try {
                        context.checking(new Expectations() {
                        {
                        allowing (mockPartyKYCCheckAdapter).performOnlineKYCCheck (with (any (KYCHistoryDTO.cl
                        ass)));
                        final KYCHistoryDTO kycCheckDTO = new KYCHistoryDTO();
                        KYCDetailsDTO automaticKYCDetails = new KYCDetailsDTO();
                        automaticKYCDetails.setKycStatus(KYCStatus.CONFIRMED);
                        automaticKYCDetails.setKycProcessStage(KYCProcessStage.Complete);
                        automaticKYCDetails.setKycComments("KYC Status maintained by Party");
                        String bankCode = (String) FCRThreadAttribute.get(FCRThreadAttribute.USER_BANK);
                        Date postingDate = new CoreService().fetchBankDates(bankCode).getCurrentDate();
```

```
automaticKYCDetails.setKycDate(postingDate);
kycCheckDTO.setAutomaticKYCDetails(automaticKYCDetails);
will(returnValue(kycCheckDTO));
}
);
} catch (Exception e) {
throw new
MockAdapterException(InfraErrorConstants.MOCK_METHOD_NOT_CONFGD,
e, PartyKYCCheckAdapterFactory.class.getName());
}
return mockPartyKYCCheckAdapter;
}
```

To override the default implementation of the KYC check, the customization developer has to implement a custom adapter and its corresponding adapter factory. Assume the same are named as *CustomPartyKYCStatusCheckAdapter* which conforms to the interface of the product KYC check adapter and

CustomPartyKYCStatusCheckAdapterFactory which would return an instance of the custom adapter. As a guideline, the custom adapter should extend the existing adapter and override the methods which need to be replaced with new functionality.

Therefore, *CustomPartyKYCStatusCheckAdapter* can override and provide an actual implementation of the methods defined in the default product adapter interface. For example, the adapter implements the method *performOnlineKYCCheck()* to reflect the desired functionality.

The entry in *AdapterFactories.properties* corresponding to the *PartyKYCCheckAdapterFactory* can to be modified to instantiate and return the *CustomPartyKYCCheckAdapterFactory*. The same is shown below.

Original entry PARTY_KYC_ADAPTER_ FACTORY=com.ofss.fc.app.adapter.impl.party.PartyKYCCheckAdapterFactory Changed entry PARTY_KYC_ADAPTER_FACTORY= com.ofss.fc.app.adapter.impl.party.CustomPartyKYCCheckAdapterFactory

This changed entry specifies the custom adapter factory class corresponding to the constant which is referred to in the product. The new entry shall ensure that the *AbstractFactory* instantiates and returns an instance of *CustomPartyKYCCheckAdapterFactory* instead of the original *PartyKYCCheckAdapterFactory* supplied by the product.

User Defined Fields

OBP application is shipped with the additional functionality where the additional data items can be added for certain objects/entities. These additional attributes needs not be part of the core product but could be the client's requirement.

For this provision of adding the user defined fields, the application is provided with the UDF task-flow fields on a screen, the UDF are useful for capturing and displaying additional data. However, it is difficult to use this additional data in the business logic. Hence, UDF are ideal for capturing data and reporting purposes. When using this way for additional capture, simple changes on client side and minimal changes (or no changes) on host side are required.

The following sections describe the changes to be done to enable the UDF for a particular screen.

6.1 Enabling UDF for a Particular Screen

This section provides a detailed explanation on enabling UDF for a particular screen.

6.1.1 UDF Metadata

Metadata for UDF are maintained in a table FLX_MD_SCREEN_BINDING. There is a facility to generate this data for the screens using a utility, but considering the complexity involved, in some cases the utility fails to generate the actual data and the metadata needs to be entered manually.

The Utility for assisting generation of the metadata are:

DomainObjectParser:

This utility gets the data of the middleware entities and creates a mapping between the DTO and the Entity fields and populates the same in a data source.

ScreenComponentDTOMapping:

This utility uses the data generated above, and by parsing the UI related files around the area where the VO Object is getting set from the DTO, and tries to arrive at the mapping between the vo attribute and the associated entity attribute.

The accuracy of this utility process isn't 100 percent as it depends directly on the different flavours/fashion of the code written and also on the way the screen has been designed. As 100 percent accuracy is not achievable using this, the generated data needs to be verified and corrected wherever necessary.

In case, the required data does not come out of this utility, the same need to be manually supplied in the table.

The above mentioned utility populates FLX_MD_SCREEN_BINDING, however, as there would be quite some cases where the utility will not supply the values when run, currently the UDF processing logic is based on another table with the same structure, which is FLX_UD_SCREEN_BINDING. The data in this table will be checked in as seed data.

The columns in this table are explained as follows:

 Table 6–1
 FLX_UD_SCREEN_BINDING

S.No	Column Name	Explanation
1	TASK_CODE	This is the task code of the screen.
2	JSFF_PATH	This is the path for the jsff relative to the root folder.
3	BINDING_CTX	There is a row in this table for every relevant UI attribute. Among these rows only the key fields are of interest in the case of UDF. This shows the page definition where this field figures. for example, com_ofss_fc_ui_taskflows_priceDefinitionMaintenancePageDef.
		And if the field happens to be inside an inner task flow, then it will be a concatenation of the wrapper jsff pagedef with the inner task flow definition id and then the inner task flow definition pageDef. (as given below).
		com_ofss_fc_ui_view_taxation_taxWaiverPageDef#partyDetailsTaskFlowDefn1#com_ ofss_fc_ui_taskflows_partyDetailsPageDef.
4	VO_ITERATOR	The Iterator name for the attribute.
5	VO_ ATTRIBUTE	The UI attribute name. This should be the attribute name inside the vo.
6	ENTITY_FIELD	This is the fully qualified entity name for the corresponding Entity.
7	PARENT_ ENTITY	If there are multiple entities mapped to the screen, then one of the entities will be the parent entity, and the same needs to be marked as 'Y'.

Figure 6–1 UDF Metadata

A UDF	TESTING~2 ×	APR_UD_SCREEN_E	BINDING.sql ×				
	🥦 🐮 🛃 i l	🗟 🖪 I 🏦 🔩 🥔	• 🗔 I				🗐 L
Workshe	eet Ouerv Bu	ilder					
5	select * fro	n flx ud screen	binding;				
6	*1						
	s (
Que	ry Result X						
1 📌 📇	🔂 🎭 SQL I	Fetched 50 rows in 0.	031 seconds				
	TASKCODE	JSFF_PATH	BINDING_CTX	VO_ITERATOR	VO_ATTRIBUTE	ENTITY_FIELD	PARENT_ENTITY
7	PR005	/com/ofss/fc	com_ofss_fc_ui_task	interestViewObj1Iterator	code	com.ofss.fc.domain.p	Y
8	PR006	/com/ofss/fc	com_ofss_fc_ui_task	PriceIdLOVV01Iterator	EffectiveDate	com.ofss.fc.domain.p	Y
9	PR006	/com/ofss/fc	com_ofss_fc_ui_task	priceDefinitionV01Iterator	priceDefinitionId	com.ofss.fc.domain.p	Y
10	PR007	/com/ofss/fc	<pre>com_ofss_fc_ui_task</pre>	pricePolicychartV01Iterator	effectiveDate	com.ofss.fc.domain.p	Y
11	PR007	/com/ofss/fc	com_ofss_fc_ui_task	pricePolicychartV01Iterator	pricePolicyChartId	com.ofss.fc.domain.p	Y
12	PR009	/com/ofss/fc	com_ofss_fc_ui_task	tierCriteriaDef1Iterator	tiercriteriaid	com.ofss.fc.domain.p	Y
13	PR015	/com/ofss/fc	com_ofss_fc_ui_task	BundleCampaignBenefitChart	priceBenefitType	com.ofss.fc.domain.p	Y
14	TDS02	/com/ofss/fc	com_ofss_fc_ui_task	TaxWaiverCriteriaV01Iterator	waiverCriteria	com.ofss.fc.domain.t	Y
15	TDS06	/com/ofss/fc	com_ofss_fc_ui_view	TDSummaryTableV01Iterator	accountNumber	com.ofss.fc.domain.t	(null)
16	TDS06	/com/ofss/fc	com_ofss_fc_ui_view	PartyDetailsV01Iterator	PartyId	com.ofss.fc.domain.t	(null)
17	TDS06	/com/ofss/fc	com_ofss_fc_ui_view	(null)	(null)	com.ofss.fc.domain.t	(null)
18	TDS06	/com/ofss/fc	com_ofss_fc_ui_view	TDSummaryTableV01Iterator	waiverStartDate	com.ofss.fc.domain.t	(null)
19	LCM52	/com/ofss/fc	com_ofss_fc_ui_task	LCCMatrixMainV01Iterator	id	com.ofss.fc.domain.l	Y
20	LCM61	/com/ofss/fc	<pre>com_ofss_fc_ui_task</pre>	ExposureLimitsV01Iterator	limitsFact	com.ofss.fc.domain.l	Y
21	LCM037	/com/ofss/fc	com_ofss_fc_ui_view	CollateralValuerPreference	valuerID	com.ofss.fc.domain.l	Y
22	LCM21	/com/ofss/fc	com_ofss_fc_ui_task	CollateralDedupeV01Iterator	collateralCategory	com.ofss.fc.domain.l	Y
23	LCM05	/com/ofss/fc	view_InsuranceProvi	InsuranceProviderV01Iterator	insuranceProviderCode	com.ofss.fc.domain.l	Y
24	LCM21	/com/ofss/fc	com_ofss_fc_ui_task	CollateralDedupeV01Iterator	collateralType	com.ofss.fc.domain.l	Y
25	PM024	/com/ofss/fc	com_ofss_fc_ui_view	InterestRuleDefinitionV011	productCode	com.ofss.fc.domain.p	N
26	CS03	/com/ofss/fc	com_ofss_fc_ui_task	BankParametersV01Iterator	BankCode	com.ofss.fc.domain.c	N
27	PMU019	/com/ofss/fc	com_ofss_fc_ui_task	ProductGroupRoleHeadV01Ite	productGroupCode	com.ofss.fc.domain.p	Y
6.1.2 Seed Data for the Task Codes

The UDF linkage to different services is done through the screen task flow codes.

The Task Flow code and a meaningful description need to be populated in FLX_UD_ AVLBL_TASK_CODES table.

Task Code LOV in the UDF Linkage screen shows the values from this table, in order to attach the different UDF codes to this.

6.1.3 Screen Changes for Incorporating UDF

The following changes should be taken care of to incorporate UDF functionality to a screen.

Changes to the UI/Middleware

There are no changes to be done to the UI/Middleware to enable UDF, except in a few special cases.

In case of a transaction screen, the screen type will be taken as input by default. If a screen happens to be an enquiry screen, then the parameter ("TransactionScreenType") should be passed accordingly. There are some other special cases which are explained in a subsequent section.

Changes to the Middleware

The design is in such a way that a transaction service saved will use the Transaction Reference number (SessionContext.internalReferenceNumber) for saving the UDF Details. This has been done because it will be difficult to link the transaction services to a single Entity. A Transaction spans multiple Entities.

Typically a Maintenance Service (domain layer) will extend "MaintenanceDomainService". Hence, the code has been put inside create(),update() and merge() to extract the key fields from the Entity instance using reflection.

MaintenanceDomainService. extractKeyFromDomainObject(AbstractDomainObject)

The above method does the following:

- 1. Take the data from flx_md_key_fields for the Entity and extract the Key values from the entity instance. (keyvalue1#keyvalue2#keyvalue3)
- **2.** Form the key attributes in a similar fashion (keyattr1#keyattr2#keyattr3)
- **3.** Take the fully qualified Entity name from the entity passed and store the same.

Finally the data is stored into the FCRThreadAttribute as shown below:

Figure 6–2 Data Stored into the FCRThreadAttribute

```
}
if(FCRThreadAttribute.get(FCRThreadAttribute.UDF_KEY)==null){
HashMap<String, String> udfMap = new HashMap<String, String>();
udfMap.put(UDFKeyThreadAttributeMapConstants.KEY_VALUE, keyAttrData);
udfMap.put(UDFKeyThreadAttributeMapConstants.KEY_ATTRIBUTES, sbAttribute.toString());
udfMap.put(UDFKeyThreadAttributeMapConstants.ENTITY_NAME, fullyQualifiedName);
if (logger.isLoggable(Level.FINE)) {
    logger.log(Level.FINE, formatter.formattMessage("Inside %s.setEntityKeyForUDF().Setting udfKey map into FC
        THIS_COMPONENT_NAME));
}
FCRThreadAttribute.set(FCRThreadAttribute.UDF_KEY, udfMap);
}
```

When the UDF service is called to persist the UDF data, the following checks happen:

- 1. If the UDFDTO contains the key in it, then it is taken to save the UDF.
- 2. If not, the FCRThreadAttribute is accessed and the key is taken from there.
- **3.** If step#2 also does not yield the key, then it saves with the internalReferenceNo from SessionContext. Transaction Services typically should not extend "MaintenanceDomainService", and hence will fall under this option to get the key.

6.1.4 Linking of UDF to a Screen (Taskflow Code)

UDF can be linked to a particular task flow code in three simple steps.

- 1. Create the Required User Defined Fields using the UserDefinedFields Definition Screen (UDF01).
- **2.** Link the Fields to the TaskFlow Code using the User Defined Fields Linkage Screen (UDF02).

(Optional) If there is a requirement to use the Associated UDF functionality, Link the Associated UDF's in this screen (UDF03). If the Associated UDF's are not required, the steps 1 and 2 will suffice.

6.2 Control Flow for UDF

This section describes the control flow of UDF.

6.2.1 Initial Screen Load

UDF task flow has been put on the template (maintenance and transaction) and it will be enabled for the screens using these templates, with an exception of a few because of the way the screen and/or service has been done.

When the screen loads the template page, Definition is initialized, and as a part of it the UDF region is initialized.

When the UDF region is initialized, the 'LinkedUDFsBean' is initialized, and from the constructor of which, the LinkedUDFsHelper is initialized.

As part of the helper initialization, UDF metadata is fetched, UDF VO is initialized and the related UDF fields (the ones linked to the task code) are shown on the screen.

6.2.2 Extracting UDF Values on Submission

The following steps are involved in extracting UDF values on Submission:

Step 1

Appx Layer is enabled for the Services that need to enable UDF. Enabling the appx for the service can be done as follows.

Step 2

From the First Layer (Proxy Façade), the data details for the UDF are extracted, with a call to **com.ofss.fc.ui.core.adfhelper.ADFProxyLayerHelper.**

Step 3

The data thus obtained into the UDF DTO is passed to the subsequent layer json client till the Service Spi, if the host application layer is APPX.

Figure 6–3 LinkedUDFDTO



Step 4

LinkedUDFsRegionHelper.getLinkedUDFsHelper() is get the instance of the LinkedUDFsHelper from the respective container, using which it the UDF DTO can be extracted.

Figure 6–4 Extracting UDF DTO using instance of the LinkedUDFsHelper



Step 5

From the ServiceSpi, the call to the actual service is made along with a call to the UDF Service.

A new property will be available in the UIConfig.properties. This is being added to get around the circular dependency which could otherwise come into existence.

Figure 6–5 UIConfig.properties

```
50 # Time interval for Common Enumeration loading
51 ENUM.RELOAD.INTERVAL=3600000
52
53 # Proxy layer helper class for client dependent method implementations
54 # like fetching UDF data in appx layer. We can add methods for supporting
55 # other input parameters also in this helper.
56 adf.proxy.layer.helper= com.ofss.fc.ui.core.adfhelper.ADFProxyLayerHelper
57
58 # Property added for temporary location for creating pdf file
59 EXPORT.LOCATION=/oracle/deployables/config/temp/
60
```

Package Level Interactions

The following diagram presents the package level interaction for the Extraction of UDF Data:





Sequence Diagram

Following is the Sequence Diagram for UDF DTO Extraction.

Figure 6–7 Sequence Diagram for UDF DTO



6.2.3 Handling the Fetch of UDF Values

UDF Values linked to a key in the screen is done through a call in the Proxy layer to fetch and hence render the values on to the screen.

Package Level Interactions

The following figure explains the package level interactions.





Sequence Diagram

The following sequence diagram explains the code flow:

Figure 6–9 Sequence Diagram



The fetch is dependent on the fact that the Key values are populated on the screen (whether Visible on the screen or not), and the attribute name as well as the VO Iterator name and PageDef name is part of the Seed Data.

6.2.4 UDF Enabling Special Cases

Because of the screen/service design, it is possible that some of the cases will need to be handled differently. These cases revolve around either the extraction of key values in the middleware for any service, or about obtaining the key (with which the UDF had been saved for the current task code (record)).

The cases as identified now are provided below:

Fetch

During fetch of the UDF, there could be situations due to which the key cannot be extracted from the screen, using the metadata provided in the FLX_UD_SCREEN_ BINDING table:

- In the case of a composite Key if one of the attributes involved in the key is not available in the UI View object. The cases in which this would happen are different.
 - An 'if' condition based on which a value is set into the DTO at the time of save from a Enum or a constant, where as the screen displays the value as something else.
 - One of the key attributes is taken from the session context in the middleware, just before the fetch happens, and that attribute is not available anywhere in the UI.

Solution:

The above two cases needs to be handled by raising an event from the screen, after setting the key value into TaskFlow.udf_key. Event Consumer class will consume this event and make a call to the middleware with this key to fetch the LinkedUDFDTO.

 In some cases, the screen is designed to list the multiple records maintained from that screen in a single grid. In this case there are multiple keys (record) available on the screen at the same time

Solution:

Same as the above. Raising an event from the screen during the process of fetch, after setting the key value into TaskFlow.udf_key. Event Consumer class will consume this event and make a call to the middleware with this key to fetch the LinkedUDFDTO.

Save

During Extraction of the key in the middleware, there could be the following cases:

A Maintenance Service calling multiple services within itself. In this case, there are
multiple entries into MaintenanceDomainService, and in all cases the code to
extract they key from the entity will be executed. Now, we don't know the right
entity from which the key needs to be extracted.

Solution:

A new mapping table (new metadata) has been put in place to maintain a mapping between Fully Qualified (FQ) Spi name and the Fully Qualified EntityName. When the call enters the key extraction method it will check if there is an entry in this table, and if yes, it will process only if it comes in with the right Entity. If there is no entry in this table, then it takes the key details from the first Entry into the key extraction method and stores it against the FCRThreadAttribute.udf_Key. Subsequently, there is a check to see if it is already available, and skip if yes.

6.2.5 Tips for Trouble Shooting

UDF panel is not appearing on the screen:

If the UDF panel is not appearing on the screen, you can perform the following checks:

- Check correct key entry is there in FLX_UD_LINKED_UDF, the taskcode linkage table.
- Check if the table (FLX_UD_SCREEN_BINDING) has rows for the task code. The rows in here are required only if it is maintenance template.
- If the screen is using transaction template, check if the TransactionType is input or inquiry.
- If it is input, then check if there are UDF's linked to the taskcode.
- If it is inquiry, then see if the event is getting raised from the grid. If the event is there, then on the grid task flow, check if MultiTabCtx is being passed as a parameter into this task flow. The event is raised after setting the UDF key is set into the respective attribute inside the TaskFlow.
- Check if the page has template binding in its page definition.

UDF field values are not persisting in to DB:

If the UDF field values are not persisting in to DB, you can perform the following checks:

- Check if Appx is enabled in HostApplicationLayer.properties
- Debug inside the extractUDF method of the ADFProxyLayerHelper, it extracts UDF fields from UDF view object, creates linkedUDFDTO and returns the same to

the proxy layer. This can be checked at the service Spi level if the UDFDTO is coming in with values.

 The case could also be that, the UDF values are saved with the user reference number, that is, in the case where the key extraction from MaintenanceDomainService has not happened fine. In this case, even though the UDF is persisted, it does not look like it has, and as a result this would seem like a persistence issue.

UDF field values are not populated in fetch:

If the UDF field values are not populated in fetch:, you can perform the following checks:

- Check if the screen has some special handling to populate the UDF by raising events.
- To raise the event, the keyvalue for the screen needs to be set in "TaskFlow" instance, and this is possible only if MultiTabContext is available in the pageflowscope where the event is getting raised.
- If not, check the call from the Proxyfacade, if the call has to AbstractProxyFacade. populateUDFData(SessionContext sessionContext)
- Check if the screen is a transaction template screen, and it happens to be a pure enquiry screen. If yes, the parameter "transactionScreenType" needs to be passed from the taskcode jsff appropriately.

UDF Data getting saved with the wrong key:

If the UDF data is getting saved with the wrong key, you can perform the following checks:

- If the metadata related to an entity is not available in FLX_MD_KEY_FIELDS, the
 actual maintenance service key cannot be used to save the UDF, and it by default
 gets saved with the SessionContext.internalReferenceNumber, and make it look
 like the UDF values were not saved.
- Similarly, in case of a transaction service, the design is to save the UDF with the transaction reference number (SessionContext.internalReferenceNumber). However, if the service (or one of the component Services happens to extend "MaintenanceDomainService"), the key will be extracted from the entity that is passed into this. This can be handled by passing the screentype parameter properly.

6.3 Limitations and Special Cases

Following is the list of the limitations and special cases:

- There are multiple records getting maintained at one time, that is during one save. There is only one instance of the UDFDTO available in the service signatures for the App services.
- UDF Panel has been added to the template. Currently, it supports Maintenance and Transaction template. UDF cannot be enabled for Dashboard type of screen where there is a collage of information fed by different services, as it is UI (taskcode) dependent rendering for the UDF. This will require coding specifically to be done on the screen where it needs to be rendered.
- Extraction of the UDF Key is dependent on the metadata generated on the Entities, and using a linkage that needs to be maintained with the UI (Vo) attributes and

the service attributes. If the UI attributes or Entity attributes change, the metadata has to be brought in sync.

- Fetching of UDF requires the task code to be supplied along with the Key value, currently though the domain entity name is also captured.
- If there is a grid on the screen, the call to render the UDF for the different keys on the grid needs to happen through an Event raising.
- Multiple fetch Calls might happen from the UI. UI might not be able to differentiate between the main fetch calls and the others when it comes to fetching the UDF values.

7

ADF Screen Customizations

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.

7.1 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:



Figure 7–1 Customization Application View

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

7.2 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:

- 1. From the main menu, choose the **File** -> **open** option. Locate and open the file
- **2.** 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 7–2 CustomizationLayerValues.xml

cor	nfig.)	cml.xml × 📙 DemoContactPoint.java × 📙 OptionCC.java × 🞯 CustomizationLayerValues.xml × 🕼 contactPoint.jsff × 🗎 د ط الع	Ð
16	10 - 1	ind 🕹 🔂	9
	94	Asusiariou.	-
1	95	>	
1	90	<pre><cust-lavers _xmlns="http://xmlns.oracle.com/mds/dt"></cust-lavers></pre>	
	98	<pre><cust-layer id-prefix="s" name="site"></cust-layer></pre>	
1	99	Generated id-prefix would be "s1" and "s2" for values</th <th></th>	
1 :	100	"sitel" and "site2">	
1	101	<cust-layer-value display-name="Site One" id-prefix="1" value="sitel"></cust-layer-value>	
	102	<cust-layer-value 10-prefix="2" display-name="site" v0"="" value="site2"></cust-layer-value>	
	103	Since no prefix was specified on the value>	
	105	ADF SiteCC always returns the value as "site"	
	106	<cust-layer-value display-name="Site" value="site"></cust-layer-value>	
	107		
	108 109 110	<cust-layer name="option" prefix="o"> <cust-layer-value display-name="demo" id-prefix="o1" value="demo"></cust-layer-value> <cust-layer-value display-name="Ubank" id-prefix="o2" value="Ubank"></cust-layer-value></cust-layer>	
	114	Customization layers that are only meant for runtime usage can</th <th></th>	
	115	be excluded in design time by defining size as "no_values">	
	116	<cust-layer name="runtime_only_layer" th="" value-set-size="no_values" 🛵<=""><th></th></cust-layer>	
d		court lawar nore "water" walks get give "large".	
	119	 cust-tayer name= user value-set-size= targe > cl Generated id-prefix would be "usl" and "usl" for values "user!" 	
	120	and "user2" since no prefix was defined per-name level>	
	121	<cust-layer-value display-name="First User" id-prefix="us1" value="user1"></cust-layer-value>	
	122	<cust-layer-value display-name="Second User" id-prefix="us2" value="user2"></cust-layer-value>	
	123	Generated 10-prefix Would be "Useradmin" and "Userquest" for</th <th></th>	
	125	laver level and name level>	
	126	<cust-layer-value display-name="Administrator" value="admin"></cust-layer-value>	
	127	<cust-layer-value value="guest"></cust-layer-value>	
	128	<th></th>	
	130	A/Cubic Cayeria	
S	ourc		

3. Save and close the file.

7.3 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.
- **3.** Go to **Project Properties** for this project and add the required **MDS** libraries in the classpath of the project.

Figure 7–3 Customization Class



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 7–4 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.

7.4 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.



Figure 7–5 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 7–6 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 7–7 Adding com.ofss.fc.demo.ui.OptionCC.OptionCC



Figure 7–8 Adf-config.xml



7.5 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**.

7.6 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 7–9 Customization Developer

8 Preferences	
(Search)	Roles
Diagrams	<u>R</u> ole:
Extensions	O Default Role
External Editor	Enables all technologies.
File Types	Customization Developer
Global Ignore List	Configures the product for customizing metadata.
Http Analyzer	O Database Edition
Java Visual Editor	Includes only features for core database development.
JSP and HTML Visual Ec Mouseover Popups	Java EE Edition Includes only features for core Java EE development.
Run	🔿 Java Edition
Shortcut Keys	Includes only features for core Java development.
TopLink	
UML Unaco Desertion	
Wersioning	
Web Browser and Prox	
WS Policy Store	
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 7–10 Selecting Always Prompt for Role Selection on Start Up

Select Role Select the role that matches your requirements. You can also change roles using the Roles page in preferences.	ê
<u>R</u> ole:	
Default Role Enables all technologies.	
 Customization Developer Configures the product for customizing metadata. 	
Database Edition Includes only features for core database development.	
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	
ОК	Cancel

3. On restarting JDeveloper, you will be prompted for role selection. Select *Customization Developer Role.*

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.

4. 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.

5. Select the Customization Context for which, the customizations you edit should be applicable.

Figure 7–11 View Customization Context

View - Customization Context ×			
View without Customizations			
 Edit with following Customization C 	ontext		
Tip layer	Name		Value
0	option		demo (demo) 🔻
			demo (demo)
		2	Ubank (Ubank)
Customization Context : option/dem	0		
Override global laver values			

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

7.7 Customization Examples

This section describes the customization examples.

7.7.1 Adding a Validator to Input Text Component

In this first example of customization, we will be adding a Validator to an Input Text Component present in a screen.

Use Case Description: The **Party -> Contact Information -> 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. For certain types, the Telephone Details tab is enabled in which the user can enter the telephone details. A custom component numericCode is used for getting the user's input for Telephone Number. We will be adding a Validator to this component which will validate the user's input against a regular expression.

Figure	7–12	Contact Point
--------	------	---------------

I <u>0</u> 41						
ntact Point						
<u>R</u> ead P Create	🖉 Updat	e				✔ O <u>k</u> 🥢 Clear 🖹 Cancel 🔗
Party Details						
		Party ID	200001929	Date of Birth	02-Jul-1986	
NO		Full Name	Lashawn Hasty	Gender	Undisclosed	NO
IMAGE	н	lome Branch	082991-Demo Bank Operat	tions Roles	Customer	IMAGE
AVAILADIE		Party Class	Others	Onboarding Date	31-Jan-2013	AVAILABLE
MYMILMDLE		Party Type	IND			AVAILADLE
Address Details						
	 Country 	US			State KY	
	* City	kentucky			 Zip 91790 	
	Line1	815			Line2	
	Line3				Line4	
	Line5				Line6	
	Line7				Line8	
Contact Point De	tails				4	
Contact F	Point Type	Mobile		Contact Preferen	ice Type Home	
Seasonal S	Start Date	_		Seasonal E	End Date	
 Allowed 	Purposes	Communi	cation			
		Alert				
Preferre	d Contact	Preferred	Contact			
Marketin	g Consent	Marketing	Consent			
larketing Consent :	start Date			Marketing Consent b	and Date	
Telephone Detai	Is					
Cou	ntry Code			An	ea Code	
	 Number 	9833445566	i	E	xtension	
Service	e Provider			vo	IP Code	
Timing Preferen	ces					

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

Step 1 Create Customization Project

- **1.** Create a project (*com.ofss.fc.demo.ui.view.party*) to hold the customization, as mentioned in the section **Customization Project**.
- 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 Validator Class

All the files which are not customizable (*for example - Java Classes*), are read only in the *Customization Developer Role*. Hence, you have to create the Validator Class in the Default Role itself. Create the class with following features:

- 1. To get a handle on the *numericCode* component of the *Telephone Number*, include a private member in this class of type *ContactPoint* which is the backing bean for this screen.
- **2.** Add a validator method with the following signature public void methodName (FacesContext facesContext, UIComponent uiComponent, Object object).

Figure 7–13 DemoValidator.java



Step 3 Create Managed Bean

After creating the validator class, you have to switch to the *Customization Developer Role*.

- 1. Select the required customization context (for example demo).
- **2.** Open the customization project's *adfc-config.xml* which is present in the *WEB-INF* folder.
- **3.** In the **Managed Beans** tab, add the validator class as a managed bean with request scope as follows.

Figure 7–14 Managed Beans

≇contactPoint.jsff ×	👸 DemoValidator.ja	wa × Madfc-config xml ×		28
General Description	S Managed B	eans		÷ ×
Activities Control Flows	Name * 🔺	Class *		Scope *
Managed Beans	DemoValidator	com.ofss.fc.demo.ui.view.party.contact	Point.validator.DemoValidator	request
Metadata Resources	Managed Pro	perties: DemoValidator		÷>
	Name * 🔺	Class	Value	

When you save the changes, JDeveloper creates a customization XML to store the changes. For the above change, JDeveloper creates the XML *adfc-config.xml* in the *WEB-INF/mdssys/cust/option/demo* folder where option is the Customization Layer Name and demo is the Customization Layer Value.

2

Figure 7–15 Creating Managed Bean - Customization XML



Step 4 Open Screen JSFF

After adding the Validator class as a managed bean, open the JSFF for the screen and perform the below mentioned steps:

- 1. In the *Application Navigator*, open the *Navigator Display Options* for *Projects* tab.
- 2. Select the *Show Libraries* option.

Figure 7–16 Opening JSFF Screen - Show Libraries



- **3.** In the navigator tree, locate the JAR that contains the screen (*com.ofss.fc.ui.view.party.jar*).
- Inside this JAR, locate the screen JSFF (com.ofss.fc.ui.view.party.contactPoint.form.contactPoint.jsff) and open it. You will notice that you cannot modify this JSFF in the editor.
- **5.** Locate the *<fc:numericCode>* component for the Telephone Number.





Step 5 Bind Validator to Component

- 1. Select the aforementioned component and open the *Property Inspector* tab.
- 2. For the property *Validator*, select the *Method Expression Builder*.
- **3.** In the pop-up, locate the *Validator Class Method* under the *ADF Managed Beans*. When you select this method and save, the component is bound to the validator.

NumericCode - Pro	perty Inspector ×		
🕄 I 📌 💁 I 🥒 👘	🚺 Find 🕹 🕄		?
e Id:	telNumber		^
Immediate:	<default> (false)</default>	• ~	
LabelAndAccessKey:]~	
MaxIntegerDigits:	20]~	
MinLength:	1]~	
PostValueChange:]~	
ReadOnly:	true	• ~	
Rendered:	<default> (true)</default>	• ~	
Required:	<default> (false)</default>	• ~	
Secret:	<default> (false)</default>	• ~	
ShortDesc:]~	
Simple:	<default> (true)</default>	• ~	-
StyleClass:	Į	\ ~	
• Validator:	#{DemoValidator.telNumberValidator}	~	
• Value:		\sim	-
Visible:	<default> (true)</default>	·] ~	~

Figure 7–18 Bind Validator to Component - Validator Property

Figure 7–19 Bind Validator to Component - telNumberValidator

Select values from methods or directly type the expression:	e method here: 🦃 🖗	in 🥔
#{DemoValidator.telNumberValidator}		
- DF Controller Objects		^
ADF Managed Beans		
DemoPartyGeneralInformation		
🖃 💊 DemoValidator		
telNumberValidator		
applicationScope		
backingBeanScope	~	
demoPartyOpenEnum		
pageriowscope		
E Faces' Resource Bundles		
		×

4. When you save the changes, JDeveloper creates a customization XML to store the changes.

For the above change, JDeveloper creates the XML *contactPoint.jsff.xml* in the *com/ofss/fc/ui/view/party/contactPoint/cust/option/demo* folder where option is the Customization Layer Name and demo is the Customization Layer Value.

Figure 7–20 Bind Validator to Component - contactPoint.jsff.xml



Step 6 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:
 - 1. Customization Project JAR (com.ofss.fc.demo.ui.view.party.jar)
 - 2. Customization Class JAR (com.ofss.fc.demo.ui.OptionCC.jar)
 - 3. All dependency libraries and JARS for the project.
- **2.** Start the application and navigate to the *Party -> Contact Information -> Contact Point* screen.
- **3.** Input a *Party Id*.
- **4.** Select a *Contact Point Type* and provide input in the *Telephone Number* input box.

If the input is invalid as per the *Validator Class Method*, an error message is displayed to the user.

PI <u>0</u> 41				
Contact Point				
🛄 Read 🕼 Create 🥒 Upda	te			✔ Ok 🧳 Clear 🕱 Cancel 🚔 Print
Party Details				j.
NO IMAGE AVAILABLE	Party ID 200001929 Full Name Lashawn Hasty tome Branch 082991-Demo Ban BR Party Class Others Party Type IND	Date of Birth Gender ak Operations Roles Onboarding Date	02-Jul-1986 Undisclosed • Customer 31-Jan-2013	NO IMAGE AVAILABLE
Address Details				
Country City Une1 Line3 Line5 Line7 Contact Point Details	US kentucky 815		State KY Zip 91790 Line2 Line4 Line6 Line8	
Contact Point Type Seasonal Start Date Allowed Purposes Preferred Contact Marketing Consect	Communication Alert Preferred Contact	Contact Preteren Seasonal E	nd Date	
Marketing Consent Start Date	Marketing Consent			
Telephone Details		Wahile Number should be 10 d	nber ligits long	
Country Code		Enter between 1 and 20 charac	cters	
Number Service Provider	96191024	E) VO	xtension IP Code	
Timing Preferences	9619	1024		
DND				N

Figure 7–21 Contact Point screen

7.7.2 Adding a UI Table Component to the Screen

In this second example of customization, we will be 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. We will be 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.

PI <u>0</u> 40	P1030												
		Party ID	00000	5296					Full Name				
		First name							Last Name				
	1	Short Name				1			Email ID				
												Search	Reset
Party Se	earch Results												Clear
View-	Detach							2	•				
then t	Detach						Date of Birth or						
irty ID		Name		Туре	Number of Role	es	Incorporation	Party Class	Email ID				
0005296	,	Daniel John	son	Individu	2		05-Dec-1980	Others	dipika.patnaik@oracl	e.com			
Account	t Details						Account Specifi	c Details					
rial Numt	ber Account	t Number A	ccount 1	Туре			Account Numb	er 000000000	0007510	Account Title	Daniel Corp		
	000000	000000751L0	ON				Account Openi	ng 15-Jan-201	6	Account Currency	AUD		
							Party	ID 000005297		Party Name	Daniel Corp		
							or	er LOF003 NA	B TAILORED HOME	Branch	082991 U Bar		
					1			COMIN CON	003			ik Operati	ons BP
							Facility Co	de FC2016015	003 0018764	Facility Name	Home Loan	ik Operati	ons BF
						•	Facility Co Total Disburs Amou	de FC2016015 ed \$200,000.0	003 0018764 0	Facility Name Last Disbursement Date	Home Loan 31-Jan-2016	ik Operati	ons BF
						8	Facility Co Total Disburs Amou Date Of Matur	de FC2016015 ed \$200,000.0 int ity 15-Jan-201	003 0018764 0 7	Facility Name Last Disbursement Date Accrual Status	Home Loan 31-Jan-2016 Normal	ik Operati	ons BR
						8	Facility Co Total Disburs Amou Date Of Matur	de FC2016015 ed \$200,000.0 nt ty 15-Jan-201	003 0018764 0 7	Facility Name Last Disbursement Date Accrual Status	Home Loan 31-Jan-2016 Normal	ik Operati	on

Figure 7–22 Adding a UI Table Component - Party Search screen

Figure 7–23 Adding a UI Table Component - Related Party screen

P1040 P1030		
Related Party		
Read Create 🖉 Update		🛷 Ok 🥜 Clear 🔣 Exit 📇 Print
Primary Party Information		
Party Details		
• Party	D 000005296 Date of Birth 05-Dec-1980	NO
NO Full Nan	e Daniel Johnson Gender Undisclosed	NO
IMAGE Home Bran	h 082991-U Bank Operations BR Roles Customer	IMAGE
AVAILARIE Party Cla	• Director	AVAILARIE
Party ly	e IND Onboarding Date 15-Jan-2016	ATAILADL
Address Details		
Relation Details		
View - R Detach		
Serial No. Party Id Related Party I	Relationship Type Direct Relation Name Name Share Collateral Share Ex	posure Share Income
1 000005295 000005296	Business Authorized Signat Authorized Signat	
L	2	
1	N°	
1		
1		
1		
Columns Hidden 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.

Figure 7–24 Creating Binding Bean Class

🕹 Demo	PartySearchMaintenance.java X
👶 - Fin	d 🛛 🕹 🚱 🖄 🤞 🖓 📽 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓
	package com.ofss.fc.demo.ui.view.party.partySearch.backing; import;
	<pre>public static final String PARTY_RELATIONSWIP_TABLE_V0 = "RelatedPartiesAndDetailsTableVOIIterator"; public static final String PARTY_SEARCH_MAINTERNAKCE_PAGE_DEFN = "com_ofss_fc_ui_view_party_BartySearchMaintenancePageDef1";</pre>
	private RichPanelGroupLayout pgll; private RichPanelBox olpol; private RichPanelCollection olpc1; private RichTable oll1; private RichOutputText olot1; private PartySearchMaintenance partySearchMaintenance;
8	public DemoPartySearchMaintenance() {
	<pre>partySearchMaintenance = (PartySearchMaintenance) ELHandler.get(PartyProxyConstants.BACKING_BEAM_PARTY_SEARCH);</pre>
8	<pre>public void setPgl1(RichPanelGroupLayout pgl1) { this.pgl1 = pgl1; }</pre>
8	public RichFamel&noupLayoul getFgll() { this.pgll = partySearchMaintenance.getFgll(); return pgll; 2
8	<pre>public void set01pb1(RichPanelBox olpb1) { this.olpb1 = olpb1; }</pre>
8	<pre>public RichPanelBox getOlpbl() { return olpbl; }</pre>
8	<pre>public void set01pc1(RichPanelCollection olpcl) { this.olpc1 = olpc1; }</pre>

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.





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 7–26 Creating Managed Bean

adfc-config.xml ×					
General Description	S Managed Beans				
Activities Control Flows Managed Beans	Name * 🔺 DemoPartySearchMaintenance	Class * com.ofss.fc.demo.ui.view.party.partySea	rch.backing.DemoPartySearchMaintenance	Scope * request	
Metadata Resources	Managed Properties: DemoPartySearchMaintenance Name * Class Value				

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.





- 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

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:

PartySearchMaintenancePageDef.xml ×						
Page Data Binding Definition	^					
This shows the Oracle ADF data bindings defined for yo	our page. Select a binding to see its relationship to the und					
Data Binding Registry: //com/ofss/fc/ui/view/party/Data	aBindings.cpx					
Bindings and Executables Contextual Events Pa	arameters					
🗆 Model						
Bindings 🐥 🖉 💥	Executables 🐥 🖉 💥					
 handleTaskCodeChangeEvent handleCustTaskCodeEvent handleAccountTaskCodeEvent AccountDetailsViewObj1 	variables taskFlow - PartySearchTaskFlow1 AccountDetailsViewObjlIterator taskFlow - accountDetailsTaskFlow1					
Select the category of components from	n which you would like to find an item:					
Select the item to be created:						
graph ist ist istofValues methodAction anavigationlist table tree intreetable						
Description:						
Tree binding for the control. Tree bin hierarchical list of attributes derived relationships as specified by the bus <u>H</u> elp	Tree binding for the control. Tree binding lets users view a hierarchical list of attributes derived from master-detail relationships as specified by the business services in your Model Help OK Cancel					

Figure 7–28 Adding View Object Binding to Page Definition - Add Tree Binding

PartySearchMaintenancePageDef.xml ×						
Page Data Binding Definition	<u>^</u>					
This shows the Oracle ADF data bindings defined for your page. Select a binding to see its relationship to the und						
Data Binding Registry: /com/ofss/fc/ui/view/party/DataBindings.cpx						
Bindings and Executables Contextual Events Parameters						
🗆 Model						
Bindings Provide Exe	cutables 👎 🖉 💥					
 handleTaskCodeChangeEvent handleCustTaskCodeEvent handleAccountTaskCodeEvent AccountDetailsViewObj1 	 variables taskFlow - PartySearchTaskFlow1 AccountDetailsViewObj1lterator taskFlow - accountDetailsTaskFlow1 					
😣 Create Tree Binding						
Select the data source for the root tree node, and decide wh display in the tree. To add additional tree level rules for child parent tree level rule andclick the Add icon. If no child collect selected node, the Add icon is disabled.	Select the data source for the root tree node, and decide which attributes you want to 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.Related	Parties V Add					
Tree Level Rules: PartyAppModuleDataControl.RelatedP	artiesAndDetailsTableV01					
📄 com. ofss. fc. ui. model. party. relatedparty. vo. RelatedPar	tiesAndDetailsTableV					
Accessor: Folder Label:	Enable Hiltering:					
Available Attributes: Display Attribute	s:					
UdfKey DeleteCheck						
DirectRelationsh	ipName n Context					
Inversekelations						
PartyType	mo (demo).					
ReadOnlyFlag RelatedPartvld	×					
	ето					
Help	OK Cancel					

Figure 7–29 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 7–30 Page Data Binding Definition - Insert Item

PartySearchMaintenancePageDef.xml ×			•
Page Data Binding Definition			~
This shows the Oracle ADF data bindings d Data Binding Registry: <u>/com/ofss/fc/ui/vie</u> Bindings and Executables Contextual Model	efined for your page w/party/DataBinding Events Paramete	e. Select a binding to see its re <u>IS.CPX</u> rS	lationship to the und
Bindings image: BindleTaskCodeChangeEven image: BindleCustTaskCodeEven image: BindleAccountTaskCodeEven AccountDetailsViewObj1 RelatedPartiesAndDetailsT	vent tt vent TableV01	Executables variables taskFlow - PartySearc AccountDetailsViewOb taskFlow - accountDet taskFlow - accountDet	hTaskFlow1 ojliterator ailsTaskFlow1 ailsTableVO1Iterator
🙁 Insert Item			
Sele <u>c</u> t the category of components	from which you wo	uld like to find an item:	1
Generic Bindings			-
Select the item to be created:			
Ingraph 네 list 데 listofValues			^
 methodAction navigationlist table tree tree treetable 	2		•
Description:			
Method binding for the control.			~ ~
Help		ОК	Cancel

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

PartySearchMainten	ancePageDef yml X				[
age Data Binding [Definition					
This shows the Oracle ADE data hindings defined for your name. Select a hinding to see its relationship to the und						
This shows the oracle Abr data bindings defined for your page. Select a binding to see its relationship to the did						
Data Binding Registry: /Com/ofss/fc/ui/view/party/DataBindings.cpx						
Bindings and Executables Contextual Events Parameters						
⊡ Model						
Bindings		🕂 🖊 🕂	Executables		- - - <i>l</i>	
🔚 hand	eTaskCodeChangeEve	nt	variables	Part Search Task Flow		
🖂 hand	eAccountTaskCodeEve	ent	AccountDe	tailsViewObjliterator		
	untDetailsViewObj1		💽 taskFlow -	accountDetailsTaskF	lowl	
💦 Relat	edPartiesAndDetailsTa	bleVO1	RelatedPa	rtiesAndDetailsTable\	/Oliterator	
😣 Create Actio	on Binding					
Select a data collect	ion and the action you	want your conf	rol to initiate. The con	trol initiates the actio	n on	
the data objects of t	he selected collection.					
Data <u>C</u> ollection:		3				
DDBankerAcc	ountConfigurationEven	tProducer			^	
	unteventProducer					
DemoPartySe	archConsumer					
🕀 🛃 DepositBasic	DetailsEventProducer					
🕀 🛃 DirectRolePar	tyDetailsConsumer					
DocumentCat	tegoryEventConsumer				~	
Select an Iterator:				• <u>N</u> e	w	
Operation: h	andleAccountTaskCode	AndPartyRelati	onshipEvent(Object) 🔻			
	Apply to all iterators i	n page defintio	n			
Parameters :	_					
Name	Type	Value		Option		
object	Java.lang.Object					
Help				OK Canc	el	

Figure 7–31 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 7–32 Edit Event Map

🔚 PartySearchMaintenancePageDef.xml - Structure ×						
📌 🗟						
⊕- 📴 Warnings (1) - 🖵 Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved. ⊖- 🔁 PartySearchMaintenancePageDef						
parameters	_	Insert inside PartySearchMaintenancePageDef	•			
		Generate Resource Bundle				
eventMap	@	Edit Authorization				
i∰… 🔰 handleTaskCodeChangeE	5 6	Edit Event Map				
handleCustTaskCodeEven	Ж	Cut	Ctrl-X			
		Copy	Ctrl-C			
		<u>P</u> aste	Ctrl-V			
	×	<u>D</u> elete	Delete			
	_	<u>G</u> o to Source				
		Find <u>U</u> sages	Ctrl+Alt-U			
		Go <u>t</u> o Properties				
		Go to D <u>e</u> claration				

- **3.** In the **Event Map Editor** panel, edit the mapping for the required event.
- **4.** Select the newly added Event Consumer's method.
Figure 7–33 Event Map Editor

💈 Event Map Edit	or			
Add, Update and Delete	event entries	3		
Events Map				🕂 🖊
Producer PartySearchTaskFlow1. PartySearchTaskFlow1. PartySearchTaskFlow1.	Event Party handl Party handl Party handl	Name eTaskCodeChang eCustTaskCodeE eAccountTaskCod	Consumer handleTaskCode handleCustTask handleAccountTa	ChangeEvent CodeEvent askCodeEvent
Modify EventMa	ap Entry			
<u>P</u> roducer: <u>E</u> vent Name: <u>C</u> onsumer:	tySearchTo handleAcco PageDef.ho	askFlow1.PartySear ountTaskCodeEvent andleAccountTaskCo	chPageDef.raise/	AccountEvent 🗸
Consumer Params			4 X	
Param Name		Param 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 7–34 Add UI Components to Screen

Part	tySear	chMaintenance.jsff 🗵 🐨
(80-F	ind	\$ (c)
	1	<af:activecommandtoolbarbutton <="" binding="#{PartySearchMaintenance.cbl}" id="cbl" text="#{rbPartySearchMaintenance.LBL_Exit}" th=""></af:activecommandtoolbarbutton>
		/af:panelGroupLayout
		<af:region binding="#{PartySearchMaintenance.r1}" id="r1" value="#{bindings.PartySearchTaskFlov1.regionModel}"></af:region>
	8	<ar:panplbox 10="01p01" t<="" text='#trone(atedFarty.LBL_FELATION_DETAILS_PANEL)"' th="" xmlns:at="http://xmlns.oracle.com/adt/faces/rich"></ar:panplbox>
		<pre><ar:panettoitection binding="#iummorartySearchwaintenancemeipter<br" log-oipti"="" xmins:ar="http://xmins.oracle.com/aof/faces/fich"><arbs <br="" billiong="" lights="" searchaintenancemeipter="">searchaintenancemeipter/searchaintenancemeipter/searchaintenancemeipter/searchaintenancemeipter/ searchaintenancemeipter/searchaintenancemeipter/searchaintenancemeipter/searchaintenancemeipter/ searchaintenancemeipter/searchaintenancemeipter/searchaintenancemeipter/ searchaintenancemeipter/searchaintenancemeipter/ search</arbs></ar:panettoitection></pre>
		<pre>saf:date xmonstal= http://xmonstale.com/adi//acce/rich vatue *pinoings.retateoraritesenderalistateori.com/adi//acce/rich vatue *pinoings.retateori.com/adi//acce/rich vatue *pinoings.retateoraritesenderalistateori.com/adi//acce/rich vatue *pinoings.retateori.com/adi//acce/rich vatue *pinoings.retateoraritesenderalistateori.com/adi//acce/rich vatue *pinoings.retateori.com/adi//acce/rich vatue *pinoings.retateoraritesenderalistateori.com/adi//acce/rich vatue *pinoings.retateori.com/adi//acce/rich vatue *pinoings.retateoraritesenderalistateori.com/adi//acce/rich vatue *pinoings.retateoraritesenderalistateori.com/adi//acce/rich vatue *pinoings.retateori.com/adi//acce/rich vatue *pi</pre>
		<pre>cafeurtert values af="http://walue.com/action/</pre>
	8	<pre><af:column adf="" faces="" headertext="#irbRelatedParty.LBL PARTY 1</pre></td></tr><tr><th></th><th></th><td><af:outputText xmlns:af=" http:="" id="olot1" rich"="" sortable="false" value="#{row.PartyId}" xmlns.oracle.com="" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:column></pre>
	8	<pre><af:column adf="" faces="" headertext="#{rbRelatedParty.LBL_RELATED</pre></td></tr><tr><th></th><th></th><td><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></pre>
	_	
	8	<arcoutumn ;="" <="" adf="" eartable="falee" faces="" headertext="sirbGelatedParty LBL DIGECT</th></tr><tr><th></th><th>-</th><th>saf output text values af this //value cracle.com/add/faces/rich" http:="" id="0.0106" le.com="" neaderiext="#/rowelate0Party.LBL_PELAIL</td></tr><tr><th></th><th></th><td><arroupdctext xmthstat= http://xmths.oracte.com/adi/faces/fich value= #frow.hetationshiptype/ 10= 01013 /></td></tr><tr><th></th><th></th><th><pre></pre>caf:column value:af=" rich"="" softable="false" th="" value="#frow.birect@lationshinName" value.org="" xmtns:at="http://xmtns.oracle.com/adv/faces/fich*"></arcoutumn>
		<pre><af:column <="" adf="" faces="" headertext="#{rbRelatedParty.LBL_INVERSE</pre></td></tr><tr><th></th><th></th><td><af:outputText xmlns:af=" http:="" id="olot2" rich"="" sortable="false" td="" value="#{rov.InverseRelationshipName}" xmlns.oracle.com="" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:column></pre>
	8	<af:column =<="" headertext="Share Collateral" id="ole6" sortable="false" td="" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:column>
		<pre><af:selectbooleancheckbox aut="" ch<="" fichos.text"="label=st/boollanchookbay" http:="" label="#{rbRelatedParty.Lbl_s/</pre></td></tr><tr><th></th><th>-</th><td></arccounts</td></tr><tr><th></th><th>-</th><td><pre><ar:column xmuns:ar= http://xmuns.oracle.com/aut/races/fich sortable= taise neader/exit= share Exposure 10= 0.05 >=
cafield.ortBoollanchookbay valuesaf=" lb="" pre="" text=" " xmlns:af="http://xmlns.oracle.com/adf/faces/rich" xmlos.oracle.com=""></af:selectbooleancheckbox></pre>
		 Selection contraction with a selection of a contraction of the contraction o
	8	<af:column headertext="Share Income" id="olc4" sortable="false" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:column>
		<af:selectbooleancheckbox #(partysearchmaintenance.pgl3)"="" 400"="" id='pgl3"' inlinestyle="neight:340px;" label="#{rbRelatedParty.Lbl sh</td></tr><tr><th></th><th></th><td></af:column></td></tr><tr><th></th><th></th><td></af:table></td></tr><tr><th></th><th></th><td></af:panelCollection></td></tr><tr><th></th><th></th><td></artipanelBox></td></tr><tr><th></th><th></th><td><ar:panetGroupLayout Dinging=" layout="scroll" style(<="" styleclass="AFStretchWi</td></tr><tr><th></th><th></th><td><pre><ar:pametsptitter ig= ps2= unging= *irartysearcnmaintenance.ps2}* splitterPosition=" td="" text=" " visible='Taise"' xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:selectbooleancheckbox>
len roa	-	STERIOUS INTERPORT
Desire		n panetnyrouphayouteppit */ anpanethoxeotpot */
Design	1 ponto	a pinoings Preview -

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.

Figure 7–35 Application Navigator



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 7–36 Party Search

	vidual											
	F	arty ID 0000	05296					Full Nan	ne			
	Firs	at name	13230					Last Nam	ne			
	Shor	t Name						Email	ID			
											Search	Rece
Party Search	h Results										Scoren	Clear
View-	Detach				-							
Party ID	Na	me	Туре	Number of Ro	les	Date of Birth or Incorporation	Party Class	Email ID				
000005296	Da	niel Johnson	Individu	2		05-Dec-1980	Others	dipika.patnaik@or	acle.com			
Relation De	tails				_							
View -	Detach				-							
Serial No.	Party Id	Related	Party ID	Relationship	Тур	Direct Relation	Inverse Relation	Share Collateral	Share Exposure Sha	re Income		
1	000005295	000005	296	Business		Authorized Signat	Authorized Signa	10	-			
Account De	tails					Account Specifi	c Details					
Account De	tails Account Nu	mber Account	Type			C Account Specifi	c Details	007510	Account Title	Daniel Corp		
Account De Serial Number	Account Nu 000000000	mber Account	Туре			Count Specifi Account Numb Account Open Di	c Details ber 0000000000 ing 15-Jan-2016 ste	007510	Account Title Account Currency	Daniel Corp AUD		_
Account Def	Account Nu 000000000	mber Account 000751LON	Туре			CAccount Specifi Account Numb Account Opi Di Party	c Details ber 0000000000 ing 15-Jan-2016 ate ID 000005297	007510	Account Title Account Currency Party Name	Daniel Corp AUD Daniel Corp		
C Account Def Serial Number	Account Nu 000000000	mber Account	Туре			Account Specifi Account Numb Account Open D Party Of	c Details er 00000000000 ing 15-Jan-2016 ste ID 000005297 fer LOF003 NAB LOAN - LOF00	707510 TAILORED HOME	Account Title Account Currency Party Name Branch	Daniel Corp AUD Daniel Corp 082991 U Ba	nk Operat	ions BR
C Account Deferrial Number	Account Nu 000000000	mber Account 000751LON	Туре			Account Specifi Account Numn Account Open D Party Of Facility Co	c Details ber 0000000000 ing 15-Jan-2016 ite ID 00005297 fer LOF003 NAB LOAN - LOF00 de FC201601500	207510 TAILORED HOME 33 D18764	Account Title Account Currency Party Name Branch Facility Name	Daniel Corp AUD Daniel Corp 082991 U Ba Home Loan	ink Operat	ions BR
Account Definition	tails Account Nu 000000000	mber Account 000751LON	Туре	_		Account Specifi Account Numt Account Open Di Party Of Facility Co Total Disburs Amon	c Details ber 0000000000 ing 15-Jan-2016 te 1D 000005297 fer LOF003 NAB LOAN - LOF00 de FC201601500 de \$200,000.00 unt	007510 TAILORED HOME 03 D18764	Account Title Account Currency Party Name Branch Facility Name Last Disbursement Date	Daniel Corp AUD Daniel Corp 082991 U Ba Home Loan 31-Jan-2016	ink Operat	ions BR
Account De	tails Account Nu 000000000	mber Account 000751LON	Туре			Account Specifi Account Numk Account Open D Party Of Facility Co Total Disburs Armon Date Of Matur	c Details er 0000000000 ing 15-Jan-2016 te ID 000005297 fer L0F003 NAB LOAN - L0F00 de FC201601500 de FC201601500 de S200,000.00 int ity 15-Jan-2017	207510 TAILORED HOME 3 218764	Account Title Account Currency Party Name Branch Facility Name Last Disbursement Date Accrual Status	Daniel Corp AUD Daniel Corp 082991 U Ba Home Loan 31-Jan-2016 Normal	nk Operat	ions BR
Account De Serial Number	tails Account Nu 000000000	mber Account 000751LON	Туре			Account Specifi Account Numi Account Open Di Party Of Facility Co Total Disbors Date Of Matur Approved Amor	c Details er 0000000000 ig 15-Jan-2016 ite ID 00005297 fer LOF003 NAB LOAN - LOF00 de FC201601500 ed \$200,000.00 int 15-Jan-2017 is \$200,000.00	207510 TAILORED HOME 33 218764	Account Title Account Currency Party Name Branch Facility Name Last Disbursement Date Accrual Status Next Installment Amount	Daniel Corp AUD Daniel Corp 082991 U Ba Home Loan 31-Jan-2016 Normal \$0.00	ink Operat	ions BR

7.7.3 Adding a Date Component to a Screen

In this third example of customization, we will be adding a Date Component to an existing screen to capture date input from the input. This input will be saved in the database.

Use Case Description: The *Party* -> *Contact Information* -> *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. We will be adding a field *Expiry Date* as a date component to this tab. We will be adding a table to the database to save the user input for this field and services for this screen will be added/modified.

PI <u>0</u> 41			
Contact Point			
🛄 Read 💠 Create 🥖 Upda	te		🛷 Ok 🥔 Clear 🔣 Exit 🚔 Print
Party Details			
NO IMAGE AVAILABLE	Party ID 00005295 Home Branch 082991-U Bank Operations BR Impany 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		Seasonal End Date	
Allowed Purpose:	Communication		
Preferred Contac	Preferred Contact		
Marketing Consen	Marketing Consent		
Marketing Consent Start Date		Marketing Consent End Date	
Telephone Details			
Country Code	1	Area Code	
Number	32577789	Extension	
Service Provide	r	VOIP Code	
Timing Preferences			
DND			
DND Star	t	DND End	
Weekdays	Weekdays		
From	1	То	
Weekends	Weekends		
From	1	То	
Hide Modification History			
Created By ofssuser	On 24-Aug-2012 12:00:00 A	Approved	2 OF 2
Approved By ofssuser	On 24-Aug-2012 12:00:00 A	AM Active	

Figure 7–37 Adding a Date Component

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

Step 1 Host Application Changes

In this use case, we need to save the input data in the database of the application, we 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 7–38 Create Table in Application Database

000	REATE TABLE "FLX_PI_CONTACT_	EXPIRY*	
ľ	"PARTY_ID"	VARCHAR2(40 BYTE) NOT NULL ENABLE,	Key Columns
	"CONTACT_POINT_TYPE"	VARCHAR2(3 BYTE) NOT NULL ENABLE,	Expin: Date Field
	"CREATED_BY"	VARCHAR2(254 BYTE) NOT NULL ENABLE,	
	"CREATION_DATE" "LAST_UPDATED_BY"	TIMESTAMP (6) NOT NULL ENABLE, VARCHAR2(254 BYTE) NOT NULL ENABLE,	Record Information Columns
	"LAST_UPDATE_DATE" "OBJECT_VERSTON_NUMBER"	TIMESTAMP (6) NOT NULL ENABLE, NUMBER (9,0) NOT NULL ENABLE.	
	"OBJECT_STATUS_FLAG"	CHAR(1 BYTE) NOT NULL ENABLE,	
	CONSTRAINT "FLX_PI_CONTACT_	EXPIRY_PK" PRIMARY KEY ("PARTY_ID", "(CONTACT_POINT_TYPE", "CONTACT_PREF_TYPE") ENABLE,
)	CONSTRAINT "FLX_PI_CONTACT_ ;	EXPIRY_FK1" FOREIGN KEY ("PARTY_ID") F	REFERENCES "FLX_PI_PARTIES_B" ("PARTY_ID") ENABLE

After creating the table, we will 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.

Figure 7–39 Create Java Project



Step 4 Create Domain Objects

We will 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.
- **2.** Add the properties, getters and setters for the key columns of the table in this class.
- 3. Implement the abstract methods of the superclass.

Figure 7–40 Create Domain Objects

IConta	ctExpiry.java 🕖 ContactExpiryKey.java 🕴 🖉 ContactExpiry.java	•
1 pac	kage com.ofss.fc.demo.domain.party.entity.contact;	
30 100	ort com.ofss.fc.enumeration.ContactPointType:	
4 imp	ort com.ofss.fc.enumeration.ContactPreferenceType:	
5 imp	ort com.ofss.fc.framework.domain.AbstractDomainObjectKey;	
6		
7 pub	lic class ContactExpiryKey extends AbstractDomainObjectKey {	
8		
90	/**	
10	* Serial Version	
11	*/	
12	private static final long serialVersionUID = -4179806027380497671L;	
13		
140		
15	Party Id	
10		
1/	private string partyle;	
10		
20	* Contact Point Tune	
21	*/	
22	private ContactPointType contactPointType:	
23		
240	/**	
25	* Contact Preference Type	
26	/	
27	private ContactPreferenceType contactPreferenceType;	
28		

4. 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 7–41 Create Interface



- **5.** 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.
- 6. Add the properties, getters and setters for *Key* object and *Expiry Date* field.
- 7. Implement the abstract methods of the superclass.

Figure 7–42 Create Class

IContac	tExpiry.java	ContactExpiryKey.java	ContactExpiry.java 23	•			
1 pack	i package com.ofss.fc.demo.domain.party.entity.contact;						
2	2 2 demonst com ofice fe datatumo Data:						
4 impo	3 import com.olss.rc.datAype.uate;						
5 impo	rt com.ofss.	fc.framework.domain.Abstr	actDomainObjectKey;				
6							
7 publ	ic class Con	tactExpiry extends Abstra	ctDomainObject implements IContactExpiry {				
8	/**						
10	* Serial Ve	rsion					
11	*/						
12	2 private static final long serialVersionUID = -4179806027380497671L;						
13							
148	* Contact F	miry Key					
16	*/	WALL NOT					
17	private Conta	actExpiryKey key;					
18							
199	/**						
20	*/	Le	Ν				
22	private Date	expiryDate;	HT.				
23							

8. After creating the domain objects, build the project. We will be using the OBP 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 -> Preferences -> OBP Development* and the set the preferences as follows.

Figure 7–43 Preferences - Service Publisher

8 Preferences				
type filter text 🛛 🗷	Service Publisher			⇔ ⇒ ⇒
 ▶ General ▶ Ant 	Output project location	Client		<u>C</u> hange
Aptana Studio	Service URL	http://localhost:8080		
AspectJ Compiler Data Management	Path of Dynamic Property file:	/home/rshanbha/Work/eclipse_workspaces/	'ngp/config	Browse
FLEXCUBE Development	Path of User Home	/home/rshanbha/Work/eclipse_workspaces/	ngp/config	Browse
GEFU GEFX JUnit				
Procedure Wrapper			b	
Reverse Engineering			ns.	
RMI				
Service Deployer				
Service Publisher				
XML/JSON Facade				
. ► Help				
Install/Update		_		
► Java		R	estore <u>D</u> efaults	Apply
0			Cancel	ОК

Figure 7–44 Preferences - WorkSpacePath

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

Figure 7–45 Preferences - XML/JSON Facade

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	<u>B</u> rowse
Aptana Studio	Sample XML Output Path:	/home/rshanbha/Work/eclipse_workspaces/generator/xml	Browse
AspectJ Compiler	Facade XSD Output Path:	/home/rshanbha/Work/eclipse workspaces/generator/xsd	Browse
Data Management	List of XSD included	(
* FLEXCUBE Development			
GEFU			Ne <u>w</u>
JUnit			Remove
Procedure Wrapper			Up
Reverse Engineering			
RMI			Down
Service Deployer			
Service Publisher			
WorkSpace Path			
XML/JSON Facade		N	
Help		kë	
 Install/Update 			
P Java			
 Java Ec Java Persistence 		Restore Defaults	Apply
- Javar er sistelite			
3		Cancel	ОК

Step 6 Create Application Service

You will 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 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 7–46 ApplicationService Generator

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

5. Click Generate. Application Service classes will be 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.





Step 7 Generate Service and Facade Layer Sources

Before generating the service and facade layer sources, you will 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.

- **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 7–48 ContactExpiryDTO. java file



To generate the service and facade layer sources, follow these steps:

- 1. Open the application service class (ContactExpiryApplicationService)
- **2.** Right-click 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. Deselect the *Overwrite Existing Files* option.

Figure 7–49 Generate Service and Facade Layer Sources

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

- 4. Click *Finish*. Service and facade layer sources will be 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.

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*.

Open ContactExpiryApplicationServiceSpi and modify the code as shown below.



Figure 7–50 ContactExpiryApplicationServiceSpi.java file before Modification

Figure 7–51 ContactExpiryApplicationServiceSpi.java file after Modification





Figure 7–52 Contact Expiry Application Service - Contact Point Transaction

The project should contain the Java packages as shown below:

Figure 7–53 Java Packages



Step 8 Export Project as a JAR

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

- **1.** Right-click 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 and click Finish.

Figure 7–54 Export Project as a Jar

S JAR Export
JAR File Specification
Define which resources should be exported into the JAR.
Select the resources to export: Image: Select the resources to export: Image: Select the resources to export: Image: Select the resources to export generated class files and resources Image: Select the resource files and resources
Export refactorings for checked projects. <u>Select refactorings</u>
Select the export destination:
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
□ A <u>d</u> d directory entries
<u>O</u> verwrite existing files without warning
? < Back Next > Cancel Finish

Step 9 Create Hibernate Mapping

You will 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.





Step 10 Configure Host Application Project

You will 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.

Figure 7–56 Adding an Entry in hostapplicationlayer.properties file



- **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.

Figure 7–57 Adding an entry in ProxyFacadeConfig.properties file



- 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 7–58 Adding an entry in JSONServiceMap.properties file



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.
- **2.** 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.

Client Application Changes

After creating database table to hold the input data and after creating the related domain objects and service and facade layers, we 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 1 Create Model Project

You will 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 7–59 Create Model Project - ADF Model

This list is filtered according to the	current project's <u>selected technologies</u> .	
<u>C</u> ategories:		Show All Descriptions
⊟ General	Generic Project	^
Applications Ant Connections 	ADF Model Project Creates a project that defines a da application using ADF Business Cor	ta model for an ADF web mponents.
Deployment Profiles	D ADF Swing Project	
Diagrams	DF ViewController Project	
Projects	EJB Project	
-Web Tier	🛅 Java Application Project	
Applet	🔁 Java Project	
JSF	Project from Existing Source	
JSP Servlets	Project from WAR File	
All Items	Reproject Template	

- **2.** Give the project a title (*com.ofss.fc.demo.ui.model.party*) and set the default package as the same.
- **3.** Click *Finish* to create the project.

Figure 7–60 Create Model Project - Click Finish

8 Create ADF Mod	lel Project - Step 2 of 2
Configure Java set	tings
Project Name	Your new project starts with a default package, a source root directory, and an output directory. Default Package: [com.ofss.fc.demo.ui.model.party]ava Source Path: [/rshanbha/Work/jdev_workspace/View/com.ofss.fc.demo.ui.model.part] Browse Qutput Directory: anbha/Work/jdev_workspace/View/com.ofss.fc.demo.ui.model.party/cle Browse
Help	< <u>Back</u> Next > <u>Finish</u> Cancel

Step 2 Create Application Module

You will 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 the model project and select *New*.
- 2. Choose *Application Module* from the dialog box that opens.

🎒 Search		
Categories:	Items:	Show All Descript
General	Business Components from Tal	bles
Applications Connections Deployment Descriptors Deployment Profiles Diagrams Java Projects Business Tier ADF Business Components	Application Module Launches the Create Application create an application module. U and organize view objects, to he business service methods. To enable this option, you must Navigator. Before you can finish module, you will be prompted to connection.	n Module wizard, which allows you to Jse application modules to assemble andle transactions, and to provide t select a project in the Application h creating the new application o select (or create) a database
Security	🔁 Association	
All Items	📋 Business Components Deploym	nent Profiles
	Business Components Diagram	1
	🍘 Default Data Model Components	s
	-	

Figure 7–61 Create Application Module - ADF Business Components

- **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 7–62 Create Application Module - Set Package and Provide Name

		and the second se	-
Name	Application mod of an applicatio	Jules are for assembling, packaging, and deploying the view objects and busines n.	s service
Data Model	Package:	com. ofss. fc. demo. ui. model. party	Brows
Application Modules	Na <u>m</u> e:	DemoParty/AppModule]
Java	Display Name:	Demo Party App Module] 🔍
Summary	E <u>x</u> tends:		Brows
	Property Set:	<none> -</none>]

- 5. Click *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 *Finish* to create the application module.

Figure 7–63 Create Application Module - Summary

😣 Create Applicat	ion Module - Finish
Summary	0x0101010101010101010101010101010101010
ų Name	You have finished the Create Application Module Wizard.
🗘 Data Model	You have selected the following options:
4 Application Modules	Application Module
🖕 lava	🕀 🦢 Name
Summary	Name: com.ofss.fc.demo.ui.model.party.DemoPartyAppModule
	Data Model
	None
	- 🕒 View Link Instances:
	- 🐼 None
	E Application Modules
	- S None
	Java
	Generate Appriodole class, Palse
< >>>	When you click Finish, the application module will be created.
Help	< <u>Back</u> Next > Finish Cancel

Step 3 Create View Object

You will need to create a view object for the newly added *Expiry Date* field. This view object will be 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 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 Next.

Figure 7–64 Create View Object - Provide Name

			Concentration of the Annual State	
R	Name	View objects ar given applicatio	e for joining, filtering, projecting, and sorting your business data for the spea In task.	cific needs of
ψ	Entity Objects	Packa <u>q</u> e:	com. ofss.fc. demo. ui. model. party. contactexpiry	Browse
Ŷ.	Attributes	Na <u>m</u> e:	ContactExpirtVO	
Ŷ.	Attribute Settings	Display Name:	Contact Expirt Vo	Q
Ŷ.	Query	Extends:		Browse
ļ	Bind Variables Java	Property Set:	<none></none>	•
ł	Application Module	Select the data	source type you want to use as the basis for this view object.	
Ċ	Summary	O Updatable a	ccess through entity objects	
		O Read-only ac	cess through SQL query	
		Rows popula	ited programmatically, not based on a query	
		O Rows popula	rred at design time (Static List)	

- **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 7–65 Create View Object - View Attribute

	Attribute				Updatable-
Name	<u>N</u> ame:	ExpiryDate			
Attributer	Type:	Date		rowse	
Attributes	Property Set:	<none></none>	•		
Attribute Settings	Value Type:				
Java	<u>V</u> alue:			<u>E</u> dit	Always
Application Module					O While New
Summary	Selected	to <u>C</u> olumn or SQL d in <u>Q</u> uery	Queryable		○ Ne <u>v</u> er
	Dis <u>c</u> rimi	nator: O View O Entity	Effective Date		
	Default V	/alue:	○ Start ○ End		
	Query Column				
	Alia <u>s</u> :	VIEW_ATTR	Type: DATE		
	Alia <u>s</u> : Expression:	VIEW_ATTR	Type: DATE		
	Alias:		Type: DATE		Canad

9. Click *Next*.

10. On the *Application Module* dialog, browse for the previously created *DemoPartyAppModule*.

Figure 7–66 Create View Object - Application Module

8 0	S Create View Object - Step 5 of 6			
Appli	ication Module		6867636363636363636363636363636363636363	
	ame ttributes ttribute Settings	Select the applicatio	e checkbox to add an instance of this view object to an application module. If the specifie n module does not exist, it will be created. t tion <u>M</u>odule	d
	va	Packa <u>g</u> e: Name:	com. ofss.fc. demo. ui. model. party	Browse
<u> </u>	ummary			Dienser
<	<i>• ></i>			
He	elp		< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cancel

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

Figure 7–67 Create View Object - Click Finish

😣 Create View Ob	oject - Finish
Summary	
Attributes Attribute Settings Java Application Module Summary	You have finished the Create View Object Wizard. You have selected the following options: Yiew Object Name You have selected the following options: Name You have selected the following options: You have selected the following option: You have selected the following opting opting You have s
Help	< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel

Step 4 Create View Controller Project

You will 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*.

Figure 7–68 Create View Controller Project - ADF View Controller Project

This list is filtered according to the	current project's <u>selected technologies</u> .	
🎁 Search		
<u>C</u> ategories:	Items:	Show All Descriptions
⊟-General	Generic Project	
Applications Ant	DF Model Project	
Connections Deployment Descriptors	DF Swing Project	
Deployment Profiles Diagrams lava	ADF ViewController Project Creates a project that defines view and an ADF web application using ADF Face	d controller components for s and ADF Task Flows.
Projects	EJB Project	
-Web Tier	Java Application Project	
Applet HTML	📴 Java Project	
JSF	Project from Existing Source	
Servlets	Project from WAR File	
All Items	Project Template	

- **2.** Give the project a title (*com.ofss.fc.demo.ui.view.party*) and set the defaults package to the same.
- **3.** Click *Finish* to finish creating the project.

8 Create ADF Vie	vController Project - Step 1 of 2
Name your project	01010101010101010101010
Project Name	Project Name: com.ofss.fc.demo.ui.view.party
Project Java Settings	Directory: /_workspace/View/com.ofss.fc.demo.ui.view.party Browse
	Project Technologies Generated Components Associated Libraries
	<u>A</u> vailable: <u>S</u> elected:
	ADF Business Components 🗠 ADF Faces
	ADF Desktop Integration ADF Page Flow
	ADE Mobile Browser
	ADF Swing
	Ant ISP and Serviets
	Database (Offline)
	EJB
	JavaBeans
	Technology Description:
	ADF Faces adds very high quality components, a dialog framework, as well as personalization and skinning capabilities. ADF Faces features include: file upload support, client-side validation, partial rendering of a
Help	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish Cancel

Figure 7–69 Create View Controller Project - Project Title

- **4.** Right-click the project and go to *Project Properties*. In the *Libraries* and *Classpath* tab, add the following:
 - The Jar containing the screen to be customized (*com.ofss.fc.ui.view.party.jar*)
 - The Jar containing the domain objects and services for Contact Expiry (*com.ofss.fc.demo.party.contactexpiry.jar*) as created in host application project.
 - All the required dependent Jars for the above Jars.
 - The Jar containing the customization class (*com.ofss.fc.demo.ui.OptionCC.jar*)

🍋 Search 📃	Libraries and Classpath	
Project Source Paths	○ Use <u>C</u> ustom Settings	Customize Settings
- ADF Model	Use Project Settings	·
- ADF View		
€- Ant	Java SE Version:	
Business Components	1.6.0_29 (Default)	Cha <u>n</u> ge
Compiler	Classpath Entries:	
- Dependencies	Evont Description	Add Library
- Deployment		A CONTRACTOR
- EJB Module	ADF DVT Faces Databinding MDS Runtime	Add JAR/Directory
- Extension		Remove
■ Javadoc		Territe
 Java EE Application 		Edit
- JSP Tag Libraries	VIEW COMMON LIB	encies jars
 JSP Visual Editor 	TASKFLOW_LIB	<u>Share As</u>
Libraries and Classpath	CORE_LIB	Movello
- Resource Bundle	VIEW_PARTY_LIB party	screens jar
Run/Debug/Profile	ADDRESS_TASKFLOW_LIB	Move Do <u>w</u> n
- Technology Scope	🗹 👖 com.ofss.fc.ui.customtags.jar	
	Com. ofss.fc. ui. components.jar	on close inr
		host domain iar
	Com. orss.rc. demo. party. contactexpiry. jar	- nose domain jar
	ADE Escas Databiadias Bustima	

Figure 7–70 Create View Controller Project - Libraries and Classpath tab

- **5.** In the Dependencies tab, browse for and add the previously created adf model project (*com.ofss.fc.demo.ui.model.party*)
- **6.** In the *ADF View* tab, check the *Enable Seeded Customizations* option to enable this project for customizations.

Figure 7–71 Create View Controller Project - Dependencies Tab

😣 Project Properties - /	'home/rshanbha/Work/jdev_workspace/View/com.ofss.fc.demo.u	i.view.party/com.ofss.fc
(🍓 Search	Dependencies	
Project Source Paths ADF Model ADF View Ant Business Components Compiler Dependencies Deployment EJB Module Extension Javadoc lava EE Application	Use Custom Settings Use Project Settings Dependent Projects and Archives: com.ofss.fc.demo.ui.model.party.jpr	Customize Settings
JSP Tag Libraries JSP Visual Editor Libraries and Classpath Resource Bundle Run/Debug/Profile Technology Scope		
Help		OK Cancel

7. Save the changes by clicking *OK* and rebuild the project.

Step 5 Create Maintenance State Action Interface

Create an interface containing the method definition for a maintenance action. This interface will be 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 7–72 Create an Interface

🔠 IDemoContactPoint.java 🗴						
💏 – Find						
<pre>1 package com.ofss.fc.demo.ui.view.party.contactPoint.backing; 2 public interface IDemoContactPoint { 4 public boolean performStateAction(DemoContactPoint demoContactPoint); 6 }</pre>						

Step 6 Create State Action Class

You will 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 7 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 7–73 Create Update State Action Class



Figure 7–74 Create Update State Action Class - Service Exception



Step 8 Create Backing Bean

You will 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 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 7–75 Create Backing Bean

🕹 DemoCo	ontactPoint.java ×
💏 🕶 Find	
45 ⊟ pu	<pre>blic class DemoContactPoint implements ICoreMaintenance {</pre>
40 47 48	<pre>private static final String WD_CONTACT_EXP = "ContactExpiryVOllterator"; private static final String EXPIRY_DATE = "ExpiryDate";</pre>
50	<pre>protected static final String CONTACT_EXPIRY_PROXY = "ContactExpiryApplicationServiceProxy";</pre>
52 53 54	<pre>private UIXGroup formData; private ContactPoint contactPoint; private ViewObject contactPointV0 = IteratorHandler.getViewObject(Constants.PAGE_DEF, Constants.V0_CONTACT_POINT);</pre>
56 57 58 59	ContactPointBusinessRules contactPointBR = new ContactPointBusinessRules(); private RichPanelLabelAndMessage plan18; private DateComponent expiryDate; private ViewCbject contactExpVD = IteratorHandler.getViewObject(Constants.PAGE_DEF, VD_CONTACT_EXP);
60 61 62	<pre>private transient Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(DemoContactPoint.class.getName());</pre>
63 64 ⊟ 65 66 67 69	<pre>public DemoContactPoint() { super(); contactPoint = (ContactPoint)ELHandler.get("#{ContactPoint}"); }</pre>
69 E 70 71 72	<pre>public void setFormData(UIXGroup formData) { this.formData = formData; }</pre>
73 ₪ 74 75 76	<pre>public 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.

Figure 7–76 Create Backing Bean - Save and Clear Method



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

Figure 7–77 Create Backing Bean - Contact Expiry DTO Method



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

Figure 7–78 Create Backing Bean - OnExpiryDateChange



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

Figure 7–79 Create Backing Bean - Value Change Event Handler

```
public void onContactPreferenceChange(ValueChangeEvent valueChangeEvent)
    if (MaintenanceHelper.getCurrentState().equals(MaintenanceHelper.READ)) {
    clearContactExpiryDetails();
        initializeContactExpVO();
        ContactExpiryDTO contactExpDTO = fetchContactExp();
if(contactExpDTO != null) {
                 setContactExpDetails(contactExpDTO);
            }
        }
    3
    contactPoint.onContactPreferenceChange(valueChangeEvent);
3
public void onContactPointTypeChange(ValueChangeEvent valueChangeEvent) {
    if (MaintenanceHelper.getCurrentState().equals(MaintenanceHelper.READ)
        || MaintenanceHelper.getCurrentState().equals(MaintenanceHelper.CREATE)) {
    clearContactExpiryDetails();
        if (MaintenanceHelper.getCurrentState().equals(MaintenanceHelper.READ)) {
            initializeContactExpVO();
        }
    }
    contactPoint.onContactPointTypeChange(valueChangeEvent);
3
```

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

Figure 7–80 Create Backing Bean - Contact Expiry Proxy Service

```
private ContactExpiryDTO fetchContactExp() {
   ContactPointType cpType = null;
    if (contactPointV0.getCurrentRow().getAttribute(Constants.CONTACT_POINT_TYPE) != null) {
       ContactPreferenceType contactPrefType = null;
   if (contactPointV0.getCurrentRow().getAttribute(Constants.CONTACT_PREF_TYPE) != null) {
    contactPrefType = (ContactPreferenceType)EnumerationHelper.getInstance().fromValue(ContactPreferenceType.class,
                           (String)contactPointV0.getCurrentRow().getAttribute(Constants.CONTACT_PREF_TYPE))
   String partyId = (String)contactPointV0.getCurrentRow().getAttribute(Constants.PARTYID);
   ContactExpiryDTO contactExpDTO = new ContactExpiryDTO();
   ContactPointDTO contactPointDTO = new ContactPointDTO();
   contactPointDTO.setContactPoint(cpType);
   contactPointDTO.setPreferenceType(contactPrefType);
   contactPointDTO.setPartyId(partyId);
   contactExpDTO.setContactPointDTO(contactPointDTO):
   SessionContext context = SessionContextFactory.getSessionContextFactory().getSessionContextInstance();
   context.setServiceCode(Constants.SERVICE_CODE)
   IContactExpiryApplicationServiceProxy contactExpProxy = null;
   ContactExpiryInquiryResponse response = null
   try {
       contactExpProxy = (IContactExpiryApplicationServiceProxy) ProxyFactory.getInstance().getProxy(CONTACT_EXPIRY_PROXY);
       if (logger.isLoggable(Level.FINE))
           logger.log(Level.FINE, "Calling fetchContactExp service");
       3
       response = contactExpProxy.fetchContactExpiry(SessionContextFactory.getSessionContextFactory()
                                                      .getSessionContextInstance(), contactExpDTO);
       if (response != null &&
            (response.getStatus().getErrorCode().equals("0"))) {
           contactExpDTO = response.getContactExpiryDTO();
   % catch (FatalException e) {
```

 Create Managed Bean - You will need to register the *DemoContactPoint* backing bean as a managed bean with a *backing bean* scope. 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:

Figure 7–81 Create Managed Bean - Register Demo Contact Point

👶 DemoContactPoint.ja	va × 🔚 adfc-config.x	cml ×					
							- 22 🤇
General Description	💊 Managed Beans						+ X
Control Flows	Name * 🔺	Class *				Scope *	
Managed Beans	DemoContactPoint	com.ofss.fc.demo.	ui.view.party.contactPoin	t.backing.DemoCo	ntactPoint	backingBean	1
Metadata Resources	Managed Propert	ies: DemoContactP	pint				+ ×
	Name * 🔺		Class		Value		
							10

• **Create Event Consumer Class** - You will 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.

Figure 7–82 Create Event Consumer Class



Step 9 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.

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 7–83 Create Data Control

	는 🗃 consumer 그 🔛 DemoPartyldChangeEventConsumer.jav	_	58 private 59 private	DateComponent e ViewObiect_cont
	Web Content Gamma adfc-config.xml Gamma faces-config.xml	×	Exclude Project Content Delete	Ctrile Altal
	rinidad-config.xml ∰ web.xml ⊕ ि age Flows		<u>M</u> ake Re <u>b</u> uild	Ctrl+Shift-F9 Alt+Shift-F9
≐-⊊ 6] com.ofss.fc.ui.view [10.180.22.95] - 🚰 Application Sources - 🚰 Resources	>	R <u>u</u> n Debug	Ctrl-F11
E	B B com B B com B C css	5	Refa <u>c</u> tor Re <u>f</u> ormat Organ <u>i</u> ze Imports	► Ctrl+Alt-L Ctrl+Alt-0
	Government G	4	Add Versioning Compare With	•
	Page Flows AboutUs.jsff 215810 Default isff 215861	_	Replace <u>W</u> ith Create <u>S</u> ervice Interface	•
	error.html 215061 login.html 227634	-	Cr <u>e</u> ate Web Service Crea <u>t</u> e Data Control	2

2. Restart JDeveloper in the *Customization Developer Role* to edit the customizations. Ensure that the appropriate *Customization Context* is selected.

Step 10 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 7–84 Adding UI to Screens

et co	ntactPoint_jsff.xml ×
(#b-	And 🕹 🌚
1	<pre><nds:customization version="11.1.1.61.92" xmlns:mds="http://xmlns.oracle.com/mds"></nds:customization></pre>
2	<pre><mds:insert after="smcAllowedpurpose" parent="formData" xmlns:af="http://xmlns.oracle.com/adf/faces/rich" xmlns:fc="/com/ofss/fc/ui/components"></mds:insert></pre>
3	<af:panellabelandnessage binding="#{DemoContactPoint.plan18}" id="plan18" label="Expiry Date" xmlns:af="http://xmlns.oracle.com/adf/faces/rich"></af:panellabelandnessage>
4	<fc:date <="" binding="#{DemoContactPoint.expiryDate}" id="expiryDate" label="Expiry Date" td="" xmlns:fc="/com/ofss/fc/ui/components"></fc:date>
5	<pre>value="#{bindings.ExpiryDate.inputValue}" autoSubmit="true" readOnly="true" postValueChange="#{DemoContactPoint.onExpiryDateChange}"/></pre>
6	
7	
8	<mds:modify backingbeanclass"]"="" element="ptl(xmlns(f=http://java.sun.com/jsf/core))/f:attribute[@name="></mds:modify>
9	<pre><mds:attribute name="value" value="com.ofss.fc.demo.ul.view.party.contactPoint.backing.DemoContactPoint"></mds:attribute></pre>
10	
11	<mds:modify element="pfl3"></mds:modify>
12	<mds:attribute name="maxColumns" value="2"></mds:attribute>
13	<pre>emds:attribute name='le(ow10th' value='00s'/></pre>
14	<pre>«mostattribute name= (soe(wigth ' Yalue= 40% /> interpret/int</pre>
15	
10	<pre>studies element= second de troiningpe ></pre>
10	<pre>cinds accration e name = valuechangeListemer = value = wipencontactroint.oncontactroint spechange; /></pre>
10	<pre>cmds.modify_secont="socContactDraft"> cmds.modify_secont="socContactDraft"> cmds.modify_secont="socContactDraft" cmds.modify_secont="socContactDraft"> cmds.modify_secont="socContactDraft" cmds.modify_secont="socContactDraft" </pre>
20	enderstrikete name "value/andelistenen" value_"#CompontantPoint onContantPreference(hanne)" />
21	
22	
23	

Step 11 Add View Object Binding to Page Definition

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

- 1. 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 7–85 Adding View Object Binding to Page Definition

8 Insert Item	
Sele <u>c</u> t the category of components from which you would like to find an item:	
Generic Bindings	•
Select the <u>i</u> tem to be created:	
🐼 action	^
🖶 attributeValues	
La button	
eventBinding	
ing graph	
method Artion	
Description	~
Supports binding to one (or more) attributes.	
	~
<u>H</u> elp 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 7–86 Create Attribute Binding

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

Step 12 Add Method Action Binding to Page Definition

You will 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.
- **2.** For the *Data Collection* option, locate the previously created *DemoPartyIdChangeEventConsumer* data control.

Figure 7–87 Adding Method Action Binding

8 Insert Item	
Select the category of components from which you would like to find an item:	
Generic Bindings	•
Select the item to be created:	
graph	^
🔚 list	
🔚 listOfValues	
📟 methodAction	
🔚 navigationlist	
table	
腺 tree	
🔚 treetable	~
Description:	
Method binding for the control.	^
	~
Help	OK Cancel

Fiaure 7–88	Addina Method	Action Bindina	- Demo Party	/ Change Event	Consumer
i igui e i ee	/ adding mounda				•••••••

	Create Act Select a data colle the selected collection: Data <u>Collection</u> : DemoParty DemoParty DepositBas DirectRolef DirectRolef Documentt Documentt	tion Binding ection and the action you ction. AppModuleDataControl AdChangeEventConsumer SicDetailsEventProducer PartyDetailsConsumer CategoryEventConsumer DetailsEventProducer ExecutionEventProducer	want your control to initiate. The	control initiates the action on the data objects of	
	Select an Iterator:			▼ <u>N</u> ew	
	Operation:	partyldChangeEvent(Obje	ect) 💌		
1		Apply to all iterators in	n page defintion		
	Parameters :				
	Name	Туре	Value	Option	
T	object	java.lang.Obje	ect		
	Help			OK Cancel	1

Step 13 Edit Event Map of Page Definition

You will 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 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 7–89 Edit Event Map of Page Definition - Edit Mapping

producer:	contactPointPageDef.partyDetailsTaskFlowDefn1.partyDetailsPageDef.raiseEvent partyIdOnChangeEvent contactPointPageDef.partyIdChangeEvent1			
vent Name:				
Consumer:				
Consumer Params			+ ×	
Param Name		Param Value		
payLoad		#{payLoad}		
Step 14 Edit Event Map of Page Definition

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 7–90 Edit Event Map of Page Definition - ContactPoint.jsff.xml

-co	ntactPointPageDef.xml.xml ×
đ0-	Find U G
1	<pre><mds:customization version="11.1.1.61.02" xmlns:mds="http://xmlns.oracle.com/mds"></mds:customization></pre>
2	<pre>-mds:insert parent="contactPointPageDef(xmlns(mds nsl-http://xmlns.oracle.com/adfm/uimodel))/mds nsl:executables" position="last"></pre>
3	<pre><iterator <="" binds="ContactExpiryV01" datacontrol="DemoPartyAppModuleDataControl" pre="" rangesize='10"'></iterator></pre>
- 4	id="ContactExpiryVOIIterator" xmlms="http://xmlms.oracle.com/adfm/wimodel"/>
5	
6	<pre><mds:insert parent="contactPointPageDef(xmlns(mds_nsl=http://xmlns.oracle.com/adfm/uimodel))/mds_nsl:bindings" position="last"></mds:insert></pre>
- 7	<attributevalues id="ExpiryDate" iterbinding="ContactExpiryVOIIterator" xmlns="http://xmlns.oracle.com/adfm/uimodel"></attributevalues>
	<attrans></attrans>
	<item volue="txp1ryoate"></item>
10	 ALTERATIVE ALTERATIVE
12	<pre></pre> //dit/iduterations/
13	<pre>valkipset carent_"contactPointPaceDef(valns(ads ns1-http://valns.orarle.com/adfs/uimodel))/ads ns1-bindings" position_"last"></pre>
14	sethodiction id='partyIdChangeFvent1' InstanceReage-"DemoPartyIdChangeFventConsumer_dateProvider' DateControl-"DemoPartyIdChangeFventConsumer'
15	RequiresLpdateModel-"true" Action-"invokeMethod" MethodName="partyIdChangeEvent" IsVievCbiectMethod-"false" xmlns-"http://xmlns.oracle.com/adfm/uimodel">
16	<nameddata ndtype="java.lang.Object" nename="object"></nameddata>
17	
18	
19	<pre><mds:replace node="contactPointPageDef(wmlns(mds_ns)+http://xmlns.oracle.com/addfm/contextualEvent))/mds_ns1:eventMap"></mds:replace></pre>
20	<pre>«mds:insert parent="contactPointPageDef" position="last"></pre>
21	<pre>wewEndp xmins="http://xmins.oracie.com/adtm/contextualEvent"></pre>
22	<pre><rvent name-partyldow.hangetwent=""></rvent></pre>
23	<pre>sprouter region= partypetatistask=towern: partypetatistask=towert > computer region= partypetatistask=towern: partypetatistask=towert > computer region= partypetatistask=towern: partypetatistask=towern: > computer region= partypetatistask=towern: > computer regio</pre>
- 53	contraction instance party contracted a
26	<pre>sparageter name="payload" value="#[payload]"/></pre>
27	
20	
- 29	
- 30	
31	
32	
33	

Step 15 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)*

- **1.** Go to *Project Properties* of the main application project and in the *Libraries* and *Classpath*, add the following:
 - View controller project Jar (com.ofss.fc.demo.ui.view.party.jar)
 - Host domain Jar (com.ofss.fc.demo.party.contactexpiry.jar)
 - Customization Class Jar (com.ofss.fc.demo.ui.OptionCC.jar)
 - 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 will be able to input data and fetch value for the newly added *Expiry Date* field.

PI041			
Contact Point			
🖵 Bead 👍 Create 🥒 Update		🕜 O <u>K</u> 🖉 Clear 🕱 Cancel 🔅) Print
S Party Details			
NO IMAGE AVAILABLE	Party ID 000005295 Home Branch 082991-U 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	
Address Details			
S Contact Point Details			
Contact Point Type Seasonal Start Date Allowed Purposes	Mobile	Contact Preference Type Home Personal Crist Data Exciting	
	Alert	510k09/2012 ag	
Preferred Contact	Preferred Contact		
Marketing Consent	Marketing Consent		
Marketing Consent Start Date		Marketing Consent End Date	
S Telephone Details			
Country Code		Area Code	
Number	32577789	Extension	
Service Provider		VOIP Code	
Timing Preferences			
DND	OND		
DND Start	-	DND End	
Weekdays	Weekdays		
From	Othersteads	TO	
From	- weekends	To	
From		1M	

Figure 7–91 Contact Point screen with Expiry Date field

7.7.4 Removing existing UI components from a screen

In this fourth example of customization, we will be removing some existing UI components present in a screen.

Use Case Description: The *Back Office -> Events -> Alert Maintenance* screen is used to define *Alerts* in the system for different types of events / activities. In this screen, there is a check box field Is *Conditional* for specifying whether there is a *Rule* to be associated with this alert and the *Effective Date* for the rule. If the check box is unchecked, the *Rule* and *Expiry Date* fields are disabled. If the check box is checked, the *Rule* and *Expiry Date* fields are enabled. We will remove these 3 fields in customization.

Figure 7–92 Remove UI Components from Alert Maintenance screen

ALQ4						
Alert Maintenance						
🛄 Bead 🏟 Create 🖉 Update	M Delete				~	Ok 🥔 Clear 📓 Cancel 🗎 Print
Activity Search	🗄 Detail					
View+ 🔯 🖃 Detach		Alert Name			Rule ID	٩
		 Activity Name 			Rule Effective Date	
Activity Name		Event Name			 Expiry Date 	1
LoanRepaymentScheduleApplic		Retries required			Alert Type	-
LoanRepaymentScheduleApplic		Number of Retries			Alexa Natura	-
LoanDisbursementApplication.p		Is Condtional 👿			Alert Neture	
DemandDepositAccountApplicat	Recipient Message T	emplate				
DemandDepositAccountApplicat		No				
DemandDepositAccountService	View- 🗶 📭 🛛	Detach	ha h			
DemandDepositAccountApplicat	Subscriber Type Relation	on Type Group Id	Message Destination	Template Rule	Rule ID Rule Effective	
DemandDepositAccountApplical	No data to dicelay		remplate Type		Date	
DemandDepositAccountApplicat	No only to only ay.					
LoanAccountSplitInstructionApp						
LoanCustomerInitiatedRollover	Columns Hidden 2					
LoanCustomerInibatedRollover	1					
LoanPartialPayOftApplicationse	Message Template					
DemandDenositAccountApplicat	Update	Read	View - 🔯 🛃 0	etach		
LoanDisburgementheolication of	Template		Data	Attribute		
BatchConditionService process	Name		Data Attribute Mask	Attribute	ID	
	 Message 		No data to display.			
	Data					
Event Search						
View 🖝 🔯 🛃 Detach						
Event Name Event ID			4			
No data to display.	BRule					Create Rule
S Hide Modification History						
Created By of	suser		On 07-Sep-2012 12:00:0	0 AM	Approved	
Approved By			On		Active 🗌	

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

Step 1 Create View Controller Project

You will need to create a view controller project to hold the customizations that need to be done on the screen. To create a view controller project, follow these steps:

- **1.** In the client application, create a new project of the type *ADF View Controller Project*.
- **2.** Give the project a title (*com.ofss.fc.demo.ui.view.ep*) and set the default package to the same.
- 3. Click *Finish* to finish creating the project.

Figure 7–93 Create ADF View Controller Project - Project Technologies

8 Create ADF View	wController Project - Step 1 of 2
Name your project	01010101010101010101010
Reproject Name	Project Name: com.ofss.fc.demo.ui.view.ep Directory: dev_workspace/View/com.ofss.fc.demo.ui.view.ep Browse
Project Java Settings	Project Technologies Generated Components Associated Libraries
	Available: Selected: ADF Business Components ADF Desktop Integration ADF Desktop Integration ADF Page Flow ADF Mobile Browser ADF Swing Ant Java Database (Offline) EJB JavaBeans XML
< <u> </u>	Technology Description: ADF Faces adds very high quality components, a dialog framework, as well as personalization and skinning capabilities. ADF Faces features include: file upload support, client-side validation, partial rendering of a < Back Next > Finish Cancel

- **4.** Right-click the project and go to *Project Properties*. In the *Libraries* and *Classpath* tab, add the following:
 - The Jar containing the screen to be customized (com.ofss.fc.ui.view.ep.jar)
 - All the required dependent Jars for the above Jar.
 - The Jar containing the customization class (com.ofss.fc.demo.ui.OptionCC.jar)
- **5.** In the *ADF View* tab, check the *Enable Seeded Customizations* option to enable seeded customizations for this project.



Figure 7–94 Create View Controller Project - Libraries and Class Path

6. Restart JDeveloper in the *Customization Developer Role* to edit the customizations. Ensure that the appropriate *Customization Context* is selected.

Step 2 Remove UI Components from Screen

Browse and locate the JSFF for the screen to be customized (*com.ofss.fc.ui.view.ep.activityEventAction.form.ActivityEventActionMaintenance.jsff*). Open the JSFF and do the required changes as follows:

- 1. Select the *Is Conditional* check box component. In the *Property Inspector* panel, set the *Rendered* property to *false*.
- **2.** Select the *Rule Id* custom component. In the *Property Inspector* panel, set the *Rendered* property to false.
- **3.** Select the *Rule Effective Date* component. In the *Property Inspector* panel, set the *Rendered* property to false.
- **4.** Set the *Rendered* property to *false* is better than completely deleting the component to avoid binding errors.
- **5.** 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

ActivityEventActionMaintenance.jsff.xml should look similar to as shown below.

Figure 7–95 Modifications in the ActivityEventActionMaintenance.jsff.xml



Step 3 Deploy Customization Project

After finishing customization changes, exit *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.ep.jar*).

- **1.** Add this Jar and customization class jar to the classpath of the main application project.
- **2.** Start the application and navigate to *Back Office -> Events -> Alert Maintenance* screen. You will notice that the fields *Is Conditional, Rule Id* and *Rule Effective Date* are not present on the screen.

Figure 7–96 Modified Alert Maintenance Screen

ALQ4						
Alert Maintenance						
🛄 Bead 🏟 Create 🥒 Lodate	3 Delete				🛩 e	🖄 🥜 Clear 😹 Cancel 📄 Print
Activity Search	E Detail					
View - 🔯 🛃 Detach	• /	Jert Name			Expiry Date	6
	• Act	vity Name			 Alert Type 	2
Activity Name	• 61	ent Name			Alert Nature	_
LoanRepaymentScheduleApplic	Retrie	s required 🗌				
LoanRepaymentScheduleApplic	Number	of Retries				
LoanDisbursementApplication.p	Instananc					
DemandDepositAccountApplicat	View- Y R Detach					
DemandDepositAccountApplical	Lodate Ceter Concer Concer					
DemandDepositAccountService	Subscriber Type Relation Type	Group Id Temp	plate Type	Template Rule Rule II	Date	
DemandDepositAccountApplicat	No data to display.					
DemandDepositAccountApplicat						
LoanAccountSolitInstructionApp						
LoanCustomerInitiatedRollover	Columns Hidden 2					
LoanCustomerInitiatedRollover	S Message Template		Create Rule Create Rule Create Rule Create Rule Create Rule Create Rule			
Demandepositiccumagena Demandepositiccumagena LoankaccumspitalmitrutionApp LoankoutomertholaedRollover LoankoutomertholaedRollover LoankoutomertholaedRollover						
DemandDepositAccountApplicat	Template Name		Data Attribute Data	Attribute Attribute ID		
LoanDisbursementApplication.p	tenance > Create > Create > Create > Priot Search > Detail - Addrify Name - Strairy Date > Strairy Date > Strairy Date > Addrify Name -					
BatchConditionService.process	Data					
Event Search						
View - 🛃 🛃 Detach			A		1	
Event Name Event ID	🗄 Rule					💠 Create Rule
No data to display.						
((
Hide Modification History						
Created By of	suser		On 07-Sep-2012 12:00:0	0 AM	Approved	
Approved By			On		Active 📋	

SOA Customizations

OBP provides the functionality for customizing the SOA composite applications. The steps to customize a SOA composite application are similar to those of customizing an ADF View Controller application with a few differences. The similarities and differences would be apparent in the examples demonstrated in the following sections.

The following section provides details about the SOA Components Customization. The detailed documentation for customizing and extending the SOA Components is also available at the Oracle website:

http://docs.oracle.com/cd/E25178_01/fusionapps.1111/e16691/ext_soaedit.htm

8.1 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.

We need to create a customization layer with name *option* and values *demo* and *another bank name*.

To create this customization layer, follow these steps:

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

Figure 8–1 Add an entry for new Customization Layer



4. Save and close the file.

8.2 Customization Class

Before customizing an application, a *customization* class needs to be created which is the interface that the *Oracle Meta-data Services* framework uses to define which customization layer should be applied to the application's base meta-data.

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.
- **3.** Go to **Project Properties** for this project and add the required **MDS** libraries in the classpath of the project.
- **4.** Create the customization class in this project. The customization class **must** extend the *oracle.mds.cust.CustomizationClass* abstract class.

Implement the following abstract methods of the CustomizationClass as follows:

- 1. 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.
- 2. getName() This method will return the name of the customization layer.
- **3. getValue()** This method will return the customization layer value at runtime.

The screenshot below depicts a sample implementation of the above methods.





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.

8.3 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 View* section, check the *Enable Seeded Customizations* option.
- **3.** In the *Libraries* and *Classpath* section, add the previously deployed which contains the customization class.

Figure 8–3 Enabling Application for Seeded Customization

Search	Libraries and Classpath	
Project Source Paths ADF Model	Use <u>⊆</u> ustom Settings	Customize Settings
ADIF View	Java SE Version:	
Business Components	[1.6.0_29 (Default)	Change
Compiler	Cjasspath Entries:	
Deployment	Export Description	Add Library
EJB Module	SOA Designtime	Add JAR/Directory.
Extension	BPEL Runtime	Bernove
ava EE Application	Modiator Runtime	E de
SP Tag Libraries	DEMO_OPTION_CC	Eur.
SP visual Editor Ubraries and Classpath		Suere wa
Resource Bundle		Move Up
Run/Debug/Profile		Move Do <u>w</u> n

- **4.** In the **Application Resources** tab, open the *adf-config.xml* present in the *Descriptors/ADF META-INF* folder.
- **5.** In the list of *Customization Classes*, remove all the entries and add the *com.ofss.fc.demo.ui.OptionCC.OptionCC* class to this list. The sections below will elaborate in detail the actual customization of a SOA process with examples.

8.4 SOA Customization Example Use Cases

This section describes the examples use cases of SOA customization.

8.4.1 Add a Partner Link to an Existing Process

In this example of SOA customization, we will be adding a Partner Link call to an Echo Service to an existing SOA process. The Echo Service will take a string input and respond with the same string as output.

The following section will explain how to create a SOA project and process with the example of Echo Service.

Step 1 Create SOA Project

You will need to create a SOA project to contain the Echo Service process. To create the SOA project, follow these steps:

- 1. In the Main Menu, go to File -> New.
- 2. In the Project Gallery that opens, select SOA Project and click OK.

Figure 8–4 Select SOA Project

Show All Description
which you create a service-oriented architecture
inents like BPEL, Business Rules, Human Tasks,
pplication, a project, or a file within a project in

- **3.** In the **Create SOA Project** wizard, enter appropriate project name (EchoService) and location for the project.
- 4. Click Next.

Figure 8–5 Enter SOA Project Name

Project Name	Project Name: EchoService
Project SOA Settings	Dirgctory: /home/rshanbha/Work/jdev_workspace/Dec10/application/OriginationAndFulfillment/process/EchoService Broms
	Project Technologies Generated Components Associated Libraries
	Available: Selected:
	ADF Business Components SOA
	ADF Desktop Integration
	ADF Faces
	ADF Library Web Application Support
	ADP Mobile Browser
	ADF Page How
	Ant
	BPM
	Database (Offline)
	EJB
	HTML
	java
	JavaBeans 🗸
	Technology Description:

- **5.** In the next dialogue of the wizard, enter appropriate name (EchoService) for the SOA composite.
- **6.** Select *Empty Composite* from the drop-down menu.
- 7. Click Finish.

Figure 8–6 Configure SOA Settings

😣 Create SOA Pro	ject - Step 2 of 2	
Configure SOA settings		
Draiact Name	Composite Name:	
Project Name	EchoService	
Project SOA Settin	Name Composite Name: EchoService 2 SOA Settir Composite Template: Empty Composite Empty Composite Composite With BPEL Process Composite With Business Rule Composite With Hediator Composite With Hediator Composite With Spring Context Composite From Oracle BPA Blueprint	
	Empty Composite	
	Composite With BPEL Process	
	Composite With Business Rule	
	Composite With Mediator	
	Composite With Human Task	
	Composite with Spring Context	
	Composite From Gracie BPA Bideprint	
< · · · · · · · · · · · · · · · · · · ·	<u>C</u> ustomizable	
Help		< Back Next > Finish Cancel

Step 2 Add Mediator Component

You will need to add a *Mediator* component to the BPEL process to process the input to the SOA process and generate an output.

To add the *Mediator*, follow these steps:

- **1.** From the **Project Navigator** tab, select and open *EchoService.bpel* in the *Design* mode.
- **2.** From the **Component Palette** tab, in *SOA Components* section, select the *Mediator* component.
- **3.** Drag and drop it onto the *bpel* process.
- **4.** In the **Create Mediator** dialogue that opens, enter appropriate name (EchoService).
- 5. From the Templates drop-down, select *Synchronous* Interface.
- 6. Check the Create Composite Service with SOAP Bindings option.
- 7. Click OK.

Figure 8–7 Create Mediator

😣 Crea	ate Me	liator	
Mediator Create	Compo a mediat	nent or component to perform routing, filtering, and transformations.	4
<u>N</u> ame:	EchoSe	vice]
Template:	🔀 Syn	chronous Interface	• @
	Crea	e Composite Service with SOAP Bindings	
	Input	{http://xmlns.oracle.com/singleString}singleString	٩
	Output:	{http://xmlns.oracle.com/singleString}singleString	Q
Help			OK Cancel

- 8. An *EchoService.mplan* file will be created. Open this file in **Design** mode.
- **9.** In the *Routing Rules* section, click the icon for *Add*.
- **10.** Select *Static Routing Rule* from pop-up menu.
- **11.** In the **Target Type** dialogue that opens, click **Echo**.

Figure 8–8 Select Target Type

Bcomposite.xm/ × EchoService.mplan ×	V
~8	
🔩 Mediator	
Name: EchoService WSDL URL: EchoService.wsdl	8 Target Type
Port Type: execute_ptt Resequence Level: operations	event, or echo a response back to the initial caller?
😑 緯 Routing Rules	Service Event Echo Cancel
Operations	€ 9 €9
🛱 execute	Priority 4 😴 🗌 Validate Syntax (XSD) 🛆 🤝 🍁 💥
Callout To < <java callout="" class="">></java>	8
Design Source History	

A Static Routing section will be added to the screen.

12. Click the icon next to the *Transform Using* drop-down.

- **13.** In the **Request Transformation Map** dialogue that opens, select the option **Create New Mapper File**.
- **14.** Click **OK**.

Figure 8–9 Request Transformation Map to create new mapper file

lame:	EchoService			
SDL URL:	EchoService.wsdl 🚷			
ort Type:	execute_ptt			
esequence l	😣 Request Transforma	ion Map		
	Transformation from request i	essage requestMessage to message replyMessage.		
🚑 Routin	Transformation to part:	epły		
Operation	Use Existing Mapper File:	9	1	₽ , ₽
	Create New Mapper File:	ingleString_To_singleString.xsl		
😂 execute				v 🕂 🗙
Callout	Help	OK Can	cel	
Static Ro	uting			
< <fi< td=""><td>lter Expression>></td><td>😵 🕉 ݿ Initial Caller*::execute:output</td><td>9</td><td>Sequential 🔻</td></fi<>	lter Expression>>	😵 🕉 ݿ Initial Caller*::execute:output	9	Sequential 🔻
	Validate	Semantic	- 8	
	Transfe	rm Using reply : < <transformation map="">></transformation>	- 80	
		a Valuar		

15. This will create a *singleString_to_singleString.xsl* file. Open this file in **Design** mode.

You will see the input parameters in tree format on the left hand side and the output parameters on the right hand side of the screen.

- 16. In our case, the input and output contain a single *string*.
- **17.** Select the input string from the left hand side and drag and drop it to the output string on the right hand side. This will create a mapping between input and output parameters.

Figure 8–10 Mapping Input and Output string



18. Save all files and build the project.

Step 3 Deploy Project to SOA Server

You will need to deploy this project to a SOA Server. From the *Admin* team, get details of the SOA Server and configure it in your JDeveloper.

After adding the SOA Server to your JDeveloper, follow these steps to deploy the *EchoService* composite to the server:

- 1. In the **Project Navigator** tab, right click the project and select *Deploy*.
- **2.** In the **Deploy EchoService** dialogue that opens, select *Deploy* to *Application Server* from the list.
- 3. Click Next.

Figure 8–11 Select Deployment Action

8 Deploy EchoSer	vice
Deployment Action	
R Deployment Action	Select a deployment action from the list below.
Summary	Deploy to Application Server Deploy to SAR
< <u>H</u> elp	Deploy this archive to SOA configured Application server(s) < Back

- **4.** In the Deploy Configuration dialogue, check the option *Overwrite any existing composites with the same revision ID*.
- 5. Click Next.

Figure 8–12 Deploy Configuration Settings

epioy configuratio	n		
Deployment Action	Composite Revis	sion ID	
Select Server	Project:	EchoService	
Summary	Current Revision ID:	1.0	
	New Revision ID:	1.0	
	🗄 SOA Configurati	ion Plan	
-	Mark composite revi Overwrite any existi	ision as default. ng composites with the same revision ID.	
- 6 6	Mark composite revi Overwrite any existin Keep running inst Use the following SC	ision as default. ng composites with the same revision ID. tances on after redeployment. DA configuration plan for all composites:	

Figure 8–13 Select Deployment Server

😣 Deploy EchoSer	vice	
Select Server		
Deployment Action Deploy Configuration Select Server SoA Servers Summary	Application Servers: IntegratedWebLogicServer 50a_server_mumbai_155 SOA_server_mumbai	* 2 ³
< <u> </u>	 Overwrite modules of the same name 	< Back Next > Finish Cancel

- **6.** Select the appropriate *Partition* of the SOA Server where the composite should be deployed.
- 7. Click Finish.

Figure 8–14 Select Target SOA Server

				0101010	
Deployment Action	noose	SOA Server:	Partition:	Status:	Server URL:
Deploy Configuration	~	soa_serverl	default	RUNNING	http://ofsmug-vm-145.
SOA Servers					
Summary					

Step 4 Test Echo Service

After deploying the *EchoService* composite to a SOA Server, you can test it through the EM console:

- 1. Log in to *em* console of the SOA Server to which the composite is deployed.
- 2. From the *SOA Domain* select the *EchoService* composite.

Figure 8–15 Select SOA Domain

m - 22 504 Infrastructure - 8 Topology							
m + 33 sow musepactne + 10 robowy	A EshaCandes II 01a				I const in at a	weblaatel blact Of	ICM ICAM
	V EchoService [1.0]				Copyee in as a	2. 2012 6:06:15 0	
com.ofss.fc.workflow.process.DeclineOffer [1.0]	SUA Composite +				age Refreshed Dec 1	3, 2012 6:06:15 PF	4 GME+05:3
com.ofss.fc.workflow.process.EditSubmission [1.0]			110				
com.ofss.fc.workflow.process.HumanTaskEventHandler	Running Instances 0 Total 0	Active Retire Sh	ut Down Tes	t Setting	s • 🚯 🗃	(PF	telated Lini
com.ofss.fc.workflow.process.InquirePartyDetails [1.0]	Dashboard Instances Faults	and Rejected Messages	Unit Tests Polic	ies			
Com.ofss.fc.workflow.process.InquiryPartyDetails [1.0]							
com.ofss.fc.workflow.process.ManageDepositOrigination	v						
com.ofss.fc.workflow.process.ManageLendingOrigination	Recent Instances						
om.ofss.fc.workflow.process.ManageOriginationFault [1	Show Only Running Instances	Runn	ing 0		Total Q		
com.ofss.fc.workflow.process.ManageUserPreferences [Instance ID Name	Conversation ID State				1	Start Time
om.orss.nc.worknow.process.NotryCustomerHub [1.0]	No composite instances found.						
com.orss.rc.worknow.process.OpenDemandDepositAcco							
com.orss.rc.worknow.process.OpenLendingAccount [1.0							
com.orss.rc.worknow.process.OpenTermDepostAccount							
com orsa.rc.worknow.process.OriginateDebitCard [1.0]							
com oftes fr workflow process Originate/ormoppicant [1	3 Show All						
com.orss.rc.worknow.process.originate/pending [1.0]	ERecent Faults and Rejected	Messanes					
im ofte fr workflow process Derform CollaberalBerfertin							
m ofte fr workflow process BerformCollateraDialuatio	Show only system faults 🕑						
om ofte fr workflow process PerformOrioinationEndful	Error Message	Recove	try	Fault Time	Fault Location	Composite Tostance ID	Logs
m ofte fr workflow process PerformOriginationAndFul	No faults found					Instance ID	
m.ofss.fc.workflow.process.PerformOriginationAndFul							
om.ofss.fc.workflow.process.PerformSettlement [1.0]							
om.ofss.fc.workflow.process.PerformSubmissionFraud							
com.ofss.fc.workflow.process.PerformTitleSearch [1.0]							
com.ofss.fc.workflow.process.ProcessLendingApplication							
com.ofss.fc.workflow.process.ProvisionIdentity [1.0]							
com.ofss.fc.workflow.process.ReferDownsell (1.0)	a filme II						
com.ofss.fc.workflow.process.StructureDepositSolution	an annam wa						
m.ofss.fc.workflow.process.StructureLendingSolution	Component Metrics						
n.ofss.fc.workflow.process.VerifyApplication [1.0]						Faulted Instance	5
m.ofss.fc.workflow.process.VerifyIdentity [1.0]	Name	Component Type	Total Instance	s Running In	stances Rec	overable Non Re	coverable
om.ofss.fc.workflow.process.WithdrawSubmission [1.0	SchoService	Mediator		0	0	0	0
choService [1.0]							
EchoServiceComposite [1.0]							
HelloWorld [1.0]							
IPMBulkUploadProcess [1.0]							
Project1 [1.0]	Services and References						
SampleSOA [1.0]	Name	Type	Useen	Faulte	Total Messager	Average Proces	ssing Time
SampleSOA [2.0]		type	Usage	reura	roun riessages		(sec)
Test [1.0]	EchoService_ep	Web Ser	vice Servic	e 0	0		0.000
ogic Domain							
idata Repositories							
rtg Sampreson (2.0) 4€ test (1.0) WebLogic Domain Metadata Repositories User Messaging Service □	S EchoService_ep	Web Ser	vice Servic	e O	0		

- **3.** On the right hand side panel, you can see the *Dashboard* which lists the instances of SOA requests to that composite and many other options.
- 4. Click the **Test** button to test the composite.

Figure 8–16 Test Web Service

SOA Comp	vice [1.0] () osite +		Logge Page Refres	d in as weblogic Host OFSMUG-VI hed Dec 13, 2012 6:08:55 PM GMT+05
est Web Jse this pag WSDL. When hat you wan	Service to test any WSD the page refresh nt to test. Specify	DL, including WSDLs that an res with the WSDL details, f any input parameters, and	e not in the farm. To test a Web service, ent first select the Service,then select the Port, a I click Test Web Service.	Test Web Service er the WSDL and click Parse and then select the Operation
WSDL	http://OFSMUG- Parse WSDL HTTP Basic Auth	VM-145:8001/soa-infra/ser	vices/default/EchoService/EchoService_ep?	WSDL
Service Port Operation	EchoService_ep execute_pt execute ▼			
Request	L http://ofsmug-	vm-145:8001/soa-infra/set	rvices/default/EchoService/EchoService_ep	Edit Endpoint URL
• Securit	y			
 Quality	of Service			
HTTP T	ransport Option	ıs		
• Additio	nal Test Option	s		
	Arguments			
Tree View	w •			
and the second division of the second divisio		Tune	Value	
Name		Type	¥ dide	

- 5. In the *Input Arguments* section, enter input and click *Test Web Service*.
- 6. You will be able to see the response in the *Response* section.

Step 5 Add Customizable Scope to SOA Application

By default, a BPEL process in itself is not customizable. In addition to the steps followed to enable customizations in a SOA application, you will need to add a *Scope* component to the BPEL process and enable it for customizations.

To demonstrate customizations of a SOA process, we will be using the BPEL process *NotifyCustomerHubProcess* present in the composite *com.ofss.fc.workflow.process.NotifyCustomerHub*.

To see the flow of the *NotifyCustomerHubProcess* before customizations:

- 1. Deploy the composite to a SOA Server.
- 2. Log in to the *em* console and select the process from *SOA Domain*.
- **3.** From the *Dashboard*, click **Test**.
- 4. Enter appropriate input and click *Test Web Service*.
- 5. From the *Dashboard*, click an *Instance* of the composite request.
- 6. Select the Flow tab to see the flow of the process.

Flow Trace > Instance of NotifyCustomerHubProcess	Data Refreshed Dec 13, 2012 6:17:34	PM GMT+05:30 (2
AInstance of NotifyCustomerHubProcess		
This page shows BPEL process instance details.	Instance ID bpel:450 Started Dec 13, 2	012 6:16:55 PM
Audit Trail Flow Sersor Values Faults		
Click an article to view the details.	Current Audit Level: production	View Raw XML
	content man server, production w	The same set
receiveInput		
betine		
50 B		
publishPartyD		
publishPartyD		
0		

Figure 8–17 Customization of SOA Application - Flow

7. Open the SOA application which contains the base composite which will be customizing. The aforementioned process is present in the *OriginationAndFulfillment* application inside the *com.ofss.fc.workflow.NotifyCustomerHub project*.

To add a customizable scope to the BPEL process, follow these steps:

- 1. Open the *NotifyCustomerHubProcess.bpel* file in **Design** mode.
- **2.** From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Scope* component and drop it on to the BPEL process as shown in the figure.
- **3.** Double-click the component and enter appropriate name (EchoServiceScope) for the component.
- **4.** Drag and drop the existing *Assign* component labeled *setTitle* on to the newly added *EchoServiceScope* component.

NotifyCustomerHubProcess.bpel × of composite.xml ×			
🖉 🥥 (x) 🖏 ஞ 🚥 🗒 🕕 🗵 • 🖗 • 😓 • 🎯 🗆 🚕 🖓		(M •	📄 🛃 BPLL 🔛 Monitor 🕞 Test 🛛
Partner Links			Partner Links 2
	B 2 Settlike		
	publishPartyOetails		
	publishPartyOetails		MDH9artsApplic
setTitle - /process/sequence/scope/assign			Zoom: 10010
esian Source History			

Figure 8–18 Customization of SOA Application - Notify Customer

- 5. Right click the *Scope* component and select *Customizable* from the context menu.
- 6. Save all the changes and restart JDeveloper in *Customization Developer Role*.

Step 6 Customize the SOA Composite

After adding a *Customizable Scope* to the base composite, you can start performing customizations in JDeveloper's *Customization Developer Role*.

When you open the *NotifyCustomerHubProcess.bpel* file in *Design* mode, you will notice that all other components in the process, except the customizable *EchoServiceScope* component, are disabled. This means that your customizations are limited to that scope.

In the following sections, we will be adding a *Partner Link* call to the previously created *EchoService* BPEL process and other required components in the *customization* mode.

Step 7 Add Partner Link Component

To add a *Partner Link* to the BPEL process, follow these steps:

- 1. From the Project Navigator, open the NotifyCustomerHubProcess.bpel file in Design mode.
- **2.** From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Partner Link* component and drop it on to the *Partner Links* section of the BPEL process.
- **3.** In the *Create Partner Link* dialogue that opens, enter appropriate name (EchoService) for the partner link.
- **4.** In the *WSDL Settings* section of the dialogue, enter the URL for the previously created *EchoService* composite.

- **5.** You will get alert notifying that there are no Partner Links defined in the current WSDL with an option to create a wrapper WSDL file with partner links defined for specified WSDL.
- 6. Click Yes.

A new *EchoServiceWrapper.wsdl* file will be created which contains the partner links.

- **7.** Select the newly defined partner link type and partner role in the *Partner Link Type* and **Partner Role** drop-down.
- 8. Select *Not Specified* option in the **My Role** drop-down.

Figure 8–19 Add Partner Link Component

Application X CBPM Project Navig X 🕞 X 😱	NotifyCustomerHubProcess.bpel X	Component Palette ×
월 🐻 OriginationAndPulfilment 🔹 🖪 •	🛩 🥥 (x) 🖏 📴 💷 🎇 🕕 I 🖉 • 🖗 • 😓 • 🥘 I 🕉 🍓 • 👘 🚮 10000 🗱 Monitor) — 🖓 Testii) 🕐	BPEL 2.0
Beglication * @ EMP Project Nang. * @ * Projects Composite Andrukhliment Projects Composite revealed and the second seco	A konforcersen keel x A konforcersen keel x Partner Links Partner Link Types r Link Types c Create Partner Link yes B9 C Create Partner Link Foreral Image Property Yes B9 C Create Partner Link Process: Process: Partner Link Partner Link Process: Partner Link Process: Partner Link Partner Link Process: Partner Link Part	BFEL 2.0 BFEL 2.0 BFEL Constructs BFEL Constructs BFEL Constructs BFEL Constructs BFEL Construct BFEL Construct BFEL BFEL
	Pactner Role:	♥ # ₩ Prk ₩ Prek ₩ Repeat Until 1 Scope 8 Sequence ₩ White

Step 8 Add Invoke Component

You will need to add an *Invoke* component to invoke the previously added partner link call to *EchoService*.

To add *Invoke component*, follow these steps:

- 1. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Invoke* component and drop it on the BPEL process inside the *EchoServiceScope* component.
- **2.** Click the *Invoke* component and drag it to the previously added *EchoService* partner link.
- **3.** Double-click the *Invoke* component.
- **4.** In the *Edit Invoke* dialogue that opens, enter an appropriate name (invokeEchoService) for the component.
- **5.** Click the icon for adding a new variable in the *Input Variable* and *Output Variable* sections.
- **6.** Click **OK** to save the changes.

Figure 8–20 Add Invoke Component

	Targets	Sources	Headers	Assertions	Skip Condition	
General		Correlat	ions	Prop	perties	Annotations
Name:	invokeEch	oService				
Conversation ID:						
Detail Label:						
	Invoke	as Detail				
Interaction Ty	ype: 🔯 Pa	rtner Link 🕶] ———			
Partner Link:	EchoServic	e				Q
Port Type:	🐺 execute	_ptt				•
Operation:	execute					-
Input Out	put					
To Parts	Inpu	t Variable				
Input: invoke	eEchoServic	e_execute_	InputVariable	e		4 Q

Step 9 Add Assign Components

An *Assign* component is used to assign values to a variable. These values can be directly assigned from one variable to another or modified using BPEL functions available.

The *EchoService* accepts a single string as an input and gives a single string as an output. The *Input Variable* and *Output Variable* defined in the previously created *invokeEchoService* component will be used to hold the input value for the *EchoService* and the output returned respectively.

In our case, we will need to add two Assign components for following purposes:

- To populate the *Input Variable* of the *invokeEchoService* component with the value returned by the existing *setTitle* component.
- To populate the *setTitle* component with the value returned in the *Output Variable* of the *invokeEchoService* component.
- To add the *Assign* components, follow these steps:
 - 1. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Assign* component and drop it on the BPEL process inside the *EchoServiceScope* component between the *setTitle* and *invokeEchoService* components.
 - **2.** Double-click the *Assign* component.
 - **3.** In the *Edit Assign* dialogue that opens, enter appropriate name (copyToEchoServiceInput) for the component.

- **4.** In the **Copy Rules** tab, select the *compositeInstanceTitle* from the left hand side tree and drag it to the *invokeEchoService_inputVariable* on the right hand side screen as shown in the figure.
- 5. Click OK.

Figure 8–21 Edit Copy Rules Variable

General Copy Rules Annotations Documentation Targets	Sources Skip Condition	
Insert New Rule After 💌		🖪 🚖 🔕 🏣 👊
NotifyCustomerHubProcess.bpel Partner Links Variables Caracter Content of the second of the seco	Drag objects here	NotifyCustomerHubProces Partner Li Variable utVariable client NotifyCustomerHubProcessReguestHeessage & hPartyDetails_inputVariable inst_publishPartyDetails40MParty & is_OutputVariable inst_publishPartyDetails40MParty & compositeInstanceTitle wed string & boService_execute_nputVariable option_nst_requestHessage & ns5:singleString compositeInstanceTitle wed string & hoService_execute_OutputVariable option_nst_replyHessage &
		+ × ± 3
From	То	
KompositeInstanceTitle	📑 \$invokeEi	choService_execute_InputVariable.request

6. Repeat the above steps for another *Assign* component labeled *copyFromEchoServiceOutput*. This component should be present after the *invokeEchoService* component.

The Copy Rules for this component should be as shown in the figure below.

Figure 8–22 Add Assign Components - Reply

Insert New Rule After Image: Imag	nsert New Rule After ▼ Not5/CustomerHubProcess.bpel Partner Links Arrables Arrocess Bro	Drag objects here	
NotifyCustomerHubProcess.bpel Drag objects here Variables Partner Links Variables Partner Links Variables Process NotifyCustomerHubProcess.poly Process Variables Process Variables Process Prom Prom To Prom To Prom	NotifyCustomerHubProcess.bpel Partner Links Variables	Drag objects here	Notify-CustomerHubProcess.bpel & Partner Linka :: Process & - Process & - Variables := Sable client Notify-CustomerHubProcessPrequestHessace (20) ⊕
From To from from	(c) publishPartyOetails_publishPartyOetails_OutputVaria (c) compositeInstanceTitle x=d atring (c) invokeEthoService_execute_noutVariable option_ns (c) invokeEthoService_execute_OutputVariable option_n (c) invokeEthoService_execute_OutputVariable (c) invokeEthoService_execute_OutputVariable (c) invokeEthoService_execute_OutputVariable (c) invokeEthoService_execute_OutputVariable (c) invokeEthoService_execute_OutputVariable (c) invokeEth		Details_inputVariable ns1:publishPartyOetailsMDMParty (2) - SputVariable ns1:publishPartyOetailsMDMPartyResponse (2) - compositienstanceTitle sx8: string (2) - sce_execute_inputVariable cotion_ns1:replyMessage (2) - vice_execute_OutputVariable cotion_ns1:replyMessage (2) -
From To To To To the score of the second of	× >		< , , , , , , , , , , , , , , , , , , ,
From To To To To To To To			4 X 9 3
🔲 🙀 \$invokeEchoService_execute_OutputVariable.reply 🙀 \$compositeInstanceTitle	From	То	
	kinvokeEchoService_execute_OutputVariable.reply	🚮 \$composi	teinstanceTitle

7. Save all the changes.

The **Design** view of the BPEL process should look as shown in the figure below:

Figure 8–23 Design View of the BPEL Process



Step 10 Test the Customized Composite

After performing the customizations, build the project and deploy it to a SOA Server. You will need to include the *Customization Class JAR* in the runtime classpath of the deployed application.

To test the customized composite, follow these steps:

- 1. Log in to the *em* console and select the composite from the *SOA Domain*.
- 2. Click **Test** and enter appropriate input.
- **3.** On the *Dashboard* panel, click the composite *Instance*. In the *Flow* panel of the screen, you will be able to see the flow of the customized composite.

Trace > Instanc	e of NotifyCustom	erHubProce		Data Refreshed Dec 14, 2012 4:06:25	PM GMT+05:30
page shows BPE	L process instance	details.	cess 🛛	Instance ID bpel:4501	974
it Trail Flow	Sensor Values	Faults		Started Dec 14, 2	012 4:03:29
k an activity to v	view the details.			Current Audit Level: production	View Raw X
			-		
			<u></u>		
			receiveInput		
			EchoServiceScope		
			BB		
			SetTitle		
			199		
			copyToEchoSer		
			invokeEchoSer		
			68		
			copyFromEchoS		
			88		
			which facts 0		
			possible of possi		
			(a)		
			publishPartyD		

Figure 8–24 Test Customized Composite - Flow

4. Click the *invokeEchoService* component from the flow to see the request and response XML for the invoke operation to the partner link.

Figure 8–25 Test Customized Composite - invokeEchoService



8.4.2 Add a Human Task to an Existing Process

In this example, we will demonstrate how to add a Human Task component mode.

In this example of SOA customization, we will be adding a *Human Task* to a BPEL process. Instead of adding the *Human Task* in customization mode, we will build a

separate BPEL process with the human task and then customize the base composite to include a *Partner Link* call to that BPEL process.

The following section will demonstrate how to create a BPEL process with *Human Task*. The human task will take the title as a *string* input and will have the outcomes *REJECT* and *APPROVE*. The BPEL process will invoke the human task passing the title as input. Based on the outcome of the human task, the title will be suitably modified and returned by the BPEL process.

Step 1 Create SOA Project

You will need to create a SOA project to contain the Echo Service process. To create the SOA project, follow these steps:

- 1. In the Main Menu, go to File -> New.
- 2. In the Project Gallery that opens, select SOA Project.
- 3. Click Ok.

Figure 8–26 Select SOA Project

🔁 Search	
Categories: 	
	Java Application Project
Business Tier ADF Business Components Rusiness Intelligence	Java Project Project from Existing Source
	Project from WAR File
—EJB —Security	- E Project Template
TopLink/JPA Web Services Client Tier	Opens the Create SOA Project dialog, with which you create a service-oriented architecture project using new or existing service components like BPEL, Business Rules, Human Tasks, Mediator, Adapters, and Web Services.
ADF Desktop Integration ADF Swing Extension Development	To enable this option, you must select an application, a project, or a file within a project in the Application Navigator.
-Swing/AWT	DopLink Project
Database Tier	Web Project

4. In the **Create SOA Project** wizard, enter appropriate project name (TitleApproval) and location for the project and click **Next**.

Figure 8–27 Create SOA Project Name

😣 Create SOA Pro	ject - Step 1 of 2
Name your project	
Project Name	Broject Name EchoService Dirgctory: /home/rshanbha/Work/jdev_workspace/Dec10/application/OriginationAndFulfillment/process/EchoService Brogge Broiget Technologies Generated Components Associated Libraries
<>	Available: ADF Desktop Integration ADF Faces ADF Dasktop Integration ADF Page Flow ADF Swing ADF Swing ADF Swing Ant BPM Database (Offline) Ej8 HTML java javaBeans Iechnology Description: SOA is the Service Oriented Architecture to build composite applications. Soa is the Service Oriented Architecture to build composite applications.
Help	< Back Next > Einish Cancel

- **5.** In the next dialogue of the wizard, enter appropriate name (TitleApproval) for the SOA composite.
- 6. Select *Composite With BPEL Process* from the drop-down menu.
- 7. Click Finish.

Figure 8–28 Configure SOA Settings

😣 Create SOA Pro	ject - Step 2 of 2	
Configure SOA set	tings	
Project Name	Composite Name:	
Dealact 505 Sattin	TitleApproval	
Project SOA Settin	Composite Template:	
	Empty Composite	
	Composite With BPEL Process	
	Composite With Mediator	
	Composite With Human Task	
	Composite With Spring Context	
	Composite From Oracle BPA Blueprint	
	Customizable	
Help		< <u>Back</u> <u>Next</u> <u>Finish</u> Cancel

8. The dialog *Create BPEL Process* will open. Enter a name (TitleApprovalProcess) for the process and select *Asynchronous BPEL Process* from the templates drop-down.

Figure 8–29 Configure BPEL Process Settings

A BPEL process is a service orchestration, based on the BPEL specification, used to describe/execute a business service), which is implemented as a stateful service.	s process (or large grained 🛛 👔 🗎
BPEL 1.1 Specification BPEL 2.0 Specification	
Name: TitleApprovalProcess	
Namespace: http://xmlns.oracle.com/OriginationAndFulfillment/TitleApproval/TitleApprovalProcess	
Template: Templa	• 0
Service Name: titleapprovalprocess_client	
Expose as a SOAP service	
Delivery: async.persist	*
Input: {http://xmlns.oracle.com/OriginationAndFulfillment/TitleApproval/TitleApprovalProcess}proce	ess 🔍
Output: [/http://www.scarle.com/OriginationAndEufillment@Hetencousl@Hetencousl@coresc3.com/	

Step 2 Create Human Task

After defining the BPEL process, you will need to add the *Human Task* component to the process. To add the *Human Task*, follow these steps:

- 1. From the **Project Navigator** tab, select and open *composite.xml* in the *Design* mode.
- **2.** From the **Component Palette** tab, in *SOA Components* section, select the *Human Task* component and drag and drop it onto the *components* section of the *composite.xml*.
- **3.** In the **Create Human Task** dialog that opens, enter a name (TitleApprovalHumanTask) for the human task.
- 4. Click OK.

Figure 8–30 Enter Human Task Name

😣 Creat	e Human Task
Human Tas A human t end-to-end	k task describes the tasks for users or groups to perform as part of an d business process flow.
<u>N</u> ame:	TitleApprovalHumanTask
N <u>a</u> mespace:	http://xmlns.oracle.com/process/TitleApproval/TitleApprovalHumanTask
Help	Create Composite Service with SOAP Bindings

- **5.** From *Project Navigator*, select and open *TitleApprovalHumanTask.task* file in **Design** mode. This file has the human task definition.
- 6. In the *General* section, specify a **Task Title** and **Description** for the human task.

Figure 8–31 Create Human Task - General Tab

🏠 General			
Data	Task Title: Description:	Text and XPath Title Approval Human Task Title Approval Human Task	
Deadlines Notification Access Events	Outcomes: Priority: Category:	APPROVE.REJECT 3 (Normal) By expression	Q

- 7. In the *Data* section, click the icon for add task parameter.
- **8.** In the **Add Task Parameter** dialog, specify the parameter type and name for the input to the human task. In our case, the input task parameter would be a string title.

Figure 8–32 Add Human Task Parameter

😣 🛛 Add Task F	arameter	
● Variable () Ent	ity	
Define this para	meter's type:	
🚺 💿 <u>T</u> ype:	{http://www.w3.org/2001/XMLSchema}string	٩,
	Includes standard simple XML types and types found in proje	ct schemas.
↔ ○ <u>E</u> lement		Q
	Define type by reference to elements found in project scheme	as.
Parameter Nam	e: title	
	Editable via worklist	
Use Collections		+ ×
Use Collections	Xpath	÷ ×
Use Collections Name Note: Collection r	Xpath ames must be unique across parameters	+ ×

Figure 8–33 Create Human Task - Data Tab

General	🗢 🏧 Data			4.1%
Data	Name	Element or Time		Editable
Assignment	title	{http://www.w3.org/2001/XMLSche	ma}string	Luitabi
presentation				
Deadlines	😑 🕱 Mapped Attributes			+/>
Deadlines Notification	B Apped Attributes	Value	Description	+/>

The *Assignment* section is used to define the *Users* or *User Groups* to which the human task should be assigned.

- 9. Double-click Edit Participant.
- **10.** In the **Add Participant Type** dialog, check the Value-based option for **Specify Attributes Using**.
- **11.** Click the icon for adding a value.
- 12. Select the User By Name option and enter the name of your user (weblogic).

Figure 8–34 Add Participant Type Details

pe: Single	-	Label:	Stagel.Participant1	
			e.g., Approval Manager	
articipant List				
Build a list of participa	nts using: Names	s and exp	pressions 🕶	
Build a list of participa	nts using: Names	and exp	pressions	
Build a list of participa	nts using: Names	and exp	pressions ¥	
Build a list of participa Specify attributes us	ing:	and exp	pressions 💌	
Build a list of participa Specify attributes us Participant Names	nts using: Names ing: ⊚ <u>V</u> alue-base	and exp	pressions ▼ Rule-based	+- X
Build a list of participa Specify attributes us Participant Names Identification Type	nts using: Names ing: <u>Value-base</u> Data Type	ed OR	Rule-based	4 - X
Build a list of participa Specify attributes us Participant Names Identification Type User	nts using: Names ing: <u>Value-base</u> Data Type By Name	and exp d R Value weble	Rule-based e logic	+- X

Figure 8–35 Create Human Task - Assignment Tab

没 General 🍑 Data	🗱 Make Parallel 👬 Make Serial 🚕 😪 🕂 🖋 Edit 🕱	Task will go from starting to final participant
S Assignment		
Presentation		
Deadlines		
> Notification		
Access		
Fvents		
🖢 Documents	I Stagel	
	Stagel.Participant1	

- **13.** From the Project Navigator, open the *TitleApprovalProcess.bpel* file in **Design** mode.
- **14.** From the **Component Palette**, select the component *Human Task* and drag-drop it on to the BPEL process.
- **15.** Click the icon to add task parameter.
- **16.** In the **Task Parameters** dialog, select the *string* input to the BPEL process.

Figure 8–36 Select Human Task Parameters

rom	
pe: Variable	
) Variables	
Process	
🖃 🖾 Variables	
InputVariable client: TitleApprovalProcessRequestMessage	
E payload	
Client process <anonymous></anonymous>	
(r) output/variable client TitleApprovalDroceesDeepopeeMessage	
⊕ Scope - HumanTask1	
Show Detailed Node Information	
Path: /client:process/client:input	

17. In the task outcomes *Switch* in the BPEL process, delete the condition for *otherwise*.

Figure 8–37 Create Human Task - Delete Condition



- **18.** From the **Component Palette**, select the *Assign* component and drag-drop it to the *REJECT* outcome of the switch.
- **19.** Enter a *name* (rejectTitle) for the component.
- **20.** In the *Copy Rules* section of the assign, use the *Expression Builder* to set the output *string variable* of the BPEL process to '<title> Rejected'.

Figure 8–38 Create Human Task - Expression Builder

concat(bpws:getVariableData('inputVariable', 'payload', '/clie	ent:process/client:input'), ' - Rejected')	
\land Insert Int	o Expression	
BPEL Variables	Functions	
Content Preview:	String Functions	io Io
bpws:getVariableData('inputVariable', 'payload', '/client:proc	ess/client:input')	
Variable XPath expression		

Figure 8–39 Create Human Task - Copy Rules

Insert New Rule After 🔻		📆 🚖 🔕 🐲 🖷		
valProcess.bpel		TitleApprovalProcess.bpel		
les		Variables 🗁		
cess		Process 💑 😑		
Variables		Variables 🗁 😑		
(x) inputVariable client: TitleApprovalProcessR		inputVariable client: TitleApprovalProcessRequestMessage (x) 🔅		
😑 🔚 payload		outputVariable client: TitleApprovalProcessResponseMessage (x)		
client process <anonymous></anonymous>		payload 🔚 🖮		
- client input string		client:processResponse <anonymous> 🍪 🖻</anonymous>		
 (x) outputVariable client: TitleApprovalProcess 		client result string 🥸		
 TitleApprovalHumanTask1_globalVariable 		oprovalHumanTaskl_globalVariable taskservice.taskMessage (X) ⊕		
<				
vac2		To XPath: /client:processResponse/client:result 🛛 🕂 🎇 🍲 🗏		

21. Save all changes to the human task. The BPEL process should look as shown in the figure below.



Figure 8–40 Create Human Task - BPEL Process

22. Deploy the SOA process to the server as mentioned in the previous example.

Step 3 Create Human Task Form

The *Human Task* is visible to assigned users in the *BPM Worklist* application. To display the task parameters and the payload for the task, you will need to create a *task-flow* with the *Human Task* Form. This task form can be auto generated through the process. Follow these steps:

- **1.** From the Project Navigator, open the *TitleApprovalHumanTask.task* file in **Design** mode.
- 2. Click the button for *Create Task Form*.
- 3. Select the *Auto-generate Task Form* option from the context menu.

Figure 8–41 Select Human Task Form

Auto-Genera	te Task Form		
Launch Task	Form Wizard	Text and XPath Title Approval Human Task	
Assignment Presentation	Description:	Title Approval Human Task	
Deadlines	Outcomes:	APPROVE,REJECT	Q
Notification Access Events Documents	Priority: Category: Owner:	3 (Normal) By expression User Static Q	

- **4.** Enter a name (TitleApprovalHumanTask) and *location* for the human task form project.
- 5. Click Finish.

The generated human task form project will have default file names. Re-Factor file names for form and page definition using appropriate naming conventions.

Step 4 Deploy Human Task Form Project

You will need to deploy the human task form project on the UI server for the previously deployed SOA process. To deploy, follow these steps:

- 1. Clean and build the project.
- 2. Right-click the project and select *Deploy* from the context menu.
- **3.** In the **Deploy** dialog that opens, select the *Deploy to Application* Server option from the list and click *Next*.
- 4. Select the appropriate UI server for the SOA server.
- 5. Click Finish.

eployment Actior	
Deployment Action	Select a deployment action from the list below.
Select Server Weblogic Options Summary	Deploy to Application Server Deploy to WAR
	Create an archive and deploy it on a remote Application Server.

Figure 8–42 Select Human Task Form Deployment Action

Figure 8–43 Select Human Task Form - Weblogic Options

eblogic Options			
Deployment Action Select Server Weblogic Options	Indicate which server instances to target for deployment. O Deploy to <u>a</u> ll instances in the domain O Deploy to <u>s</u> elected instances in the domain O List Standalone Servers and Clusters		
· <u></u>	Name AdminServer soa_server1 Ui_server bam_server1	Servers	Status RUNNING RUNNING RUNNING SHUTDOWN
	This module can optionally Deploy as a standalone	/ be registered as a shared li Application	brary

Step 5 Add Customizable Scope to SOA Application

To demonstrate customizations of a SOA process, we will be using the BPEL process *NotifyCustomerHubProcess* present in the composite *com.ofss.fc.workflow.process.NotifyCustomerHub*.

Open the SOA application which contains the base composite which will be customizing. The aforementioned process is present in the *OriginationAndFulfillment* application inside the *com.ofss.fc.workflow.NotifyCustomerHub project*.

To add a customizable scope to the BPEL process, follow these steps:

- 1. Open the *NotifyCustomerHubProcess.bpel* file in **Design** mode.
- **2.** From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Scope* component and drop it on to the BPEL process as shown in the figure.
- 3. Double click the component and enter appropriate name for the component.
- **4.** Drag and drop the existing *Assign* component labeled *setTitle* on to the newly added *Scope* component.

Figure 8–44 Add Customization Scope to SOA Application

RotifyCustomentubProcess.bpel × in composite xml ×			•
🖉 🥥 (/) 🗞 ஞ 📾 🗒 🕕 🕄 • 🖗 • 👶 • 🍓 🗆 🖧 📲		(8-	🚺 🛃 🖾 Monitor 🕞 Test 🕖
Partner Links	Receivingut		Partner Units 🛆 🗅
	■ 20 0 SetTide SetTide		
	publishd any dealis		
	publishPargOrtails		NOHP any Applic
	0		
SetTitle - /process/sequence/scope/assign			Zoom. 100 0 0
Design Source History			

- 5. Right-click the *Scope* component and select *Customizable* from the context menu.
- 6. Save all the changes and restart JDeveloper in *Customization Developer Role*.

Step 6 Customize the SOA Composite

After adding a *Customizable Scope* to the base composite, you can start performing customizations in JDeveloper's *Customization Developer Role*.

When you open the *NotifyCustomerHubProcess.bpelf* ile in Design mode, you will notice that all other components in the process, except the customizable Scope component, are disabled. This means that your customizations are limited to that scope.

In the following sections, we will be adding a *Partner Link* call to the previously created *TitleApproval BPEL* process and other required components in the customization mode.

Step 7 Add Partner Link Component

To add a *Partner Link* to the BPEL process, follow these steps:

1. From the Project Navigator, open the NotifyCustomerHubProcess.bpel file in Design mode.
- From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the Partner Link component and drop it on to the *Partner Links* section of the BPEL process.
- **3.** In the *Create Partner Link* dialogue that opens, enter appropriate name (TitleApproval) for the partner link.
- 4. Select the following options:
 - 1. TitleApprovalProcess as the Partner Link Type.
 - 2. TitleApprovalProcessProvider as the Partner Role.
 - 3. TitleApprovalProcessRequester as the My Role.

Figure 8–45 Add Partner Link Component

ame: TitleAppr	roval
rocess:	
✓ Initiali	ze Partner Role
SDL Settings	
	🚱 i 🔊 🖉
SDL URL:	45:8001/soa-infra/services/default/TitleApproval/titleapprovalprocess_client_ep?WSDL
artner <u>L</u> ink Type	: 🚏 TitleApprovalProcess 🔹
	S TitleApprovalProcessProvider
arther Role:	

Step 8 Add Invoke Component

You will need to add an *Invoke* component to invoke the previously added partner link call to *TitleApproval*. To add *Invoke component*, follow these steps:

- 1. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Invoke* component and drop it on the BPEL process inside the *Scope* component.
- Click the *Invoke* component and drag it to the previously added *TitleApproval* partner link.
- **3.** Double-click the *Invoke* component.
- **4.** In the *Edit Invoke* dialogue that opens, enter an appropriate name (invokeTitleApproval) for the component.
- 5. Click the icon for adding a new variable in the *Input Variable* section.
- **6.** Click **OK** to save the changes.

Figure 8–46 Add Invoke Component

General		Correlations	Properties	Annotations
		contendents		- Milletations
ame: ir	vokeTitleAppro	val		
onversation ID:				
etail Label:				
] Invoke as Det	ail		
Interaction Type	e: 🚳 Partner L	ink▼		
	la fanancial			
	Texpproval			~
Port Type:	TitleApprovalP	rocess		-
Operation:	process			-
Input Output				
O To Parts	Input Varia	ble		
-		Br J.L.		
Input: Invokeln	put_process_in	outvariable		

Step 9 Add Receive Component

You will need to add a *Receive* component to receive output from the previously added partner link call to *TitleApproval*. To add *Receive* component, follow these steps:

- 1. From the *Component Palette* panel on the right side, in the *BPEL Constructs* section, drag the *Invoke* component and drop it on the BPEL process inside the *Scope* component.
- **2.** Click the *Receive* component and drag it to the previously added *TitleApproval* partner link.
- **3.** Double-click the *Receive* component.
- **4.** In the *Edit Receive* dialogue that opens, enter an appropriate name (receiveTitleApproval) for the component.
- 5. Click the icon for adding a new variable in the *Output Variable* section.
- 6. Click **OK** to save the changes.

Documentation	Targets	Sources	Headers	Asser	rtions	Skip Condition	Timeout		
General	r in the second s	Corre	lations			Properties		Annotations	
<u>N</u> ame:	receiveTit	eApproval							
Conversation ID:									
	Create	Instance							
		Inoconce	6						
 Interaction Ty 	pe: 🔯 Pa	rtner Link▼							
Partner Link: T	itleApprova	I							Q
Port Type:	TitleAppr	ovalProces	sCallback						-
Operation:	processi	Response							-
O From Parts	 Variable 								
Variable: receit	veOutput p	rocessResp	onse InputV	/ariable					4 Q
-									

Figure 8–47 Add Receive Component using BPEL functions

Step 10 Add Assign Components

An *Assign* component is used to assign values to a variable. These values can be directly assigned from one variable to another or modified using BPEL functions available.

The *TitleApproval* accepts a single *string* as an input and gives a single *string* as an output. The *Input Variable* and *Output Variable* defined in the previously created *invokeTitleApproval* and *receiveTitleApproval* components will be used to hold the input value for the *TitleApproval* and the output returned respectively.

In our case, we will need to add two Assign components for following purposes:

- To populate the *Input Variable* of the *invokeTitleApproval* component with the value returned by the existing *setTitle* component.
- To populate the *setTitle* component with the value returned in the *Output Variable* of the *receiveTitleApproval* component.
- Add the two required Assign components and save all changes.

The customized process should look as shown in the figure below.

Figure 8–48 Add Assign Component



Step 11 Deploy and Test Customized SOA Composite

After performing the customizations, build the project and deploy it to a SOA Server. You will need to include the *Customization Class JAR* in the runtime classpath of the deployed application.

To test the customized composite, follow these steps:

- 1. Log in to the *em* console and select the composite from the *SOA Domain*.
- 2. Click **Test** and enter appropriate input.
- **3.** A *Human Task* will be created and assigned to the user as specified in the human task definition.
- 4. Log in to the *BPM Worklist* application with the appropriate user.

You will be able to see the previously created task on the dashboard.

5. Select the task from the list and click *Approve* or *Reject* button to perform approve or reject action on the task.

Worklist Views	+/%	My Tasks Initiated	Tasks Administr	ative Tasks						
Inbox		Actions - 🚷 Assign	nee Me & Group *	State Assig	ned *	Search		Advanced		
My Work Queues		Title	Number	Priority	Assignees	State	Created	Expires		
Due Soon		Title Approval Human T	Task 201609	3	weblogic (U)	Assigned	Dec 21, 2012 10:5			
Past Week Past Month Past Quarter New Tasks Mry Views Proxy Work Queues Shared Views										
			n Tack						Acti	 ontve Reie
Task Status	/	Title Approval Human							1000	
Task Status Assigned		Title Approval Human								
Task Status Assigned 1 Suspended		E B Details								
Task Status Assigned 1 Suspended Completed 4		Title Approval Human								
Task Status Assigned 1 Suspended Completed 4 Expired		Title Approval Human								
Task Status Assigned I Suspended Completed Expired Withdrawn		Title Approval Human Details Contents Title 000007918								
Task Status Assigned 1 Suspended Completed Expired Withdrawn Alerted Frozend		Title Approval Human								
Task Status Assigned 1 Suspended Completed Expired Withdrawn Alerted Errored Deleted		Title Approval Human Contents Title 000007918 History								

Figure 8–49 Deploy and Test Customized SOA Composite - My Tasks Tab

6. On the *Dashboard* panel of the *em*, click the composite *Instance*.

In the *Flow* panel of the screen, you will be able to see the flow of the customized composite.

Figure 8–50 Deploy and Test Customized SOA Composite - Flow

Now Trace > Instance of NotifyCustomerHubProcess		Data Refreshed Dec 21, 2012 3:42:56 PH GMT+05:30 €2
This page shows BPEL process instance details.		Instance ID bpel:451387 Started Dec 21, 2012 3:41:56 PM
Audit Trail Flow Sensor Values Faults		
Click an activity to view the details.		Current Audit Level: production Wiew Raw XML
	receiveInput	
	□ Scope1	
	Section	
	copyToInvokeI	
	invokeInput	1
	receiveOutput	
	copyFromRecel	
	· · · · · · · · · · · · · · · · · · ·	
	publishPertyD	

7. Click the *invokeTitleApproval* component to see the request xml for the partner link call to *TitleApproval* process.

Figure 8–51 Deploy and Test Customized SOA Composite - Invoke Input



8. Click the *receiveTitleApproval* component to see the response xml for the partner link call to *TitleApproval* process.

Figure 8–52 Deploy and Test Customized SOA Composite - Receive Output



9

Batch Framework Extensions

Most of the enterprise applications require bulk processing of records to perform business operations in real-time environments. These business operations include complex processing of large volumes of information that are most efficiently processed with minimal or no user interaction. Such operations would typically include time-based events (for example, month-end calculations, notices or correspondence), periodic application of complex business rules processed repetitively across very large data sets (for example, rate adjustments). All such scenarios form a part of batch processing. Thus, batch processing is used to process billions of records for enterprise applications.

There are few primary categories in OBP Batch Processes:

- Beginning of Day (BOD)
- Cut-off
- End of Day (EOD)
- Internal EOD
- Statement Generation
- Customer Communication

Additional categories can also be configured as per the requirement.

9.1 Typical Business Day in OBP

The following graphic describes a typical business day in OBP:





9.2 Overview of Categories

This topic describes the categories in OBP Batch Processes.

9.2.1 Beginning of Day (BOD)

The activities for a new day of the bank / branch begin with the BOD (beginning of day). This is a batch process which executes a group of shells (programs) which are required to be performed before the normal day-to-day operations at the branch can be started. The BOD typically includes

- TD Maturity and Interest Processing
- Standing instructions execution (Based on setup)
- Loan Charging, Drawdown and Auto-Disbursement
- Value date processing of cheques (Based on the setup)
- Reports Generation

9.2.2 Cut-off

Cut-off is a process that sets the trigger for modules to start logging transactions with a new date.

It also marks cut-off for the channel transactions.

9.2.3 End of Day (EOD)

Once all the operations for the current working date of the branch are completed and all the transactions are posted the Branch EOD batch is started. This batch executes a

group of shells (programs) which are required to be performed before the Business Date of the branch is changed to the next working date. It marks the end of a business day. The EOD typically includes:

- DDA Sweep-Out Instruction
- Loan Rate Change
- Term Deposit Lien Expiry and Interest Capitalization
- DDA Balance Change, Rate Change, Interest Capitalization and Settlement
- Account and Party Asset Classification
- Loan Interest Computation
- Accounting Verification

9.2.4 Internal EOD

This category performs all the activities which do not affect the customer account but are related to bank internal processing. Internal EOD typically includes:

- Interest Accrual and Compounding
- Deferred Ledger Balance Update
- Balance Period Creation
- Financial Closure

9.2.5 Statement Generation

This category performs different statement generation activities on the monthly or yearly basis. It typically includes:

- Periodic PL balance history Generation
- CASA Statement Generation
- Loan Statement Generation
- TD Statement Generation

9.2.6 Customer Communication

This category performs different communications which needs to be done with the customer on the regular basis. It typically includes:

Regular Account Balance Notification On Specified Date

9.3 Batch Framework Architecture

This section describes the architecture of the Batch Framework.

9.3.1 Static View

The static view of batch framework shows the architecturally significant classes included in the batch framework being developed. It is in line with the overall design and development guidelines and principles. This section shows the class diagrams representing the static model of the batch framework emphasizing the static structure of the system using objects, attributes and relationships.

Class Diagram

The following diagram depict details about the different classes of the code which are involved in the batch execution.





9.3.2 Dynamic View

This section emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects.

Sequence Diagram

The following diagram depicts the sequence diagram for Batch framework. It provides details about the flow of control during the batch execution.





State Diagram of a Shell

When the end of day batch starts, every shell is reset to Not Started. During the course of the batch, the shell status will change till the shell is completed. The transitions of shell execution are explained in the state diagram below:

Figure 9–4 State Diagram of a Shell



9.4 Batch Framework Components

This section describes the batch framework components.

9.4.1 Category Components

This section describes the category components.

CategoryListenerMDB

This MDB listens to the FCBBatchRequestQ and delegate to CategoryHelper for further processing.

CategoryHelper

This class starts or restarts a category depending upon the request received.

It will validate the input xml Request, validate the prerequisites for starting/restarting a category, get the list of shells that can be initiated on a category start/shell

completion, prepare the Batch XML Message for each of the shell and send a message to FCBBatchShellQ for each Shell to be started.

It also services requests initiation of the next shell after a shell has been successfully completed.

9.4.2 Shell Components

This section describes the shell components.

ShellListenerMDB

This MDB listens on ShellRequestQ and delegate to ShellProcessHelper for processing.

ShellProcessHelper

This class validates the input request and calls appropriate batch handler to start the shell. It will call:

- BatchFrameworkShellHelper for non-report Java Bean Based Shell
- ProcedureShellHelper for Procedure based shell
- BatchReportShellBean for report shells
- BatchReportRestartShellBean for report epilogue shells

After successful completion of shell, it sends an 'InitiateNext' request to the CategoryHelper to initiate subsequent shells. If the shell is aborted, this class will mark the shell as aborted.

ShellRootHelper

This is the base class which is required for each shell processing. It Implements the IBatchHandler Interface. All the batch handlers extend this class.

This class contains the common methods which need to invoked for processing each shell for example, method to parse the request, methods used to acquire and release lock for shell, method to initiate the shell and mark the shell as complete upon successful completion.

BatchFrameworkShellHelper

This SSB extends ShellRootHelper. It is responsible for executing non report Java Bean based shells. This class will validate the process date of the request, prepare a BatchContext entity encapsulating the batch run details and call BatchJobHandler to run the shell.

BatchJobHandler

This class is responsible for putting the stream requests in queue. It will get the Batch Processes (1 Batch Process per stream) by calling BatchProcessManager and post them to the Stream Queue.

After posting the stream requests, it will start polling on the status of the streams till either all streams are completed or any one of the streams is aborted. If the streams are completed, it will return 'Success' as the status else it will return the status as 'Failure'.

BatchProcessManager

This component acts as a manager for the complete batch process. The functionalities include finding the pending batch processes and creating batch processes and returning the list of batch processes to be initiated.

If the shell is being restarted, this class will fetch the aborted batch processes, reset them and return list of reset Batch Processes to be re-initiated.

If the shell is being started, it will call BatchJobHelper to populate the driver table and create the batch processes and return the list of batch processes to be initiated.

BatchJobHelper

This class is responsible for populating the driver table and creating the Batch Processes.

ProcedureShellHelper

This class is used to process DB procedure based shells. This class will fetch the procedure to be executed from the 'flx_batch_job_shell_master' table and execute it.

BatchReportShellBean

This class is responsible initiating the generation of reports. It will call ReportJobRequestor to fetch the reports to be generated, prepare the generation request and post the requests to the Report Queue.

After the successful posting of requests, the report shell will be marked as complete. The report generation will be done in parallel to the execution of subsequent shells.

BatchReportRestartShellBean

This class is used for the epilogue shell in each category which has reports generation.

This class will check whether all the reports have been generated or not. This class will call ReportJobRequestor which will poll on the status of the reports till all the reports are completed or aborted.

If the aborted reports are to be regenerated, it will also post the messages to regenerate aborted reports.

9.4.3 Stream Components

This section describes the stream components.

StreamListenerMDB

This MDB is responsible for listening to the stream queue. It delegates the processing to StreamProcessHelper.

StreamProcessHelper

This class is responsible for starting the batch process. It calls RecoverableBatchProcess to start the process.

BatchProcess

This component is the base class for processing the batch process. The StreamProcessHelper calls this class for starting the batch process. This class will initialize the BatchShellResult, clear the StaticCacheRegistry (if the BatchProcess is the first BatchProcess of a category), process the BatchProcess, retry the processing of the BatchProcess (if the earlier failure was due to StaleState or PKDuplication) and finalize the BatchShellResult status depending on success/failure.

The call to process a batch request is routed through this class to the subclass.

RecoverableBatchProcess

This component processes the batch data and inherits the BatchProcess class. This class will process all the records in the sequence number range specified in the BatchShellResult.

This class will fetch the records from the driver table and process them sequentially.

To execute each record, it will call service method of the service class stored in the BatchShellDetails table using reflection. If there is any exception, it will call the exception handler method of the service class if the service class implements the IBatchExceptionHandler interface.

It will commit the transaction at the end of commit size. If all the records are executed successfully, the stream is marked as complete. If any record fails, the stream is marked as aborted.

Recoverable Batch Process can handle the failure of a record in the following ways depending upon the set up.

Recoverable Batch Process with Recovery Mode ON

When a record fails, the previous records in the commit size will be committed and marked as success, the failed record will be marked as failed and the execution of batch process resumes from the record after the failed record. Hence in this mode all the successful records are committed and the failed records are marked as failed.

Recoverable Batch Process with Recovery Mode OFF

In this mode, when a record fails the earlier records in the commit size are marked as skipped for the current run, the failed record is marked as failed and execution of batch process resumes from the record after the failed record.

Simple Batch Process

While executing the shell as a Simple Batch Process, the stream will be executed till the first failed record. When a record fails, the previous records in the commit size will be committed and the shell will be aborted. The records after the failed record will be skipped in the current run.

SimpleBatchProcess class is no longer used

The functionality of SimpleBatchProcess is executed through RecoverableBatchProcess by specifying the FLG_PROCESS_TYPE as "SBP" in the flx_batch_job_shell_dtls table. In the flx_batch_job_shell_dtls table:

- FLG_PROCESS_TYPE column indicates whether it is RecoverableBatchProcess (RBP) or SimpleBatchProcess (SBP).
- FLG_RECOVERY_MODE column indicates whether the Recovery mode is ON or OFF
- Simple Batch Process should have Recovery Mode as ON.

Example 9–1

```
Total Number of records =20;
Commit Frequency = 10
Failed Records = 5, 18
```

The shell will be executed as follows:

Recoverable Batch Process with Recovery Mode ON:

- Records 5 and 18 will be skipped and rest all the records will be committed successfully
- Recoverable Batch Process with Recovery Mode OFF:
 - Records 1 5 will be skipped.
 - Records 6 15 will be committed successfully.
 - Records 16-18 will be skipped
 - Records 19 20 will be committed successfully
- Simple Batch Process:
 - Records 1- 4 will be committed successfully. Rest of the records will be skipped.

9.4.4 Database Components

The Database Server houses the following components:

Batch Framework Tables	Description
flx_batch_job_category_ master	This table contains details of each of the category per branch group. This table contains the description, last run date and the multi run flag for the category. The status, state flag and the last Run Date for each category is maintained and validated from this table during batch run.
flx_batch_job_grp_ category	This table contains the previous, current and the next run date for each category per branch group.
flx_batch_job_category_ depend	This table contains the category dependencies.
flx_batch_job_shell_ master	This table contains details of each shell per branch group. Shell wise status, Last Run Date, process category and frequency of shell execution are the critical attributes of this table.
	This table will also specify whether the shell is Java Bean based shell or Procedure Based shell.
flx_batch_job_shell_ depend	This table contains the dependencies of and for each shell in flx_batch_job_shell_ master.
flx_batch_job_shell_dtls	This table will contain the details for executing Java Bean Based shell.
flx_ <module>_drv_ <action></action></module>	This driver table contains the batch execution details for the particular action
flx_ <module>_actions_b</module>	This table defines the action type, action name and action executor which gets mapped to the driver table. The action type value is populated as action sequence in the driver table.
flx_batch_job_shell_ results	This table contains execution details of each stream of each shell for each batch run per branch group.
flx_batch_job_brn_grp_ mapping	This table will contain the mapping between the branch group and the branches
flx_batch_job_grp_brn_ xref	This table will contain the list of branches for which a category is being run. This table will be used only when a category is running.

Table 9–1Database Server Components

9.5 Batch Configuration

The following section defines the configuration which needs to be done in order to create a new category or add a new batch shell for batch execution using the batch framework.

9.5.1 Creation of New Category

The following steps explain the creation of new category:

1. Create an entry in flx_batch_job_category_master:

This contains the new category name and category code along with branch group code to be defined here.

Columns	Description
DAT_EOD_RUN	This column specifies the date on which the category was last run
COD_EOD_STATUS	This column specifies the Status of the last category run. 0 - Successful Completion ; 1 - The process was aborted after start
COD_PROC_CATEGORY	This column specifies the category code. 1 - EOD, 2 - BOD etc. Any number of process categories can be defined
FLG_MULTI_RUN	This column specifies whether this category can be run multiple times. 0 - Multi-Run is disabled; 1 - Multi-Run is enabled.
FLG_EOD_STATE	This column specifies the flag indicating the state of the category. R - Running ; C - Completed (i.e. not running)
TXT_CATEGORY	This column specifies the category description
COD_BRANCH_ GROUP_CODE	This column specifies the code of the Branch Group of the category
OBJECT_VERSION_ NUMBER	This column specifies the version number of the category
NAM_PROD_REP_DB	This column mentions about the database repository

Table 9–2 FLX_BATCH_JOB_CATEGORY_MASTER

2. Create an entry in flx_batch_job_grp_category:

This contains branch group code, new category code, bank code and dates relating to run the category.

Table 9–3 FLX_BATCH_JOB_GRP_CATEGORY

Columns	Description
BRANCH_GROUP_ CODE	This column specifies the Branch Group Code
COD_PROC_CATEGORY	This column specifies the procedure category
DAT_LAST_PROCESS	This column specifies the date on which the category was last run
DAT_PROCESS	This column specifies the current date of the category
DAT_NEXT_PROCESS	This column specifies the next date of the category

3. Create an entry in **flx_batch_job_category_depend** (if required):

This table will contain the category dependency. If the category does not depend on any other category, no entry in this table is required.

Columns	Description
COD_PROC_CATEGORY	This column specifies the procedure category
COD_BRANCH_ GROUP_CODE	This column specifies the branch group code
COD_PROC_REQD_ CATEGORY	This column specifies the dependency of the required procedure category which needs to be run before this category
COD_PROC_ VALIDATION_DATE	This column defines the validation time. It can be Current/Previous.

Table 9–4 FLX_BATCH_JOB_CATEGORY_DEPEND

4. Create bean or procedure based shells:

New shells (bean/procedure based, as shown in the section below) are created and linked to the category by populating the cod_proc_category column in those tables with the new category code created in the flx_batch_job_category_master.

5. Add enumeration:

In the middleware code, add an enum value in the ProcessCategoryType.java for the category.

6. Add category code in the property file:

In the middleware code, add the entry for the category in the ProcessCategoryType_en.properties file.

7. Middleware Changes

If any validations required or any dependency on other categories we can make changes in EODShellProgressManager.java file accordingly.

Figure 9–5 Creation of New Category



9.5.2 Creation of Bean Based Shell

In this batch execution (Type "B"), the business logic is provided in the service method of the java class.

1. Create an entry for Shell Parameters in the table FLX_BATCH_JOB_SHELL_ MASTER.

Columns	Description
COD_EOD_PROCESS	Process code. This is the name of the program module that will be started as a process by the EOD monitor.
TXT_PROCESS	Process name to be displayed in the new UI screen
FRQ_PROC	Frequency at which this process is to be run.
	1 - Daily 2 - Weekly 3 - Fortnightly 4 - Monthly 5 - Bi-monthly 6 - Quarterly 7 - Half-yearly 8 - Yearly.
COD_PROC_STATUS	Process Status Code 0 - Complete 1 - Started 2 - Not Started 3 - Aborted 4 - Prerequisite Aborted 5 - Prerequisite Absent
NUM_PROC_ERROR	Last error returned by this process
FLG_RUN_TODAY	Flag indicating whether process to be run today Y/N
COD_PROC_CATEGORY	Category code to which this shell belongs to e.g.: 1 - EOD, 2 - BOD and so on.

Table 9–5 FLX_BATCH_JOB_SHELL_MASTER

Columns	Description
SERVICE_KEY	Service method to be executed
NAM_COMPONENT	Name of the Procedure (if procedure based batch execution) or fully qualified class name of the Batch Handler (if bean based).
	1. com.ofss.fc.bh.batch.BatchFrameworkShellHelper - java bean based shell
	2. com.ofss.fc.bh.batch.BatchReportShellBean - procedure based shell for reports
	3. com.ofss.fc.bh.batch.BatchReportRestartShellBean - procedure based for report epilogue shell
TYPE_COMPONENT	This indicates whether the specified nam_component is Java class or Function. P stands for Function and B Stand for the Java Class.
NAM_DBINSTANCE	The DB instance for PROD or REP(reports)
COD_BRANCH_ GROUP_CODE	Specifies the branch group code that a branch is part of.
OBJECT_VERSION_ NUMBER	This column specifies the version number of the category

Table 9–5 (Cont.) FLX_BATCH_JOB_SHELL_MASTER

2. Create an entry for Shell Details in the table **FLX_BATCH_JOB_SHELL_DTLS**.

This table contains the following parameters;

Columns	Description
COD_SHELL	A unique code for batch shell.
SHELL_NAME	Provide a name to batch shell
SHELL_DESCRIPTION	Description about the batch shell
COMMIT_FREQUENCY	Provide the commit frequency thus, after every this no of records have been processed the framework would commit those set of records
FLG_RECOVERY_MODE	Flag indicating whether recovery mode is ON or OFF. Possible values are 'Y' and 'N' only. This would be only used by Batch Processes which support recovery mode functionality but there might be batch processes which would ignore this flag (e.g.: SBP)
FLG_STREAM_TYP	Define the type of stream for the batch shell. This would have three possible values ('S' - fixed no of streams, 'R' - fixed no of rows, 'N' - no streams)
STREAM_COUNT	Define the no of streams to be created for the batch shell. This is only applicable if the StreamType is marked as 'S' or 'R'
INPUT_DRV_NAME	Define the fully classified class name mapped to the driver table
INPUT_SHELL_PARAM	Define the name for the shell parameter
SERVICE_CLASS_NAME	Define the fully classified class name for the service class. This class is the starting point of the business logic execution.
	In case of service class name as ActionSetProcessor, the action sequence column is populated in the driver table. The execution is done corresponding to those actions
SERVICE_METHOD_ NAME	Define only method name of the service. The service method should have input parameter as driver table entity
DRV_POP_PROC_NAME	Defines the name procedure used for driver table population. The procedure should have three input parameters branch group code, process date and next process date. Use only procedures instead of packages for data population. Because db2 will not support Package

 Table 9–6
 FLX_BATCH_JOB_SHELL_DTLS

Columns	Description
FLG_PROCESS_TYPE	It defines the type of process RBP or SBP. In RBP (Recoverable Batch Process) if any records fails in batch it will continue and execute rest of the records in the stream. But in case of SBP (Simple Batch process) it will abort the stream
HELPER_CLASS_NAME	It defines the helper class for caching big queries
BATCH_NO	Indicates the batch number for the shell

Table 9–6 (Cont.) FLX_BATCH_JOB_SHELL_DTLS

3. Create an entry for Shell Execution Order in the table **FLX_BATCH_JOB_SHELL_ DEPEND**.

Table 9–7	FLX_BATCH_JOB_SHELL_DEPEND
-----------	----------------------------

Columns	Description
COD_EOD_PROCESS	Process code This is the name of the program module that will be started as a process by the EOD monitor
COD_REQD_PROCESS	Required process code after which the framework will run process code
COD_PROC_CATEGORY	Category of the Process Code. 1 - EOD, 2 - BOD and so on.
COD_REQD_PROC_CAT	Category of the required process code. 1 - EOD, 2 - BOD and so on.
COD_BRANCH_ GROUP_CODE	This column specifies the branch group code

If the shell is not dependent on any other shell or category then no need to keep an entry in this table.

4. Create a new driver table (the name of the table prefix by FLX_<ModuleCode>_ drv_<>) for the Batch Shell. This is the table from which the data will be picked up for processing by the defined batch process. This table should be populated by the procedure written for population of the driver table. This table would contain the following parameters:

Column	Description
DATE_RUN	Defines the date on which the batch job was run (process date). Value in this column needs to be populated by the driver table population procedure.
SEQ	Sequence no for the data present in the table i.e. simple sequence from 1 to maximum number of records present in table. Value in this column needs to be populated by the driver table population procedure.
PROCESS_RESULT	Define the column which would contain the result of processing of each record of this table. This column would be updated the framework with values 0,1, 2,3 or 4 indicating not processed, processing of record successful, failed with business exception , failed with framework exception or failed with SQL exception respectively.
ERROR_CODE	Define the column for error code. This would be updated the framework with the error code returned by the processing logic (currently updating the exception if any occurred).
BRANCH_CODE	Attribute specifies the branch code in which the shell is executed
BRANCH_GROUP_ CODE	Attribute specifies the branch group code that a branch is part of.

Table 9–8 Driver Table

Column	Description
ERROR_DESC	Attribute specifies error description. This will populated by the batch framework in case the record aborts
ACTION_SEQUENCE (Optional)	In case of service action as ActionSetProcessor, the batch execution is done through the executor framework defined in the action table of the module. The details of this action table in mentioned below.
	If user want to execute multiple actions, then the comma separated action_type can be defined in this column. They will be executed based on the defined priorities.
<custom_columns></custom_columns>	Define the other columns required which would contain the data required by the processing logic. Typical examples would be a column containing accountNo (if the main logic is per account) or customerId or txnRefNo etc. We can have multiple such columns which are used for per record processing for e.g. we can have two columns branchCode, accountNo.

Table 9–8 (Cont.) Driver Table

Note: DATE_RUN, SEQ, BRANCH_GROUP_CODE columns are part of the Unique key. e.g.: flx_in_drv_eod_actions

5. Add the entry of the action in the actions table (FLX_<ModuleCode>_actions_b) for the shell where the service method is defined as ActionSetProcessor in the details table. This table would contain the following parameters, for example, flx_td_actions_b

Table 9–9	Actions	Table
-----------	---------	-------

Column	Description
ACTION_TYPE	Stores the type of action to be performed. The defined action type is populated in the action sequence column of the driver table.
ACTION_LEVEL	Stores the action level of the action as 0,1,2 based on the execution status.
PRIORITY	Stores the priority of the action.
ENTITY_STATUS	Stores the status of the entity.
ACTION_NAME	User friendly name of the action.
ACTION_DESC	Stores the description of the action.
ACTION_EXECUTOR	Stores the name of the action executor which needs to be executed when the service action is populated as ActionSetProcessor.
HOLIDAY_TREATMENT	Stores the holiday treatment of the action.
HOLIDAY_EPOCH_TYPE	Stores the holiday epoch type of the action.

- 6. Create a procedure (the name of the proc prefixed with ap_<Module Code>_pop_ drv) which would populate the data in the driver table, created above. This procedure would be called at the first time when the Batch shell is run. The procedure will have only three arguments branch group code, process date and next process date. e.g.: ap_in_pop_drv_eod_actions
- **7.** Create an entity which extends **AbstractBatchData** and map this entity to the driver table. This entity name would be the one which will carry the data to be processed for batch processing. This should be provided in the InputDataName column of flx_batch_job_shell_dtls table. e.g.: InterestEODActionSetBatchData

- 8. Map the entity to the driver table in the hbm. The entity attributes should represent only Extra columns added in the driver table. They shouldn't be mapped to the seq, date_run, error_code, process_result columns. For example, InterestEODActionSet.hbm.xml
- **9.** Make additions in **batch-mapping.cfg** file for the new hbm entities created for BatchData. For example, account-mapping.cfg.xml
- Create Helper Class for caching big queries in Application layer. The fully qualified class name of the helper class needs to be defined in the HELPER_ CLASS_NAME column of the FLX_BATCH_JOB_SHELL_DTLS table. For example, InterestEODActionSetBatchDataHelper.java
- **11.** Create a **service processor class** with the **service method** which processes the batch application. For example, ActionSetProcessor

The fully qualified class name of this service processor class need to be defined in the **SERVICE_CLASS_NAME** column of the FLX_BATCH_JOB_SHELL_DTLS table.

This processing method defined in this class should be specified in the **SERVICE_METHOD_NAME** column of the FLX_BATCH_JOB_SHELL_DTLS table.

The service method should have two input arguments - ApplicationContext and AbstractBatchData.

If the shell needs to handle the batch exceptions, the service processor class should implement IBatchHandler interface.

Note: The above steps would suffice for creating a batch shell to be run using the new Batch Framework. The Results of the shell will be present in the FLX_BATCH_JOB_SHELL_RESULTS table.

9.5.3 Creation of Procedure Based Shell

In this batch execution (Type "P"), the business logic is provided in the Stored Procedures.

- 1. Create an entry for **Shell Parameters** in the table **FLX_BATCH_JOB_SHELL_ MASTER**. Same as described in the above section.
- Create an entry for Shell Execution Order in the table FLX_BATCH_JOB_ SHELL_DEPEND. Same as briefed in the above section if there is any dependency with any other shell.
- **3.** Create a **function** in Database which contains the Business logic. This function will be used for batch procedure based execution and the signature of the function must have the arguments as shown in the example:

```
CREATE OR REPLACE FUNCTION ap_as_batch_verify

(var_pi_cod_brn_grp_code VARCHAR2,

var_pi_cod_user_no NUMBER,

var_pi_cod_user_id VARCHAR2,

var_pi_dat_process DATE,

var_pi_nam_bank VARCHAR2,

var_pi_cod_stream_id NUMBER,

var_pi_cod_eod_process VARCHAR2,

var_pi_cod_proc_category NUMBER) RETURN NUMBER AS

VAR_L_RETCODE NUMBER;

BEGIN

VAR_L_RETCODE := 0;
```

```
-----1. Init Restart------
 BEGIN
   plog.error('var_pi_dat_process : ' || var_pi_dat_process);
   var_l_ret_code := ap_ba_init_restart(var_pi_cod_eod_process,
                                 var_pi_cod_brn_grp_code,
                                 var_pi_cod_proc_category);
   IF (var_l_ret_code != 0) THEN
    BEGIN
      IF (var_l_ret_code = -2) THEN
        RETURN var_l_ret_code;
      ELSE
        ora_raiserror(SQLCODE, 'Error in executing Init Restart ', 53);
        RETURN 95:
      END IF;
    END;
   END IF;
 END;
-----2.Bisuness Logic-----
...we can write a piece of code ...or a new proc which contain all the business
logic...
------3.Finish Restart-----
 BEGIN
   var_l_ret_code := ap_ba_finish_restart(var_pi_cod_eod_process,
                                   var_pi_cod_brn_grp_code,
                                   var_pi_cod_proc_category,
                                   var_pi_dat_process);
   IF (var_l_ret_code != 0) THEN
    ora_raiserror(SQLCODE, 'Error in executing Finish Restart ', 76);
    RETURN 95;
   END IF;
 END:
 _____
 return 0;
EXCEPTION
 WHEN OTHERS THEN
   ora_raiserror(SQLCODE,
              'Execution of ap_as_batch_verify Failed',
              37);
   RETURN 95;
END;/
```

9.5.4 Population of Other Parameters

The following procedures describe the population of other parameters:

1. Create database credential details for Lock Connection in the jdbc.properties file

Figure 9–6 Population of Other Parameters

O Ja File	sva EE - config/properties/jdbc.properties - Eclipse Erit Navigate Search Project Run Window Help	- F ×
9	ταν παγματέ στατά πορες παις παιροποιούν πουρ γ φριν Ο × Ο ₄ × Π' × Π' ⊗ Δ' φριμ A' × Π Π ⊗ A Ω Ω × Έριν Έριν Φ × Φ × γ ⊕ ×	11 🗉 🖏 🗉 🎄 🕄 🔞
94 🏧 🛃 🕫	<pre>idbcpropettis 3 iditpropettis 3 iditprox.LCCK.CONNFOLDEPY iditprox.LCCK.CONNFOLDEPY iditprox.LCCK.NDEINAME_iddo/FCEDataSourceNIB iditprox.LCCK.USER=orig1 iditprox.LCCK.LDB.DRIVER=oracle.jdbc.OracleDriver iditprox.LCCK.LDB.DRIVER=oracle.jdbc.OracleDriver iditprox.LCCK.LDB.URL=jdbc:oracle:thin:810.180.22.245:1521:DEVD8 iditprox.LCCK.LDB.URL=jdbc:oracle:thin:810.180.22.245:1521:DEVD8 iditprox.LCCK.LDB.URL=jdbc:oracle:thin:810.180.22.245:1521:DEVD8 iditprox.LCCK.LDB.URL=jdbc:tracleDriver iditprox.LCCK.LCCK.LCCK.LCCK.LCCK.LCCK.LCCK.LCC</pre>	

2. Create datasource on the host server where the batch needs to be executed

Home >Summary	of Deployme	nts >Summar	v of JDBC	Data Sources	>LockDatasour	9 0	
ettings for Loc	kDatasouro	e	, 0. 5000		Contractor		
Configuration	Targets	Monitoring	Contro	ol Security	Notes		
General Cor	nection Pool	Oracle	ONS	Transaction	Diagnostics	Identity Options	
Save							
Applications ge provides the co This page enab	t a database nnection to t les you to de	connection f he applicatio fine general	from a da n from its configura	ta source by lo s pool of datab ation options fo	ooking up the da ase connections or this JDBC data	ta source on the J a source.	ava Naming and Directory Interface (JNDI) tree and then requesting a connection. The data source
lame:					LockE	atasource	A unique name that identifies this data source in the WebLogic domain. $\ensuremath{{\rm M}}$ Info
JNDI Name	: ataSourc	eNIB			1		The JNDI path to where this data source is bound. By default, the JNDI nar the name of the data source. More Info
🗌 街 Row Pro	efetch Enab	led					Enables multiple rows to be "prefetched" (that is, sent from the server to the client) in one server access. More Info
Row Prefe	tch Size:				48		If row prefetching is enabled, specifies the number of result set rows to prefetch for a client. More Info
🖰 Stream Ch	unk Size:				256		Specifies the data chunk size for steaming data types. More Info
ave							

Figure 9–7 Population of Other Parameters - General Tab

Figure 9–8 Population of Other Parameters - Connection Pool

← → C ☆ [] 10.180.25.93:700	1/console/co	onsole.portal?	nfpb=t	rue& pa	agel abel	= JdbcData	sourcesJDBCDat	aSourceConfi	ConnectionPoolTabPage&handle=co	om.bea.console.his?		
👻 Jira 🗢 Grok 🗋 UI 💽 IUT 1 💽 ST1	. 🖸 ST2 🦲		onsole S	OBP22I	ит 👻 ог	BPBMARK	NGP Design Debt	🗅 Est Tracker	Dev Tracker	C Other bookmarks		
ORACLE WebLogic Server®	Administration	n Console			-				-	0		
Change Center	🙆 Home Log Out Preferences 🗠 Record Help											
View changes and restarts	Home >Sum	mary of Deployment	ts >Summar	y of JDBC I	Data Sources	>LockDatasou	rce					
No pending changes exist. Click the Release	Settings for	Settings for LockDatasource										
Configuration button to allow others to edit the domain.	Configurati	ion Targets 1	Monitoring	Control	Security	Notes						
Lock & Edit	General (Connection Pool	Oracle	ONS	Transaction	Diagnostics	Identity Options					
Release Configuration	Save											
Domain Structure												
host_domain	main The connection pool within a DPEC data source contains a group of DPEC connections that applications reserve, use, and then return to the pool. The connection pool and the connections within it irrorment irrorment are created when the connection pool is registered, usually when starting up WebLogic Server or when deploying the data source to a new target. Joyments Use this page to define the configuration for this data source's connection pool.								the connections within it			
Decurry reality Diagnostics	idbc:oracle:thin:@10.180.22.						D.22.245:1521:DEVDB The URL of the database to connect to. The format of the URL v. driver. More Info					
	niver Class Name:		ora	acle.jdbc	.xa.client.(DracleXADat	aSource		The full package name of JDBC driver class used to create the physical database connections in the connection pool. (Note that this driver class must be in the classpath of any server to which it is deployed.) More Info			
	街 Propert	ties:						The list of properties passed to the JDBC driver that are used to create				
	user=orig1								physical database connections. For example: server=dbserver1. List each property=value pair on a separate line. More Info			
How do I												
Configure testing options for a JDBC data source												
 Configure the statement cache for a JDBC connection pool 												
Configure credential mapping for a JDBC data source	ta System Properties: The list of System Properties anaec passed to the JBG Christ Hat to create hybrical database connections. For examples server-doo each property=value pair on a separate line. More linfo						IDBC driver that are used ple: server=dbserver1. List e Info					
System Status												
Health of Running Servers												
Failed (0)												
Overloaded (0) Warning (0)	👍 Passwo	ord:	•••		•••••				The password attribute passed to the JDBC driver v database connections. More Info	when creating physical		

3. Enable Node Affinity for Batch Processing (Optional)

This feature can be used for Clustered Database environment. In this feature connections taken by threads are pinned to a particular database node explicitly in order to reduce Cluster Wait events.

4. To enable this feature, set IS_DB_RAC = true in jdbc.properties file and specify the number of DB nodes.

Figure 9–9 Population of Other Parameters - Set IS_DB_RAC

```
41 #Denotes if the data base is running in cluster mode.
42 IS_DB_RAC=true
43 #Denotes the number of nodes in the db cluster.
44 NO_OF_DB_NODES=2
45
```

5. Create a separate data for each node in the cluster. Each of these connections will have the IP of an individual node instead of the SCAN-IP. Specify the data source configuration per node in the cluster in jdbc.properties.

Figure 9–10 Population of Other Parameters - Specify Data

```
109 #Used in Clustered env for pinning connection to stream
110 FCON.BATCH1.CONNPOOLED=Y
111 FCON.BATCH1.JND1.NAME=jdbc/FCBDataSourceN1
112 FCON.LOCK.USER=orig1
113 FCON.LOCK.PASS=A61FB928642DE0130000000000000000
114 FCON.LOCK.LDB.DRIVER=oracle.jdbc.OracleDriver
115 FCON.LOCK.LDB.URL=jdbc:oracle:thin:@10.180.22.245:1521:DEVDB
116 #Used in Clustered env for pinning connection to stream
117 FCON.BATCH2.CONNPOOLED=Y
118 FCON.BATCH2.JNDI.NAME=jdbc/FCBDataSourceN2
119 FCON.LOCK.USER=orig1
120 FCON.LOCK.LDB.PRIVER=oracle.jdbc.OracleDriver
121 FCON.LOCK.LDB.DRIVER=oracle.jdbc.OracleDriver
122 FCON.LOCK.LDB.URL=jdbc:oracle:thin:@10.180.22.245:1521:DEVDB
123
```

9.6 Batch Execution

The user can execute the batch process from the task code EOD10 screen. User needs to select the process category, job type and job code. The corresponding shells get populated in the table below which can be started by clicking on the start/restart button.

User can also monitor the performance by clicking on the Refresh button available in the Category Details section. The execution of the batch takes care of shell dependencies and the dependent shells are run once their dependencies are executed.

Figure 9–11 Batch Execution

← → C	10.180.2	25.249:8001/com	.ofss.fc.ui.vi	ew/faces/m	ain.jspx?_a	afrLoop=153	87030866726108	_afrWindowMode=	0&_adf.ctrl	-state=u5u961f	5_4		\$
🎽 Jira 🛛 🖸	Grok 🗋 UI 🧰 IUT	1 🦲 ST1 💽 ST.	2 🖸 INT [INT Console	👻 ОВР2	210т 👻 оврі	BMARK 👻 NGP De	sign Debt 📄 Est Track	er 📄 Dev Tr	acker		📋 Other b	ookmari
	TFORM					AdminUser, Las	t Login 02-Jan-2013 05	16:29 PM , Posting Date 31	Dec-2013 🕨	Busine	ss Unit BU01 💌 👔	? i 🗢	ڻ <u>د</u>
Account 🔻	Back Office 👻 CAS	A 👻 Channel 👻	Collection •	r LCM - L	Loan 👻 O	rigination 🔻	Party 🔻 Payment A	nd Collection 👻 Secu	rity 🔻 Term	c 🗱 🕏	🖌 🎽 Fast Path		Þ
eot	010												
End of D	ay												
											e	Clear 🛛 Exit	🚔 Print
-													
Categor	y Details												
	Proc	ess Category Forwa	rd Health Check	< 🔍				Category S	itatus Complete	ed			
		Job Type GROU	Р					Process	Date 05-Jan-	2014			
		Job Code BRN_0	SRP_1	Q				Next Process	Date 06-Jan-	2014			
	Catego Click be	ry Start Time U7-Jar	1-2013 14:57:54	+				Category End Last Refresher	Time 07-Jan- Time 07-Jan-	2013 14:58:14 2013 15:02:01			
	P	olling Interval											
	0											Pactart St	tart
E Process	3											Restart Si	
Clear All Fi	lters												
View 🕶	Export To Excel	Detach											
][][
Name Of Si	hell	State	Health	Module Code	Streams	Number Of Streams	Start Time	End Time	Execution Time	Duration	Wait Time	# of Aborts	
Health Chec	kup Shell	complete	1	LN	false	1	2013-01-07 14:57:57	2013-01-07 14:58:00		00:00:20			

Uploaded File Data Processing

In Banks, there are multiple times when the bulk load of data is available in the form of files which needs to be uploaded and processed in the banking application. An example for the same can be salary credit processing. The salary credit data is provided by the organizations in the form of files where employer account needs to be debited and the multiple accounts of the employees needs to be credited for the provided data in the files.

In OBP, file upload and file processing are two independent processes where the upload of file is done as one process and the processing on the uploaded data is done as another process. Every upload provides a unique field for the uploaded file. The processing is then done for each uploaded file and the final status is then provided at the end of the processing in the form of ProcessStatus.

The below section, from the extensibility perspective, provides the detailed understanding of the steps involved in the business logic processing of the files once the files are uploaded from the upload services. After the upload of the data, the data gets populated in the temporary tables in the database with the unique file id, which is then used for the processing of the uploaded file for the required business logic.

In the above mentioned salary credit example, the employer account details (in the form of header records) and the multiple employee account details (in the form of detail records) can be uploaded in OBP through the file upload, functionality which can then be processed for debiting the employer account and crediting the multiple salary accounts of the employees.

The framework of the uploaded file processing is shown in the sequence diagram below:



Figure 10–1 Uploaded Data File Processing Framework

From the implementation perspective, the following sections describe the configuration and processing of uploaded file.

10.1 Configuration

The configuration part of the uploaded file processing requires definition of the following components.

10.1.1 Database Tables and Setup

In case of file processing, there is one master table and individual record process tables for the recordType.

• FLX_EXT_FILE_UPLOAD_MAST

Column Name	Description
COD_FILE_ID	This defines the primary key identifier as file id for each specific file.
COD_XF_SYSTEM	This identifies the system to which the file type is associated. This indicates that the file is received from or sent to the particular system indicated by the system code.
FILE_TYPE	This identifies the type of file that is being uploaded. For every file type the format is defined. The file type can be like TXN
NAM_HOFF_FILE	Name of the uploaded file.
TXT_NRRTV	File Narration for the uploaded file.
COD_ORG_BRN	This stores the originating branch code from where the file is uploaded.
CTR_BATCH_NO	This identifies the batch number of the file upload. This is generated internally.
DAT_FILE_PROCESS	The process date as specified while uploading a file.
COD_FILE_STATUS	Indicates the current status of the file.
DAT_FILE_UPLOAD	Indicates when the file was uploaded.
DAT_TIM_PROC_START	The start time indicates the time the processing starts.
DAT_TIM_PROC_END	The end time indicates the time the processing ends.
DAT_FILE_REVERSE	Indicates when the file was reversed.
CTR_TOTAL_REC	This value indicates the total records in the file.
CTR_PROCESS_REC	This Value indicates the number of records processed for a file.
CTR_REJECT_REC	This Value indicates the number of records rejected for a file.
FILE_SIZE	This value indicates the size of the file in bytes.
COMMENTS	The file Comments for the uploaded file if the processing fails
FILE_CHECK_SUM	This column is used to store check sum of the file
FROM_ODI	This flag is used to indicate whether upload is happening from ODI
CURR_RECORD_TYPE	This column denotes the current record type being processed, updated after every record Type is successfully processed

Table 10–1 FLX_EXT_FILE_UPLOAD_MAST

- FLX_EXT_<<Process>>_HEADERRECDTO e.g. FLX_EXT_SALCREDIT_ HEADERRECDTO
- FLX_EXT_<<Process>>_DETAILRECDTO e.g. FLX_EXT_SALCREDIT_ DETAILRECDTO

The field and record Id together as the key forms the record identifier in the record tables. The mandatory fields in the record tables are mentioned below. The additional required fields should be defined as the additional columns in the record tables.

Column Name	Description			
RECORDID	This defines the primary key identifier as record id in the table. This is generated for every record.			
FILEID	This is the primary key identifier as file id for the specific file.			
RECORDTYPE	The type of record; possible values 'H', 'D' and 'F'			
RECORDNAME	Name of the record type; possible values 'Header', 'Detail' and 'Footer'			
DATA	Stores the complete data of each row of the file. This is populated for inquiry purposes that the user can view the contents of the record as it was read from the file.			
LENGTH	Total length of DATA. This value is populated after the record is parsed.			
COMMENTS	Comment update at the time of GEFU Upload and Processing of record			
RECORDSTATUS	List of Record Status : 1-UPLOADED, 2-FAILED_UPLOAD, 3-CANCELLED, 4-INPROGRESS, 5-PROCESSED, 6-FAILED_PROCESS, 7-REVERSED, 8-FAILED_ REVERSED, 9-ABORTED, 10-MARKED_FOR_PROCESS			
DATE_RUN	This column holds the value of batch job's run date			
SEQ	This column holds the value of batch job's sequence number			
PROCESS_RESULT	This column holds the value of batch job process result			
ERROR_CODE	This column holds the value of batch job's error code			
ERROR_DESC	This column indicates the Error Description			
BRANCH_CODE	This column holds the branch code of the branch			
BRANCH_GROUP_ CODE	This column holds the value of branch Group code			

Table 10–2 Mandatory Fields in Record Tables

FLX_EXT_FILE_PARAMS

This table contains the information about the file definition template which is used to define the handlers, DTO and other details required for the processing of the uploaded file.

Column Name	Description
COD_XF_SYSTEM	This identifies the system to which the file type is associated. This indicates that the file is received from or sent to the particular system indicated by the system code.
FILE_TYPE	This identifies the type of file that is being uploaded. For every file type the format is defined. The file type can be like TXN
NAM_XF_SYSTEM	Name of the system to which the file type is associated. This indicates that the file is received from or sent to the particular system indicated by the system code.
NAM_FILE_TYPE	This is name of the type of file that is being uploaded. For every file type the format is defined. The file type would be like PYMT (Payment File) or SAL (Salary Upload).
NAM_UPLOAD_TMPL	XFF file definition template name
FLG_OUTPUT_REQD	Once the processing of all the records is complete, a check is made if its value is 'Y' and then the response file is generated accordingly.
FLG_FILE_ TRANSACTIONAL	Used to decide, whether File level validation is required or not.
CTR_COMMIT_SIZE	Used to commit records in batch while processing, so it's the batch size.

Table 10–3 FLX_EXT_FILE_PARAMS

Column Name	Description
RELATIVE_PATH	If provided, this searches for xff file in the path: base_folder/folder_name_ mentioned_here.
COD_ADHOC_ REQUEST_CLASS	Adhoc request class name
CTR_UPLOAD_ COMMIT_SIZE	Used to commit records in batch while validation, so it's the batch size.
FLAG_DUPLICATE_ FILE_CHECK	This flag is used to indicate whether duplicate file check is required or not
FLAG_FROM_ODI	This flag is used to indicate whether upload is happening from ODI

Table 10–3 (Cont.) FLX_EXT_FILE_PARAMS

• FLX_BATCH_JOB_SHELL_DTLS

This table contains the information about the batch processing with bean based shell mechanism as described in the 'Batch Framework Extension' section. The sample values are provided below:

Table 10–4 FLX_BATCH_JOB_SHELL_DTLS

Columns	Description	Sample Values		
COD_SHELL	A unique code for batch shell. For example, 'upld_batch_shell_ <processtype>'</processtype>	upld_batch_shell_SalCredit		
SHELL_NAME	Name for batch shell	GEFU Processing Shell For Salary Credit		
SHELL_ DESCRIPTION	Description about the batch shell.	GEFU Processing Shell For Salary Credit		
COMMIT_ FREQUENCY	Commit frequency	100		
FLG_RECOVERY_ MODE	Recovery mode - ON / OFF	Y		
FLG_STREAM_TYP	Type of stream : 'S' - fixed no of streams, 'R' - fixed no of rows, 'N' - no streams	S		
STREAM_COUNT	No of streams for the batch shell. Applicable only for StreamType as 'S' or 'R'	2		
INPUT_DRV_NAME	Fully classified class name mapped to the driver table	com.ofss.fc.entity.upload.AbstractRecordDT O		
INPUT_SHELL_ PARAM	Name for the shell parameter	AbstractRecordDTO		
SERVICE_CLASS_ NAME	Fully classified class name - starting point of the business logic execution	com.ofss.fc.upload.processor.batch.BatchReco rdProcessor		
SERVICE_ METHOD_NAME	Method name of the service	processRecord		
DRV_POP_PROC_ NAME	Defines the name procedure used for driver table population	ap_gefu_pop_drv_gefu_rec		

Columns	Description	Sample Values
FLG_PROCESS_ TYPE	RBP (Recoverable Batch Process) if any records fails in batch, it will continue and execute rest of the records in the stream or SBP (Simple Batch process) it will abort the stream	RBP
HELPER_CLASS_ NAME	Helper class for caching big queries	com.ofss.fc.upload.processor.batch.GEFUBatc hJobHelper
BATCH_NO	Batch number for the shell	1

Table 10–4 (Cont.) FLX_BATCH_JOB_SHELL_DTLS

10.1.2 File Handlers

File Handler class is written for processing of the uploaded file and should extend the AbstractFileHandler. The class name of the File Handler is mentioned in the File Definition XML. In this class, the following abstract methods should be implemented:

- isValid(): To check if the particular uploaded file is valid. Validations such as, is the file uploaded duplicate or not, or are the header details valid or not are done as part of file level validations.
- processFile() : To write the actual processing business logic where the functionality is implemented, if required, or else a default blank implementation is executed.

Figure 10–2 File Handlers

🕑 Java EE - com.ofss.fc.extensibility.samples.upload/src/com/ofss/fc/extensibility/upload/salarycredit/handler/SalCreditFileHandler/java - Eclipse 🕞					
File Edit Source Refactor Navigate Search Project Run Window Help					
📑 🕶 🗒 💼 🕸 🕶 💽 🕶 💊 🗰 💷 📾 🐼 🗩		.e =	5 😿 🖏 + 68 + 129 🖨 🖉 + 1 9 128 19 📝 🕸 🗐 🖬 📲 👷 + 🏹 + 🏷 🔶 + -> + 1 🗃		
			Ouick Access	: •	
Project Explorer 💥 📄 🔄 👘 🍸 🖓 🗋		SalCr	ditFileHandler.java 🙁 🖹 SalCreditFileDefn.xml 🛛 💭 DetailRecHandler.java		
▲ 🚰 com.ofss.fc.extensibility.samples.upload		⊕ *	Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.	*	
🔺 🞥 src		ра	kage com.ofss.fc.extensibility.upload.salarycredit.handler;		
com.ofss.fc.extensibility.upload.adapter		⊕im	port java.util.logging.Level;		
III InploadDDAAdapter.java					
com.ofss.fc.extensibility.upload.salarycredit.dto		pu	plic class SalCreditFileHandler extends AbstractFileHandler {		
DetailRecDTO.java			, ·		
DetailRecDTOKey.java		æ	<pre>public static final String THIS_COMPONENT_NAME = "SalCreditFileHandler"; * This is an instance variable and not a class variable (static or static final). This is</pre>	require	
HeaderRecDTO.java		Ű	<pre>private Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(THIS_COMPONENT_NAMA</pre>	E);	
I HeaderRecDTOKey.java		Ð	* Create instance of multi entity logger.		
Com.orss.rc.extensibility.upioad.salarycredit.nandier	S		private static String <u>IS PROC USED</u> = "PROCEDURE";		
Betainteerint			private long returnValue;		
SalCreditFileHandler.java		۲	* @see com.ofss.fc.bh.upload.AbstractFileHandler#isValid(String fileId)		
JRE System Library [JavaSE-1.6]		Θ	<pre>public boolean isValid(String fileId) {</pre>		
com.ofss.fc.app.xface.jar - D:\Work\StagingArea\obp.host			<pre>if (logger.isLoggable(Level.FINE)) { logger_fine(formatter_formatMessage(THIS_COMPONENT_NAME"isValid() entry"));</pre>		
com.otss.tc.appcore.dto.jar - D:\Work\StagingArea\obp.n			<pre>>> competition matched and costage(mail_competition, ', ', ', ', ', ', ', ', ', ', ', ', ',</pre>		
com.ofss.fc.common.iar - D:\Work\StagingArea\obp.host	M		<pre>String METHOD_NAME = "isValid"; if (lagger islagger)] flucture for the state of the state o</pre>		
com.ofss.fc.datatype.jar - D:\Work\StagingArea\obp.host.			<pre>logger.fine(formatter.formatMessage(THIS COMPONENT NAME, "", "isValid() exit"));</pre>		
com.ofss.fc.enumeration.jar - D:\Work\StagingArea\obp.l			}		
com.ofss.fc.framework.dto.jar - D:\Work\StagingArea\obj			return true;		
com.ofss.fc.ann.adanter.internal.interface.iar - D:\Work\StagingArea\obp.nost.dom		_			
com.ofss.fc.framework.batch.jar - D:\Work\StagingArea\c		•	//This is a dummy method which is not to be removed.		
com.ofss.fc.framework.domain.jar - D:\Work\StagingArea			}		
com.ofss.fc.module.commonservice.jar - D:\Work\Stagini		æ	* @see com ofes fo bb unload AbstractFileHandler#processFile(String fileId)		
com.ofss.fc.upload.core.jar - D:\Work\StagingArea\obp.h		Θ	public ProcessStatus processFile(String fileId) {		
Velocity-1.0.2-dep.jar - D:\Work\StagingArea\obp.thirdpail			<pre>if (logger.isLoggable(Level.FINE)) { learner fire(formation formation formati</pre>	· · ·	
v 🖉 comg			<pre>iogger.time(formaccer.formacmessage(fris_component_wave, , processFile() entry }</pre>));	
			//strmethod used "PROCEDURE"		
	Và		<pre>String MEIHOD_NAME = "processile"; if (logger.isLoggable(Level.FINE)) {</pre>		
			<pre>logger.fine(formatter.formatMessage(THIS_COMPONENT_NAME, "", "processFile() exit")</pre>);	
			} return ProcessStatus fromString(String valueOf(returnValue));		
			}		
			* 0		
	8	•	public ProcessStatus reverseFile(String fileId) {[]		
		_			
	S	⊕ ⊕	wese com.orss.rc.on.upload.AbstractFileHandler#populateSummaryTable(String fileId)[]		
		_			
		Ð	public void CLASS() {[]		
		Ð	<pre>public void PROCEDURE() {</pre>		
		}			
4 III >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>		•	III	•	
	1				

10.1.3 Record Handlers for Both Header and Details

This class provides the methods for record level validations and processing. It should extend the AbstractRecordHandler. The class name of the Record Handlers are also mentioned in the File Definition XML. The following abstract method needs to be implemented in this class:

- isValid(): To check if the particular uploaded record is valid for the processing purpose
- process(): To write the actual processing business logic where the functionality is implemented. It is called once the file is successfully validated.





10.1.4 DTO and Keys Classes for Both Header and Details

This is a persistent class for the particular process. This class provides the fields which represents the characteristics of the record data. This class is defined for each record type of a file.
Figure 10–4 DTO and Keys Classes for Both Header and Details - HeaderRecDTOKey

Java EE - com.ofss.fc.extensibility.samples.upload/src/com/ofss/fc	/extensibility/upload/salarycredit/dto/HeaderRecDTO.java - Eclipse	
<u>File Edit Source Refactor N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un 1	<u>₩indow</u> <u>Help</u> · · · · = = = :::::::::::::::::::::::::	
Project Explorer 🛛 📄 🗳 🗖 🗖	1 DetailRecHan 1 HeaderRecHan 1 HeaderRecDT 🔀 1 IUploadDDAA 1 HeaderRecDT	"2 E
 Project Explorer 12 com.ofss.fc.extensibility.upload.adapter f) UploadDDAAdapter.java f) UploadDCAdapter.java f) UploadDCAdapter.java f) DetailRecDTO/sya f) DetailRecTorouter.java f) DetailRecTorouter.java f) DetailRecTorouter.java f) DetailRecTorouter.java f) DetailRecTorouter.java f) DetailRecTorouter.java f) DetailRecthandler.java f) SafCreditFileHandler.java f) Comofss.fc.appcore.jar - D.Work/StagingArea/obp.hcs f) Comofss.fc.app.apdapter.internal.interface.jar - DNWork/StagingArea/obp.hcs f) Comofss.fc.app.adapter.internal.interface.jar - DNWork/StagingArea/obp.hcst.dc f) Comofss.fc.framework.domain.jar - D.Work/StagingArea/obp.hcst.dc f) Comofss.fc.app.adapter.internal.interface.jar - D.Work/StagingArea/obp.hcst.dc f) Comofss.fc.app.adapter.internal.interface.jar - D.Work/StagingArea/obp.hcst.dc f) Comofss.fc.app.adapter.internal.interface.jar - D.Work/StagingArea/obp.hcst.dc f) Comofss.fc.app.adapter.internal	<pre>Quick Access Quick Access Quick Access DetailRecHan</pre>	
	<pre></pre>	
	}	
۰ III ۲	4	

Figure 10–5 DTO and Keys Classes for Both Header and Details - AbstractDTORec

10.1.5 XFF File Definition XML

The xff file contains all the information about the different record type DTOs, the fields in those DTOs and the handlers pertaining to the uploaded file. The name of the xff file is mentioned in the FLX_EXT_FILE_PARAMS table. The file details are read from each tag in xff file and interpreted as described below in the table. The record element can occur N number of times based on number of record types present, for example if a particular upload has three record types Header, Detail and Trailer then there will be three elements for Record, each describing the three record types.

There are two one-to-many relationship in the file definition xml file:

- One 'File' element can have many 'Record' elements, depending upon the number of recordType present for this upload.
- One 'Record' element can have many 'Field' elements, depending upon the number of fields present for this recordType of upload.

Elements	Attributes	Description
File		Contains all details about the FileHandler, there is only once
		occurrence of this element.
	fileName	This denotes logical name of the file
	validationClassName	Fully qualified name of the FileHandler class
	encryptionClass	This denotes the name of the class that is used for encryption (optional).
	charSet	This denotes the Charset of the file.
	delimiter	This denotes delimiter coming in the file.(optional)
	comments	This is used to store comment on the file.(optional)
	lengthInBytes	This Boolean variable is used to denote whether the file's length has to be calculated in bytes.
	xffSystem	Name of xff file system, name should be same as mentioned in COD_ XF_SYSTEM in table FLX_EXT_FILE_PARAMS
	fileType	Name of file type, name should be same as mentioned in FILE_TYPE in table FLX_EXT_FILE_PARAMS
Record		Child element of "File" can have any number of occurrences depending upon number of RecordType for a particular Upload.
	recordHandlerClassName	Fully qualified name of the Handler class for this RecordType
	recordType	This denotes record type which can be "Header", "Detail" or "Trailer"
	streamingAllowed	Indicates if the streaming is allowed for the record; Possible values are true or false
	dtoClassName	Name of DTO for this particular recordType
	recordName	Name of this record.
	multiplicity	This denotes whether this record type will appear only once in the file or multiple times. Value of this field will be either 1 (for only once) or -1 (for multiple times)
	maxFields	This denotes the maximum number of fields coming in the record type.
	comments	This stores comments.(optional)
	maxLogicalRecords	This denotes maximum number of records that may come of this record type.
	parent	
	lastFieldOfVariableLength	This denotes whether the last field of the record is variable or not. This value can be either "true" or "false"
Field		Child element of "Record" can have as many occurrences as the number of fields in a particular recordType
	name	Name of the field
	type	This denotes field type. E.g.:- 'CHAR', 'NUMBER' and so on.
	length	Length of field
	format	This denotes format of the field
	recordIdentifier	This denotes whether this field is used to identify the record. Value of this field can be either true or false.
	nullable	This denotes whether this field can be null or not

Table 10–5 XXF File Definition XML

Elements	Attributes	Description
defaultValue Default value		Default value of this field if any.
comments		This stores the comment on the field. (optional)
	crossReferenceID	If another field wants to refer to this field then this id will be used.

Table 10–5 (Cont.) XXF File Definition XML

Figure 10–6 XXF File Definition XML

Java EE - config/xff/SalCreditFileDefn.xml - Eclipse						
File Edit Navigate Search Project Run Desi	gn Window Help					
🔁 🕶 🖫 🕼 🖻 🕸 🕶 💽 🕶 隆 💌	비 🗏 21 (3. () 26 🗏 📆 🔞	• 63 • 😂 🗁 🔗 • 🚱	4 2 8 2 .	- 🖓 - 🏷 🔶 -	· 🔿 🔻 🛃	
					Quick	Access
Project Explorer 💥 📃 🗌	🔊 SalCreditFileHandler.java	alCreditFileDefn.xml 🔀				- E
Project Explorer S2 Project Explorer S2 Com.ofss.fc.extensibility.samples.upload Com.ofsst.fc.extensibility.samples.upload CompositeFileDefn.xml NPACProvsnFileDefn.xml StafcraftifiteDefn.xml BULD.INFO CompositeNameMapping.properties CompositePickList.properties CompositePickList.properties CompositePickList.properties Nonweblogic.logging.xml Preferences.xml StafdowDefn_element.csv weblogic.logging.xml	SalCredifielHandlerjava SalCredifielHandlerjava vilidationClassName=' encryptionClassName=' recordIdentific crossReferent crossReferent	alCreditFileDefnxml 😒	Larycredit.handLa "," comments="" leType="TXN" d.pc.salarycredit ycredit.dto.Head ields="2" comment fv=riableLength the"1" format="" se" defaultValue " length="16" format Lse" defaultValue format="" ue" defaultValue format="" ue" defaultValue format="" ue" defaultValue format="" ue" defaultValue format="" ue" defaultValue format="" se" defaultValue th="16" format=" Lse" defaultValue th="16" format=" se" defaultValue format="" se" defaultValue format="" se" defaultValue format="" se" defaultValue format="" se" defaultValue format="" " length="3" for lse" defaultValue format="" ",,,,,,,	er.SalCreditfil t.handler.Heade erRecDTO" ts="" ="false"> "comments=" "mat="" ="" comments=" "" er" comments=" t.handler.Detai liRecDTO" ts="" er" comments=" "formets=" "" formets="" er" comments=" "" t.handler.setai ""	eHandLer" rRecHandLer" LRecHandLer"	
	0 errors, 15 warnings, 0 others					
	Description	Resource	Path	Location	Туре	
	🔊 🔺 Iava Ruild Dath Drohlems (1 ite	m				
🗴 SalCreditFileDefn.xml - config/xff					1	

10.2 Processing

Processing of an uploaded file is done on two levels, one on file level and the other on Record level. The processing is initially triggered when a message is sent on to a JMS Queue. The message is then picked up by an MDB which parses the message into a key value pair, and then passes it on to the FileProcessor by passing the processor type as an input. Based on the processor type, that is, header or detail record, the file processor initiates respective processing by invoking specific business logic written as file or record level handlers.

The processing of the business logic to different Service APIs of different modules are carried in the handler classes of the records. The processForRecordType() method of

the FileProcessor invokes the respective handler classes that is, if the Header section is being processed, it invokes the HeaderHandler class.

As per the process, the headers are processed first and then the details records. Each and every record is processed individually. As soon as a file is picked for processing, its status is changed to InProgress so that the same file is not picked by any other process for processing. Individual records are processed based on its record type.

10.2.1 API Calls in the Handlers

The API calls of different exposed application services are called from the handlers. The respective method call from the adapter will return the response object which can be further used for another adapter call as the input value or for the validation purpose. In the following example, it is shown that the salary account is debited for the user and then the returned response summary is used for validation purpose before raising the accounting for that account.

```
<Responsel>=Adapter1.<method call>(<method parameters>)

If(<Validation on Responsel>) {

<Response2>=Adapter2.<method call>(<method parameters containing Responsel>) }

Example:

executionResponse = adapter.debitSalaryAccount()

if(executionResponse.getSummary().getIsSuccessful()) {

adapter.raiseAccounting(); }
```



Java EE - com.ofss.fc.extensibility.samples.upload/src/com/ofss/fc.	/extensibility/upload/salarycredit/handler/HeaderRecHandler.java - Eclipse	
<u>File Edit Source Refactor Navigate Search Project Run</u>	<u>Window</u> Help	
🖸 ▼ 🗒 🕼 🗁 ; 🅸 ▼ 🕗 ▼ 💁 ▼ ; 🖉 🕨 💷 🖉 3	▙ ۞k ≒ 死:; ゐ ▾ ♂ ▾: ❷ @ ৵ ▾: ♥ : ♣:♥ Ø ♥ ॿ ॿ :! : 뵘 ▾ 줘 ▾ や 수 ▼ → ▼ ヹ	
	Quick Access	EP 🔁
Project Explorer 🛛 📄 🔄 🖓 👘	D SalCreditFil SalCreditFil D DetailRecHan D HeaderRecHan D HeaderRecDT N P HeaderRecDT N P	- 6
 Project Explorer B Comofoss fc.extensibility.samples.upload Sirc Comofoss fc.extensibility.upload.adapter Di Upload/DAdapter.java Upload/DAdapter.java Comofoss fc.extensibility.upload.salarycredit.dto DetailRecDTO(ky.java DetailRecDTO(ky.java DetailRecHondler.java Comofoss.fc.extensibility.upload.salarycredit.handler DetailRecHondler.java Comofoss.fc.extensibility.upload.salarycredit.handler DetailRecHondler.java Comofoss.fc.extensibility.upload.salarycredit.handler DetailRecHandler.java SalCreditFileHandler.java SalCreditFileHandler.java SalCreditFileHandler.java Comofos.fc.app.cre.jar D:Work/StagingArea\obp.hc Comofos.fc.app.cre.jar D:Work/StagingArea\obp.hc Comofos.fc.app.cre.jar D:Work/StagingArea\obp.hc Comofos.fc.app.adapter.internal.interface.jar - D:Work/StagingArea\obp.hc Comofos.fc.framework.dtojar - D:Work/StagingArea\obp.hc Comofos.fc.framework.dtojar - D:Work/StagingArea\obp.hc Comofos.fc.framework.dtojar - D:Work/StagingArea\obp.hc.dt <	<pre>SalCredifFil. () DetaiRecHam. () HeaderRecHam. () HeaderRecDT. ''' try {</pre>	□ [1] [1] [1] [1] [1] [1] [1] [
	return processStatus; }	
	⊕ public void recProcStubCreator() {[]	-
4 III >	✓ III	•

10.2.2 Processing Adapter

The processing adapters needs to be implemented for invoking the required application service API. In the example, the new methods as creditSalaryAccount(), debitSalaryAccount() and raiseAccounting() are implemented by the user based on their requirements.

Figure 10–8 Processing Adapter

Java EE - com.ofss.fc.extensibility.samples.upload/src/com/ofss/fc.	/extensibility/upload/adapter/IUploadDDAAdapter.java - Eclipse	
<u>File Edit Source Refactor Navigate Search Project Run</u>		
	ĸ○ư≂vĩe@▼®▼₽₽₽₽₽₩₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽₽	
	Quick Access	
Project Explorer 💥 📄 🔄 🖻 🗆	🕼 SalCreditFil 👔 SalCreditFil 👔 DetailRecHan 👔 HeaderRecHan	- 6
 2 com.ofss.fc.extensibility.samples.upload 3 5 src 	package com.ofss.fc.extensibility.upload.adapter;	^
 isrc implicatDDAAdapterjava imp	<pre>pitage tom.ofss.fc.app.accounting.dto.AccountingEventAmountTagDetailsDTO; public interface TuploadDDAAdapter { public static final String GEFU_DDA_ADAPTER = "GEFU_DDA_ADAPTER"; /** * @param accountId *@param serviceCode *@param accountId *@param accountId *@param transactionResponse debitSalaryAccount[String accountId, String bsbCode, throws fatalException */* *@param transactionReferenceNumber *@param transactionReferenceNumber *@param transactionReferenceNumber *@param transactionReferenceNumber *@param transactionReferenceNumber, *fring transactionRefere</pre>	String eve String ever
۰ ا	۲	

10.3 Outcome

In case of header or footer, there is only one Record for these record types, hence based on Record Level Status returned, the processing status is set, if RecordLevelStatusType is SUCCESS or WARNING, the PROCESSING_STATUS will be marked as SUCCESS else FAILURE.

In case of detail records, processing status is decided based on the criteria that is, if NumberOfRecords with record processing status as FAILED is equal to totalNoOfRecords then overall ProcessStatus is FAILED or if less than totalNoOfRecords then overall ProcessStatus is WARNING and if zero then overall ProcessStatus is SUCCESS. Also, in case there is error in insertion of any record to the working table then overall ProcessStatus is FAILED.

Each record on processing can have any one of the three process status. If process status is success it moves to the next record. If process status is warning then it moves to the next record but marks the record as failed. If process status is failure then an Exception is raised and the file is marked as Failed.

Status Name	Value	Description
SUCCESS	0	Processing of this record is a success. Further record processing should continue.
FAILURE	1	Processing of this record has failed. Further record processing should not continue.
WARNING	2	Processing of this record has failed. Further record processing should continue.

Table 10–6 Process Status

On successful processing, the record will get persisted into the respective table and return a status of '5' to the invoked method.

But, in case of failure, the status is returned as '6' for that particular record and it continues with the next record for processing. Also the exceptions raised during a failure can be appended into the "comments" column of the respective table.

10.4 Failure/Exception Handling

There can be processing failure in case of any validations failure caused by the service. In case of any exceptions raised, it will be handled in the handler class.

While invoking an API when the SessionContext variables are not passed properly it would result in null. 'Invalid user id' will be added in the comments column and the processing will not happen.

The exceptions raised during processing can be logged into the comments column of the respective table by calling the setErrorMessage() method. In case of process failure in file handling, this method can also be invoked from inside the catch block of the processFile() method:

```
this.setErrorMessage(errorMessage);
processStatus = ProcessStatus.FAILURE;
```

11

Alerts Extension

OBP has to interface with various systems to transfer data which is generated during business activities that take place during teller operations or processing. OBP Application is, therefore, provided with the framework which can support on-line data transfer to interfacing systems.

The event processing module of OBP provides a mechanism for identifying executing host services as activities and generating or raising events that are configured against the same. Generation of these events results in certain actions that can vary from dispatching data to subscribers (customers or external systems) to execution of additional logic. The action whereby data is dispatched to subscribers is termed as *Alert*.

The following sections provides an overview of what the developer needs to do in order to add a new *Activity* and an *Event* which will be raised on execution of the said that activity. We will be using a sample activity and event to illustrate the steps.

Use Case: In the *Party -> Contact Information -> Contact Info* screen, user can create or update the contact details for a party. This screen has many attributes like *telephone number*, *email*, *do not disturb info* and so on. We will be registering this *update* transaction as an *Activity* and creating *Events* which will be raised on this activity.

11.1 Transaction as an Activity

This section describes how existing or new online transactions can be supported and recognized as activity for the events that are setup in the system with action, subscriber and dispatch configuration already in place. A transaction can be either financial or maintenance executing in the application server middleware host environment. This kind of setup is particularly useful when we have external systems like CEP, BAM to which data needs to be dispatched online.

The procedure for creating activities and events for a *financial* transaction is a subset of the same for a *maintenance* transaction. The aforementioned use case describes a maintenance transaction.

11.1.1 Activity Record

You will need to create a record for the activity in the table FLX_EP_ACT_B which stores all the recognized activities. This table has the following columns:

Column Name	Use	Example		
COD_ACT_ID	The unique activity id for the activity. This id will be used in the activity - event mapping as well	'com.ofss.fc.app.party.service.contact.ContactPoi ntApplicationService.updateContactPoint.dndIn fo'		
TXT_ACT_NAME	Activity name	'ContactPointApplicationService.updateContact Point.dndInfo'		
TXT_ACT_DESC	Meaningful description of the activity	'DND Info Change'		
MODULE_TYPE	Module code for the module of which the transaction is a part off	'PI'		
CREATED_BY	User id of the user creating this record	'SYSTELLER'		
CREATION_ DATE	Creation date of this record	to_date('20110310', 'YYYYMMDD')		
LAST_ UPDATED_BY	User id of the user last updating this record	'SYSTELLER'		
LAST_UPDATE_ DATE	Last update date of this record	to_date('20110310', 'YYYYMMDD')		
OBJECT_ VERSION_ NUMBER	Version number of this record	1		
OBJECT_STATUS	Status of this record	'A'		

Table 11–1 FLX_EP_ACT_B

Sample script for Activity Record:

Figure 11–1 Sample script for Activity Record

```
--for insertion of activity record

DELETE FROM FLX_EP_ACT_B WHERE COD_ACT_ID =

'com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint.dndInfo';

INSERT INTO FLX_EP_ACT_B (COD_ACT_ID, TXT_ACT_NAME, TXT_ACT_DESC, MODULE_TYPE, FLG_IP_REQD, FLG_OP_REQD, FLG_LOG_REQD, TXT_LOG_CLASS,

CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATE_DATE, 00JECT_VERSION_NUMBER, 00JECT_STATUS)

VALUES ('com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint.dndInfo',

'ContactPointApplicationService.updateContactPoint.dndInfo', 'DND Info Change', 'PI', null, null, null, null, 'SYSTELLER', to_date

('20110310', 'YYYYMMDD'), 'SYSTELLER', to_date('20110310', 'YYYYMMDD'), 1, 'A');
```

11.1.2 Attaching Events to Activity

Recognized events can be attached to recognized activities. The mapping in this case can be many-to-many viz., an activity can raise multiple events and an event can be raised by multiple activities.

11.1.3 Event Record

You will need to create an event record in the table FLX_EP_EVT_B which stores all the recognized events. This table has the following columns:

Column Name	Use	Example
COD_EVENT_ID	The unique event id for this event. This id will be used in the activity - event mapping as well	'PI_UPD_DND_INFO'
TXT_EVENT_ TYP	The type of event	'ONLINE'
TXT_EVENT_ DESC	Meaningful description for the event	'DND Info Updated'
EVENT_ CATEGORY_ID	The category code for this event	2

Table 11–2 FLX_EP_EVT_B

Sample script for Event Record:

Figure 11–2 Sample script for Event Record

```
--for insertion of event record

DELETE FROM FLX_EP_EVT_B WHERE COD_EVENT_ID = 'PI_UPD_DND_INFO';

INSERT INTO FLX_EP_EVT_B (COD_EVENT_ID, TXT_EVENT_TYP, TXT_EVENT_DESC, EVENT_CATEGORY_ID)

VALUES ('PI_UPD_DND_INFO', 'ONLINE', 'DND Info Updated', 2);
```

11.1.4 Activity Event Mapping Record

You will need to create an activity event mapping record in the table FLX_EP_ACT_EVT_B which stores the mapping between all activities and events. This table has the following columns:

Column Name	Use	Example
COD_ACT_ID	The unique activity id as specified in the activity table	'com.ofss.fc.app.party.service.contact.Cont actPointApplicationService.updateContact Point.dndInfo'
COD_EVENT_ID	The unique event id as specified in the event table	'PI_UPD_DND_INFO'
TXT_ACT_EVT_DESC	Meaningful description for the activity event mapping	'DND Info Updated'
TXT_EVT_TYP	The type of event	'OTHER'
TXT_ACT_EVT_TYP	The type of activity event mapping	'ONLINE'

Table 11–3 FLX_EP_ACT_EVT_B

Sample script for Activity Event Mapping Record:

Figure 11–3 Activity Event Mapping Record

--for insertion of activity - event mapping DELETE FROM FLX_EP_ACT_EVT_B WHERE COD_ACT_ID = 'com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint.dndInfo' AND COD_EVENT_ID = 'PI_UPD_DND_INFO'; INSERT INTO FLX_EP_ACT_EVT_B (COD_ACT_ID, COD_EVENT_ID, TXI_ACT_EVT_DESC, TXI_EVT_TYP, TXI_ACT_EVT_TYP) VALUES ('com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint.dndInfo', 'PI_UPD_DND_INFO', 'DND Info Updated', 'OTHER', 'ONLINE');

11.1.5 Activity Log DTO

In order to transfer activity data to the actions defined for the event, you need to develop data objects to contain the activity data. The DTO should implement the interface *com.ofss.fc.xface.ep.dto.IActivityLog*. Module specific activity log DTO's which already implement the *IActivityLog* interface are present. These DTO's contain the application specific and module specific activity data. You can extend the module's DTO class and add the transaction specific activity data.

For party module, the class *com.ofss.fc.app.party.dto.alert.IndividualPartyTypeDatalogDTO* is one of the classes that implement the *IActivityLog* interface. For the aforementioned activity, the activity log DTO can be as follows:

Figure 11–4 Activity Log DTO



11.1.6 Alert Metadata Generation

This section describes the different types of alert metadata generation.

Metadata Generation

To generate metadata for alerts you need to have plugin.

Once you have plugin you need to set properties in preferences in windows tab for Service Publisher, Service Deployer and Workspace Path.

- **1.** Go to your DTO class and right-click that class and click the following : *Oracle Banking Platform -> Generate DTO Metadata*.
- 2. This will generate the insert scripts for following two tables:
 - FLX_MD_DATA_DEFN
 - FLX_MD_FIELDS_DEFN

These scripts will be generated in your config folder by default. The path of this script is :

WorkspaceDirectory -> config -> meta-data-scripts -> incr-meta-data.log

Figure 11–5 Metadata Generation

· 🖂 🕲 🖻	b · O · Q · # 0	- 😕 🗁 🛷 🖓 📝	P 🛛 🗊 🗐 🔹	i + ⊕ + ⊕ +	🗈 🛃 Java 📌 Java
OfferActiv	ityOTOjava E3				- ø
* * Copy	yright (c) 2012, Oracle and/	or its affiliates. All ri	ghts reserved.[]		
 Copy package import /** Thii RE3 rel: imode (builty) (builty)	<pre>yright (c) 2012, Oracle and/ e com.ofss.fc.app.origins java.util.logging.ievel; s class represents 00 for for Ores 0,000 for ared to all this events for order for event processin thor Virendra0 pe(manespace = "http://ac ctlass OfferactivityOTO e " Generated SerialVersion</pre>	or its affiliates. All rå Undo Executile Sove Open Declastion Open Type Hearchy Open Gal Hierarchy Open Gal Hierarchy Open With Show In Cut	Ctri-Z Ctri-Z Ctri-S F3 F4 Ctri-Att-H Att-Strit-B Ctri-X Ctri-X Att-Strit-W Att-Strit-W	n,RIJECT_OFFER,DERCUTE_OFFER_DOC(br> to transfer and carry the data(br) ith all required data (br) accurring.	Ì
0 0 0	Vote static final long s ivate static final String Indicates the date at wh ivate Date offerietterGen Indicates the date at wh ivate Date offerietterExp Denotes the details for	Copy Copy Qualified Name Paste Quick Fix Source Refactor Local History References Declarations	Ctrl+C Ctrl+V Ctrl+1 Alt+Shift+S Alt+Shift+S	Ettane () ; ; Generate Validator Validate JavaDoc Generate Factory Pattern Generate Sesion Bean Generate GEFU Helpers Generate Repository	
• /•	Indicates the date at wh	Add to Snippets		Generate ApplicationService	
• /	ivate Date rejectionDate; Indicates the reason for	Debug As Profile As	:	Generate EIB VO Facade Generate EIB Facade	
e /	ivate String rejectionRea	Validate Team Compare With	:	Generate Denormalized Entity MetaData Generate DTO MetaData Generate Stocios Mitilidate MetaData	
e /*	ivate Date acceptanceDate	Replace With Oracle Banking Platform	•	Generate Service and Facade Layer Sources	
. /	ivate Date executionDate; Preturn the offerLetterGene /	Remove from Context	Ctrl+Alt+Shift+Down]	
			Weinshie	Smart Insert 2-1	

Service Data Attribute Generation

After generating metadata, we need to generate service attribute which will be mapped with facts which will be used in data bindings in Alert Maintenance screen AL04.

To generate we need to activity ID class for specific event, DTO is used for this activity ID.

1. Right-click that service and select *Oracle Banking Platform -> Generate Service Attribute Metadata*.

In this case also insert scripts will be generate in same location as metadata attributes.

- 2. This will generate the insert scripts for following tables:
 - FLX_MD_SERVICE_INPUTS
 - FLX_MD_SERVICE_OUTPUT
 - FLX_MD_SERVICE_ATTR

There are some steps in generating of service attribute which are as follows:



Figure 11–6 Service Data Attribute Generation

FLX_MD_SERVICE_ATTR is used to map the alert activity attribute with the fact code and to map the alert activity attribute with the DTO field to extract the data from.

As an example, the key fields in FLX_MD_SERVICE_ATTR for an alert activity attribute have been listed below:

Table 11–4 Key Fields in FLX_MD_SERVICE_ATTR

Column	Description
COD_SERVICE_ATTR_ID	The Unique ID for the Attribute of any Activity configured for an alert. For example, com.ofss.fc.app.account.service.accountaddresslinkage.AccountAddressLinkageAppl icationService.createAccountAddressLinkage.Alert.Party.Address.City.DTO
TYP_DATA_SRC	Indicates the Data Source(entity/input/DTO) for the Attribute of the Resource

Column	Description
COD_ATTR_ID	This field indicates the Fact Code. For example, Alert.Party.Address.City
COD_SERVICE_ID	This field indicates the Activity ID. For example, com.ofss.fc.app.account.service.accountaddresslinkage.AccountAddressLinkageAppl icationService.createAccountAddressLinkage
REF_FIELD_DEFN_ID	This field indicates the DTO leaf field from which the data is extracted. For e.g.: com.ofss.fc.app.dda.dto.alert.AccountAddressLinkageAlertDTO.Address,com.ofss.fc. datatype.PostalAddress.City
	Data for this column is interpreted /extracted as follows.
	com.ofss.fc.datatype.PostalAddress address = com.ofss.fc.app.dda.dto.alert.AccountAddressLinkageAlertDTO.getAddress();
	String city = address.getCity()

Table 11–4 (Cont.) Key Fields in FLX_MD_SERVICE_ATTR

11.1.7 Alert Message Template Maintenance

User will maintain template format and template ID to be used for the alerts definition.

These messages need to be defined only if the same template is going to be used for multiple events. Else there is a provision to define the message template during the definition of the alert itself.

All data elements defined within the '#' symbol will be defaulted in the panel below as data attribute.

For example, your account Number #Acct No# has been credited with #currency# #transaction amount# being cash deposited

The user can Mask certain digits in data elements that are preceded with '#' under the 'Attribute Mask' column.

Figure 11–7 Alert Message Template Maintenance

Ø	Oracle Banking Platform - Win	ndows Internet Explorer					- 6 .
G	🗸 🗢 🗖 http://10.180.2	5.183:8001/com.ofss.fc.ui.view/faces/main.jspx?_afrLoo	p=3279837999766968c_afrWindowMode=08	k_adf.ctrl-state=13zqxg6c7	Γς_4 - ← ← ★ X D Bing		+ م
	r Favorites 🛛 👍 🖸 Consult	ing Env 猪 Google 🙋 Directory Manager 퉬 Links	🕶 퉬 Env Links 👻 📸 Security 💽 ST01	🔊 ST SOA 🚼 Google (2)		
8	🗄 🔻 <i>6</i> Re-issue card	🖹 Issue Navigator 🙀 [#NGPR2BEPT-722 🥖	Directory Manager 🛛 🖸 Oracle Banking	Pla 🖸 Oracle Banking	x 🔄 🏠 🔻 🖾 🖛 🖾	▼ Page ▼ Safety ▼ To	ols 🕶 🔞 🕶 🧉
ВА	Posting D	ate 31-Mar-2017 Local Currency AUD Last Login 13-Aug	-2012 12:12:05 PM Branch U Bank Operation	is BR Busir	ness Unit UBank Business Unit 👻 Access	ibility About Help Logout	HardikA O
	Account - Back Office - CAS	SA - Channel - Collection - LCM - Loan - Origin	ation - Party - Payment And Collection -	Security - Term Deposit	_ % 4	» Fact Dath	
					· · · ·		
	AL <u>o</u> S Message Template						
	Read Create / Up	date 💥 Delete				🛷 Ok_ 🏈 Clear 📓 Exit	Print
	Template Search						
	* Template Id ATM Cash Depo	sit					
	🗉 Message Template Deta	sil					
E	* Template Nane * Message Data	ATM Cash Deposit Deer Customer, Vor Account no CASAAcountNumber # has been Your Account of the Casual Annuarts #Accounturrency # on date #Valuedate # from ATM #ATMd# and Transaction (# #ATMHonid # The present net balance in the account is #netBalance# Regards, Bank		CA ATT Tra De Val Acc CC/C Ne	New Detach		3
	☐ Hide Modification History Created By Of Approved By Of	SSUBBEY On SSUBEY On	15-Jun-2012 07:03:25 PM 15-Jun-2012 07:04:52 PM	Approved V Active V			2 >
	Approved By OF	Suber On	15-Jun-2012 07:04:52 PM	Active 🗹			
ST	R 2. BUILD DATE : 2012-08-09 TN	S Details : ngpst = (DESCRIPTION = (ADDRESS = (PROTOCC	L = TCP)(HOST = 10.180.22.245)(PORT = 152	1)) (CONNECT_DATA = (SER	VER = DEDICATED) (SERVICE_NAME = ngp	ost))). DB Details : st02rd/st02	rd @ ngpst.
Dor	ne				🕵 Local intranet Protected Mod	le: Off 🛛 🖓 🔻 🤅	100% 👻

11.1.8 Alert Maintenance

Given below is the Alert Maintenance screen.

Figure 11–8 Alert Maintenance

Ø	Oracle Banking Platform - Windows Internet Explorer		×
0	http://10.180.25.183:8001/com.ofss.fc.u	i.view/faces/main.jspx?_afrLoop=237875854650423&_afrWindowMode=0&_adf.ctrl-state=12e1pv70uj_4 🔹 😽 🗙 🔯 Bing 🔎	•
ý	💡 Favorites 🛛 🖕 🧰 Consulting Env 🔧 Google 🖉	Directory Manager 🎳 Links 🔻 🎳 Env Links 👻 😹 Security 🧰 ST01 🔊 ST SOA 💡 Google (2)	
8	🗄 💌 🏉 Oracle Test Manager 🙀 Issue Navigator	🍸 [#NGPCONREGRE 🖸 Oracle Banking 🗙 🍘 Security - base_do 🧿 Oracle Banking Pla 👘 🔻 🔝 🔻 🖃 👻 Page 🔻 Safety 🔻 Tools 🕶 🚱 🖛	»
В	ANKING PLATFORM Posting Date 31-May-2017 Local C	urrency AUD Last Login 27-Aug-2012 12:38:51 PM Branch U Bank Operations BR Business Unit UBank Business Unit 🗸 Accessibility About Help Logout HardkA 🖉	Þ
r	Account - Back Office - CASA - Channel - Collect	ion 🗸 LCM 👻 Loan 👻 Origination 👻 Party 👻 Payment And Collection 🚽 Security 👻 Term Deposit 🚽 🛛 🙀 🎃 😹 🙀 🐘 Fast Path	
	AL05 AL04		
	Alast Maintenness		1
	Alert Maintenance		
	🛄 Read 🖶 Create 🧪 Update 💥 Delete	🔗 Ols 🏈 Clear 😹 Exit 🗎 Print	
	Alert Search	🖸 Detail	
	View - Detach	* Alert Name ATM Cash Withdrawal Remote Rule Effective Date * Activity Name ATM Cash Withdrawal * Expiry Date 12-Dec-2099 * Event Name Cash Withdrawal Remote-Onus Alert Type Clustomer Subscribed	
	Alert Name	Retries required 🗹 Alert Nature Critical	
	Chaque Book Return Alert	Number of Retries 3	
	Modification of Advance against Lindear Funds	Is Condtional	
	Define Advance against Unclear Funds	Recipient Message Template	
	Adhoc Statement Generation on Printing		1
	Re-pricing CASA account	view - X By En Detach	
	Modification of Account Address Linkage	Subscriber Type Relation Type Group Id Message Template *Destination Type Template Rule * Rule ID Rule Effective Date	
	Create Account Address Linkage	PARTY PARTY ATM Cash Withdra EMAIL	
	Cheque Book Issued		
	Hold Funds- Creation		
	POS Purchase Reversal Remote		
	POS Purchase Remote	Columns Hidden 2	
Ľ	POS Purchase Onus		1
	ATM Funds Transfer Reversal	Hessage Template	- 11
	ATM Funds Transfer	Update Read View 🗸 🔛 Detach	
	ATM Cash Withdrawal Reversal Onus	* Template ATM Cash Withdrawal Data Attribute and the To	
	ATM Cash Withdrawal Remote	Name Data Attribute ID Mask Attribute ID	
	ATM Cash Deposit Reversal	*Message Dear Customer, AccountAmount D Alert.Channel.bxnAmount	
	deletion of an account from the group	Data Your Account no #CASAAccounthlumber # has been Accounthlumber XXD Alert.Channel.accounthlumber	
	addition of an account to the group	depiced with amount #Accountumount# date D Alert.Channel.dateOffransaction *	
	Change in the system defined status (User define Deletion of Account Address Linkage	withdrawal from ATM #ATMid# and Transaction Id	
		#AIMONIG#. The present net balance in the	
		۲. III III III III III III III III III I	1
	Hide Modification History		
	Created By kishorem	On 20-May-2011 12:18:39 PM Approved 🗹 🔇 10F	
	Approved By kishorem	On 20-May-2011 12:18:39 PM Active	
SI	R2. BUILD DATE : 2012-08-24 TNS Details : nanst = /DESCRI	PTION = (ADDRESS = (PROTOCOL = TCP)(HOST = 10, 180, 22, 245)(PORT = 15211) (CONNECT DATA = (SERVER = DED)(CATED) (SERVICE NAME = const1)). DB Detals : sti 2/disti 2/d db const	
Do	ne	🕵 Local intranet Protected Mode: Off 🛛 🦓 🔻 🕄 100% 💌	

We can define the alert name, expiry date, alert type (Customer Subscribed/ Mandatory) and link this with predefined activity and event. These entries are fed to table "flx_ep_act_evt_acn_b".

Now, we need to link a Recipient Message Template/s with this alert. For this we drag recipients from the Recipient Panel on to the Recipient Message Template Panel. In this setup, we define the kind of recipient and link this to predefined Message Template and Destination Types. The entry for this goes to table "flx_ep_evt_rec_b".

Finally, we need to complete the Message Template Mapping Configuration for each Recipient Message Template. For this, we map each data attribute of each Recipient Message Template with a corresponding attribute (Fact Code) from the drop down. This drop down populates fact codes configured for this activity id in the metadata table FLX_MD_SERVICE_ATTRIBUTE. The entry for this goes to table "flx_ep_msg_ src_b"

11.2 Alert Subscription

Subscription can be done for alerts at **account level** or at **application level** (called as subscription level)

Figure 11–9 Alert Subscription

🥖 Oracle Banking Platform - Windows Internet Explorer			- d -
- http://10.180.25.183:8001/com.ofss.fc.ui.view/faces/main.jspx	_afrLoop=237875854650423&_afrWindowMode=0&_adf.ctrl	-state=12e1pv70uj_4 🔹 😽 🗙 📴 Bir	ng 🔎 🔻
🖕 Favorites 🛛 🖕 🖸 Consulting Env 🔮 Google 🖉 Directory Manager 🌗	Links 🕶 🌗 Env Links 👻 🚵 Security 🛄 ST01 🕖 ST S	OA 👌 Google (2)	
🔠 🔹 🏈 Oracle Test Manager 👻 Issue Navigator 👻 [#NGPCONREGR	E 🖸 Oracle Banking 🗴 🌈 Security - base_do 🧲	Oracle Banking Pla 👌 🔻 🔝 👻 🖃	🖶 💌 Page 🕶 Safety 🕶 Tools 💌 🕢 🎽
DANKING PLATFORM Posting Date 31-May-2017 Local Currency AUD Last Logi	27-Aug-2012 12:38:51 PM Branch U Bank Operations BR	Business Unit UBank Business Unit 👻 A	accessibility About Help Logout HardikA 🔘
		Terr Desert	
	Origination		🕻 👩 » Fast Path
AL05 AL04			
Alert Subscription			
La Read de Create Dooate & Delete La Close La Ke Open			Viear 🗶 Cancel 📇 Print
🗉 Unsubscribed Alerts	scription		lê l
View 🗸 Select all 🔄 🚮 Detach	Subscription Level O Application		
Sweep out Execution (Failure)	Account		
Cheque Book Request CASAChequeBookDispatch	* Subscriber 000008119	Alert ATM C	ash Withdrawal Remote
Subsequent Disbursement execution	* Subscription Start 22-Aug-2017	* Subscription End 29-Au	g-2017 🔯
Principal offset Linkage	* Account Number 000000000014921	Account Number for SC	
Manual reschedule-Recalculate Installment Advance Payment by Cash	•		
MC_Alert on print			
Cheque Book Return Alert E Subscript	ion details		
Cheque Book Issued POS Purchase Reversal Remote	d Alerts	Recipients	=
POS Purchase Remote View View	Detach	View 🗸 🔛 Detach	1
ATM Cush Withdrawal Reversal Remote	Subscription Start Subscription End	Recipient Destination Relation Typ	e
BP_Alert On periodic bill payments successful POS Merchant Return Reversal Remote	22*Aug*2017 25*Aug*2017 40		
POS Merchant Return Remote			
Subscribed Alerts			
View - Detach	r Preferences		
Vo data to display. View -	🗶 📝 Detach		
Redpier	t Contact Name Destination Destination	Id Amount Priority	
PARTY	EMAIL newOD@abc	.com HIGH 👻	
Hide Modification History			
Created By HardikA	On 27-Aug-2012 12:00:00 AM	Approved	
Approved by	UII	Active	
STR2. BUILD DATE : 2012-08-24 TNS Details : ngpst = (DESCRIPTION = (ADDRESS = (P	ROTOCOL = TCP)(HOST = 10.180.22.245)(PORT = 1521)) (CONN	ECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME	= ngpst))). DB Details : st02rd/st02rd @ ngpst.

11.2.1 Transaction API Changes

You will need to modify the transaction API to support the newly registered activities. This section gives an overview of how the developer needs to modify the transaction API.

The entry point for activity business logic would be the service call for the transaction. In the aforementioned use case, the service call would be *com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint(...)*.

Figure 11–10 Transaction API Changes - Service Call



If the activity needs to be conditional, then the logic for evaluating the conditions should be present inside the service call. This should be followed by the invocation of the routine to register the activity. In the aforementioned use case, the activity should be registered only if the *update* transaction updates the attributes associated with *DND Information*. Following code snippet shows the conditional evaluation and invocation of the call to register activity.

Figure 11–11 Transaction API Changes - Conditional Evaluation

The *persistActivityLog(..)* routine primarily takes the *Activity Id, Event Id* and *Activity Log DTO*. This routine first calls a helper routine to populate the activity log DTO with the activity data and then passes on the DTO to the appropriate *Event Processing Adapter* which will register the activity and generate associated events.

Figure 11–12 Transaction API Changes - persistActivityLog(..)

```
/**
   This method logs/registers the activity log DTO
    "
   This method logs/registers the activity log DTO
    "
    @@fcb.param NI_String.activityId, The ActivityId for which we need to log the data for the further processing of
        alerts.
    @@fcb.param NI_String.partyName, holds the new data, sent as DTO's.
    @@fcb.param NI_String.partyName, holds the new data, sent as DTO's.
    @@fcb.param NI_String.partyName, holds the Party Name.
    @@fcb.param NI_String.eventId, holds the Party Name.
    @@fcb.param NI_String.eventId, holds the event Id.
    @@fto.wara MI_String.eventId, holds the event Id.
    @@fto.wara MI_String.eventId, holds the event Id.
    @fto.wara KI_String.eventId, holds the eventId.
    @fto.wara KI_String.eventId, holds the eventId.
    @fto.wara KI_String.eventId, holds the eventId.
    @fto.wara KI_String.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eventId.eve
```

You will need to add the business logic to populate the activity log DTO with the data specific to the transaction and the activity. This logic can be present inside the activity helper class for the module. Module specific activity attributes can also be populated in this logic. Following code snippet shows the activity log DTO population with activity data for the aforementioned activity.

Figure 11–13 Transaction API Changes - Activity Log

```
private IActivityLog populateActivityLogForDNDInfoChange(Object object, String partyName) {
```

```
ContactPointDT0 contactPointDT0 = (ContactPointDT0) object;
PartyDNDInfoChangeDatalogDT0 activityLog = new PartyDNDInfoChangeDatalogDT0();
activityLog.setCustomerId(contactPointDT0.getPartyId());
activityLog.setPartyId(contactPointDT0.getPartyId());
activityLog.setFullName(partyName);
activityLog.setUpdatedIsDnd(contactPointDT0.isDnd());
activityLog.setUpdatedIsDnd(contactPointDT0.getDndStartDate());
activityLog.setUpdatedDndStartDate(contactPointDT0.getDndStartDate());
activityLog.setUpdatedDndEndDate(contactPointDT0.getDndEndDate());
activityLog.setCriticalNotification(true);
return activityLog;
```

```
}
```

Figure 11–14 Transaction API Changes - Register Activity

```
* Used to register an Activity with an associated Event
  * @param activityID
   @param eventID
  % @param eventProcessingDate
% @param activityLog
  * Greturn
  * @throws FatalException
public String registerActivityAndGenerateEvent(ApplicationContext applicationContext,
                                                                String activityID,
String eventID,
Date eventProcessingDate,
                                                                 Object logObject) throws FatalException {
     ActivityLog activityLog = (ActivityLog) logObject;
     ActivityRegistrationApplicationService activityManager = new ActivityRegistrationApplicationService();
SessionContext sessionContext = AdapterContextHelper.fetchSessionContext();
if (sessionContext = null) {
    sessionContext = AdapterContextHelper.fetchBasicSessionContext(applicationContext);
     ActivityEventKeyDTO activityEventKeyDTO = new ActivityEventKeyDTO();
     activityEventKeyDTO.setActivityId(activityID);
     activityEventKeyDTO.setEventID(eventID);
ActivityRegistrationResponse response = activityManager.registerActivityAndGenerateEvent(sessionContext,
                                                                                                                                 activityEventKeyDT0,
                                                                                                                                 eventProcessingDate,
                                                                                                                                activityLog):
     return response.getActivityDataId();
}
```

The *Event Processing Adapter* contains the logic to register the activity and generate events. You can use the existing adapter class *com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapter* or write your own custom adapter which must implement the interface *com.ofss.fc.app.adapter.impl.ep.IEventProcessingAdapter*.

All the above steps would suffice to support a transaction as an activity and raise events on the activity.

On successful completion of the transaction and the activity registration and event generation, you can view the activity log in the table FLX_EP_ACT_LOG_B and the generated events log in the table FLX_EP_EVT_LOG_B.

Actions associated with the activity events would pick up the activity and event data from these tables for processing.

11.3 Alert Processing Steps

For any modules the starting point is EventProcessingAdapter method named 'registerActivityAndGenerateEvent'.

Through this we call 'registerActivityAndGenerateEvent' method of ActivityRegistrationApplicationService which marks actually registration of your activities and events.

During this activity the entries are made in table FLX_EP_ACT_LOG_B and FLX_EP_EVT_LOG_B with appropriate comments depending on type of Alerts whether it is Mandatory (M) or Customer Subscribed (S).

There is one flag maintained in FLX_EP_EVT_LOG_B viz. FLG_PROCESS_STAT, which specifies status of event.

In this step various validations are also performed such as checking if email Id of recipient is mentioned and so on.

However, the final processing of alerts is managed in 'Interaction.java' when it is about to close that is, call is made in 'manageLastInteraction'.

Figure 11–15 Alert Processing Steps



EventProcessStatusType

This shows status of event throughout cycle of event processing from Registration of event to Dispatch of Alert. (It is maintained in FLX_EP_EVT_LOG_B table as "flg_ process_stat").

The various statuses of events are as follows:

- GENERATED("G")
- COMPLETED("C")
- NO_SUBSCRIPTION("N")
- ABORTED("A")
- INITIATED("I")
- REINITIATED("R")

For any event online or batch, when it is logged for first time it is marked as Generated "G" in flx_ep_evt_log_b table.

Figure 11–16 Event Processing Status Type



JMS (Java Messaging Service) is used for dispatch of alerts.

For Online Alerts:

- Direct Approach: If alert gets send in first try, flg_process_stat is as "G" in FLX_ EP_EVT_LOG_B and alert is dispatched through JMS, and then entry for that event record is moved to FLX_EP_EVT_LOG_HIST_B and flg_process_stat is marked as "C".
- EventPoller: If alert gets failed in first retry it will mark status as "R". In this case EventPoller will pick the failed event and complete its processing and mark status as "A" and then entry for that event record is moved to FLX_EP_EVT_LOG_HIST_ B and flg_process_stat is marked as "C".
- For Batch Alerts: In case of batch alerts as no Interaction.close() is called, the direct approach is not used in Batch Alerts. In this case only EventPoller approach is used.





11.4 Alert Dispatch Mechanism

The dispatch mechanism is triggered by the *AlertHandlerService* for dispatching subscribed actions of type *Alert*. The processing is implemented as part of the respective handlers. The handler services delegate the call to the *Dispatcher* based on the type of *DestinationType* configured in the *Recipient* at the time of *ActivityEventAction* maintenance which involves *RecipientMessageTemplate* setup.

The module provides definition of multiple dispatch detail configurations on the basis of *SubscriberType* and various configuration parameters like *UrgencyType*, *ImportantType* in the AlertTemplate.

The dispatcher uses the *DispatchDataConverter* to convert the data captured as part of activity registered in the system into data which can be dispatched to the target subscriber.







Figure 11–19 Alert Dispatch Mechanism - Dispatcher Factory





The various Destination Types are coded as per the above diagram. This existing framework makes it further extensible as per the requirements that is, you can add more destination types.

11.5 Adding New Alerts

To add a new alert:

- **1.** Implement the Service Extension Interface for the application service of the method for which alert is to be raised.
- **2.** Use either the preServiceMethod() or postServiceMethod() hook for the method in the implemented service extension class depending on the requirement.
- **3.** The method should call the registerActivityAndGenerateEvent() of the EventProcessingAdapter class. In case a custom adapter is required the custom adapter method should call registerActivityAndGenerateEvent() of ActivityRegistrationApplicationService.

4. New Activity ID, Event ID and implementation of IActivityLogDTO have to be created.

11.5.1 New Alert Example

This example will explain the above points in detail.

Use Case: A new alert has to be added after updating a party name.

The class PartyNameApplicationService has a method updateIndividualName() that does this activity.

Create the extension class, say PartyNameApplicationServiceExt, for this application service by implementing its extension interface IPartyNameApplicationServiceExt. Since the alert should be raised after updation of party name we will use the postUpdateIndividualName() method.

Within the method a call to registerActivityAndGenerateEvent() in EventProcessingAdapter should be made.

Code snippet for the call:

```
com.ofss.fc.app.adapter.IAdapterFactory adapterFactory =
AdapterFactoryConfigurator.getInstance().getAdapterFactory(ModuleConstant.EVENT_
PROCESSING);
IEventProcessingAdapter adapter = (IEventProcessingAdapter)
adapterFactory.getAdapter(EventProcessingAdapterConstant.MODULE_TO_ACTIVITY);
adapter.registerActivityAndGenerateEvent(applicationContext, activityId, eventId,
new Date(), activityLog);
```

In case a new customer adapter has to be used, a call to registerActivityAndGenerateEvent() in ActivityRegistrationApplicationService should be made from within the adapter. A class called ActivityEventKeyDTO is used which captures the event ID and activity ID.

Code snippet for the call:

```
ActivityRegistrationApplicationService activityManager = new
ActivityRegistrationApplicationService();
ActivityEventKeyDTO activityEventKeyDTO = new ActivityEventKeyDTO();
activityEventKeyDTO.setActivityId(activityID);
activityEventKeyDTO.setEventId(eventID);
ActivityRegistrationResponse response =
activityManager.registerActivityAndGenerateEvent(sessionContext,activityEventKeyDT
0,eventProcessingDate, activityLog);
```

The signature for the method is:

public String registerActivityAndGenerateEvent(ApplicationContext applicationContext,

> String activityID, String eventID, Date eventProcessingDate, Object logObject) throws

FatalException;

Create new activityID, eventID and logObject to be passed to this method.

ActivityID and EventID as explained in detail in the above section have to be added in the following database tables. If data is not added in the tables, a runtime exception will occur while displaying the alert.

FLX_EP_ACT_B stores all the recognized activities.

FLX_EP_EVT_B stores all the recognized events.

FLX_EP_ACT_EVT_B which stores the mapping between all activities and events.

The Activity ID denotes the actual action that should raise the event within the application service and hence for ease of understanding it should ideally be the fully qualified name of the method.

 $\label{eq:com.ofss.fc.app.party.service.contact.PartyNameApplicationService.updateIndividualName$

The Event ID can be anything that denotes the event

For example, UPDATED_PARTY_NAME

The logObject is an implementation of IActivityLogDTO. For the new alert a new implementation has to be created. The DTO should have fields mapped to the placeholders in the new alert to be added

For example, for the alert "Your name has been updated from #previous_Name# to #new_Name# successfully."

the following DTO has to be made. The variables have to map to the placeholders in the alert template.

```
public class PartyNameChangeLogDTO implements IActivityLogDTO {
    private static final long serialVersionUID = -3492413059506052931L;
    private String updatedName;
    private String registeredOldName;
    //getters and setters for the variables
}
The DTO has to be populated with relevant data
E.g.:. private IActivityLog populateActivityLogForIndividualPartyNameChange() {
    PartyNameChangeLogDTO activityLog = new PartyNameChangeLogDTO();
    activityLog.setUpdatedName("Andrew Matthew");
    activityLog.setRegisteredOldName("Andy Matthew");
    return activityLog;
}
```

11.5.2 Testing New Alert

JUnit test cases can be used to test the alert created by supplying sample input data. The example below shows how the above new alert can be tested.

```
public void testPartyUpdateName() throws IOException {
   String testCase = "PartyUpdateName";
   ActivityRegistrationApplicationService activityRegistrationApplicationService
          = new ActivityRegistrationApplicationService();
   ActivityEventKeyDTO activityEventKeyDTO = new
ActivityEventKeyDTO("com.ofss.fc.app.party.service.contact.
    PartyNameApplicationService.updateIndividualName "," UPDATED_PARTY_NAME");
   Date date = new Date();
   SessionContext sessionContext = getSessionContext();
   com.ofss.fc.app.party.dto.alert.PartyNameChangeLogDTO activityLog
         = new com.ofss.fc.app.party.dto.alert.PartyNameChangeLogDTO ();
   activityLog.setUpdatedName("Andrew Matthew");
   activityLog.setRegisteredOldName("Andy Matthew");
    trv{
          ActivityRegistrationResponse response
activityRegistrationApplicationService.registerActivityAndGenerateEvent(
```

```
sessionContext, activityEventKeyDTO, date, activityLog);
TransactionStatus result= response.getStatus();
dumpTransactionStatus("ActivityRegistrationApplicationService", "
testPartyUpdateName ", result);
logger.log(Level.FINER, "The ErrorCode is: "+ result.getErrorCode());
} catch (FatalException e) {
logger.log(Level.SEVERE, "FatalException from"+THIS_COMPONENT_NAME+".
testPartyUpdateName ",e);
fail("Unexpected failure from " + THIS_COMPONENT_NAME + ".
testPartyUpdateName ");
}
```

For testing with the JUnit test cases we need to update the PoolType property in the AlertPollerPool.properties as follows:

```
PoolType=JDK
```

The value should be JDK for testing with JUnit (standalone application) and JMS if the application is run on WebLogic server.

Creating New Reports

Oracle's Business Intelligence Publisher Enterprise is a standalone reporting and document output management solution that allows companies to lower the cost of ownership for reporting solutions. BI Publisher Enterprise's (hereafter known as BIP) strength is that it separates the data model from the actual report formatting/layout. BIP relies on 2 fundamental components to create reports, XML data and a template that represents the look and feel of the report. The XML data can be generated from any number of sources and BIP makes accessing data in the proper format easy. Templates can be created in Microsoft Word and Adobe Acrobat allowing almost anyone familiar with these desktop applications the ability to create reports.





The following sections will give an overview of Oracle's *BI Publisher*. The developer will be able to add and configure an *Adhoc* report to OBP using the BI Publisher.

Use Case: The OBP application has a batch framework using which a developer can easily add batch processes, also known as *batch shells*, to the application. The batch framework executes all the batch shells defined in the system as per their configuration. The results of these batch shell executions are stored in the database. We will be adding a report using BIP for the execution results summary for batch shells.

12.1 Data Objects for the Report

The *Data Model* of the report invokes the database to fetch the data for the report through certain data objects that we will need to create. The primary data objects needed for the reports are as follows:

Global Temporary Table

You will need to create a *Global Temporary Table* based on the fields required for the report data. This table should mandatory have the field *SESSION_ID* of *NUMBER* type. The naming convention followed in OBP for the global temporary table's name is *RPT_<Module_Code>_R<Report_Number>*.

For the aforementioned use case, the script for creating the global temporary table would be as shown below.

Figure 12–2 Global Temporary Table

```
-- Global temporary table for the report

DROP TABLE RPT_PI_R007;

CREATE GLOBAL TEMPORARY TABLE RPT_PI_R007

(

COD_SHELL VARCHAR2(30),

TXT_PROCESS_NAME VARCHAR2(120),

COD_PROC_CATEGORY NUMBER(3),

TXT_CATEGORY VARCHAR2(20),

DATE_RUN CHAR(8),

STREAM_START_TIME DATE,

STREAM_END_TIME DATE,

PROCESSED_COUNT NUMBER(38),

COD_BRANCH_GROUP_CODE VARCHAR2(10),

EXECUTION_DURATION NUMBER,

SESSION_ID NUMBER

)

ON commit preserve rows;
```

Report Record Type

You will need to create a *Type* object with the fields present in the global temporary table. This type will represent a single row of data for the report. The naming convention followed in OBP for the report record type's name is *REP_REC_<Report_Id>*.

For the aforementioned use case, the script for creating the report record type would be as shown below.

Figure 12–3 Report Record Type

```
-- Record type for the report

CREATE OR REPLACE TYPE REP_REC_PI007 AS OBJECT

(

COD_SHELL VARCHAR2(30),

TXT_PROCESS_NAME VARCHAR2(120),

COD_PROC_CATEGORY NUMBER(3),

TXT_CATEGORY VARCHAR2(20),

DATE_RUN CHAR(8),

STREAM_START_TIME DATE,

STREAM_END_TIME DATE,

PROCESSED_COUNT NUMBER(38),

COD_BRANCH_GROUP_CODE VARCHAR2(10),

EXECUTION_DURATION NUMBER,

SESSION_ID NUMBER

);
```

Report Table Type

You will need to create a *Type* object which will be a table of the previously created report record type. This type will represent the set of rows of data for the report. The naming convention followed in OBP for the report table type's name is *REC_TAB_* <*Report_Id>*.

For the aforementioned use case, the script for creating the report table type would be as shown below.

Figure 12–4 Report Table Type

```
-- Table type for the report
CREATE OR REPLACE TYPE REP_TAB_PI007 AS TABLE OF REP_REC_PI007;
```

Report DML Function

You will need to create a DML function which will be invoked to populate the previously created global temporary table with the data required to be displayed in the report. This function can have parameters as per the developer's requirements with filtering the data or inserting additional data. The naming convention followed in OBP for the report DML function's name is *AP_DML_<Report_Id>*.

For the aforementioned use case, the script for the report DML function would be as shown below.

Figure 12–5 Report DML Function

```
-- DHL function for the report
CREATE OR REPLACE FUNCTION AP_DHL_PI007(var_l_session_id IN NUMBER,
var_bank_code IN VARCHAR2,
var_cod_shell IN VARCHAR2)
var_l_cod_shell VARCHAR2(30);
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
          input parameter
   IF (var_cod_shell IS NULL or length(trim(var_cod_shell)) = 0) THEN
var_l_cod_shell := '%';
   var_l_cod_shell := var_cod_shell;
END IF;
    --delete existing data for the session
DELETE FROM RPT_PI_R007 WHERE SESSION_ID = var_l_session_id;
      -insert data into the table
    INSERT INTO RPT_PI_R007
(COD_SHELL, TXT_PROCESS_NAME, COD_PROC_CATEGORY, TXT_CATEGORY, DATE_RUN, STREAM_START_TIME,
STREAM_END_TIME, PROCESSED_COUNT, COD_BRANCH_GROUP_CODE, EXECUTION_DURATION, SESSION_IO)
    STREAM_ENU_ITHE, PROCESSED_COUNT, COU_BRANCH_CHOUP_COUE, EXECUTION_DURATION, SESSION_ID)

SELECT DISTINCT

BJSR.COD_SHELL, BJSM.TXT_PROCESS_NAME, BJSM.COD_PROC_CATEGORY, BJCM.TXT_CATEGORY, BJSR.DATE_RUN, BJSR.STREAM_START_TIME,

BJSR.STREAM_END_TIME, BJSR.PROCESSED_COUNT, BJSR.COD_BRANCH_CROUP_CODE, BJSR.EXECUTION_DURATION, var_lsession_id
       FLX_BATCH_JOB_SHELL_RESULTS BJSR, FLX_BATCH_JOB_SHELL_MASTER BJSM,
FLX_BATCH_JOB_CATEGORY_MASTER BJCM, FLX_BATCH_JOB_BRN_GRP_MAPPING BJBGM
    HHERE

BJSR.COD_SHELL = BJSM.COD_EOD_PROCESS AND BJSR.COD_SHELL LIKE var_l_cod_shell AND

BJSM.COD_PROC_CATEGORY = BJCM.COD_PROC_CATEGORY AND BJSM.COD_BRANCH_GROUP_CODE = BJBGM.BRANCH_GROUP_CODE AND

BJBGM.BANK_CODE = var_bank_code

ORDER BY DATE_RUN;
     -- commit
    COMMIT;
RETURN 0;
    EXCEPTION
WHEN OTHERS THEN ORA_RAISERROR(SQLCODE, 'Execution of AP_DNL_PI007 failed', 500);
END:
```

Report DDL Function

You will need to create a DDL function which will be invoked to fetch data required to be displayed in the report from the global temporary table and wrap it in the previously created report table type. The naming convention followed in OBP for the report DDL function's name is *AP_DDL_<Report_Id>*.

For the aforementioned use case, the script for creating report DDL function would be as shown below.

Figure 12–6 Report DDL Function

```
-- DDL function for creating the report
CREATE OR REPLACE FUNCTION AP_DDL_PI007(var_bank_code IN VARCHAR2,
var_cod_shell IN VARCHAR2)
RETURN REP_TAB_PI007 AS
                               REP_TAB_PI007;
v_ret
var_l_session_id NUMBER;
dml_function_result NUMBER;
BEGIN
   var_l_session_id := USERENV('SESSIONID');
dml_function_result := AP_DML_PI007(var_l_session_id, var_bank_code, var_cod_shell);
   SELECT
      CAST
      (
        MULTISET
           SELECT
              COD_SHELL, TXT_PROCESS_NAME, COD_PROC_CATEGORY, TXT_CATEGORY, DATE_RUN, STREAM_START_TIME,
STREAM_END_TIME, PROCESSED_COUNT, COD_BRANCH_GROUP_CODE, EXECUTION_DURATION, SESSION_ID
           FROM RPT_PI_R007
WHERE SESSION_ID = var_l_session_id
           ORDER BY DATE RUN
        AS REP_TAB_PI007
      INTO v_ret
     FROM DUAL;
   RETURN v_ret;
   EXCEPTION
      WHEN OTHERS THEN ORA RAISERROR(SOLCODE, 'Execution of AP DDL PI007 failed', 500);
END;
```

Data Model for the Report

Once you have created the data objects for the report in the database, you can start adding and configuring the report using BIP. Log in to the BIP application and follow these steps.

You can log in to the BIP application deployed on http: //<IP ADDRESS><PORT>/xmlpserver/ with the credentials *weblogic/weblogic1*.

12.2 Catalog Folder

Before creating the data model or the layout for the report, you should create a folder to save the model and layout. You can find the link for the Catalog tab on the home screen. Click it and create a folder for your report at an appropriate location.

For the aforementioned use case, you can create a folder *PI007* at the location /*My Folders/FC Module/Demo* as shown below.

Figure 12–7 Catalog Folder

ORACLE BI Publisher Enterprise	Search All	*	•				Ç
Catalog		Home Catalog	New 🗸	Dpen 🗸 📔	Signed In A	s weblogic	• •
🕐 🖤 🐏 😹 🗇 🗈 💥 🛠 🗸 👘 Location /My Folders/FC Modules/Demo							(
Polders Polders PD07 [List Modified 31/8/22 11:32 AM Created By weblogic Expand Morev Expand Morev PD07 List Modified 31/8/22 11:32 AM Created By weblogic Expand Morev PD07 List Modified 31/8/22 11:32 AM Created By weblogic Expand Morev PD07 List Modified 31/8/22 11:32 AM Created By weblogic PD07 List Modified 31							

12.3 Data Source

You will need to add the data source from which the data will be fetched to be displayed in the report. The data source can be a *JDBC Connection, JNDI Connection, File, LDAP Connection* and so on. You can find the link for the *Administration* tab on the home screen. Click it and choose the appropriate data source connection type. Enter the required parameter values and validate the connection. Save the data source with an appropriate name.

For the aforementioned use case, you can add the JDBC Connection data source as show below.

Figure 12–8 Data Source

ORACLE	BI Publisher Enterprise		Search All	×	0	Administration	Help v	Sign Ou	t Ç
Administration				Home Catalog	New~	🔁 Open 🛩	Signed In A	: weblog	lc ~
Administration > J	DBC > Update Data Source: FCRHDEVSAILS40								
Update Data Sour	ce: FCRHDEVSAILS40								_
							Apply	Cance	el
General									
	ike sure to install the required JDBC driver classes. e Fusion Middleware Security Model, select the Use Data Source Name * Driver Type	System User checkbox to use the BI S PCRHDEVSAILS40 Oracle 11g	lystem User for your Bi Server Database Conr	nection.					
	Database Driver Class	(oracle.pdbd.OracleDriver (Example: oracle.dbc.OracleDriver.)							
	* Connection String	jdbc.oracle.thin:@10.180.22.245:152	1.DEVD8						
	Use System User								
	* Usemame	FCRHDEVSAIL840							
	Password								
	Pre Process Function								
	Post Process Function								
		Use Proxy Authentication Test Connection							

12.4 Data Model

You will need to create a data model to back the report. This data model represents the report data fetched using the data objects and formatted into XML data. You can find the link to *Create Data Model* on the home screen of BIP. Click it and follow these steps:

- **1.** Enter an appropriate *description* for the data model.
- 2. Choose the previously created *data source* from the list displayed.
- **3.** Check the Enable Scalable Model option.

- 4. Check the Include Parameter Tags option.
- 5. Check the Include Empty Tags for Null Elements option.
- 6. Check the Include Group List Tags option.
- 7. You can leave the rest of the options to default.

For the aforementioned use case, you can create data model as shown below.

Figure 12–9 Data Model

ORACLE	BI Publisher Enterprise		Search All	*	0	Administratio	n Help ~	Sign Out 🥥
P1007				Home Catalog	New 🗸	Dpen 🗸	Signed In A	s weblogic ~
Data Model Data Model Data Sets Data Sets Data Sets	Properties Description	Demo Report RohanS Data Model						
 Plexifields List of Values Parameters Bursting 	Default Data Source Oracle DB Default Package Database Fetch Size	FCRHDEVSAILS40 y	Refresh Data Source List					
	Backup Data Source	Enable Scalable Hode Enable Backup Connection Switch to Backup Data Source on Use Backup Data Source on	e when Primary Data Source is unavailable v					
	XML Output Options	S Include Parameter Tags Include Empty Tags for Null Elen Include Group List Tag	nents					
	XML Tag Display Attachment	Upper Case						
	Sample Data Schema Data Ries	sample.xml Delete						

Data Set

After creating the data model, you will need to create a data set of the fields required to be displayed in the report. You can find the link for *Data Sets* on the left side pane of the screen. To create the data set, follow these steps:

- 1. In the Create Data Set icon, choose the option Create Data Set from SQL Query.
- 2. Enter an appropriate *name* for the data set.
- **3.** Choose the previously created *data source* from the list displayed.
- **4.** Enter the SQL query which will be used to fetch the data for the report. The results returned should be of the *Report Table Type* previously created.

For the aforementioned use case, you can create the data set as shown below.
Figure 12–10 Data Set

DRACLE BI Pu	blisher Enterprise	Search All		Administration Help - Sign Out
1007			Home Catalog 🧕 New 🗸	Dpen 🗸 Signed In As weblogic 🗸
				💀 i 🖬 🖬 🕐
Data Model	Diagram Structure Code			
E Data Sets	B- / X			
B P1007	Edit Data Set	×		
E Event Triggers	* Name Pi007			
E Flexifields	* Data Source			
E List of Values	Default Data Source			
Parameters	CONDEVSALS40 Kerresh Data Source List			
	SELECT T. P FROM TABLE(CAST(AP_COL_P1007(:P_COD_BANK. :P_COD_S REP_TAB_P1007)) T	HELL) AS		
		Les Levent 1		
	Incp	UK Cancel		

On click of OK, a data set will be created with all the fields as defined in the previously created *Report Record Type*.

You can group the fields as per the requirements of the report:

- 1. Select the field on which you want to group and choose *Group By*.
- 2. After creating a group, you can move fields between the groups.
- 3. You can also set field which will be used to sort the data displayed in a group.

For the aforementioned use case, you can group the fields as shown below.

Figure 12–11 Group Fields

RACLE BI Pu	blisher Enterprise		Search /	All 🕑	Administration Help -	Sign Out
7				Home Catal	og 🔄 New 🗸 📄 Open 🗸 🛛 Signed In	As weblogic
					1	
ata Model	Diagram Structure Code					
lata Model						
Data Sets	Table View Output					
200 PHON 7		XML View			Business View	
Event Triggers	Data Source	XML Tag Name	Sorting	Value If Null	Display Name	Data Typ
Flexfields	S Report Data		_			
List of Values	🗟 Data Structure	P1007	_			
Parameters	S P1007	G_1	_		G_1	_
Busiles	COD_SHELL	COD_SHELL	02		COD_SHELL	-
bursing	TXT_PROCESS_NAME	TXT_PROCESS_NAME	02		TXT_PROCESS_NAME	-
	COD_PROC_CATEGORY	COD_PROC_CATEGORY	02		COD_PROC_CATEGORY	
	TXT_CATEGORY	TXT_CATEGORY	02		TXT_CATEGORY	-
	S P9007	G_2			G_2	
	DATE, RUN	DATE_RUN	20		DATE_RUN	
	STREAM START TIME	STREAM_START_TIME	1		STREAM_START_TIME	
	STREAM END TIME	STREAM END TIME	Q ₀		STREAM END TIME	
	PROCESSED COUNT	PROCESSED COUNT	20		PROCESSED COUNT	-
	COD BRANCH GROUP CODE	COD BRANCH GROUP CODE	Q2		COD BRANCH GROUP CODE	
	EXECUTION DURATION	EXECUTION DURATION	Q.		EXECUTION DURATION	
	Therease in	EFEEDON ID	0.		SESSION ID	

You can view and edit the XML structure and labels of the report data in the *Structure* tab in a tabular format.

For the aforementioned use case, the structure would be as shown below:

Figure 12–12 XML Structure and Labels

ORACLE BI Publis	sher Enterprise		Search All	2 00	Administration Help + Sign Out O
P1007				Home Catalog SNew ~	Dpen v Signed in As weblogic v
					8 2 6 2
Data Model	Diagram Structure Code				
E Data Model	and a second cone				
E Data Sets	18· / ×				
Sa #4007					
E Event Trippers	🗉 📰 Global Level Functions 🗮				
E Plexfields	Drop here for aggregate function				
List of Values					
Parameters					
E Buration		COD SHULL			
a correct		TXT PROCESS NAME			
		COD PROC CATEGORY			
		TXT_CATEGORY A >			
		Drop here for aggregate function			
		= R.c.2 =			
		DATE BUN			
		STREAM START_TIME			
		STREAM_END_TIME			
		PROCESSED_COUNT ->			
		COD_BRANCH_GROUP_CODE			
		EXECUTION_DURATION HT>			
		SESSION_ID			
		Drop here for aggregate function			

You can view the actual XML code in the *Code* tab.

For the aforementioned use case, the XML code would be as shown below.

Figure 12–13 XML Code

ORACLE	BI Publisher Enterprise Search 🗛 💽 📀 Administration Help 🗸 Sign Out. 📿
P1007	Home Catalog 🖹 New v 🍉 Open v Signed in As weblogic v
Data Model Data Model Data Model Data Shoel Data Shoel Data Ses D	Duyam Southur Cede Duyam

Input Parameters

You can define the *Input Parameters* required by the report in the *Parameters* tab present on the left hand side pane of the screen. To define input parameters, follow these steps:

- 1. In the **Parameters** tab, click the icon for *Add Parameter*.
- 2. Enter the name, type, display label and default value for the parameter.
- 3. Repeat the above steps to define as many parameters as required.

For the aforementioned use case, you can add parameters as shown below:

Figure 12–14 Add Input Parameters

Dirubit	sher Enterprise			Search All	<u> </u>		Administration	n Help v	Sign Out
					Home Catalog	New 🗸	Dpen 🗸 🔁	Signed In a	is weblogic v
a Model	Parameters								
Model	+ X								
vata pers	*Name	Data Type	Default Value	Parameter Type	Reorder				
55 P1007	P_USER_ID	String	Kishorem	Text	00				
Ivent Trippers	P_DAT_PROC	String	01-JAN-2011	Text	80				
lexfields	P_COD_BRANCH	String	· 082991	Text	00				
lat of Values	P_COD_BANK	String	. os	Text	80				
lariameters	P NAM BRANCH	String	Melbourne	Text	00				
P_USERCID	P_NAM_BANK	String	National Australian Bar	Text	89				
P_DAT_PROC	P_COD_SHELL	String	1	Text	 Image: Second sec				
P_COD_BRANCH						_			
P_COD_BANK									
P_NAM_BRANCH	P_COD_SHELL: Typ	e: Text							
D_NAM_BANK	Display	Label Shell Code							
P_COD_SHELL	Text Field	d Size							
lucition .									
	0;	ptions Text field contr	ins comma-separated valu	es					
		Refresh other	arameters on change						

12.5 XML View of Report

After following the above steps, save the data model in the previously created catalog folder with an appropriate name. You can view the report without the layout in the XML form by clicking on the icon for *XML View*.

In the XML view, you will see input fields for the previously defined *input parameters*. Enter appropriate values in those fields and click *Run*. You will be able to see the XML representation of the report data.

For the aforementioned use case, the XML representation of the report data would be as shown below.

Figure 12–15 XML View of Report

ORACLE	BI Publisher Enterprise		S.	earch All	*	0	Administrat	ion Help v	Sign Out 🧲
P1007					Home Catalog	New ~	Dpen 🗸	Signed In A	s weblogic ~
Number of rows to	User Id Kishorem Branch Code 062991 Branch Name Melbourne Shell Code return S C Run		Process Date 01-) Bank Code 08 Bank Name Nati	AN-2011 onal Australian B				Re	turn 🗄 🥑
This XML file	does not appear to have any sty	le information associated	l with it. The document tree	e is shown be	low.				
<fgenerated by<br="">- <p1007> <p_user_i <p_dat_p1 <p_cod_b <p_cod_b <p_nam_b <p_nam_b <p_nam_b <p_cod_s1 - <list_g_1 - <g_1></g_1></list_g_1 </p_cod_s1 </p_nam_b </p_nam_b </p_nam_b </p_cod_b </p_cod_b </p_dat_p1 </p_user_i </p1007></fgenerated>	Gracle & Publisher 11.1.1.5.6 D>Kishorem 00C-01-JN-2011(/P_DAT_PROI RANCH>08:2991 IRANCH>MelbourneANK>08:4P_COD_BANK> IRANCH>MelbourneANK>National Australian Bank HILL/>	C> iCH> BRANCH> 							
<pre><cod <txt_ <cod <txt_ - dLST - dLST - dC < < < < < < < < < < </txt_ </cod </txt_ </cod </pre>	SHELJ->ac, action_relog, sh <cto PROCESS, NAME>Relog actions PROC, CATEGORY>1-('COD_PR CATEGORY>End of Day6, 2> >> ATTE, RUN>20160215TREAM, START, TIME>2012-06- TREAM_END_TIME>2012-06- TREAM_END_TIME>2012-06- TREAM_END_COUNT>0-(PROCE OD_BRANCH_GROUP_CODE>E ROCESSED_COUNT>0-0-(FX) ESSION_ID>885390272></cto 	DD_SHELL> on holiday<(TXT_PROCE oc_CATEGORY> CATEGORY> JN> 13T15:22:09.000+05:30 T15:22:09.000+05:30 SEED_COUNT> RN_GRP_1ECUTION_DURATION> N_ID>	SS_NAME> STREAM_START_TIME> REAM_END_TIME> CH_GROUP_CODE>						
বা ও ও	ATE_RUN>20160229TREAM_START_TIME>2012-06- TREAM_END_TIME>2012-06-15	UN> 15T17:12:11.000+05:30 5<br T17:12:11.000+05:30 5T</th <th>STREAM_START_TIME> REAM_END_TIME></th> <td></td> <td></td> <td></td> <th></th> <th></th> <th></th>	STREAM_START_TIME> REAM_END_TIME>						

12.6 Layout of the Report

A report needs to be presented in an appropriate format. The format can vary from report to report and client to client. BIP separates the data model from the layout making it convenient for the developer.

Anybody familiar with using Microsoft Word or Adobe Acrobat can use the corresponding plug-ins for these tools to create a layout for a report. You can create a rich layout using these standalone applications with BIP plug-ins and then upload them to the BIP application for use in your report.

The BIP application can generate a very basic layout for your report from the data set. You can download the generated layout, modify it as per your layout requirements and upload it to the BIP application for use in your report.

The BIP application also allows the user to create a layout on the web. It has a rich set of tools to with drag and drop features and a ready link to the data set fields. You can create a layout in this fashion and use it in your report.

You can find the link to *Add New Layout* on the right side of the screen. Click it to get the options to *create*, *generate* or *upload* a layout.

Figure 12–16 Layout of the Report - Create Layout



Choose from the *Basic Templates* to create a layout from a template. The layout editor screen will open. The previously created data set fields are present on the left pane of the screen. The toolbar present on top of the layout has tools to insert *Layout Grid*, *Data Table*, *Repeating Section*, *Text Item*, *List*, *Image*, *Page Break*, *Page Number*, elements.

You can drag and drop the layout and data set elements on to the layout as per your requirements. After making the required modifications, save the layout and return to the previous screen.

For the aforementioned use case, the layout for the report would be as shown below.

1007 : PI007 C1				Home Catalo	a 🛛 🔍 New 🗸 🛛 🗁 O	loen v Sig	hed in As weblogic					
Data Source	90 XDD	X - Insert Pare Laws	4			Re	um 💽 😂 🖻					
PI007		Components	Par	e Flements								
- P_USER_ID		Laura Gold										
- P_COD_BRANCH	Cayou Gre	Usa rabe 🛄 Unan 🛄 Pivot rat	De E un Pre rage pres	K M Hage Kumber								
P_COD_BANK	Repeating Section	Text Item 🙆 Gauge 🔜 Image	Total Page	15								
P_NAM_BRANCH	50 100	150 200 250 300	350 400 450	500 550 600	650 700	750 800						
P_COD_SHELL]					
2 G_1 (a) 2 G_1												
- COD_SHELL			Batch Job Resu	Its								
TXT_PROCESS_NAME												
- M TXT_CATEGORY		The Research C. 1										
8 20 9.2	T Start Repeat	Start Repeating - G_1										
- RUN	Shell Code	COD_SHELL	Shell Name 1	XT_PROCESS_NAME								
STREAM_START_TIME	Category Co	Category Code COD_PROC_CATEGORY Category Name TXT_CATEGORY										
PROCESSED_COUNT	Run Date	Start Time	End Time	Processed Cou	nt Execution Duration							
- HI EXECUTION DURATION	20160430	6/25/12 11:19 AM	6/25/12 11:19 AM		0	0						
HI SESSION_ID	20160315	6/20/12 10:36 AM	6/20/12 10:36 AM		0	0						
	20160215	6(13)12 9:52 AM	6/13/12 9:52 AM		0	0						
	20120731	6(1/12 2:22 PM	6/1/12 2:22 PM		0	0						
	20120630	6/1/12 6:42 AM	6/1/12 6:42 AM		0	0						

Figure 12–17 Layout of the Report - Batch Job Results

12.7 View Report in BIP

After saving the *Data Model* and *Layout*, you can view the report in BIP. Click the **View Report** link on the top right corner of the screen to open the report screen.

You will be able to see the input fields for the input parameters defined for the report. Enter appropriate values in these fields and click **Apply**. The report will be generated and displayed on the screen with the applicable data returned by the previously created *Data Model* and formatted as per the previously created *Layout*.

For the aforementioned use case, the final report would be as shown below.

ORACLE	BI Publisher En	terprise			Search A	All .	*		Administration	Help ~	Sign Out 📿
P1007						H	ome Catalog	New 🛩	🎦 Open 🛩 S	igned in Ar	i weblogic ~
Biooz Cl	r ld Kishorem ode 08 ode	Apply		Process Date 01-) Branch Name Mell	AN-2011 bourne		Bra Br	nch Code Ink Name	082991 National Australia	n 8	
				Ratch Job Re	enlte						
		Shell Code Category Code	ac_stl_pyt_eod_shell	G_1 Row 5 Shell Name Category Name	Account Settleme	ent Payout	EOD Shell	~			
		Run Date	Start Time	End Time	Proce	essed Count	Execution Duration				
		20160430	6/25/12 11:37 AM	6/25/12 11:40 AM		0		0			
		20160415	6/23/12 9:55 AM	6/23/12 9:55 AM		7	78	5			
		20160330	6/21/12 12:25 PM	6/21/12 12:25 PM		4	11	2			
		20160315	6/20/12 10:38 AM	6/20/12 10:38 AM		10	120	11			
		20160229	6/15/12 11:43 AM	6/15/12 11:43 AM		6	1	46			
		20160215	6(13/12 9:53 AM	6/13/12 9:53 AM		1	26	0			
		20160131	6/8/12 12:22 PM	6/8/12 12:22 PM		1		0			
		20160115	6/6/12 10:35 AM	6/6/12 10:35 AM		2		0			
		20151231	6/4/12 7:02 AM	6/4/12 7:02 AM		0		0			
		20151215	6/2/12 10:04 AM	6/2/12 10:04 AM		0		0 -			
		1									

Figure 12–18 View Report in BIP

You can export the report in *HTML*, *PDF*, *Excel*, *RTF* or *PowerPoint* formats by clicking on the icon for *Export* on the right top corner of the screen and choosing the corresponding export option.

12.8 OBP Batch Report Configuration - Define the Batch Reports

Entries are required in three tables as given below to generate reports during EOD.

insert into FLX_BATCH_JOB_SHELL_MASTER (COD_EOD_PROCESS, TXT_PROCESS, TXT_PROCESS, NAME, FRQ_PROC, DAT_LAST_RUN, DAT_SCHEDULED_RUN, TXT_PROC_PARAM, COD_PROC_STATUS, NUM_PROC_ERROR, FLG_RUN_TODAY, COD_PROC_CATEGORY, FLG_MONTH_END, FLG_MNT_STATUS, COD_MNT_ACTION, COD_LAST_MNT_MAKERID, COD_LAST_MNT_CHKRID, DAT_LAST_MNT, CTR_ UPDAT_SRLNO, COD_MODULE, DAT_PROC_START, DAT_PROC_END, TXN_KEY, SERVICE_KEY, NAM_ COMPONENT, TYPE_COMPONENT, NAM_DBINSTANCE, RETRY_COUNTER, NON_RETRY_COUNTER, COD_ UNSTREAMED_PROCESS, COD_BRANCH_GROUP_CODE) values ('ch_eod_report_shell', 'CASA EOD Reports', 'CASA EOD Reports', '1', to_ date('15-02-2012', 'dd-mm-yyyy'), to_date('15-12-2007', 'dd-mm-yyyy'), '99', 0, 0, 'Y', 1, 0, 'A', ' ', 'SETUP1', 'SETUP2', to_date('09-02-2002', 'dd-mm-yyyy'), 2, 'CH', to_date('21-08-2008 09:54:57', 'dd-mm-yyyy hh24:mi:ss'), to_date('28-02-2011 05:02:41', 'dd-mm-yyyy hh24:mi:ss'), 'DUMMY', 'execute', 'com.ofss.fc.bh.batch.BatchReportShellBean', 'B', 'PROD', 0, 0, 'ch_eod_report_ shell', 'BRN_GRP_1');

Cod_proc_category = 1, for EOD; 2, for BOD and 16 for Internal System EOD

Nam_component is the same for all report shells.

Also we are using Branch_Group_Category ='BRN_GRP_1' for all these report shells.

12.9 OBP Batch Report Configuration - Define the Batch Report Shell

Insert into FLX_BATCH_JOB_SHELL_DEPEND (COD_EOD_PROCESS, COD_REQD_PROCESS, COD_ PROC_CATEGORY, COD_REQD_PROC_CAT, FLG_MNT_STATUS, COD_MNT_ACTION, COD_LAST_MNT_ MAKERID, COD_LAST_MNT_CHKRID, DAT_LAST_MNT, CTR_UPDAT_SRLNO, COD_BRANCH_GROUP_ CODE)

Values ('ch_eod_report_shell', 'dd_eod_action', 1, 1, 'A', ' ', 'SETUP', 'SETUP', to_date('30-06-1995', 'dd-mm-yyyy'),2, 'BRN_GRP_1');

Here, in the first column is the report shell name and second is the name of the shell after which this shell should run. So 'ch_bod_report_shell' runs after 'dd_bod_action'. The remaining columns are self explanatory.

COD_PROC_CATEGORY=1 , for EOD; 2, for BOD and 16 for Internal System EOD COD_REQD_PROC_CAT=1, for EOD; 2, for BOD and 16 for Internal System EOD

Also we are using Branch_Group_Category = 'BRN_GRP_1' for all these report shells.

12.10 OBP Batch Report Configuration - Define the Batch Report Shell Dependencies

Insert into flx_ba_report_ctrl (COD_REPORT_ID, FLG_REP_ADV, COD_MODULE, NAM_ REPORT, TYP_REPORT, FRQ_REPORT, FLG_PRINT, FLG_DELETE, CTR_REP_COPIES, COD_ PRIORITY, COD_ACCESS_LVL, COD_FILEID, BUF_INV_VAR1, BUF_INV_VAR2, BUF_INV_VAR3, BUF_INV_VAR4, BUF_INV_VAR5, FLG_MNT_STATUS, COD_MNT_ACTION, COD_LAST_MNT_MAKERID, COD_LAST_MNT_CHKRID, DAT_LAST_MNT, CTR_UPDAT_SRLNO, FLG_SOURCE, FLG_SPLIT, FLG_ PROD_REP, COD_REPORT_DB_PREFIX, FLG_APPLY_SC, REF_UDF_NO, XPATH, FLG_REPORT_ SERVER) values ('CH318', 'R', 'CH', 'CASA BALANCE LISTING', 'E', '1', '1', '0', 1, 0, 0, 10047, ' ', ' ', ' ', ' ', 'A', ' ', 'PHASE_2', 'PHASE_2', to_ date('01-11-1999', 'dd-mm-yyyy'), 2, 'P', 'Y', 'P', 'PROD', '', '', '', 'B'); Entry for each report should be here with typ_report = 'I' for Internal System EOD; 'E' for EOD and 'B' for BOD.

Currently, for EOD and BOD eod_report_shell and bod_report_shell will take care of all non CASA and TD EOD and BOD reports respectively.

No separate module specific shell is required during EOD and BOD. That is to mention Entry 3 alone is sufficient during EOD and BOD categories for any module. However, entries are needed for all three entries for batch report generation during any other category.

12.11 OBP Batch Report Configuration

This section describes the OBP batch report configuration.

12.11.1 Batch Report Generation for a Branch Group Code

During Batch Process, a report should be generated for all branches linked to the respective Branch Group Code.

For any Batch Report to make use of the Branch Group Code getting passed by the application, a parameter 'P_COD_BRANCH_GRP' has to be defined in the Data Model.

The Data Model should pass this parameter to the Report Related DDL Function.

The Report Related DML Function filters all branch codes from FLX_BATCH_JOB_ RESULTS_FILTERED that belong to the same Branch Group Code.



Figure 12–19 Batch Report Generation for a Branch Group Code

12.11.2 Batch Report Generation Status

At the end of all batch processes BA_REPORT_RESTART gets logged with the generated report status as D -> Done or F->Failed.

12.11.3 Batch Report Generation Path

The reports (for example, 30th September 2008) will be generated as shown in the host side screen-shot.

Locate these reports at this location in the host server.

/oracle/deployables/batch/08/runarea/rjsout/09/30 which actually is of the format

/config/../<BankCode>/runarea/rjsout/<MM>/<DD>

9 30 - NEW_MT@10.180.9.143 - WinSCP									- d -	
Local Mark Files Commands Session (Options Remote	Help								
A III - A - I - A IIII - A D	• IIII III IIII	* 2 2 Defe		. 15.						
	a suus suus puxus s		unc							
🎽 🗎 NEW_MT@10.180 🔹 🅍 👹 📲										
🏭 C: Local Disk 🔹 🤄 🗸 🔿 🔹 🔁	🖾 🚮 🛃 🚞	te l		🔒 30 🛛 👻	🔶 🔹 - 🔁 🔁 🚰 🛃	🔄 te				
C:\Users\siladityad\Documents				/oracle/deployables/batch/l	08/runarea/rjsout/09/30	-				
Name Ext	Size	Туре	*	Name Ext	Size	Changed	Rights	Owner	1	
1		Parent directory		2		28-06-2011 20:06:19	TWXT-XT-X	oracle		
335		File folder		DMSInputFiles		29-06-2011 11:02:09	rwxr-xr-x	oracle	d	
3351		File folder		AT002.pdf	1,342	29-06-2011 10:53:45	rw-r	oracle		
-administrator		File folder		CD198.pdf	17,436	29-06-2011 10:53:42	rw-r	oracle		
hbm .		File folder		CD395.pdf	17,456	29-06-2011 10:53:45	rw-r	oracle		
Fax		File folder		CD396M.pdf	46,596	29-06-2011 10:53:49	rw-r	oracle		
Scanned Documents		File folder		CD396W.pdf	60,314	29-06-2011 10:53:42	rw-r	oracle		
Remote Assistance Logs		File folder		CH117.pdf	0	29-06-2011 11:08:42	rw-r	oracle		
My Music		File folder		CH122.pdf	10,245	29-06-2011 10:53:44	rw-r	oracle		
E My Pictures		File folder		CH123.pdf	3,933	29-06-2011 10:53:44	rw-r	oracle		
🗃 My Videos		File folder	=	CH127.pdf	1,435	29-06-2011 10:53:44	rw-r	oracle		
OBIP 11G Report Creation.docx	6,636,931	Microsoft Offic		CH318.pdf	43,502	29-06-2011 10:53:48	rw-r	oracle	=	
~\$IP 11G Report Creation.docx	162	Microsoft Offic		CH321.pdf	45,405	29-06-2011 10:53:54	rw-r	oracle		
Notes and the second se	2,052	Remote Deskto		CH500.pdf	2,072	29-06-2011 10:53:39	rw-r	oracle		
STReports.xlsx	13,639	Microsoft Offic		E GL102.pdf	1,506	29-06-2011 10:53:41	rw-r	oracle		
CH318.rtf	537,474	Rich Text Format		GL205M.pdf	6,644	29-06-2011 10:53:50	rw-r	oracle		
CH321.rtf	492,600	Rich Text Format		GL205W.pdf	4,783	29-06-2011 10:53:53	rw-r	oracle		
🔁 CH123.rtf	442,483	Rich Text Format		GL206A.pdf	26,233	29-06-2011 10:53:51	rw-r	oracle		
🛃 CH123[1].rtf	442,112	Rich Text Format		GL206B.pdf	43,933	29-06-2011 10:53:52	rw-r	oracle		
xdocore.jar	6,215,008	WinRAR archive		GL208.pdf	17,311	29-06-2011 10:53:52	rw-r	oracle		
SILA38079.pdf	21,182	PDF File		OR001.pdf	1,705	29-06-2011 10:53:57	rw-r	oracle		
ch122_en_us%252e.rtf	577,832	Rich Text Format		PL001.pdf	359,628	29-06-2011 11:02:07	rw-r	oracle		
at001_en_us%252e.rtf	219,096	Rich Text Format		RE203.pdf	15,433	29-06-2011 10:53:51	rw-r	oracle		
OracleFLEXCUBE_BWB.ear	26,266,880	EAR File		RE204.pdf	5,844	29-06-2011 10:53:55	rw-r	oracle	-	
OracleFLEXCUBE.ear	26,237,343	EAR File		RS396D.pdf	4,568	29-06-2011 10:53:53	rw-r	oracle		
EODEODEOD.txt	26,562	Text Document		RS396G.pdf	14,428	29-06-2011 10:53:50	rw-r	oracle		
com.ofss.fc.reports.core.jar	202,833	WinRAR archive		TD003R1.pdf	2,702	28-06-2011 23:37:46	rwxr-xr-x	oracle		
CH318164912.995.pdf	87,016	PDF File		TD003R2.pdf	1,658	28-06-2011 20:07:47	rwxr-xr-x	oracle		
client.pdf	551,298	PDF File		TD102.pdf	2,495	28-06-2011 20:07:47	rwxr-xr-x	oracle		
ofss.log	5,429,049	Text Document		TD103.pdf	6,057	28-06-2011 20:07:48	rwxr-xr-x	oracle		
com.ofss.fc.module.taxation.jar	339,551	WinRAR archive		TD126.pdf	22,598	28-06-2011 23:37:46	rwxr-xr-x	oracle		
TD126.pdf	7,254	PDF File		AT002.txt	2,933	29-06-2011 10:53:45	rw-r	oracle		
TD126.txt	104,248	Text Document		CH117.txt	0	29-06-2011 11:08:42	rw-r	oracle		
22330_DasguptaSiladitya_IFS1222.docx	75,740	Microsoft Offic		CH122.082991.txt	79,834	29-06-2011 10:53:48	rw-r	oracle		
Deployed Reports.xlsx	13,671	Microsoft Offic		CH122.089999.txt	9,673	29-06-2011 10:53:47	rw-r	oracle		
R2 - TD Reports V1.0.xlsx	137,014	Microsoft Offic		CH122.txt	90,450	29-06-2011 10:53:44	rw-r	oracle		
Reports-CASA.xlsx	266,299	Microsoft Offic		CH123.082991.txt	41,985	29-06-2011 10:53:47	rw-r	oracle		
CH12815299.502.pdf	2,268	PDF File	-	CH123.txt	42,513	29-06-2011 10:53:44	rw-r	oracle		
✓ III		+		CH127.txt	2,903	29-06-2011 10:53:44	rw-r	oracle		
0 B of 93,261 KB in 0 of 62				0 B of 11,416 KB in 0 of 62						
- F2 Rename 2 F4 Edit 1 F5 Conv	F6 Move CF F7 C	reate Directory ×		elete 🕾 F9 Properties 👖	F10 Ouit					
Friedmine Er (1997) - Friedronken M. (1998) - Friedmine Friedmi										
									anin-3 🧤 2,0:15:13	
🔁 🌗 🥔 🥭		1 📲	W.		🖄 🌟 🙋				▲ 12:36 ▲ 12:36 29-06-2011	

Figure 12–20 Batch Report Generation Path

12.12 OBP Adhoc Report Configuration

This section describes the OBP adhoc report configuration.

12.12.1 Define the Adhoc Reports

Define the adhoc reports as follows:

Insert into flx_ba_report_ctrl (COD_REPORT_ID, FLG_REP_ADV, COD_MODULE, NAM_ REPORT, TYP_REPORT, FRQ_REPORT, FLG_PRINT, FLG_DELETE, CTR_REP_COPIES, COD_ PRIORITY, COD_ACCESS_LVL, COD_FILEID, BUF_INV_VAR1, BUF_INV_VAR2, BUF_INV_VAR3, BUF_INV_VAR4, BUF_INV_VAR5, FLG_MNT_STATUS, COD_MNT_ACTION, COD_LAST_MNT_MAKERID, COD_LAST_MNT_CHKRID, DAT_LAST_MNT, CTR_UPDAT_SRLNO, FLG_SOURCE, FLG_SPLIT, FLG_ PROD_REP, COD_REPORT_DB_PREFIX, FLG_APPLY_SC, REF_UDF_NO, XPATH, FILE_DESC, FLG_ REPORT_SERVER) values ('CH318', 'R', 'CH', 'CASA BALANCE LISTING', 'A', '1', '1', '0', 1, 0, 0, 10047, ' ', ' ', ' ', ' ', 'A', ' ', 'PHASE_2', 'PHASE_2', to_ date('01-11-1999', 'dd-mm-yyyy'), 2, 'P', 'Y', 'P', 'PROD', '', '', '', 'Savings Listing Reports', 'B');

12.12.2 Define the Adhoc Report Parameters

Define the adhoc report parameters as follows:

INSERT INTO flx_ba_report_params (COD_REPORT_ID,FLG_REP_ADV,COD_SERIAL,NAM_PROMPT,

```
COD_FLD_TYP,LEN_FLD,FLG_DELETE,DAT_LAST_MNT,NAM_VAL_ROUTINE,REQD_DESC) VALUES
('CH318','R',1,'Branch Code',0,0,'N','01-NOV-99','','Y')
/
INSERT INTO flx_ba_report_params (COD_REPORT_ID,FLG_REP_ADV,COD_SERIAL,NAM_PROMPT,
COD_FLD_TYP,LEN_FLD,FLG_DELETE,DAT_LAST_MNT,NAM_VAL_ROUTINE,REQD_DESC) VALUES
('CH318','R',2,'Product Code',0,0,'N','01-NOV-99','','Y')
/
INSERT INTO flx_ba_report_params (COD_REPORT_ID,FLG_REP_ADV,COD_SERIAL,NAM_PROMPT,
COD_FLD_TYP,LEN_FLD,FLG_DELETE,DAT_LAST_MNT,NAM_VAL_ROUTINE,REQD_DESC) VALUES
('CH318','R',3,'From Date(DD-MMM-YYYY)',8,0,'N','01-NOV-99','','Y')
/
```

Also COD_FLD_TYP = 8 will ensures the host side date format validations.

COD_FLD_TYP = 0 is for string type parameters.

Corresponding to each of the above sequence of parameters appearing in screen, a mandatory parameter 'FUNC_PARAM<Parameter Sequence Number>' should be defined in BIP Data Model. So the input parameter 'FUNC_PARAM2' defined in data model should correspond to Product Code as defined above.

12.12.3 Define the Adhoc Reports to be listed in Screen

Define the group name as follows:

For Adhoc Report, column FILE_DESC of report master table FLX_BA_REPORT_ CTRL contains the name of the group under which the report will be listed in 7775 screen.

12.12.4 Adding Screen Tab for Report Module

For adding a Screen Tab do the following:

```
com.ofss.fc.ui.view.brop.jar@
/com/ofss/fc/ui/view/brop/reportRequest/backing/ReportRequest.java
private RichCommandNavigationItem cnil1;
   Add following accessors:-
   public void setCnil1(RichCommandNavigationItem cnil1) {
      this.cnil1 = cnil1;
   }
   public RichCommandNavigationItem getCnil1() {
      return cnil1;
   }
```

Also modify the selection tab highlighting portion of the code.

com.ofss.fc.ui.view.brop.jar@

/com/ofss/fc/ui/view/brop/reportRequest/rb/ReportRequest_en.properties

LBL_Reconciliation = Reconciliation

12.13 Adhoc Report Generation – Screen 7775

The adhoc report can be generated using the following screen:



Account Back Office CASA Channel Origination Party Payment And Collection Product Generic Security Service Request Term Deposit	📩 🚖 👻 Fast Path	0
ms		
Report Request	🥔 Clear 🕱 Exit 🛛 😂 Print	
🗟 Report Criteria		<u>^</u>
Advoc Reports O Batch Reports		and i
General Ledger Savings Term Deposit Cards Product Ledger Payments and SetSement Origination ATM		
Vio data to display. No data to display.		
<		
Report Parameter Details		
No data to display.		
		•

Figure 12–22 Adhoc Report Generation - Report Generated

Account 🗸 Back Office 👻 CASA 👻	Channel 🗸 Origination 👻 Party 🗸	Payment And Collection Product Generic Security Service Request Term Deposit	📩 🊖 👻 Fast Path 🚺
7775			
Report Request			🖉 Cjear 🕱 Exit 🔛 Print
Savings Dely Transaction Reports Savings INP and Domarcy Reports Saving Interst-Ta-si-CR Report Savings Dely Exception Reports Savings Exception Reports Savings Exception Reports Savings Advected Statements Savings Advected Statements Savings Advected Reports	Ori318 CASA BALANCE	() Information (x) Report Request Generaled (X)	
Report Parameter Details Parameter Name	Parameter Value		
Branch Code	ALL		
From Date/DD AMM-2000)	21.Mar.2009		
To Date(DD-MMM-0000)	31-Mar-2008		
Waived Service			
Charge			
Conorsto (

On filling the parameters and clicking on 'Generate' the report request gets successfully posted.

At the end of Adhoc report generation, RJS_REQUESTS gets logged with the generated report status as D -> Done, F-> Failed.

12.14 Adhoc Report Viewing – Screen 7779

The adhoc report can be viewed using the following screen:

Figure 12–23 Advice Report

Ø	Oracle FLEXCUBE - W	/indows Internet Explorer										
C	No 🗢 🙋 http://	/10.180.9.106:7001/com.ofss.fc.ui.view/fi	aces/main.ispx? afrLoop	=62392076004	4776& afrWindow	Mode=08L adf.c	trl-state=iih9p7	aa8 4	- ++ 🗙 🗔 Bing	- م		
0			51		_			55	· · · · · · · · · · · · · · · · · · ·			
X	O McAfee'	•										
	👷 Favorites 🛛 🏤 🙋 Oracle BI Publisher Enterp 🏈 Suggested Sites 👻 🙋 Get More Add-ons 👻											
	Oracle ELEVOURE								🍐 🛪 🗟 🛪 🖻 📥 🛪 🖻	age 🗙 Safety 🗙 Tools 🗙 🚳 🛪 ≫		
	OTACIE PLEXCOBE									ige sarely roots		
	Account - Back Office	eCASAChannelOrigination	Party - Payment And	Collection -	Product Generic -	Security - Se	nice Dequest -	Term Derv	seit — 🔶 🔶 — E	ant Bath		
			Turcy • Tuyment And	COLCEDON	Troduce denene 🗸	occurry + oc	mee request •	Terminocepa		astraul		
	7775 7779											
									1			
	Advice Report								V O <u>k</u> V Clear	🗶 Exit 🔚 Print		
	Drint on Concel A	duise /Deport										
	E Phile or Calicer A	dvice/Report										
	Advice/Report											
	User Id X00	0071 👻										
	Advice/Report L	ist										
	View - View R	eport Archive Report 🔜 🖼 Detad	1									
			-									
	ID	Descet/Advice Name	Chabur	Char								
	LD CH318		Done	201								
	CH318	CASA BALANCE LISTING	Done	201								
	CH318	CASA BALANCE LISTING	Pending	201								
	CH318	CASA BALANCE LISTING	Failed	201								
	CH321	Dormant Accounts Statistics	Faled	201								
	CH321 CH318	CASA BALANCE LISTING	Failed	201								
	CH318	CASA BALANCE LISTING	Failed	201								
н	CH318	CASA BALANCE LISTING	Done	201								
	CH321	Dormant Accounts Statistics	Failed	201						1		
	OR001	Origination Application Failed Report	Failed	201								
	ORUUI	Origination Application Failed Report	Faled	201 +								
	•	-		•								
	Columns Hidden	2										
				_								
									🗣 Local intranet Protected Mode: Off	🖓 🔻 🔍 100% 🔻		
	🦻 🦪 🛛) 🧭 📋 😡	1 🐴 👹			⇒ 🔶		8		▲ 12:45 ▲ 29-06-2011		

On selecting the correct user id that generated the report we get the reports generated by that user.

Now sort the Transaction Number (right most column) in the descending order.

Select the top record and click 'View Report'.

Oracle FLEXCUBE - Win	dows Internet Explorer						
🕞 🗢 🙋 http://10	0.180.9.106:7001/com.ofss.fc.ui.view/faces/	main.jspx?_afrLoop=623920	76004776&_afrWindowMode=0&_ac	f.ctrl-state=jjh9p7gg8_4	👻 🍫 🗙 📴 Bing		
⊙ McAfee' /							
Favorites 🛛 👍 👩 O)racle BI Publisher Enterp 🏾 🏉 Suggested !	Sites 🔻 👩 Get More Add	-ons 🔻				
Dracle ELEXCUBE					ሕ 🕶 🗟 🗉 🖨 🕶	Page - Safety -	Tools -
Indie TEENCODE						,	
nk : 08	National Australia Bank				FLEXCUBE		
anch : 082991	U Bank Operations BR				CASA BALANCE LISTING		
. Id : X00007 dule : CASA	1				For:29-Feb-2008		
count Number	Customer Name	Account	Account Status		Clear Balance	Uncleared	н
		Currency				Balance	
duct Code : C	\$100	Product Name	U Saver	Acco	ant Type:ASSET		
047845	Keith Watson	AUD	Inactive		54 21	0.00	
047861	franklin joseph	AUD	Regular		420.00	20.00	
047896	franklin joseph	AUD	Regular		712.74	0.00	
024125	Brad Pitt	AUD	Regular		0.00	0.00	
024176	Randy Orton	AUD	Regular		0.00	0.00	
024192	John GGG Cena	AUD	Regular		22,189,61	0.00	
024205	Atul KKK Sinha	AUD	Regular		0.00	0.00	
024256	Kanh Do	AUD	Regular		993,838.02	0.00	
024301	Andy Flower	AUD	Regular		26,810.07	0.00	
024408	Shane Watson	AUD	Regular		3,016.62	0.00	
024491	Aaron Lo	AUD	Regular		10,079.18	0.00	
024504	JJJJJJJJJJJJ RRRRRRR	AUD	Regular		0.00	0.00	
024627	jay more	AUD	Regular		110,263.88	0.00	
024686	Harry Jonto	AUD	Regular		0.00	0.00	
024889	Shane Watson	AUD	Regular		5,021.16	0.00	
024897	Shane Watson	AUD	Regular		14,063.40	0.00	
024918	Shane Watson	AUD	Regular		41,169.39	50,000.00	
025013	John GGG Cena	AUD	Regular		25,227.97	0.00	
025144	franklin pearl	AUD	Regular		2,322.47	2,795.00	
025179	ansdnn asnasnsn	AUD	Regular		0.00	0.00	
025320	brad hopes	AUD	Regular		1,108.71	0.00	
025347	HHHHHHH MMMMMMM	AUD	Regular		0.00	0.00	
025363	adam gilchrist	AUD	Regular		0.00	0.00	
025435	Charlotte Collins	AUD	Regular		0.00	0.00	
025443	Charlotte Collins	AUD	Regular		100,491.50	0.00	
048098	Darryl Molley	AUD	Regular		102,275,320.27	0.00	
048100	ice ice	AUD	Regular		0.00	0.00	
048119	ice 1	AUD	Regular		50,000.00	0.00	
048127	iceice ice	AUD	Regular		0.00	0.00	
048135	Aishwarya ram	AUD	Regular		100,008.81	0.00	
048151	ice ice	AUD	Regular		0.00	0.00	
0048207	Martin Berchmans	AUD	Regular		95,680.26	0.00	
							Ca
					State Local intranet Protected Mode: O	ff 🕼 🔻	100
		M _ M					12:
/ III /		Te Ne 🎽	e 🖄 🗇 🏓				

Figure 12–24 View Generated Adhoc Report

The report is rendered in the front end.

Security Customizations

OBP comprising of several modules has to interface with various systems in an enterprise to transfer or share data which is generated during business activity that takes place during teller operations or processing. While managing the transactions that are within OBP's domain, it is needed to consider security and identity management and the uniform way in which these services need to be consumed by all applications in the enterprise.

This is possible if these capabilities can be externalized from the application itself and are implemented within products that are specialized to handle such services. Examples of these services include authentication against an enterprise identity-store, creating permissions and role-based authorization model that controls access to not only the components of the application, but also the data that is visible to the user based on fine-grained entitlements.

The following security functions are provided with the extensibility features:

- Attributes participating in access policy rules
- Attributes participating in fraud assertion rules
- Attributes participating in matrix-based approval checks
- Account validator
- Customer validator
- Business unit validator
- Adding validators
- Customizing user search
- Customizing of a 'Send OTP | Validate OTP' logic
- Customizing Role Evaluation
- Customizing Limit Exclusions
- Adding approval checks





- Oracle Identity Manager (OIM) is used for managing user provisioning.
- Oracle Access Manager (OAM) is used for managing declarative authentication and SSO.
- Oracle Platform Security Services (OPSS) is used for runtime evaluation of authn / authz.
- Oracle Adaptive Access Manager (OAAM)/Oracle Adaptive Risk Manager (OARM) is used for step-up authentication and fraud management.
- Authorization Policy Manager (APM) is used to manage access policy definitions.
- Oracle Internet Directory (OID) is used as the identity/policy store.

A high-level security use case has the following access checks and assertions.



Figure 13–2 Security Use Case with Access Checks and Assertions

13.1 OPSS Access Policies – Adding Attributes

OBP uses OPSS to assert role-based access policies. Access policies are rules-based to give more flexibility.

Example of an access policy rule:

```
Grant
Role = RetailBranchOperationsExecutive
Service=com.ofss.fc.app.dda.service.transaction.DemandDepositCashTransactionServic
e.depositCash
Action = perform
IF DepositCash_IsEmployeeAccount=false AND DepositCash_IsRestrictedAccount=false
```

In the above example, the following facts (attributes) make up the access policy rule:

DepositCash_IsEmployeeAccount DepositCash_IsRestrictedAccount

The security framework allows for addition to the facts that can be used in rules. The steps to do this are mentioned in the next section.

13.1.1 Steps

Following steps are needed to add an extra attribute to an access policy rule.

1. Add attribute in OID under the 'Attributes' entry.

Figure 13–3 Add Attributes to Access Policy Rule

Oracle Entitiements Server : Authorization Policy Manager - Windows Internet Explorer		00
🔪 💭 http://101389.25.45.7001./spm/faces/AuthPolicyMgr.jps/_sticeps/S37962812798518_stiffinde=Mode=Ads_adf_ctri-state=glpixt1pm_P	🔹 😽 🗙 🚷 Google	\$
G Malar -		
Factors A Planete Year Researce -		
a server a first server -		
Chade Entitlements Server : Authorization Policy	0 · 0 · · · · · · · ·	age + Safety + Tools + 💕
DRACLE Entitlements Server		Accessibility Help Sign Out
		Signed in as weblic
Anthonization Hanagement Synthese Conferences		
Search 2		10
For Attributes til		
In OradePLEIDURE A DradePLEIDURE Entensional Attributes		
Untitled		Save - Cancel
E Grouse Vis Search Results		
Actors - Vev - 🕐 🗁 🔺 Deplay Name		Required
Name Digitizy Name CredifficationApplicationService_Per		
I sn_skljetdroven 🖉 📩 * Name Gred DecisionApplicationService_Per		
enusti Description Credit/Accessiv/apikation/arvice_P		
I Mar CustomerRegistration_Drawnel		
Mile Difer CustomerRepire store, Difer Code		
BillerCustomeRegistration_DifferName		
Type Startig + Starting - +		
In Oxford Street Service, Payeeld		
California Contraction		
ma hariyalihibuta, Ownyelbetas		
In Construction of the second s		
We remend a standard with the second se		
We of the function of the second		
in the state of the state state is a state of the state o		
W Extended and the intervention of the interve		
III Outcoinclinaved Preventione		
We Foul Attribution Payeehane		
We Postaterhotruction Accountriance		
We Findulate Instruction, Withdrawabilitationce 👔 *		
H H		
ouright © 2011, Onade and/or its affiliates. All rights reserved.	On	ade Entitiements Server 11.1.1.5
1/10 100 75 85/000 (sour Terrer Mathematication and add sour XYNXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	B Local Internet (Restarted Moder Off)	Q + 81005
	- The second sec	

- This can be done directly in OID or by using APM, as shown above.
- 2. Add the attribute under 'AllowedPolicyAttributes' against the particular resource.

Figure 13–4 Attribute to Access Policy Rule - Authorization Management

http://10.180.25.85:7001/apm/faces/Aut	thPolicyMgr.jspx?_afrLoop=53790208177	98518c_afrWindowMode=08c_adf.ctrl-s	tate: g2piut1pm_9 🔹 😽 🗙 🚷 Google
• McAfee'			
Favorites 🏾 🎪 🔊 Upgrade Your Browser 👻			
Dracle Entitlements Server : Authorization Policy			🏠 👻 🔝 🔟 📾 🖛 Page 🖬 Safety 🖛 Tools 🖛 🍯
			Accessibility Help Sign Out Signed in as we
Authorization Hanagement System Configur	ration		
earch (1)	Home @ com.ofss.fc.app.	pc.core.L.	
Resources I OradeFLEXCUEE OradeFLEXCUEE	OradeRLEXCUBE OradeRLEXCUBE	Resource Catalog Resources	3
Browse Search Results	com.ofss.fc.app.pc.core.Inc	omingClearingService.save	Apply Revert
Actions • View • 💽 🛅	Description Pc - IncomingClearingS	ervice Save	Regired
com.ofss.fc.app.sms.service.imit.CustomerO com.ofss.fc.app.sms.service.imit.CustomerO com.ofss.fc.app.sms.service.thirdParty.Party			
🕽 com.ofss.fc.appx.sms.service.limit.Customert 🕕 🐩			
com.ofss.fc.app.worklist.service.approvals.gr 🕕 💷	Actions • View • 4 Add	Remove	
com.ofss.fc.app.pc.bilpay.service.biler.biler	Display Name	Name	Values
Com.offst.fc.cz.nab.app.integration.updatepa	> Transaction Check Point	Transaction Check Point	
con ofte fr anny dia service transaction De	D Samuellactor	Sacodiarty	N Contraction of the second seco
com ofte fc app. ac service accountwriteoff. (> Enseistedester	Enserialtedenter	
com.ofss.fc.aco.oc.core.incomingClearingSer	P Printigation Control	- Handazi di cator	
com.ofss.fc.app.account.service.core.Accour	ApplicationCategory	ApplicationCategory	re - services
com.ofss.fc.app.origination.service.lending.cc	> OAAM_Transaction_ID	OAAM_Transaction_ID	ni
com.ofss.fc.appx.sms.service.lmit.Customer(AllowedPolicyAttributes	AllowedPolicyAttributes	CommonSecurity_Channel, DemandDepositAccount_IsAccount_IsEmployeeAccount, GobalAttributesAdapter_BankerId, G
com.ofss.fc.app.pc.core.IncomingPaymentSe ()	UDenasEvaluatorikequired	UlbenaisEvaluationRequired	PASE
com.ofss.fc.app.dda.service.transaction.Derr 🕕	ApplicationValidation	ApplicationValidation	N
com.ofss.fc.app.pricing.service.contract.Con/ 🕕			
com.ofss.fc.appx.dda.service.account.instrux 🕕			
🕽 com.ofss.fc.app.channel.parser.ISOFacade.p 🕕 👻			
(
	4 -		
	P		
ht @ 2011, Orade and/or its affiliates. All rights reserved	h -		Orade Entitlements Server 11.1.1

This can be done directly in OID or by using APM, as shown above. Adding this attribute under 'AllowedPolicyAttributes' ensures that the security framework executes a specified adapter to fetch the attribute value and make it available to the execution context.

3. Develop custom adapter to retrieve attribute value. Attribute should be structured along similar lines as the other adapters used for the same purpose.

```
Example -
Attribute - CreditDecisionMatrix_OverallAggregateApplicationAmount
Adapter -
public com.ofss.fc.app.adapter.impl.sms.CreditDecisionAttributesAdapter {
    public String getOverallAggregateApplicationAmount () {
        //Logic to fetch overall aggregate amount
    }
}
```

Note: The naming convention of the attribute should be as follows:

The first part of the attribute till the '-' delimiter identifies the transaction. The remaining part with CamelCase is prefixed with a 'get' to form the method in the adapter.

4. Add entry in ConstraintAttributeHelper.properties to link the attribute to the adapter.

CreditDecisionMatrix_OverallAggregateApplicationAmount= com.ofss.fc.app.adapter.impl.sms.CreditDecisionAttributesAdapter Add/Modify access policy/rule in APM to use the extra attribute.

Figure 13–5 Add or Modify Access Policy Rule

5.

* Attp://10.180.25.85/7001/apm/faces/AuthPolicyMgr.jspi?_afrWindowMode:108_afrLoop:1539914152538	3918c_adf.ctrl-state=c0kpnisuq_9	- + + ×	Google J
• McAfee			
Favorites 🛛 👍 🔊 Upgrade Your Browser 👻			
racle Entitlements Server : Authorization Policy		🗄 + 🛯	🗉 🖶 👻 Page 🕶 Safety 🕶 Tools 🕶 🚷
RACLE' Entitlements Server			Accessibility Help Sign Out
relievention Management			Signed in as wed
arch 3	Select Operand Value		
Appkaton Roles # (CreditDecisionNatrix_OverallAggregateApplicationAmount	Attributes Functions Literals		
NO NO	ns = 80000 Name Display Nam SYS_SUBJECTOROUPS sys_subject	te Data Type tgroups String 🕕	PR Apply Revert X Delete
CreditDecisionNatrix_LTVCategoryGroup1 400 MO CreditDecisionNatrix_LTVCategoryGroup1 400 Anno	SYS_OBJ Sys_obj BilerCustomerRegistration_Channel BilerCustom	String 0 I	0
> 20 22xu AND > 20 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	BilerCustomerRegistration_BilerCode BilerCustom BilerCustomerRegistration_BilerName BilerCustom BilerCustomerRegistration_BilerCreatio DilerCustom	rerRegistration_billerName String () rerRegistration_billerName String () rerRegistration_billerCreatio String ()	
Image: Text of the second s	OutgoingPaymentService_PayeeId OutgoingPay OustomerOverrideLimtAdapter_DallyTh CustomerOv	ymentService_Payee3d String () verrideLimitAdapter_DalyThr String ()	* *
ADD A	PartyAttributes_ChangedDetails PartyAttribu GobalAttributesAdapter_RequestPara GobalAttribu	utes_ChangedDetails String () utesAdapter_RequestParam String ()	
CreditDecisionMatrix_AppregateCreditCardLimit CreditDecisionMatrix_AppregateCreditCardLimit AD	FundTransfer_AccountISB FundTransfer_ FundTransfer_AccountISB FundTransfer_ FundTransfer_TransactionType FundTransfer	ercepuarbiller String () ier_Account(ISB String () fer_TransactionType String ()	
CreditDecisionMatrix_BusinessOnit ['UBank_1','	TransferFundsToBeneficiariesSpi_PayeeId TransferFun Private , pe DepositCashSpi_JsEmployeeAccount DepositCash	ndsToBeneficiariesSpi_Payee3d String ()	A Edit Condition
) =	RateAndAmountWidtVariance_Account RateAndAm OutgoingClearingService_PayeeName OutgoingCle	auntWithVariance_AccountC String () saringService_PayeeName String ()	
	PayLaterInstruction_PayeeName PayLaterIns PayLaterInstruction_AccountHame PayLaterIns	Jtruction_PayeeName String () struction_AccountName String ()	
	Add	stocon_metaranabebaance string	
к	• ²	Done Cars	
nt © 2011, Orade and/or its affiliates. All rights reserved.			Orade Entitlements Server 11.1.1
0.180.25.85:7001/apm/faces/AuthPolicyMgr.jspx?_afrWindowMode=0&_afrLoop=539914152538391&_adf.ctrl-state=	c0kpnisuq_9#	😪 Local intranet P	Protected Mode: Off 🛛 🖓 👻 🔍 100%

13.2 OAAM Fraud Assertions – Adding Attributes

OBP uses OAAM to assert fraud policies consisting of rules to identify potentially fraudulent transactions.

Attributes used in fraud identification rules:

payee_id, account_number

The security framework allows for addition to this list of facts. The steps to do this are mentioned in the next section.

13.2.1 Steps

Following steps are needed to add an attribute to an existing OAAM transaction:

- 1. Add the attribute under 'AllowedPolicyAttributes' against the particular resource.
- 2. Add attribute in OID under the 'Attributes' entry.
- **3.** Develop custom adapter to retrieve attribute value.
- **4.** Add entry in ConstraintAttributeHelper.properties to link the attribute to the adapter.

The above steps are exactly the same as mentioned in the previous section.

- **1.** Add seed data in the following tables to persist the mapping between OID attributes and OAAM attributes.
 - flx_sm_fraud_txn_attributes (stores OAAM transaction key to OAAM attribute mapping) and
 - flx_sm_fraud_assert_attributes (stores OBP attributeName oaamAttributeName mapping.

```
Example -
insert into Flx_Sm_Fraud_Txn_Attributes (TRANSACTION_KEY, ATTRIBUTE_NAME)
values ('payment', 'is_2fa_completed')
/
insert into flx_sm_fraud_assert_attributes (ATTRIBUTE_KEY, FRAUD_ATTRIBUTE_
NAME)
values (OutgoiOBPaymentService_Is2FACompleted', 'is_2fa_completed')
```

2. Add/Modify fraud rules in OAAM to use the extra attribute

Figure 13–6 Add or Modify Fraud Rules in OAAM - Data Tab

> McAfee' L •								
evorites 🛛 🍰 🔊 Upgrad	le Your Browser 🔹							
ud Prevention					🗄 = 🖾 -	r 🖂 🛞 👻 Page	 Safety • T 	iools +
ACLE							Help	Sign Ö
ces							Signed i	in as caa
and Prevention	Conditions	cies 🍡 📴 Payme	it to Paye 🖩 Over Daily Tran 🖩 Rules 📲 25A	Active I Intities I Transactions	Payment	_		
ons • View • 🔁 🎴	A Payment						Apply	Revent
Dashboard								
Cases	General Entities (5)	Data (25)	ource Data (33) Happing					
Policy Set								
Policies	Transaction data is unique for each	transaction occurrence	ind therefore not reusable across different transactions. For example, the total	I dollar amount of a transaction would not be reused in multiple t	ansactions so it	should be transaction of	lata not an entity.	
Rules		Lower and						
Conditions	ALLOYS THEN T	Celauri III		le le	a to d	and the second	la manala da	-
Datterne	Row Transaction Data Name	Internal ID	Terrenter Terre		iquirea	China data ta a	Is the ypteor	-
Entities	A (Indication Type	earsacoon_type	Indiación rype		- 90e	String data ty •	False	-1
Transactions	2 Application 10	appecation_id	Appreciation 10		ase •	String data ty •	Fase	-
Configurable Actions	3 Channel 10	channe_id	Channel to		aoe •	Soring data ty 👻	rase	-
Action Templates	4 Is Puture Dated	is_future_dated	Is Puture Dated		856 .	String data ty 💌	Paise	-
Action Instances	5 Is Offine	is_offine	Is Offine		alse 👻	String data ty 💌	False	•
Duestions	6 Amount	amount	Amount		856 *	Numeric data 💌	Palse	-
Validations	7 Is PreAuthorized Payee	is_preauthorizedpa	Is PreAuthorized Payee		alse 💌	String data ty 💌	False	•
Gategories	8 Is Intraffex	is_intrafiex	Is IntraFlex		858 *	String data ty 💌	False	-
Registration Logic	9 2FA Status	2fa_status	2PA Status		85e •	String data ty ·	False	•
Answer Logic	10 Payee List	payee_list	Payee List		alse 💌	String data ty •	False	•
System Snapshots	11 Daily Transaction Limit	transaction_lmit	Dely Transaction Limit		eloc •	String data ty 💌	False	•
m Properties	12 Daily Threshold Limit	threshold_limit	Daily Threshold Limit		alse 💌	String data ty 💌	False	
	13 Banker Id	banker_id	Banker Id	1	alse 🔹	String data ty 💌	False	•
	14 Account 1d	account_id	Account Id		alse 🔹	String data ty 💌	False	•
	15 Phone CLI	phone_di	Phone CL1		alse 💌	String data ty 💌	False	•
							Total R	lows: 25
Policy Tree							Total R	low
								_

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	W/mann admin Faces/mans/home.icm? add one-80624073028242028. addedoubledea.08. add.tdd.tatae.unremates.14	- to Y Manala
Contraction in a		- i v i v i v i v i v i v i v i v i v i
• mance	tou Researce a	
🖌 Favorites 🦷 🖉 Upgrade Y	four Browser *	
Fraud Prevention		😭 🕶 🔯 👻 🖙 🦛 👻 Page 👻 Safety 👻 Tools 👻
DRACLE"		Help Sign Ou
ferences		Signed in as oaar
Fraud Prevention		
Navigation	🛠 Conditions 🚺 Policies 🚺 Payment to Paye 🗒 Over Daily Tran 🗐 Rules 📲 2/A Active 😳 Entitle	s 🗐 Transactions 🗐 Payment 🔳
Actions • View • 🔁 🤮 🎂	🖳 🏹 Over Daily Transaction Limit	Copy Rule Apply Revert
Deshboard		
Cases	Commune Descenditions (1) Desch	
Policy Set	Action + View + A Y TO A B A	
Potoes		
Conditions	TRANSACTION: Check Transaction Aggregrate and Count using filter conditions and parameter Check Transaction Aggregrate and Count using filter of	onditions and parameter
Configurable Actions		
Configurable Actions Action Templates Action Instances		Total Rows: 1
Configurable Actions Action Templates Action Instances Action Instances Image: Action Instaction Instances Image: Action	TRANSACTION: Check Transaction Aggregizate and Count using filter conditions and parameter	Total Rame 1
Configurable Actions Action Templates Action Instances ISBA Questions Validations Categories	TRANSACTION: Check Transaction Appropriate and Count using filter conditions and parameter Select the appropriate function Sum	Ted Rever 1 Sere Const
 Configurable Actions Action Templates Action Instances WAL Questions Validations Categories Registration Logic Ansere Logic 	TRAIRACTION: Check Transaction Aggregrate and Count using lifer conditions and parameter Select the aggregate function Sum Select Sele	Total Roves 1 Same Const
Configurable Actions Configurable Actions Action Templates	TRANSACTION: Check Transaction Aggregate and Count using filter conditions and parameter Select the up rough function Sum • Select Birtly or Binner to work Annual • Seconded Constants for Aggregate Grants Than •	Trid Ases 1
() → Constantiation → Constantiations → Constantiations → Constantiations → Constantiations → Constantiation → C	TEANSACTION: Check Transaction Aggregizate and Count using filter conditions and parameter Select Teau aggregate function 5 m • Select Drity or Dement to count Annuart • Second Condition Kargongite Creater Than • Specified Onder Valers for Aggregate (Transactor).jmit	Teal Assoc 1
Configurable Actions Configurable Actions Configurable Action Instances Construine Table Construine Construine Construine Construine Construine Aquitations Advitation Advitati	TRANSACTION: Check Transaction Appropriate and Count using lifer conditions and parameter Select the appropriate function - Sam • Select Entry or Dennert to court . Anaurt • Seconded Constants fra Appropriate Counter Than • Seconded Constants for Court	Teld Revis, 1 Sere: Career
Configurable Actions discher Impolates discher Instances discher Instan	TRANSACTION: Check Transaction Aggregate and Count using filter conditions and parameter Select the gargets function Sum Generation State State State State Second Condition for Aggregate Generation Second Condition for Court Generation Management Second Condition Court Generation Second Co	Tead Assoc 1
 Gordynale Actors Acton Treaders Acton Treaders Acton Treaders TRA Questrons Valatons Valatons Valatons Valatons RegistratoLogo RegistratoLogo Revener Logo Sintere Signosts 	TRANSACTION: Check Transaction Aggregeriste and Count using Titter conditions and parameter Select Te aggregate function 5 m • Select Editory or Desert to count • Seedled Condition For Gaze Team • Specified Ock Values for Aggregate Teamstorm, Jant • Specified Ock Values for Aggregate Teamstorm, Jant • Specified Ock Values for Count • Specified Ock Values for Count • 0 g Duration 12 • 0 g Transaction Using • hours • 0 • • • • • • • • • • • • • • • • •	Teal Assec 1
Gordynale Astree Gordynale Astree Gordynale Astree Gordynale Gor	TRANSACTION: Check Transaction Aggregate and Count using Ree conditions and parameter Select the aggregate factors Sum • • Select Britly or Binner to sust . Anourt • • Specified Check Value for (* Aggregate : Envanchur, Jack Specified Check Val	Teal face 1
	TRANSACTION: Check Transaction Aggregente and Count using filter conditions and parameter Select The suproper function Sum • Select Entry or Element to count. • Specified Condition for Aggregate Transactory. Entry The Select • Specified Check Visite For Fund Aggregate Transactory. Entry The Select • Specified Check Visite For Count: • Specified Check Visite For Count: • Specified Check Visite For Count: • Transactory States Success •	Teld Rose 1
G Configuration Actions G Action Treatmess C Action Treatmess	TEMISACTION: Check Transaction Aggregate and Count using filter conditions and parameter Select Ed supgraph: function 5m • Select Ed toxity or Elevent to count Select Ed condition for Aggregate Selected Condition for Gaser Specified Deck Value for Gargergate Specified Deck Value for Gasergate Transaction State Specified Deck Value Agergate Value for Gasergate Transaction State Specified Deck Value Agergate Value for Elevent Agergate	Teal Asses 1

Figure 13–7 Add or Modify Fraud Rules in OAAM - Conditions Tab

13.3 Matrix Based Approvals – Adding Attributes

OBP uses OPSS to assert matrix-based approvals. The matrix comprises of various facts.

Example of a matrix-based rule:

```
Grant
Role = CreditAnalyst
Service=com.ofss.fc.app.origination.service.lending.core.credit.decision.CreditDec
isionApplicationService.approveDecision
Action = performWithoutApprovals
IF CreditDecisionMatrix_Margin > 1
AND CreditDecisionMatrix_CustomerExposure > 10000000
```

In the above example, the following facts (attributes) make up the access policy rule:

```
CreditDecisionMatrix_Margin
CreditDecisionMatrix_CustomerExposure
```

The security framework allows for addition to the facts that can be used in rules.

The steps to add facts are same as described in above section.

Note: The only difference between the policy semantics in the example mentioned under this and last action is the 'Action'. ['perform' versus 'performWithoutApprovals']

13.4 Security Validators

In addition to OPSS access policies, there are additional validators that perform security checks. The sole purpose of these validators was to give hooks to enable site

specific security logic that would be complicated enough and hence cannot be provisioned as rules.

Note: These additional validators come into effect only when the following property is set.

```
APPLICATION_SECURITY_VALIDATOR=true
```

The role, channel, service and the attributes available in the execution context are passed to the validators.

The validators implement the interface com.ofss.fc.app.adapter.impl.sms.validator.IExtendableApplicationValidator

There are three types of security-validation categories:

- Customer validators
- Account validators
- Business unit validators

There can be multiple validator classes contributing to each individual category.

The package structure of the validators is required to be:

'com.ofss.fc.app.adapter.impl.sms.validator'

13.4.1 Customer Validators

This validator returns a Boolean signifying whether the logged-in user can perform a transaction on the customer.

Step 1

Add property in ApplicationValidators.properties

com.ofss.fc.app.dda.service.account.core.DDAInquiryApplicationService.fetchBasicDe
tails.CustomerValidators=RestrictedAccountApplicationValidator,EmployeeAccountAppl
icationValidator

Step 2

Develop custom validator along the lines of existing adapters.

13.4.2 Account Validators

This validator returns a Boolean signifying whether the logged-in user can perform a transaction on the account.

Step 1

Add property in ApplicationValidators.properties

com.ofss.fc.app.dda.service.account.core.DDAInquiryApplicationService.fetchBasicDe
tails.AccountValidators=RestrictedAccountApplicationValidator,EmployeeAccountAppli
cationValidator

Step 2

Develop custom validator along the lines of existing adapters.

13.4.3 Business Unit Validators

This validator returns a Boolean signifying whether the logged-in user can perform a transaction on the business unit.

Step 1

Add property in ApplicationValidators.properties

```
APPLY_BUSINESS_UNIT_VALIDATION_TO_ALL_SERVICES=false
com.ofss.fc.app.dda.service.account.core.DDAInquiryApplicationService.fetchBasicDe
tails.BusinessUnitValidators=BusinessUnitApplicationValidator
BusinessUnitValidators=GlobalBusinessUnitApplicationValidator
```

Step 2

Develop custom validator along the lines of existing adapters.

Note: BusinessUnit validation can be global, in which case the following property is set.

APPLY_BUSINESS_UNIT_VALIDATION_TO_ALL_SERVICES=true

13.5 Customizing User Search

OBP application services use SessionContext as an input parameter to set the context of the user interacting with the system. The session-context is populated out of the user's details in OID. Across implementations, the user metadata (objectclasses, attributes) is expected to be different resulting in the requirements to have a custom user search capability.

The security framework provides an extension point to inject a custom search. The steps are given in the next section.

13.5.1 Steps

SecurityConstants.properties contains attributes that enable custom user searches.

Step 1

Add properties in SecurityConstants.properties.

```
CUSTOM_SEARCH_
CLASS=com.ofss.fc.domain.ixface.sms.service.utils.CustomUserSearchAdapter.retrieve
UserUsingExtendableAttributes
CUSTOM_SEARCH_PARAM=nagactualaccessid
```

Step 2

Develop custom user search adapter.

13.6 Customizing One-Time-Password (OTP) Processing Logic

OBP uses OAAM for step-up authentication and fraud assertions. Customer is asked to enter a one-time password (OTP) if OAAM suspects the transaction to be fraudulent. The logic to send or validate an OTP is implemented using a custom hook. Details of the extension are given in the next section.

13.6.1 Steps

OAAM.properties contains a property that provides an extension for second factor password generation / dispatch.

Steps:

1. Add property for the class implementing 2FA in OAAM.properties

TWO_FACTOR_AUTH_SERVICE=com.ofss.fc.domain.ixface.oaam.TwoFactorAuthService

2. Develop custom class.

13.7 Customizing Role Evaluation

OPSS can be configured to add a user in multiple groups (enterprise roles), as a result of which a user can have multiple application roles. OBP uses the most significant role amongst this list to query the user's severity configuration.

The default role-evaluator can be overridden to provide custom role evaluation logic. The steps to do this are given in the next section.

13.7.1 Steps

SecurityConstants.properties contains an attribute that provides an extension for a custom role evaluator.

Step 1

Replace property value in SecurityConstants.properties

ROLE_

```
EVALUATOR=com.ofss.fc.domain.sms.entity.user.roleEvaluationCriteria.SimpleRoleEval uator
```

Step 2

Develop custom role evaluator.

Currently, the default role evaluator returns the role that has the maximum limits for the service.

13.8 Customizing Limits Exclusions

OBP application services evaluate transaction limits for various services. The assertion logic excludes limits checks for certain conditions. Example, if the customer is transferring funds to his own accounts. Banks have site-specific requirements to exclude transactions from limits checks. The security framework provides an extension point to inject a custom limits exclusions adapter. The steps are given in the next section.

13.8.1 Steps

LimitsExclusions.properties contains a property that enables custom limit exclusions logic for a particular service.

Step 1

Add properties in LimitsExclusions.properties

EXCLUSION_PACKAGE_NAME=com.ofss.fc.app.adapter.impl.sms.exclusions com.ofss.fc.app.dda.service.transaction.DemandDepositFundsTransferService.transfer FundsToBeneficiaries=TransferFundsExclusionValidator

Step 2

Develop custom limits exclusions adapter.

13.9 Customizing Business Rules

BPEL approval process business rules can be configured and it is based on input authorizations raised during transaction processing at OBP host. The steps for configuring the business rules of the approvals are given in the below section.

13.9.1 Steps to Update the Business Rules by Browser

Following are the steps to update the business rules by browser.

1. Log in to BPM Worklist application of the OBP.

Figure 13–8 Log in to BPM Worklist Application screen



2. Select the 'Task' in the select box from the 'Task Configuration' tab in 'Administration'.

BPM Worklist ×									
← → C 10.180.5.20:8001/integration/w	vorklistapp/faces/a	dministration.jsp:	x?_adf.ctrl-state=o	v9nod5jc_28				~	값 =
ORACI E RDM Worklist	Market Firms		2	Loggodin ar	wohlogis	Homo Administrati	an Departs	Drafaransas	Help Legant
CRACLE BPM WORKIS			U.	Logged In as	s weblogic	Administratio	on Reports	Freierences	nep togout
Administration Fordence Search Approv. ♥ Tasks to be configured ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	Task Task Elitr, DemandDa Schut, On Er Assignment and Rout	popsitCashTrans regation None or Notify Allow all a Allow the Pable Complete Complete Complete Complete Scalation Policy	actionServiceSpi_ articipants to invite other incipants to edit future par- ticipants to edit future par- ticipant of edit to dein taki inter participant of early completion of paral lete parent tasks of early	DepositCash participants ticipants voses Reject el subtasks completing subta	ssks	iven Configuration			
	Vitotification Sett Teak Status Assgn Complete Error	Ings Recipient Assignees Initiator Owner	Notification Head	☐ Make not ☐ Make not ☐ Send task	ifications secur ification actiona k attachments v	e (exclude details) bble with email notifications			
	2) Task Access								

Figure 13–9 Task Configuration tab

3. In the 'Rules' tab of the 'Task Configuration' screen, select the stages of approval where the condition in rule is required to be updated.

Figure 13–10 Stages of Approval

🕒 BPM Worklist 🛛 🗙 💽				- 0 ×
← → C [] 10.180.5.20:8001/integration/	worklistapp/faces/administration.jspx?_adf.ctrl	-state=ov9nod5jc_28		☆ =
🗋 Oracle Important Sites 📋 NGP URLs 📄 Java 🗎	Market Firms			C Other bookmarks
ORACLE BPM Worklist		Logged in as weblogic	Home Administration Reports	Preferences Help Logout Q
Administration Evidence Search Approv	al Groups Task Configuration			
🗵 Tasks to be configured 🖉 🖓 🗖 🔯	Task Rules			
HT_DemandDeposit Image: Control of the section of the se	Data driven configuration Configure list builder properties via task data-driven rules Select Ruleset Stage1:ST1_PT1_RS			Add variable
법) H _ DemandDeposit.can / Innaction _ Jeposit.can (L) 题 H _ DemandDepositFundsTransaction _ Veborat.can (L) M T_ DemandDepositFundsTransferServiceSpi _ Transferf 원 HT_ DemandDepositFundsTransferServiceSpi _ Transferf 원 HT_ DemandDepositFundsTransfer _ TransferfUndsToBe	ST1_PT1_R Stape: ST2_PT1_RS Modification Rules Wew IF/THEN Rules	1-3 of 3		Ignore this participant
	≥ ¥ Rule1 🕂 🖓 🖓			
	≥ ¥ Rule2 🕹 🖓 🕹			
	≥ × Rule3 💠 🛠 🏠 🖓			
۲				

4. After stage selection, select the specific rule where the condition needs to be updated. The existing condition can be updated or the new test condition (simple/variable) can be added.

🕒 BPM Worklist 🛛 🗙 📃					- 0 ×
← → C 10.180.5.20:8001/integration/	vorklistapp/faces/administration.jspx	?_adf.ctrl-state=ov9r	od5jc_28		🔂 🔳
ORACLE' BPM Worklist	Warket Films	65	Logged in as weblogic	Home Administration	Reports Preferences Help Logout
Administration Fuidence Search Annrov	I Groups Task Configuration	44			
✓ Tasks to be configured 🖉 🖓 🗟 🔯	Task Rules				
HT_DemandDeposit HT_DemandDepositCashTransactionServiceSp HT_DemandDepositCashTransactonServiceWindraw HT_DemandDepositCashTransacton_DepositCash(10 HT_DemandDepositCashTransacton_DepositCash(10 HT_DemandDepositCashTransetor_WindrawCash(1) HT_DemandDepositCashTransferServiceSpi_Transfer HT_DemandDepositFundTransferServiceSpi_Transfer HT_DemandDepositFundTransferServiceSpi_Transfer HT_DemandDepositFundTransferServiceSpi_Transfer HT_DemandDepositFundTransferServiceSpi_Transfer HT_DemandDepositFundTransfer_TransferFundTroBe	Data driven configuration Configure list builder properties via task data-dri Select Ruleset Stage 1: ST1_PT1_RS ST1_PT1_RS	iven rules			Add variable
	View IF/THEN Rules ▼ ♣	 ↓ ↓ ↓ ↓	•		
	Selected Tests 📃 🖵	()₂ ()₃ ᠿ 小 Q isn't	v nul	e	🛦 and 🕶
< •	Task.payload.reasonCodes	Q contains	UUAL_CONTROL		simple test variable
	List Builder Resource Response Type Required FYI				
	Users Task.payload.manag Particpants Groups null Application Role null	erid Q			
	≥ ¥ Rule 1 ♣	X 🕆 🖟			
	≥ ¥ Rule2	🗙 🔂 🕹			
	N ¥ Rule3	X 🔒 🖟			

Figure 13–11 Select Test Condition

5. After selection of the test condition, the new expression row appears where the variable, the operator and the expression value can be selected.

Figure 13–12 Select Values

🗅 BPM Worklist 🛛 🗙 💽						_ @ X
← → C 🗋 10.180.5.20:8001/integration/	worklistapp/faces/administration.jsp:	x?_adf.ctrl-state=ov9r	nod5jc_28			☆ =
🗋 Oracle Important Sites 📋 NGP URLs 📋 Java 📋	Market Firms					C Other bookmarks
ORACLE' BPM Worklist		612	Logged in as weblogic	Home Administration	Reports Pre	eferences Help Logout Ϙ
Administration Evidence Search Approva	al Groups Task Configuration					
Tasks to be configured ✓ ✓ ✓ ✓ ✓ ✓ ✓	Task Rules					
HI_DemandDeposit	Data driven configuration	li un alla				Add variable
HT_DemandDepositCashTransactionServiceSpi_Withdra HT_DemandDepositCashTransactionService_Withdrawt	Select Ruleset Stage1: ST1 PT1 RS	inventules				
HT_DemandDepositCashTransaction_DepositCash (1.0 HT_DemandDepositCashTransaction_WithdrawCash (1	ST1 DT1 DS					Innore this participant
HT_DemandDepositFundsTransferServiceSpi_TransferF HT_DemandDepositFundsTransferServiceSpi_TransferF	311_F11_K3					
HT_DemandDepositFundsTransfer_TransferFundsToBe	View IF/THEN Rules 💌 🔮	- <= <u>1-4 of 4</u>				
	⊻ ¥ Rule1	• 🗙 🔂 🥾				
	IF.					
	Selected Tests 🖉	()과 ()공 合 🕹				
	Task	🔍 isn't	💌 nul		💊 and 🗸	
	Task.payload.reasonCodes	🔍 contains	UUAL_CONTROL		and -	
		🔍 is	•		2 -	
	THEN List Builder Resource Response Type	9erid Q Q • X ☆ ↓ • X ☆ ↓				
ł					_	
0.180.5.20:8001/integration/worklistapp/faces/administrati	on.jspx?_adf.ct					

6. On selection of the search button next to the variable select box, the condition browser opens where the specific task can be selected.

Figure 13–13 Select Specific Task

7. Update the variable, operator and value of the expression in a row.

Figure 13–14 Update Values



8. Save the updated rule using the save button in the left side menu.

C 10.1805.20.8001/integration/worklistapp/faces/administrationjaps?_adf.ctrl-state=ov9nod5jc_28 Oracle Important Site NoP URs watet firms C 10.1805.20.8001/integration/worklistapp/faces/administrationjaps?_adf.ctrl-state=ov9nod5jc_28 Oracle Important Site NoP URs watet firms C 10.1805.20.8001/integration C 10.1805 Oracle Important Site NoP URs watet firms C 10.1805.20.8001/integration C 10.1805 Oracle Important Site NoP Umportant Site Important Site C 10.1805.20.8001/integration C 10.1805 Important Site NoP Umportant Site Important Site D 20.1805 T 20.1805 C 10.1805 Important Site Important Site D 20.1805 D 20.1805 D 20.1805 C 10.1805 Important Site Important Site D 20.1805 D 20.1805 D 20.1805 C 10.1805 Important Site Important Site D 20.1805 D 20.1805 D 20.1805 D 20.1805 D 20.1805 Important Site Important Site D 20.1805	🕒 BPM Worklist 🛛 🗙 📃				- 0 ×
In cack important Site: MAP UIL: Java Market Firms Important Site: Market Firms <	← → C 🗋 10.180.5.20:8001/integration/	worklistapp/faces/administration.jspx?_adf.ctrl-state=0	ov9nod5jc_28		☆ =
CPRACLE BPM Worklist Is upged in as webgig: Home Administration Reports Preferences Heb La Ministration Vederace Search Approval Course Task Configuration Is de configured The Demandlopositie Configured Image: Configured Image: Configured If Demandlopositie Configured Image: Configured Image: Configured Image: Configured If Demandlopositie Configured Image: Configured <th>🗀 Oracle Important Sites 📄 NGP URLs 🛅 Java 📋</th> <th>Market Firms</th> <th></th> <th></th> <th>📋 Other bookmark</th>	🗀 Oracle Important Sites 📄 NGP URLs 🛅 Java 📋	Market Firms			📋 Other bookmark
Administration Vedeoce Search Approved Groups Task Configuration If T_DemandbookSinkt masschondernates Configuration Add ventable If T_DemandbookSinkt masschondernates Singer 1: STL_PTI_PS Ispace this balder properties vis task data driven nules If T_DemandbookSinkt masschondernates Singer 1: STL_PTI_PS Ispace this partial If T_DemandbookSinkt masschondernates Singer 1: STL_PTI_PS Ispace this partial If T_DemandbookSinkt masschondernates Singer 1: STL_PTI_PS Ispace this partial If T_DemandbookSinkt masschondernates Singer 1: STL_PTI_PS Ispace this partial If T_DemandbookSinkt masschondernates Ispace this partial Ispace this partial If Task Ispace this partial Ispace this partial Ispace this partial If the partial masschondernates Ispace this partial Ispace this partial Ispace this partial If the partial masschondernates Ispace this partial Ispace this partispartial Ispace this partiali	ORACLE BPM Worklist	Į.	Logged in as weblogic	Home Administration Reports	Preferences Help Logout 🧲
Image: State to be configured Image: State and the configuration Add variable Image: State to be configured to the configuration Configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configured to the configure to the table driven rules Image: State to be configure to the table driven rule to the t	Administration Evidence Search Approva	al Groups Task Configuration			
IT Demandlepost Image: Strate Configuration Add variable If The Demandlepost Chaft TransactionServices With Demandlepost Chaft TransactionServices Strate Configuration If The Demandlepost Chaft TransactionServices Strate Configuration Strate Configuration If The Demandlepost Chaft Transaction Peopost Chaft Transact Transfer Fund Total If The Demandlepost Chaft Transaction Peopost Chaft Transaction Peopost Chaft Transfer Fund Total If The Demandlepost Chaft Transaction Peopost Chaft Transfer Fund Total If The Demandlepost Chaft Transaction Peopost Chaft Transfer Fund Total If The Demandlepost Chaft Transfer Fund Total If The Demandlepost Chaft Transfer Fund Total If The Demandlepost Chaft Transaction Peopost Chaft Transfer Fund Total If The Demandlepost Chaft Transfer Fund Total <th>✓ Tasks to be configured</th> <th>Task Rules</th> <th></th> <th></th> <th></th>	✓ Tasks to be configured	Task Rules			
Image: Image	HT_DemandDeposit	Data driven configuration Configure list builder properties via task data-driven rules Select Ruleset [Stape1:ST1_PT1_RS] ST1_PT1_RS Information View × Rule1 If Selected Tests (bp (bg ∩))	are saved successfully OK		Add variable
	x	Task-payload reasonCodes Task-payload reasonCodes Task-payload transactionBranch THEN Let Builder Resource Response Type Required Pri Participants Groups nul Application Role nul	▼ 704L_CONTROL* ▼ 704L_CONTROL* ▼ 78542*		

Figure 13–15 Save the Updated Rule

9. Commit the changes in the rule by clicking the commit button.

Figure 13–16 Commit the Changes



Note: 'Ignore this participant' check box is available on the screen for ignoring the specific stage. The particular stage is then ignored while consideration of the rules implementation in the approval process.

13.9.2 Steps to Update the Business Rules in JDeveloper

Following are the steps to update the business rules in JDeveloper.

Step 1

Configure the JDeveloper in the customization option and the particular process jar import as the SOA project in the customizable mode. The details of this step are explained in this document in the section SOA customization.

Step 2

Expand the Business Rules folder of your project. You will see two .rules files in it. One will be <<HumanTaskName>>Rules.rules file and the other will be <<HumanTaskName>>RulesBase.rules file. Double Click and open the <<HumanTaskName>>Rules.rules file. The existing approval stages and rulesets will be available in the file.



Figure 13–17 Expand Business Rules

Step 3

Create a new stage in the format 'ST<Stage Number>_PT1_RS' by clicking the '+' button in the Rulesets. The new rules/decision table can be added to a stage.





Step 4

Add the new rule by clicking the '+' button on the stage. The existing rule can also be added/modified in the existing stage.


Figure 13–19 Add New Rule

Step 5

Populate the rule with the conditions in 'if' and 'then' block.



Figure 13–20 Populate the New Rule

Step 6

Deploy the project jar as explained in this document in the section SOA customization.

Note: All the rules should have the final 'THEN' statement with the return type as 'retract Task'. 'retract Task' makes sure that if the condition of the rule is satisfied then the second rule should not be evaluated else the flow will execute the entire ruleset. It is also mandatory to have the last rule with the final 'THEN' statement as 'call IgnoreParticipant'. This is done to bring the control out of the ruleset.

	• 🖉 • 🔝	esk × 🎯HT DemandDepositCashTransaction WithdrawCashRules.rules 🛪 الکات
Restensibility ▼ ™ Projects ⊕ @ com.ofss.fc.approval.demanddepositcashtransaction ⊕ @ SOA Content	 ✓ (□) (□) (□) (□) (□) ✓ (□) (□) (□) (□) ✓ Facts f_x Functions (□) (□) (□) (□)) 내 생 ③
Com ofss. fr. approval demanddeposit.cashtransaction Com of the figure of the	(X) Globals C Bucketsets Links Decision Functions Rulesets ModificationRules SubstitutionRules St1_PT1_RS St5_PT1_RS St3_PT1_RS St4_PT1_RS St4_PT1_RS	IF Vale1 DUAL_CONTROL with Manager IF Task != null and Task.payload.reasonCodes contains ("DUAL_CONTROL") <insert test=""> ▼ THEN call CreateResourceList(users : Task.payload.managerid , groups : null , approles : nu retract Task <insert action=""></insert></insert>
Web Content Web Content Web WEB-NF To com.ofss.fc.extensibility.samples.OptionCC Application Resources Connections Descriptors META-INF META-INF	2	<pre>insert action></pre>

Figure 13–21 Deploy Project Jar

Loan Schedule Computation Algorithm

OBP application provides the extensibility by which the additional loan schedule computation algorithm can be added or customized based on client's requirement.

14.1 Adding a New Algorithm

For adding a new algorithm following additions need to be done.

LoanCalculationMethodType.properties contains list of available loan stage algorithms in the system in the form of key-value pairs. For example, ARM=ARM

This list is used as part of screen LNM43 to populate a drop down called Computation Formula.

An entry has to be made in this file to appear as a choice in the drop-down list.

Figure 14–1 Add New Algorithm

					OFS	ISUser, La	st Login 22-Apr-2013 02:3	3:32 PM , Posting Date	0-Jun-2013 🕨	Business Un	IL SUNCORP	BANK	# ?	i o	0	2
A	count = Back (office - CASA -	Channe	I - Collection -	LCM -	Loan •	Operational Services •	Origination • Party	Payment And Colleg	tion = Sex I	8 4 🕯 -	FastPath				•
1	nstallment Ru	LNMR8 le Details						G Beac	+ Crigate 📬 Update	Close D	Reopen #	à Print 🗸	ok 🖌	Cjear 🕻	Cancel	
	Pule Informatio	n	Rula ID	ABC					• Rule Name	ABC						1
	Computation		THE REAL							- av						i
6	Dates	Computation	Formula	ARM												
	• Date Basis	for installment or	Principal	Calendar 💌					Date Basis for Interest	Calendar						
•	Rounding Rules	• Roundin	g Method	No Round 💌		-		Near	ist Amount to Round To		-	-				8
																1

This screen is used to create a new Installment Rule. For example: ABC. We can choose the new algorithm for the new rule.

Figure 14–2 Create New Installment

	ATFORM			OFSSUser, I	Last Login 22-Apr-2	2013 02:33:32 PM , Po	sting Date 30-	Jun-2013 🕨	Business	Unit SUNCOF	RP BANK 💌	8 ?	i o	0
Account •	Back Office •	CASA - Chann	el • Collection •	LCH - Loan	Operational Se	envices • Origination	• Party •	PaymentAnd	Collection - Sev	- 16 1	 FastPat 	n 📃		_
🖀 🕠	M43 UNI	198												
Loan Sc	hedule Type	,				1 O Burd	A Custa	Hiladaha IV D	alata 🗃 Cinca	a	and a		Cierce #	Case
						ren Dana	4 cillan	# Zhowe # P	inene 🖬 crose	L' neogen	Bill Print	• 05 •	oless.	Carlo
Schedu	le Details													
		Schedule Code	sch					 Schedule Nam 	sch .					
🗟 Definitio	on											+ Cri	nate × D	leiete
• Maximur	n Interest Only 1	ferm Allowed		Months										
View •	🔤 i 🛃 D	etach												_
Stage Namber	Stage Name	Installment Pale	Default Payments or Term	Installment Frequency	Interest Repayment Frequency	Interest Capitalization Frequency								
1		102		None Or At Matur	ity None Or At Matu	nity None								
🖸 Details														
Stage ty	pe													-1
		 Stage Name 						Stage Numb	er 1					- 1
		 Installment Rule 	102	Q			Insta	Iment Rule Nam	e MOR					- 1
	Treatm	Instalment Type ent for Periodic Fee	Capitalize				Default	Payments or Ten						
harment	Oraca carina													=
inceresc	Processing													-1
	• End	of Period Treatment	Capitalize Arreans	+ UnCharged Inte	nest 💌	• 6	terest Compo	unding Frequen	ty Monthly 💌)				
	Re	st Period Frequency	None			^	means Compo	unding Frequen	ty None					
Installine	ent Details													
Show M	edification Histo	ну												

Screen LNM98 is used to create new schedule codes from existing instalment rules. A new schedule code can be made using the new instalment rule created above.

A schedule generator class is created to implement a schedule code. The property file **ScheduleCalculator.properties** stores this relation in the form:

Schedule_Type_Code=Schedule_Calculator_Class

If a new schedule generator class is created for the new schedule code above, an entry in this file has to be made.

Example: IOI-EIPI-PMI_IntOnly-Month_Pr-Month_Ann= com.ofss.fc.domain.schedule.loan.generator.NewPrincipalRepaymentScheduleGenerator;

The key is the SCHEDULE_CODE column in the table FLX_SH_SCHEDULE_TYPE_B.

The **PrincipalRepaymentScheduleGeneratorFactory** reads this property file and creates an instance of the schedule generator class associated with the schedule type code passed. The following code snippet shows how it is done

```
IPrincipalRepaymentScheduleGenerator calculator = null;
String calculatorClassName = calculators.get(scheduleTypeCode);
calculator = (IPrincipalRepaymentScheduleGenerator) ReflectionHelper.getInstance()
.getClassInstance(calculatorClassName);
```

// If schedule calculator is not found then do nothing

```
if (calculator == null) {
```

```
calculator = new PrincipalRepaymentScheduleGenerator();
}
```

Currently, in the application this property file is empty and hence an instance of PrincipalRepaymentScheduleGenerator is returned by default.

The new schedule generator class has to implement the interface IPrincipalRepaymentScheduleGenerator which is the base for all schedule generators.

The important methods in it are:

public SortedMap<Integer, PrincipalRepaymentPeriod> defineStages(SortedMap<Integer, PrincipalRepaymentPeriod> repaymentStages, AccountScheduleAttributesDTO accountParameters, Money amountForScheduleGeneration, Date onDate); public LoanScheduleCalculatorOutputData defineSchedule(Date definitionDate, SortedMap<Integer, PrincipalRepaymentPeriod> repaymentStages, AccountScheduleAttributesDTO accountParameters, SortedMap<LoanInterestType, List<NetRateDTO>> interestRates, Money mountForScheduleGeneration); public LoanScheduleCalculatorOutputData generateRepaymentRecords(Date generationDate, SortedMap<Integer, PrincipalRepaymentPeriod> repaymentSchedule, AccountScheduleAttributesDTO accountParameters, Money totalPrincipalToRepay, SortedMap<LoanInterestType, List<NetRateDTO>> interestRates, List<PrincipalScheduleTransaction> scheduleTransactionHistory, SortedMap<Date, PrincipalScheduleInterestBase> interestBaseHistory, SortedMap<Date, Money> feeDetails):

The method generateAndSavePrincipalSchedule() of ScheduleApplicationService creates and processes the instance of a schedule generator as follows:

IPrincipalRepaymentScheduleGenerator scheduleGenerator =
PrincipalRepaymentScheduleGeneratorFactory.getInstance().createScheduleGeneratorIn
stance(accountParameters.getScheduleTypeCode());

The methods in the schedule generator call the business logic for the instalment rules (stage algorithms) part of the schedule code. This logic is written in a Stage generator class. New Stage generator class has to be created for the new algorithm created above.

For example, EPIARMRepaymentStageGenerator.class is created for EPI and ARM.

This class has to implement interface **IPrincipalRepaymentPeriodGenerator** which is the base for all stage generators. The important methods in it are:

public void defineStage(LoanRepaymentStageDTO repaymentStage); public void define(LoanRepaymentStageDTO repaymentStage,AccountScheduleAttributesDTO accountParameters,Date definitionDate, List<NetRateDTO> interestRates, SortedMap<Integer, LoanRepaymentStageDTO> allRepaymentStages, SortedMap<Date, PrincipalScheduleInterestBase> interestBaseHistory, List<PrincipalScheduleTransaction> scheduleTransactionHistory); public SortedMap<Date, LoanRepaymentRecordDTO> generate(LoanRepaymentStageDTO repaymentStageToBeGenerated, AccountScheduleAttributesDTO accountParameters, Date generationDate, List<NetRateDTO> interestRates, SortedMap<Integer, LoanRepaymentStageDTO> allRepaymentStageS, SortedMap<Date, RepaymentDate> repaymentDates, SortedMap<Date, LoanRepaymentRecordDTO> allRepaymentRecordS, SortedMap<Date, PrincipalScheduleInterestBase> interestBaseHistory, List<PrincipalScheduleTransaction> scheduleTransactionHistory, SortedMap<Date, Money> feeDetails); The entry for the new Stage generator class has to be made in **StageCalculator.properties**.

For example, ARM=com.ofss.fc.domain.schedule.loan.generator.EPIARMRepaymentStageGenerator

The **PrincipalScheduleRepaymentPeriodGeneratorFactory** class reads this property file and based on the installment rule passed as parameter creates an instance of its corresponding stage generator class. The following code snippet shows it

IPrincipalRepaymentPeriodGenerator stageGenerator =
PrincipalScheduleRepaymentPeriodGeneratorFactory.getInstance()
.createStageGeneratorInstance(repaymentStage.getInstallmentRule())

14.2 Consuming Third Party Schedules

As mentioned above the PrincipalRepaymentScheduleGeneratorFactory reads the property file ScheduleCalculator.properties which has entry for the schedule generator class to be used for a schedule code. For using third party schedule algorithms, an entry in this file has to be made against the required schedule codes.

IOI-EIPI-PMI_IntOnly-Month_Pr-Month_Ann=
com.ofss.external.ScheduleAlgoExt.XYZScheduleGenerator;

15

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.

15.1 Prerequisite

Following are the prerequisites for receipt printing.

15.1.1 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 15–1 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 15–2 Build Path of Utility

Properties for VO	OAttributes			
type filter text 🛛 🕱	Java Build Path	(> ▼ ⊂> ▼		
 Resource Builders 	Bource BProjects Libraries Source Source Projects Source So	rt		
Java Build Path	JARs and class folders on the build path:			
▶ Java Code Style	CUSTOM_LIB/commons-io-1.4.jar - /home/vishal/	Add JARs		
 Java Compiler Java Editor 	 CUSTOM_LIB/tld/xalan.jar - /home/vishal/eclipse CUSTOM_LIB/tld/xmlparserv2.jar - /home/vishal/ 	Add External JARs		
Javadoc Location	JRE System Library [JavaSE-1.6]	Add Variable Add Library Add Class Folder		
PMD				
Project References				
Run/Debug Settings Server		Add External Class Folder		
Subversion		Edit		
Task Tags		Remove		
 Validation WikiText 		Migrate JAR File		
?		Cancel OK		

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

Figure 15–3 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.

Fiaure	15–4	Excel	Generation

A	В	С	D	E
L 🛛 Task Code 🖓	View Object 💽	Attribute Name 🛛 💌	Attribute Type 🔹	RTF Node 💌
15 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	accountNo	java.lang.String	FundTermDepositVO_accountNo?
16 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	principalBalance	java.math.BigDecimal	FundTermDepositVO_principalBalance?
17 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	payinAmount	java.math.BigDecimal	FundTermDepositVO_payinAmount?
18 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 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	productCode	java.lang.String	FundTermDepositVO_productCode?
22 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	partyld	java.lang.String	FundTermDepositVO_partyId?
23 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 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	secondaryReason	java.lang.String	FundTermDepositVO_secondaryReason?
26 TD002	com.ofss.fc.ui.model.td.mixedpayin.vo.FundTermDepositVO.xml	narrative	java.lang.String	FundTermDepositVO_narrative?
Ŷ	$\hat{\mathbf{U}}$	Ŷ	Ŷ	Ŷ
Task Code	View Object Path	VO Attribute	Attribute Type	Reference in
	(Screen/taskflow)	Name		RTF template

15.1.2 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 15–5 Receipt Format Template

	Document in D:\SVN\extens	ibility\OBP-Extensibility_Guide.docx [Compatibility Mode] - Microsoft Word	_ = X
Home Insert	Page Layout References Mailings Review	View	A A God y
	imes New Roman 🔹 10 🔹 🗛 👘 🗄 🗄 🗄	「示」 連連計 「 AaBbCcI AaBbCcI AaBb(AaBbCc	AaBbt
Clipboard	B Z <u>U</u> * abe X, X [*] Aa* <u>** A</u> * Font G	Paragraph F Styles	Title Styles + Select + Editing
L	1 + 1 + 1 + 2 + 1 + 3 + 1 + 4 + 1 + 5 + 1	· 6 · 1 · 7 · 1 · 8 · 1 · 9 · 1 · 10 · 1 · 11 · 1 · 12 · 1 · 13 · 1 · 14 · 1 · 15 · 1 · 16	1 • 17 4 • 18 • 1 • 19
N			^
-			
-			
-	Bank Name	BankShortName?	
-	Branch Address	BranchName?	
	Posting Date & time:	PostingDate?	
- 	Transaction Ref No:	TransactionRefNo?	
- -	Event Name:	Fund Term Deposit	=
	Account Number:	AccountDetailsVO_accountNo?	
5	Deposit No:	PayinDepositNo?	
10 -	Account Title:	AccountDetailsVO_accountTitle?	
·	Account Currency:	AccountDetailsVO_currency?	
-	Payin Amount:	PayinAmount?	
	Value Date:	ValueDate?	
	Net Interest Rate:	TDPayinDetailsVO_netRate? %	_
- 10	Maturity Date:	MaturityDate?	
- 11	Deposit Term:	TDPayinDetailsVO_years? years :	
- 13 -	Interest Payout Erequency:	<2TDPayinDetails/O_intPayoutErog2>	
13 -	Narrativo:	<2TDPayinDetailsVO_intrayouti req:>	
4 - -	ivarative.		
51 12			
- 16 -			
			* •
Page: 1 of 1 Words: 62 En	glish (India)		

15.2 Configuration

This section describes the configuration details.

15.2.1 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.

15.2.2 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 15–6 Receipt Print Reports

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

15.3 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 15–7 Sample of Print Receipt

🔁 Acroba	at Docum	ient - Adi	obe Reader				- d 🔀
File Edit	View	Window	Help				×
4 🗟) 📝	4 E	🖞 🖨 🖂	27% 💌 🔚 📴 🤛 🔛	Fools	Sign	Comment
							<u>^</u>
		в	ank Name	SUN			
		В	ranch Address	CEO SUNCORP BANK			
		P	osting Date & time:	28-Feb-2013			
		т.	represention Def No.	050003283602			=
			ransaction Rel No.	039002382003			
		E	vent Name:	Fund Term Deposit			
		A	ccount Number:	0000012110			
		D	eposit No:	30			
		A	ccount Title:	Derk Zing			
		^	count Currency:	AUD			
			count currency.	A00			
		P	ayin Amount:	10,000,000.00			
		V	alue Date:	28-Feb-2013			
		N	et Interest Rate:	4 %			
		М	laturity Date:	28-Sep-2014			
		П	enosit Term:	1 years 17 months 10 days			
			eposit renn.	r years . r montais . o days			
		In	terest Payout Frequency:	YEARLY			
		N	arrative:	Bank			

15.3.1 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

15.4 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().getInternalReferen
ceNumber());
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.

Facts and Rules Configuration

This chapter explains the facts and rules configuration details.

16.1 Facts

Fact (in an abstract way) is something which is a reality or which holds true at a given point of time. Business rules are made up of facts.

A fact can be classified in two ways:

- Literal Fact Any number, text or other information that represents a value. It is a fixed value. For example, 100, 2.95, "Mumbai"
- Variable Fact A fact whose value keeps changing over a period of time For example, Account Balance, Product Type.

For example, If a customer maintains an Average Quarterly Balance of Rs.10,000 then waive off his quarterly account maintenance fees. Here, the Average Quarterly Balance is a variable fact while the Rs.10,000 is a literal fact.

16.1.1 Type of Facts

There are two types of facts:

- Direct Facts with input name value pair
- Derived Facts

Services will be exposed for various operations on the facts. These services are broadly classified into two types:

- Fact Inquiry Service
- Fact Derivation Service

For deriving the fact value, different type of datasource can be used:

- Java DataSource Derivation from Java class
- HQL DataSource HQL Query column
- JDBC DataSource SQL Query column
- DbFunction DataSource Derivation from database function

Fact Definition can be further categorized into:

 Fact Value Definition - Definition to Derive Fact Value, It is used mostly in Rule Execution

16.1.2 Facts Vocabulary

Facts Vocabulary is a list or collection of all facts pertaining to a specific field or domain. A standard vocabulary of facts aids users in defining their business rules. For example, the Facts Vocabulary of the Banking domain can contain common and familiar facts such as Account Balance, Customer Type, Product Type, Loan-To-Value Ratio. The Facts Vocabulary of the Cards domain may contain common facts such as Total Credit Limit, Available Credit Limit, Available Cash Limit.

A vocabulary is defined for variable facts. Each fact has a definition and can have source information.

Definition

The fact definition indicates common properties of the fact such as its name, its data type, which domain, domain category and group it belongs to, key for retrieving value and a brief description.

Variable facts would be defined for a domain and a domain category. Domain categories are the sub-systems inside a domain. For example, Lending, Term Deposits, Demand Deposits are the categories of Banking domain. There are some variable facts which would be common across all the categories in a given domain. For example, Customer and Bank data is common for all the categories of Banking domain. Such facts can be classified under a special category called "Global".

The facts are further categorized under various groups. One fact can belong to one or more Groups. For example, In a Banking domain, Customer Type, Birth Date, Gender are Global facts belonging to the group Individual Customer Details. Account Balance, Account Opening Date are facts in Lending category belonging to the group Account Details. Loan-to-value (LTV) ratio, Sanctioned Amount are Facts in Lending category and belong to multiple groups such as Consumer Loan, Home Loan, Agriculture Loan. There are some variable facts which do not really fall into any specific group, such facts are classified under a special group called "Others".

A variable fact value can be received as input from the consumer of eRules in the form of key-value pair, the key here is defined as *RetrievalKey*. The fact will also have a data source for value derivation in case the fact value is not an input.

Some variable facts can have a permissible list of values defined and the rule creator will be restricted to use only those values which are defined in the permissible list of a given fact. All facts will have a *FactValueType* defined as either *Enumerated* (indicates that the fact has a permissible list of values) or *OpenEnded* (indicates that the fact value is a free text). For facts with *FactValueType* as *Enumerated*, data source information will be defined in the vocabulary to derive the list of values.

Variable facts will have a grouping based on BusinessDataType. For example, Variable facts like Transaction Amount, Sanctioned Amount, and Disbursed Amount can be grouped under "Amount". Variable facts like Available Balance, Book Balance belong to "Balance" BusinessType and so on.

These BusinessDataType will in turn have PrimitiveDataType. For example, Amount and Balance will have PrimitiveDataType as double.

With the help of BusinessDataType grouping a list of facts belonging to a particular group can be displayed for user selection while defining rules, rate charts, policies and so on. During actual rule execution the respective *PrimitiveDataType* (that is, int, double, String and so on) of the BusinessDataType will be used.

Literal facts will only have a PrimitiveDatatype.

Source

Some facts can be derived, if they are not received as input. Also associated with some facts is a list of permissible values for the fact at the time of rule/policy definition. All these information forms the part of source data. The Fact Derivation layer is responsible for deriving the value of a fact and the list of permissible values for the fact based on source information defined in the vocabulary.

Deriving Enumeration (applicable list of values) for a Fact

A Variable fact can hold any value at a given point of time. But some can have a standard set of applicable values defined and the value held by such facts would be always within the range of this list of values.

For example, Account Balance as a variable fact can hold any value at a given point of time, a set of values cannot be defined for such facts. Hence, no list of permissible values will be defined for Account Balance. However, the variable fact Customer Gender can have only one of two possible values namely - Male or Female.

While defining the rules, the permissible list of values will be derived for such facts and user selection will be restricted to this list.

Deriving Value for a Fact

During rule execution, a set of fact information will be sent by the consumer of eRules in the form of key-value pair. But this might not be a complete set of fact information required for executing pricing rules. Hence some facts will have to be derived if they are not received as input.

During rule execution, the required facts would be determined, value will be fetched from *RetrievalKey* of the fact if received as input else the value will be derived.

16.1.3 Generation of Facts using Eclipse Plug-in

The fact objects can be generated either by populating the database tables directly or by using the eclipse plug-in. This plug-in is created for fact generation purpose in OBP application.

A local host server needs to be configured in eclipse before processing for configuration of the fact plug-in. For fact generation purpose, the following steps need to be followed.

Get the Fact Plugin from the development team.

Put the plugin (com.ofss.fc.util.plugin.fact_1.0.0.jar) in the plug-in folder of eclipse.

Restart Eclipse

1. In eclipse, go to Window -> Preferences.

Java EE - config/properties/UiConfig.properties - Eclips File Edit Navigate Search Project Run Window Help New Window - 🔲 🗊 🚳 🖧 🎄 🕶 🚺 😭 🖆 Team Synchr... 🤬 Java EE 📑 🕶 🔚 🖷 🇁 New Editor - -**Open Perspective** 陷 Project Explorer 🛛 🦳 📄 🐄 🗄 Tomcat v6.0 Server a 🛛 🔊 ApplicationEnterpris 💦 🎇 ×. Show View × 012, Oracle and/or its affiliates. All rights reserved. ScheduleGenerator.properties 235 * rnate configuration file path pntext details for <u>servlet</u> implementation B Search.properties 221450 5/28/12 Customize Perspective... SecurityAnnotations.properties 2 Save Perspective As... SecurityConstants.properties 229 serviceextensions properties 2150 Reset Perspective... de=08 ServiceExtensionsExecutor.proper de=BRN **Close Perspective** ServiceMap.properties 215052 5/ h=089999 Close All Perspectives hit=BU01 ServiceProviderExtensions.prope ity=ME01 SettlementPreferences.properties Navigation up.code=BRN_GRP_1
ormation in status to client, for example : Error Reference number
.url=http://localhost:9090/com.ofss.fc.channel.branch/HTTPListener
.url=http://localhost:9090/com.ofss.fc.channel.branch/HTTPListener StageCalculator.properties 23310 Web Browser . StageTypeMap.properties 215052 protocol=EJB Preferences StandingInstructionConsumer.pr 16 host.invocation.protocol=JSON brotocol=SOAP StatementPreferences.properties 2150 StateTransitionClassFactory.properties 2 Statistics.properties 215052 5/12/12 4:46 18 # Indicates the layer exposed 0 the host side which will be consumed by the ADF UI. 19 # The layer can be either APP or APPX TaskInitializerConfig.properties 215052 ! 20 host.invocation.layer=APP TaxEvent.properties 215052 5/12/12 4:46 21 TermDeposit.properties 215052 5/12/12 22 # Menu provider: FILE/OPSS TransactionCodes.properties 233888 7/1 23 ui.menu.provider=FILE TrendConfig.properties 225350 6/9/12 1 24 UiConfig.properties 229340 6/20/12 3:59 4 Upload.properties 235046 7/23/12 11:34 - 0 📮 Console 🚮 Proble 🛛 🖏 Progres 🆓 Servers 🕸 🛛 🎁 SVN Re े 🎯 Internal 🔝 Markers 🔲 Properti UserDefinedFields.properties 215052 5/1 ∇ 🖻 🏇 🚺 🖉 🔳 😰 HebServiceConfiguration.properties 218 B WhiteList.properties 230845 6/25/12 3:20 📇 Tomcat v6.0 Server at localhost [Stopped] Workflow.properties 235174 7/24/12 12: 1 00 Writable Insert 36:66

Figure 16–1 Select Window Preferences

2. Now in Preferences Window, go to **OBP Plugin Development -> Fact**.

ype filter text	OBP Plugin Development	⇔ ← ⇔ ≁
General		
Ant		
Data Management		
Help		
Install/Update		
👂 Java		
👂 Java EE		
Java Persistence		
JavaScript		
⊳ Mylyn		
OBP Plugin Development		
Fact		
GEND		
GEFX		
JUnit		
Procedure Wrapper		
Reverse Engineering		
RMI		
Service Deployer		
Service Publisher		
WorkSpace Path		
XML/JSON Facade		
Plug-in Development		
Remote Systems		
Run/Debug		
Server		
Team		
Terminal		
Usage Data Collector		
Validation		
Web		
Web Services		
> XML		

Figure 16–2 Window Preferences - OBP Plugin Development

- **3.** Enter the values as mentioned:
 - Application Server URL: Local Host Server Listener URL

Example: http://localhost:9090/com.ofss.fc.channel.branch/HTTPListener

Presentation Server URL: Token Generator Application URL

Example: http://127.0.0.1:8001/TokenGenerator/HTTPListener

If using the plug-in in local eclipse workspace, it will not be used, but a value must be provided, you can use it from example value.

For security configured environment, it will be used, and then it should be provided properly.

- Bank Code: Bank code (Example: 08)
- Branch Code: Branch Code (Example: 089999)

- User Id: username (Example: ofssuser)
- **Password**: Password (Example: welcome1)

Figure 16–3 Enter the Preferences Fact values

pe filter text	Fact	⇔ • ⇔ •
General Ant Data Management Help Install/Update Java Java EE Java Persistence JavaScript Mylyn OBP Plugin Development Fact GEFU GEFX JUnit Procedure Wrapper Reverse Engineering RMI Service Deployer Service Publisher WorkSpace Path XML/JSON Facade Plug-in Development Remote Systems Run/Debug Server Team Terminal Usage Data Collector Validation Web Web Services XML	Fact Application Server URL: Presentation Server URL: Bank Code: Branch Code: User Id: Password:	http://localhost:9090/com.ofss.fc.channel.branch/HTTPListener http://127.0.0.1:8001/TokenGenerator/HTTPListener 08 089999 ofssuser ********

- 4. Now click **Apply**, and then click **Ok**.
- 5. Open Fact.properties and modify:
 - **aggregateCodeFilePath**: The location of host workspace.
 - **sourceFilePath**: The location of src directory of com.ofss.fc.fact project.

Figure 16–4 Fact Properties - aggregateCodeFilePath

Java EE - config/properties/Fact.properties - Eclipse File Edit Navigate Search Project Run Window	Help	No. 1 alter		
°° · □ @ @	🖥 • 🞯 • 🛛 😕 😂 .	🖋 - 🔲 🖬 🗋 🎱 .	후 🕆 🛱 Team Synchr 😰 Ja	iva EE
🎦 Project Explorer 🛛 📄 🔄 🐨 🗖 🗋	🗄 Tomcat v6.0 Server a	ApplicationEnterpris	📄 *Fact.properties 🕴 🔭	= =
DMSCoreConfig.properties 235950 7/27 DMSFailureEventConfig.properties 21944 DMSHostConfig.properties 229038 7/3/ DMSHostNestedConfig.properties 22906 DMSUIConfig.properties 232008 7/3/12 DomainCache.properties 215052 5/12/1 DomainPubSub.properties 215052 5/12/1 DomainPubSub.properties 215052 5/12/12 4:46 ExtendableSecurityConstants.properties 215052 5/12/12 FacilityUtilizationStrategy.properties 215052 5/12/12 FinancialStatementItemUtil.properties 21515 HoldFundSTansaction.properties 21505 hostapplicationlayer.properties 21505 hostapplicationConfig.properties 215052 5/12/12 IntegrationConfig.properties 215052 5/12 IntegrationConfig.properties 215	3 4 #Strategy used to 5 #values can be DA' 6 #default is DATA' 7 8 #DerivationPath.St 10 #DerivationPath.St 11 #DerivationPath.St 11 #to Kncybcybcybcy 14 15 DerivationPath.St 16 #path of the file 17 #prepatecodeFile 18 19 #Path of the sour. 20 SourceFilePath=D: 21 22 #insert type of c. 23 #fileImportType=JA 25 #export Fact pack. 26 exportPackageType Console Proble	select derivation pat rASOURCE_PATH_SELECTION. SOURCE_PATH_SELECTION. trategy=DEFAULT_AGGREGA trategy=AGGREGATE_PATH_ to be created for agg "attegy=AGGREGATE_PATH_ to	h. N_AGGREGATE_PATH_SELECTION,DEFAUL To know the details go com.ofss.f H_SELECTION TE_PATH_SELECTION _SELECTION erties.Property will be taken int com.ofss.fc.domain.fact.derivatic SELECTION.FactHitCount = 2 regate 	T_AGGREGATE c.enumeratic c.enumeratic m.path.selec Properti
□◆	Writable	Insert 17:1		

Figure 16–5 Fact Properties - sourceFilePath

Java EE - config/properties/Fact.properties - Eclipse	
ी कि कि कि कि कि कि कि कि कि कि कि कि	
Project Explorer 🛛 📄 🔄 🗐	🗄 Tomcat v6.0 Server a 🕼 ApplicationEnterpris 📄 *Fact.properties 😂 🔭 🗖 🗖
DMSCoreConfig.properties 235950 7/27 DMSFailureEventConfig.properties 21944 DMSHostConfig.properties 223038 7/3/ DMSHostNestedConfig.properties 22490 DMSUIConfig.properties 232040 7/3/12 DomainCache.properties 215052 5/12/1 DomainCubSub.properties 215052 5/12/1 DEBConfig.properties 215052 5/12/1 EBConfig.properties 215052 5/12/12 4446 EndpointConfig.properties 215052 5/12/12 4446 EventList.properties 215052 5/12/12 4446 ExtendableSecurityConstants.properties ExternalSystem.properties 215052 5/12/12 FacilityUtilizationStrategy.properties 215 Fact.properties 215052 5/12/12 4446 AM FactFactories.properties 215052 5/12/12 ExtractorStatedy.properties 215052 5/12/12 ExtractorStates.properties 215052 5/12/12 FacilityUtilizationStrategy.properties 215 Fact.properties 215052 5/12/12 4446 AM FactFactories.properties 215052 5/12/12 ExtractorStates.properties 215052 5/12/12 FacilityUtilizationStrategy.properties 215 Fact.properties 215052 5/12/12 446 AM FactFactories.properties 215052 5/12/12 ExtractorStates.properties 215052 5/12/12 FinancialStatementHemUtil.properties 215 HoldFundSTransaction.properties 215052 5/12/12 IntegrationAdapters.properties 215052 5/12/12 IntegrationStrategies.properties 225031 6 IntegrationStrategies.properties 22031 7 IntegrationStrategies.properties 22031 7 Inte	3 4 #5trategy used to select derivation path. 5 #Values can be DATASOURCE_PATH_SELECTION, AGGREGATE_PATH_SELECTION, DEFAULT_AGGREGATE 6 #default is DATASOURCE_PATH_SELECTION. To know the details go com.ofss.fc.enumeration 7 8 #DerivationPath.Strategy=DEFAULT_AGGREGATE_PATH_SELECTION 9 DerivationPath.Strategy=DATASOURCE_PATH_SELECTION 9 DerivationPath.Strategy=AGGREGATE_PATH_SELECTION 9 DerivationPath.Strategy=AGGREGATE_PATH_SELECTION 10 #DerivationPath.Strategy=AGGREGATE_PATH_SELECTION 11 12 #AGGREGATE_PATH_SELECTION strategy properties.Property will be taken into considerat 13 #to kncvbvvvcvvv more details refer com.ofss.fc.domain.fact.derivation.path.selec 14 15 DerivationPath.Strategy.AGGREGATE_PATH_SELECTION.FactHitCount = 2 16 #path of the file to be created for aggregate 17 aggregateCodeFilePath=D:/Workspace/HOST_Workspace/I 18 19 #Path of the source file of FactDerivationClass. 20 SourceFilePath=D:/Workspace\\LOST_Workspace\\Com.ofss.fc.fact\\srcl\} 21 22 #insert type of classFile 23 #fileImportType=D8 24 fileImportType=D8 24 fileImportType=JAR 25 #export Factaging type 26 exportPackageType =ZIP <
	Writable Insert 20:1

6. Now start the Host server.

7. In eclipse, go to Window -> Open Perspective -> Other.

Figure 16–6 Start Host Server

🔘 Java EE - Eclipse	
File Edit Navigate Search Project Run Window Help	
Image: Second	👻 🥥 🍂 🖹 🏇 Debug 🖆 Team Synchr 😰 Java EE
Project Explorer 🛛 📄 🤹 Open Perspective	🍅 Debug 🖓 🗆
com.ofss.fc.module.commonservice [tru Show View	▶ ♣J Java
com.ofss.fc.module.config [trunk/core/ com.ofss.fc.module.content [trunk/core com.ofss.fc.module.datiantegration [tru com.ofss.fc.module.datiantegration [tru com.ofss.fc.module.datiantegration [tru	Image: Web
com.ofss.fc.module.dsl [trunk/core/mid com.ofss.fc.module.ep [trunk/core/mid com.ofss.fc.module.fact [trunk/core/mid com.ofss.fc.module.fact [trunk/core/mid	Otper
com.ofss.fc.module.integration [trunk/c Navigation com.ofss.fc.module.integration.documa com.ofss.fc.module.interest [trunk/core, Web Browser	• • • • • • • • • • • • • • • • • • •
com.ofss.fc.module.je [trunk/core/midd Preferences	
com.ofss.fc.module.lcm [trunk/core/middlewa	
com.ofss.fc.module.loan [trunk/core/middlew.]	Problems 🖷 Progress 🎋 Servers 👩 SVN Repositori 🥥 Internal Web 📝 Search 🕮 🥂 🗖
com ofes fo module mis [trunk/core/middlewa	🛷 🔳 📴 🔫 🕇 🎽
com ofss fc module ons (trunk/core/middlewa	s available. Start a search from the search dialoct
com ofss fc module origination (trunk/core/mi	
com.ofss.fc.module.party [trunk/core/middlew	
com.ofss.fc.module.pc [trunk/core/middlewar	
com.ofss.fc.module.pm [trunk/core/middlewa	
com.ofss.fc.module.pricing [trunk/core/middle	
com.ofss.fc.module.rule [trunk/core/middlewa	
Com.ofss.fc.module.schedule [trunk/core/mide	
com.ofss.fc.module.settlement [trunk/core/mi	
🚮 com.ofss.fc.module.sms [trunk/core/middlewa	

- 8. Now in Open Perspective window select Fact.
- **9.** Click **Ok**.

Figure 16–7 Select Open Perspective value

CVS Repository Exploring	-
隊 Database Debug つ Database Development 参 Debug	
E Fact	
🐉 Java 🕵 Java Browsing 📌 Java EE (default)	E
¦g Java Type Hierarchy JavaScript	
Planning	
Plug-in Development	
Besource	
SVN Repository Exploring	-

It will open **Fact Explorer** perspective, where **Fact Vocabulary** is available.

Figure 16–8 Fact Explorer



10. Now refresh and expand **Fact Vocabulary**. Expanding Fact Vocabulary will show the **Domain** names.

Figure 16–9 Fact Vocabulary

Fact - com.ofss.fc.infra/src/com/ofss/fc/infra/das/orm/hibernate/	fibernateSessionWrapperjava - Eclipse	×
File Edit Source Refactor Navigate Search Project Run	Window Help	
📫 • 🗟 🗟 🔅 • 🔕 • 🎥 🕹 🛷 •	🍄 🌶 ə 🗐 🗊 🗄 = 🖏 = 🏷 💠 =	EE
SectExplorer	🧧 🗖 🕼 RuleChangeService.ja 🕼 RuleChangeObserver.j 🕼 AbstractRepository.j 🕼 HibernateSessionWrap 🛛 🐣	- 8
 Fact Vocabulary m Sanking Domain m Collection m Colobal m Restriction Framework 	<pre>619 /** 619 /** 620 /** 620 /** 620 /** 620 /** 620 /** 621 /** 621 /** 621 /** 621 /** 621 /** 622 /** 623 /** 624 /** 625 /** 626 /** 627 /** 628 /** 629 /** 629 /** 629 /** 629 /** 620 /** 620 /** 620 /** 620 /** 620 /** 621 /** 621 /** 621 /** 621 /** 622 /** 623 /** 624 /** 625 /** 626 /** 627 /** 628 /** 629 /** 629 /** 629 /** 620 /** 6</pre>	E Resol
	· •	,

Each Domain contains its **Domain Category** names.

Facts



Each Domain category contain its Fact Groups

Figure 16–11 Fact Groups

<pre>te te voie Make Make Mayer Ando Mayer A</pre>	Fact - com.ofss.fc.infra/src/com/ofss/fc/infra/das/orm/hibernate/Hibernate	eSessionWrapper.java - Eclips	se			- 0 ×
Totalization Image: Statistics		w menp se जि. जि. • ठी •	*5 6 - 6 -		The Fact St Debug 40 Team Syn	nchr 🤏 Java FF
<pre>pice for Verschafty Basing Domain Basing Domain Account BaseCount Account BaseCount Account BaseCount</pre>	FactExplorer	RuleChangeService.ia	RuleChangeObserver.i	AbstractRepository.i	HibernateSessionWrap 2 ***	
[International Payment Interset Interset	↓ RuleChangeServezja 619 619 620 /** 621 égee 5 622 égee 5 622 égee 5 622 égee 5 625 fry 6 626 try 6 627 f2 638 f2 631 631 633 f3 634 gaes 5 635 cat 641 grope 7 643 gaes 5 644 grope 7 655 goverride 656 goverride 657 esse 658 goverride 659 esse 650 foreride fillowarmigs 655 goverride 658 try 6 659 errors (5110) Ø Errors (51 terns) Ø Warnings (000 d2	<pre>[J] KulchangeObservery [J] KulchangeObservery ession#saveOrUpdate(java. guid {737E0748-0039-4C79- id saveOrUpdate(IHibernat 'etchSessionInterceptor(). f (1sBatchNode()) { f etchActionQueue(). f etchActionQueue() { retokationQueue() { f etchActionQueue() { f e</pre>	<pre>[1] AbstractKepositoryj [ang.object] 80C3-7E9D0EA0C461} 80C3-7E9D0EA0C461} setEntity Object) thrown markEntityOirty(Object ge(this, object, object ge(this, object, object ge(this, object, null) { anslator.translate(e); setEntity object) the giver is of the identifier pr iect) setCass mapping containing new or upd tityName, IHibernateEr markEntityOirty(object m 1165.teme) </pre>	<pre>#J HibernatebessionWrap 22 5 s PersistenceException { t; ; ct.getClass().getHame()); class())) { ; th instance, depending upon the value his behaviour, may be adjusted by spe dated state htty object) throws PersistenceExce t);</pre>	of its identi ifying an ption { Resource Resource Reso
D°						
		•	ш			•
	□◆					

Each Fact Groups contains its **Facts**.

Figure 16–12 Facts



11. To see the details of any fact, just double-click it. The details will be shown in a fact window containing some tabs. Move to each tab to show the details.

Figure 16–13 Business Definition Tab



Figure 16–14 Value Definition Tab

Edit Navigate Search Project Run Window Help							E Carl & Dahua - F	*1) Taama Cumaha 🔋 🕫 Jawa	
* 🔲 📹 👘 * 🔛 🛷 * 🖄 * 👘 * 🗸		T RuleCh	angeSenviv	a ia 🕢 AbstractP	enoriton i 💭 Hibern	ateSerrionWran	an Account Contribu	je ream synchr 🖓 Java	
Banking Domain		E Nulcelli	ingebervit	ela anacio	epository.j		anaccount.contribu 18	v	_
Account		Fact Va	lue De	rivation					
Accounting									
Alert		DataSour	гсе Туре						
Asset Classification									
ATM		DataSour	ce Type	HQL 👻					
📓 Bank Draft									
📓 BankGlobal		Input Bin	ndina						
📓 Bank Loan		2							
H LoanAccount									
LoanAccount.AccountCloserDate		S.No.	Туре	Name	Variable Fact	Literal Data Type	Literal Value	Add	
LoanAccount.AccountCurrencyCode	-							Demons	í.
LoanAccount.AccountStatus								Kemove	k.
CoanAccount.AccrualStatus									
LoanAccount.Agreementid			1						
ConAccount.AgreementsignDate									
Loan Account. Amount Disbursed To Settle		Figels co	asource	et Ouenu					
LoanAccount applicationId		single co	Jumin Sele	ct Query.					
				SELECT contribution FR	OM com.ofss.fc.domain.lo	an.account.LoanAccount			
LoanAccount Contribution				WHERE id.id=:id AND s	status = 'A'				
LoanAccount.ContributionReceived		Query St	tring				V	alidate Quenr	
LoanAccount.CrrCode		daci) a						maare query	
LoanAccount.CurrentInstallmentNo									
LoanAccount.CurrentTermMonths									
LoanAccount.CustomerId									
LoanAccount.CustomerShortName									
LoanAccount.DayBalance									
LoanAccount.Deferment.term									
LoanAccount.DisbursementCount									
LoanAccount.DrawdownRetries									
LoanAccount.EconomicCost									
LoanAccount.EMIToNMIRatio									
CoanAccount.ExternalUnclearedCreditAmount									
LoanAccount.ExternalUnclearedDebitAmount		Business D	efinition	Value Definition Enum	Definition Aggregate Defi	nition LoanAccountAggree	gate.java		
LoanAccount.Failurekeason	(1	Problem	18 20	Tasks @ lavadoc				59	~
LoanAccount FixedPeriod		errore 211	160	ingr () others (Filter me	tched 105 of 211165 it-ma)				
LoanAccount HardshinReliefType	2	errors, 211	.,100 warn	ings, o others (Filter ma	coned 105 of 211105 items)	*			-
LoanAccount.BanNumber		Description	n						
▲ LoanAccount.ld		😔 Erro	ors (5 item	s)					
A LoanAccount.Installmentnumber		🚯 Wa	mings (10	0 of 211160 items)					
A LoanAccount.InterestOnArrearFrozen									
LoanAccount.IoaWaiveDate									
Y									
LoanAccount.IsAdditionalAddressAdded									
CoanAccount.IsAdditionalAddressAdded CoanAccount.IsPMIFreeze									
CoanAccount.IsAdditionalAddressAdded CoanAccount.IsPMIFreeze CoanAccount.IsRateChangeAllowed									

Figure 16–15 Enum Definition Tab

• -								
* - · ·						100 million -		
						🖹 🖽 Fac	t 🕸 Debug 📲 Team Syr	chr 🤮 Java EE
- 🗆 👔	RuleCh	angeService.ja	🕢 AbstractRe	pository.j 🛛 🖉	HibernateSessionWrap	LoanAccou	int.Contribu 🛛 🏾 🍾 🎢	- E
^ F	a et Eu		*!					
	act En	um Denva	uon					
6	Data Cau	Time						
18	Datasou	ice type						
		-						
	DataSou	rce Type	•					
	Input Bi	nding						
	S No.	Tume	Mama	Variable Ex	ct Literal Date	Tune	Literal Value	Add
=	3.140.	Type	ivanie	valiable i av	ct citerar bau	атуре		
	-							Remove
	-							
B	usiness D	efinition Value	Definition Enum (Definition Aggre	gate Definition LoanAccou	ntAggregate.java		
(10)	Dealst		(المعاد الم	2				Fa ♥ □ 1
84	Propier		asks @ Javadoc					87
5 e	rrors, 21	,100 warnings,	U others (Filter mat	ched 105 of 21110	b5 items)			
D	escriptio	n						Res
	😣 Ern	ors (5 items)						
	🙆 Wa	mings (100 of 2	11160 items)					
		Business D Problem Serversion Business D Business	Business Definition Value Problems 2 Business Definition Value Problems 2 Description 2 Business Definition Value Problems 2 Description 2 Business Definition Value Warnings (100 of 2)	Image: Sevice ja Image: AbstractRe Fact Enum Derivation DataSource Type DataSource Type Image: Sevice Type Image: Type Image: Sevice Type Image: Sevice Type	 RuleChangeServiceja RuleChangeServiceja AbstractRepositoryj Fact Enum Derivation DataSource Type DataSource Type Input Binding S.No. Type Name Variable Fa Variable Fa Serors, 211,60 warnings, 0 others (Filter matched 105 of 2111 Description Errors (Fitms) Warnings (100 of 211150 items) 	Image: Service ja AbstractRepository ja HibernateSessionWrap Fact Enum Derivation DataSource Type Image: Service ja DataSource Type Image: Service ja Image: Service ja Imput Binding Imput Binding Imput Binding S.No. Type Name Variable Fact Literal DataSource Type Imput Binding Imput Binding Imput Binding Imput Binding S.No. Type Name Variable Fact Literal DataSource Type Business Definition Value Definition Enum Definition Aggregate Definition Business Definition Value Definition Enum Definition Aggregate Definition LoanAccource Problems Imput Service Imput Service Imput Service Imput Service Imput Service Description Imput Service Imput Service Imput Service Imput Service Imput Service Imput Service Imput Service Imput Service Imput Service Imput Service Imput Service Description Imput Service Imput Service Imput Service Imput Service Imput Service Imput Service Imput Service	Image: Service ja AbstractRepository ja HibernateSessionWrap LoanAccount Fact Enum Derivation DataSource Type Image: Service ja DataSource Type Imput Binding Imput Binding Imput Binding Imput Binding S.No. Type Name Variable Fact Literal Data Type Imput Binding Imput Binding Imput Binding S.No. Type Name Variable Fact Literal Data Type Service Type Imput Binding Imput Binding Imput Binding Business Definition Variable Fact Literal Data Type Service Type Imput Binding Imput Binding Business Definition Variable Fact Literal Data Type Service Type Imput Binding Imput Binding Business Definition Value Definition Enum Definition Business Definition Value Definition	Image: Service ja AbstractRepository; Image: HibernateSessionWrap LoanAccount.Contribu (2) ** Fact Enum Derivation DataSource Type Image: Service ja Image: Service ja Imput Binding Imput Binding Imput Binding Imput Binding SNo. Type Name Variable Fact Literal Data Type Literal Value SNo. Type Name Variable Fact Literal Data Type Literal Value SNo. Type Name Variable Fact Literal Data Type Literal Value Service James Service James Imput Binding Service James Service James Business Definition Value Definition Enum Definition Aggregate Definition LoanAccountAggregate.java Problems Service Ji Ja60 warnings, 0 others (Filter matched 105 of 211165 items) Description Service Ji Javadoc Service Ji Jabourings (000 of 211160 items) Service Ji Javadoc Service Ji Javadoc Service Ji Javadoc

Figure 16–16 Aggregrate Definition Tab

Edit Navigate Search Project Run Window Help	a •			ER Fact 🕸 Debug 🖧 Team	Synchr 🧐 Java FF
actExplorer 🛛 📲 Package Explorer	RuleChangeSe	rvice.ja	HibernateSessionWrap	LoanAccount.Contribu	-
Nanking Domain					
Account	Fact Aggreg	ate Derivation			
📓 Accounting	Concernation of the second sec				
📓 Alert	Aggregate Path				
Asset Classification					
ATM	A service to	Lass Consumb		Passage	
📓 Bank Draft	Aggregate to	LoanAccount		BIOWSE	
a BankGlobal	CI				
📓 Bank Loan	Class Name	com.ofss.fc.fact.aggregate.banking	.loan.LoanAccountAggregate		
El LoanAccount		Ter TH St			
CoanAccount.AccountCloserDate	Access Path	Contribution			
CoanAccount.AccountCurrencyCode	-				
LoanAccount.AccountStatus					
LoanAccount.Accruaistatus					
LoanAccount.Agreementid					
LoanAccount.AgreementSignDate					
LoanAccount.AmountDisbursed i osettie					
LoanAccount.AmountUtilized I oday					
ConAccount.applicationid					
CoanAccount.BranchCode					
Lean Account: Contribution					
A Loop Account Concede					
Loan Account. Curcode					
Lean Account. CurrentInstallimentitio					
LoanAccount Curtomerid					
LoanAccount CustomerName					
LoanAccount Deferment term					
LoanAccount DisbursementCount					
LoanAccount.DrawdownRetries					
A LoanAccount EconomicCost					
LoanAccount.EMIToNMIRatio					
LoanAccount.ExternalUnclearedCreditAmount					
LoanAccount.ExternalUnclearedDebitAmount	Business Definitio	n Value Definition Enum Definition	Anneaste Definition	ountAggregate java	
LoanAccount.FailureReason	Dusiness Derinido		Segure Derintion Connect	Sunorggregatejava	
LoanAccount.FirstDisbursementDate	Problems 🖾	🖉 Tasks) @ Javadoc			<u>6</u> 9. ▽
LoanAccount.FixedPeriod	5 errors, 211,160 w	arnings, 0 others (Filter matched 105 of	211165 items)		
LoanAccount.HardshipReliefType	Description		~		
💠 LoanAccount.IBanNumber	G Errors (5 it	ems)			
💠 LoanAccount.Id	A Warnings	(100 of 211160 items)			
💠 LoanAccount.Installmentnumber	C trainings				
💠 LoanAccount.InterestOnArrearFrozen					
💠 LoanAccount.IoaWaiveDate					
💠 LoanAccount.IsAdditionalAddressAdded					
LoanAccount.IsPMIFreeze					
LoanAccount.IsRateChangeAllowed	* (11		
-					



12. Creating **New Fact**: Right-click any domain Category in which Fact is to be created. Go to Maintenance -> Add.

Figure 16–18 Creating New Fact - Add



13. Enter required details for the facts in the new fact window.

All fields of Business definition tab are required for creation of any fact.

Fields of other tabs may be or may not be required. It depends on the fact to be created.



Figure 16–19 Creating New Fact - Fact Business Definition


Figure 16–20 Creating New Fact - Domain Group

14. Enter the values in the fields and press CTRL+S, click **Yes** to save and fact will be created.

Figure 16–21 Saving New Fact



Figure 16–22 Saving New Fact - Fact Added

Flexcube Development Plug-in	×
Fact Added Sucessfully.	OK

15. Modification of **Existing Fact**: To modify an existing fact, right-click the fact -> Maintenance -> Modify.

It opens the fact details in editable mode. Change whatever required and then save it using 'CTLRL+S'.

Fact Perspective also provide following facilities:

- Maintenance Operations on Fact
- Add
- Modify
- Inquire
- Fact Derivation Test
- Fact Value Derivation Test
- Fact Enum Derivation Test
- Fact Import Import Fact from File Store to Database store
- Fact Export Export Fact from Database store to File store.

16.2 Business Rules

Business Rules are defined for improving agility and for implementing business policy changes. This agility, meaning fast time to market, is realized by reducing the latency from approved business policy changes to production deployment to near zero time. In addition to agility improvements, Business Rules development also requires far fewer resources for implementing business policy changes. This means that Business Rules not only provides agility, it also provides the bonus of reduced development cost.

16.2.1 Rules Engine

A rule engine is a mechanism for executing 'business rules'. Business rules are simple business-oriented statements that encode business decisions of some kind, often phrased very simply in an if/then conditional form.

For instance, a business rule for a Banking system might be: Given a Customer and his location, if all of the following conditions are met:- The Customer is High Net worth Individual (HNI) - The Location is Metro - The Location is not Delhi{_}. The consequence is a 20% Discount in Application fee for Home loan. These business rules are not new: they are the business logic that is the core of many business software

applications. These rules are expressed as a subset of requirements. They are statements like "give a twenty-percent discount to non-Delhi Metro HNI Customers"

The primary difference with a rule engine is the way these rules are expressed; instead of embedding them within the program, these are encoded in business rule form.

Rule engines are not limited to execution; they often come with other tools to manage rules. Enterprise Rule Engine has all the options such as creation, deployment, storage, versioning and other such administration of rules either individually, or in groups.

16.2.2 Rules Creation by Guided Rule Editor

Any kind of rule can be created using this tool. User can freely enter business rules in text area, throughout the rule creation tool.

Standard Rule created in GRE comprises of following elements:

```
[mandatory]
Τf
               [condition] {AND/OR [condition]}*
Then
               [Action]+
[optional]*
Else If
              [condition] {AND/OR [condition]}*
Then
              [Action]+
[optional]?
Else
              [Action]+
where
* = 0 or more Occurrence
?= 0 or 1 Occurrence
+= 1 or more Occurrence
```

Features of Guided Rule Editor (GRE)

The features of GRE are:

- The 'if' block is mandatory block at the beginning of the structure.
- If (true) kind of condition is not supported. The condition should be comprised of 'LHS operator RKH'. There is parenthesis support in the UI. But you have to add it manually. Validation of parenthesis is supported.
- Nested 'if' is not supported from UI as of now.
- Conditions and actions are added by clicking the '+' button.
- After adding Condition user can add 'AND/OR Condition' by clicking '+' button at the End of Condition
- Different types of Actions can be added under 'Then'.
- Any number of 'Else if' can be added after 'If'.
- The condition for 'Else if' should differ from its previous 'if' or 'Else if' condition. Warning should be shown to user in this case.
- At most one 'Else' condition can be added to this 'if-else if-else' structure.
- No 'Else if' can be added after 'Else'.
- Real time rule structure preview in the bottom panel.

- Rule template / fragment for re usability.
- Facts will be used to create the rules

16.2.3 Rules Creation By Decision Table

Decision tables are a precise yet compact way to model complicated logic. Decision tables, like if-than-else, associate conditions with actions to perform. But, unlike the control structures found in traditional programming languages, decision tables can associate many independent conditions with several actions in an elegant way.

Example:

Table 16–1 Example of a Decision Table

Conditions & its alternatives			Actions
Customer Type	Location Type	Location	Discount
HNI	Metro	Mumbai	20% of App. fee
HNI	Metro	Delhi	No discount
HNI		Jaipur	No discount

The features of Decision Table are:

- The decision table contains rows and columns. Each row is considered to be a rule. In normal circumstances, the decision table is evaluated from top to bottom sequentially evaluating the various rules. It does not stop even if a rule fires. However, there is an option to stop processing of the decision table in case a rule is satisfied. There should be a special fixed column in the decision table (towards the right) which allows the decision table author to stop further evaluation of rules in case the current rule fires.
- Decision table should be expandable, that is, Rows and columns can be added dynamically.

Various functions for column and row manipulation should be available:

- Add Column After
- Add Column Before
- Add Row Above
- Add Row Below
- Delete Column
- Delete Row
- Move Column
- Move Row
- Sort Column Data Ascending
- Sort Column Data Descending
- Column Headers indicate condition / action
- Decision table should be editable to input data/conditions/actions

If a condition or action has range the column should be split in to two columns to accept the minimum and maximum values. Option to automatically fill series of

values. When clicked on row, a brief description about the condition should appear. Decision table will have brief description for the conditions and actions setup. Import and export data between Decision Table and Excel Spread Sheet.

16.2.4 Rules Storage

Rules created are stored in database tables as conditions and actions first, then these database tables are used to create executable rule in java programming language and compiled.

ActionID	Outvariable	Expression	Datatype
ACTION1	Discount Fee	0.2*App Fee	Double
ACTION2	Discount Fee	0	Double
ACTION3	Discount Fee	0	Double

Table 16–2 Actions

Table 16–3 Conditions

Conditio nID	LeftExpression	Relation alOperat or	RightExp ression	LinkedC onditionI D	LinkedC ondition alOperat or	ActionId	RuleID	Version
CON1	CustomerType	==	HNI	CON2	&&	ACTION 1	RULE1	1
CON2	LocationType	==	METRO	CON3	&&		RULE1	1
CON3	Location	==	MUMBAI				RULE1	1
CON4	CustomerType	==	HNI	CON5	&&	ACTION 2	RULE1	1
CON5	LocationType	==	METRO	CON6	&&		RULE1	1
CON6	Location	==	DELHI				RULE1	1
CON7	CustomerType	==	HNI	CON8	&&	ACTION 3	RULE1	1
CON8	Location	==	JAIPUR				RULE1	1

16.2.5 Rules Deployment

Rules are put together in compiled java class which are stored in jar file and deployed on the server at runtime. This deployed jar is available for applications which are going to execute the rules.

16.2.6 Rules Versioning

Each time rule is modified new version is created for the rule and stored.

Table 16–4 Rules Versioning

RuleID	Version	Name	Effective Date
RULE1	1	DiscountRule	01/01/2009
RULE1	2	DiscountRule	31/03/2009

16.3 Rules Configuration in Modules

Rules can be configured for multiple modules and multiple screens. The list of screens where the rule definition taskflows are used is mentioned below:

 Facts are used by configuring the fact context. Fact Context contains information about interacting Module. This need to be set to interact with Fact layer. Fact Context has been categorized at Domain Level.

For example, BankingFactContext will be used in Banking domain. This context has setters method for Facts which are generic in that domain. For example, BankingFactContext has *setAcountId* method. Interacting module need to fill maximum information available. These methods are setters for Facts which will always has input like *AccountId*, *PartyId*, *TransactionAmount* and so on.

- It is possible that at the time of interaction, Module already has some derivable Facts which are not going to change in the interaction. For example, *LnAccountProduct* at the time of Interest calculation.
- Module will send such Facts using *addFact* method, *using _retrievalKey* of the Fact
 referring Fact vocabulary. The benefit of sending such facts is these Facts won't get
 derived again. At the time of Fact Derivation, if *RetrievalKey* is present in the input
 FactMap, same value will be returned as a Fact value. If *RetrievalValue* is not
 present the Fact will be derived.
- Module will send maximum Fact information available at the time of interaction for better performance.

For example, at the time of Loan Account Opening, Pseudo code will look like:

```
// create fact context.
BankingFactContext lnFactContext = new BankingFactContext("LN");
lnFactContext.setPartyId(001);
// Set max available information
lnFactContext.addFact("LnAppliedAmount",2000);
lnFactContext.addFact("LnProductType","Home");
lnFactContext.addFact("LnRiskCategory",1);
lnFactContext.addFact("CustType","VIP");
```

At the time of CashTransaction Event, code will look like:

```
// create fact context.
BankingFactContext casaFactContext = new BankingFactContext("CASA");
casaFactContext.setPartyId(003);
casaFactContext.setAcountId("11111111111");
casaFactContext.setTransctionAmount(new BigDecimal(122));
casaFactContext.setTransactionCurrency(104);
casaFactContext.setTransactionAmountInAcy(new BigDecimal(122));
// Set max available information
casaFactContext.addFact("CustType", "VIP");
casaFactContext.addFact("CASAAccountType", "Saving");
```

16.3.1 Generic Rules Configuration

Generic Rules can be configured through the screen RL001 where the new rule can be defined or the existing rule can be updated for multiple domains and domain category. The authoring mode of rule creation can be chosen as GRE or Decision Table.

ount * Back Office * CASA * Channel * Collection * LCM * Loan * Operational Servi	ces 🔻 Origination 🎽 Party 🎽 Payment And Collection 🥻 🌟 🛉 👘 Fast Path 👘 👘 👘 Path
RL <u>0</u> 01	
ale Author	
	🔲 Read 🕂 Create ち Update 🛛 Brint 🗸 Ok 🖋 Cle
Rule Details	
* Domain lo Banking * Domain Cotegory Id. CR	Domain Name Banking Domain
* Bulaid OD BL 1	* Name OD Rule
* Effective Date 02-Jan-2013	* Description OD Rule Desc
Authoring Mode GRE	Version 1
(AssetClassification, rees. Days greater than 0)	
or (AssetClassification Interact Days restartion 0)	
And (AssetClassification Interest Days less than equal to 8	
r (AssetClassification TOD Days greater than 0)	
And (AssetClassification TOD Days less than equal to 8	
or (AssetClassification.Overline.Days greater than 0)	
And (AssetClassification.Overline.Days less than equal to 8)	
or (AssetClassification.Suspended.Fees.Days greater than 0)	
And (AssetClassification.Suspended.Fees.Days less than equal to 8)	
or (AssetClassification.Suspended.Interest.Days greater than 0)	
And (AssetClassification.Suspended.Interest.Days less than equal to 8)	
Then	
Classification Code equal to 101	
And (AssetClassification Interest Days rest interest return 24) And (AssetClassification Interest Days less than equal to 24) And (AssetClassification Interest Days less than equal to 24) And (AssetClassification TOD Days greater than 16) And (AssetClassification Overline Days less than equal to 24) or (AssetClassification Overline Days less than equal to 24) or (AssetClassification Overline Days less than equal to 24) or (AssetClassification Suspended Fees Days greater than 16) And (AssetClassification Suspended Fees Days greater than 16) And (AssetClassification Suspended Interest Days reset than 16)	
And (AccetClassification Suspended Interest Days greater trian (6)	
Then	
Classification Code equal to 103	
Else If (Accet/Classification Ecco Days) greater than 24	
or (AssatClassification Interact Days greater than 24)	
or (AssetClassification.TOD.Days greater than 24)	
or (AssetClassification.Overline.Days greater than 24)	
or (AssetClassification.Suspended.Fees.Days greater than 24)	
or (AssetClassification.Suspended.Interest.Days greater than 24)	
Then	
Classification Code equal to 104 AC.ClassificationReason equal to D	
de Modification History	

Figure 16–23 Generic Rule Configuration

☆ **Janž** = 🗲 ightarrow C 🗋 10.180.4.114:8001/com.ofss.fc.ui.view/faces/main.jspx?_afrLoop=9157467154191&_afrWindowMode=0&_adf.ctrl-state=eij6fg61s_117 🗀 Oracle Important Sites 📋 NGP URLs 🦳 Java 🦳 Market Firms Dther bookmarks BANKING PLATFORM OFSSUser, Last Login 17-May-2013 07:11:02 PM , Posting Date 02-Jul-2013 🕨 Business Unit SUNCORP BANK 💌 🏦 ? 🗄 🛊 🖄 🕴 📿 1 Account 🔻 Back Office 👻 CASA 🎽 Channel 👻 Collection 👻 LCM 🎽 Loan 🎽 Operational Services 🎽 Origination 🎽 Party 🎽 Payment And Collection 🛛 🎇 🆕 👻 RL<u>0</u>01 Rule Author 🛛 💭 Read 🕂 Create 🔩 Update 🛛 🚔 Print 🗸 Ok 🥔 Clear 🛽 Exit Rule Details Domain Name Banking Domain * Domain ID Banking Domain Category ID CS Domain Category Name CS *Rule Id Regulated_OD *Name Regulated OD * Effective Date 04-Feb-2013 *Description Regulated OD Authoring Mode Decision Table Version 3 Decision Table Serial Party Party Ty.. Number Facility Limit. Ž X Ž true Individual 1 🔲 X false Show Modification History

Figure 16–24 Rule Author - Decision Table

Different expressions can be defined in the expression builder screen. The expression once defined can also be used as one of the expressions in GRE.

🗧 ightarrow C 🗋 10.180.4.100.8001/com.ofss.fc.ui.view/faces/main.jspx?_afrLoop=56354649575372&_afrWindowMode=0&_adf.ctrl-state=1arn4d0ine_4 ☆ 🌆 \Xi 🗀 Oracle Important Sites 📋 NGP URLs 🦳 Java 🗀 Market Firms 📋 Other bookmarks ORACLE OFSSI lear Last Lonin 17-May-2013 12:04:44 PM Posting Data 01-San-2013 k Rusinaes Linit SLINCORP BANK 🖵 🛝 🕐 : 1 : 🖈 : 🍌 : 🗸 **.** Account * Back Office * CASA * Channel * Collection * LCM * Loan * Operational Services * Origination * Party * Payment And Collection 🌟 🐲 * Past RL<u>0</u>04 Expression Builder 🛛 🖓 Read + Create ち Update 🛛 🚔 Print 🗸 Ok 🖋 Clear 🛽 Cancel Expression Details * Expression Name CCIAAPPLYTERM * Description CCI APPLY TERM Data Type long Expression Editor (Party.Individual.Age 🛛 + 🛥 ProposedFacility.FacilityTermDetails.RequestedTerm.NumberOfYears 🛥) » 🅑 Show Modification History

Figure 16–25 Rule Author - Expression Builder

16.4 Rules Migration

This section describes the rules migration.

16.4.1 Rules Configured for Modules

Rule taskflows can be added to different modules. User can set up different rules based on the screen requirements.

Module	Screen	Rule Type	Rule Description
Alerts	AL04 - Alert Maintenance	GRE	User can create the new message template rule or use the existing rule. In this rule, the message template of the alert is selected based on the selected rule criteria.
			For example, if there is a particular party id, then the specific alert needs to be sent.
Content	CNM03 - Document Policy Definition	Decision Table	There are two types of rules (Inbound Rule and Outbound Rule) defined for each event in the document policies. These rules primarily define the checklist of documents based on different input values. The inbound rule are defined for the scenario of the documents being inputted to the system and the outbound rule are defined for the scenario of the documents being retrieved from the system and displayed to the end user.
			For example, In document policy of new applications, there is a event for identity verification. The inbound rule can be defined for the category of the documents which are required to be uploaded for the verification purpose on the basis of the Party Agency Type and the Party Type.
Pricing	PR006 - Price Definition	Generic Rule Author	Price can be rule based that is, amount of fee to be charged or price code to be charged comes from rule
Pricing	PR005 - Interest/Mar gin Index Code Definition	Generic Rule Author	Interest Index can be Rule Based i.e. Interest rate to be applied comes as outcome of rule.
Pricing	PR004 - Rate Chart Maintenance	Generic Rule Author	Rate Chart can be Rule Based i.e. Interest index to be used comes as outcome of rule.
Pricing	PR007 - Price Policy Chart Maintenance	Decision Table	Price policy chart internally gets stored as Rule. It basically defines Prices/RateCharts applicable when criteria is satisfied which is mentioned in rule.
Pricing	PR040 - Fee Computatio	PR040 - Fee Generic Rule Computatio Author	This screen provides analysis as how the fee for particular transaction (happened in past) was computed.
	n Analysis		In case of Rule Based Fees charged in transaction, this screen displays details of that rule along with input fact values used during rule evaluation.
Pricing PR017 - Interest Rat Derivation Analysis	PR017 - Interest Rate	7 - Generic Rule est Rate vation ysis	This screen provides analysis as how the interest rate for particular account was computed.
	Derivation Analysis		In case of Rule Based Rate Chart and Rule Based Index, this screen displays details of that rule along with input fact values used during rule evaluation.
Tax	TDS01 - Tax Parameter Maintenance	Decision Table	This rule is used to maintain the exemption limit and that exemption limit will be used at the time of tax computation.

 Table 16–5
 Details of Configured Rules in Modules

Module	Screen	Rule Type	Rule Description
Product Manufacturi ng	PM011 - Define Interest Rule	GRE/ Decision Table	In the Rule and Expression task flow is consumed to create Rule or Expression which is used to derived the BaseForInterest for Calculation of Interest.
			During EOD, module send facts which is used derive the BaseForInterest by executing the Rule or Expression whichever is attached to the IRD.
Asset Classificatio n	RL001 - Rule Author	GRE	This rule is used to derive the Asset Classification code of an account during the Account level classification batch shell. The facts will be the days past due date of various outstanding arrears. The rules will be created under 'LN' and 'CS' and linked to a plan in Asset Classification Plans (NP002).
			Rule for Facility-level classification: This rule is maintained only if the 'Applicability level' in NP001 is 'Facility'. This rule is used to derive the Classification code for a Facility during the Facility-level batch classification. The rule will be created under the Domain Category 'AC' and is linked via Asset Classification Preference (NP001).
Collections	RULE01 - RuleSet	GRE/Decisi on Table	Collection module's rules are defined as RuleSet. The RuleSet can be incorporated for the batch processing to filter accounts coming to collection.
			In RuleSet screen, multiple rules can be combined together as a single object called ruleset. The RuleSet functionality in rule engine provides the user with the facility to design the sequence of execution of rules where multiple rules need to be asserted for the same set of inputs. User would be able to select and wire the already existing rules and their sequence as per his/her requirement.
			There can be output dependent rules defined. For example,
			Rule 1 is: If(FACILITY_ID equal to TEST_FACILITY_ID)
			Then Account Type equal to FIXED
			Else If (FACILITY_ID equal to AAA)
			Then Account Type equal to 0
			Rule 2 is: If (ACCOUNT_TYPE equal to FIXED)
			Then ARS_ASSESSED_AMOUNT equal to 70000
			In the above case, rule 2 will be executed only if rule 1 satisfies the condition.

 Table 16–5 (Cont.) Details of Configured Rules in Modules

<u>17</u>

Composite Application Service

OBP Application provides with the functionality of adding composite application services which call multiple application services in one request. The transactions in these composite application services are called composite transactions and are made by composing the single transaction out of the multiple APIs transaction that gives the effect of single transaction.

Using APIs, single transaction can be composed of multiple transactions using very little effort. However, this cannot be done at run time. Following points have to be taken in to account while making a new composite transaction out of existing API transactions:

- Both the transactions should be passed in the same session context except overridden warnings. Overridden warnings from one transaction are passed as an input to next transaction.
- Decision of whether to commit the transaction or rollback the same must be explicitly handled by the composite transaction. The beginning and closing of interaction should be handled by the composite transactions.

For the transaction control of the transaction manager, there are two defined patterns:

- With Interaction.begin
 - * The interaction begins to ensure that the transaction reference number is maintained same across all participating APIs
 - * Required for supporting reversal of composite financial APIs
 - * Context information for entire call is maintained and used.
 - * Similar to any other API
- With TransactionManager
 - * Scope restricted to database transaction
 - * All APIs in the composite have the same commit scope
 - * Unique transaction reference generated for each API
 - * Can be thought of as a workflow with APIs participating in the same DB commit scope
 - * The composite transactions can be handled in two scenarios:
 - Calling multiple APIs in the same module
 - Calling multiple APIs in different modules by making the adapter call

17.1 Composite Application Service Architecture

The following depicts the sequence diagram for the composite transactions where two of the domain service calls are shown which can be extended to multiple domain service (1..N) calls. After every domain service call, 'isTransactionFailure()' call needs to be made to check the transaction status before proceeding for the next domain service call.





17.2 Multiple APIs in Single Module

For writing the composite service API which calls multiple services API, the following Java classes are needed with respect to new services as mentioned in the below table:

Table 17–1 Java Classes

Class Name	Description
Composite Service Interface	This provides the method definitions for the composite services.
Composite Service Class	This provides the implementation class for the composite services. In this class, we write methods which make the calls to different service APIs. The response of one service API can be used for making calls in another service APIs. The final response of the composite service is then created with the response objects of other service APIs and then transferred back to the adapter calls.

Class Name	Description
Executor Interface	This provides the extension pre-hook and post-hook method definitions for the service calls.
Executor Classes	This provides the implementation class for the executor interface.
Composite API Response Object	This provides the final response object which is passed to the adapter calls.

Table 17–1 (Cont.) Java Classes

One of the sample composite service method 'TDAccountPayinApplicationService. openAccountWithPayin' is shown below. In this service method, there are two methods of two different services:

- tdAccountApplicationService.openAccount
- tdDepositApplicationService.openDeposit

These service methods are called where the new account is created and then the returned account id from first service is used to do the payin by creating a new deposit for that account.

package com.ofss.fc.app.extensibility.td.service.composite; import java.util.logging.Level; import java.util.logging.Logger; import com.ofss.fc.app.AbstractApplication; import com.ofss.fc.app.Interaction; import com.ofss.fc.app.agent.dto.agent.AgentArrangementLinkageDTO; import com.ofss.fc.app.context.SessionContext; import com.ofss.fc.app.extensibility.td.dto.composite.TDAccountPayinResponse; import com.ofss.fc.app.extensibility.td.service.composite.ext.IExtendedTermDepositApplica tionServiceExtExecutor; import com.ofss.fc.app.td.dto.account.TermDepositAccountOpenDTO; import com.ofss.fc.app.td.dto.account.TermDepositAccountResponse; import com.ofss.fc.app.td.dto.deposit.PayinResponse; import com.ofss.fc.app.td.dto.transaction.payin.PayinTransactionDTO; import com.ofss.fc.app.td.service.account.ITermDepositAccountApplicationService; import com.ofss.fc.app.td.service.account.TermDepositAccountApplicationService; import com.ofss.fc.app.td.service.deposit.DepositApplicationService; import com.ofss.fc.app.td.service.deposit.IDepositApplicationService; import com.ofss.fc.common.td.TermDepositTaskConstants; import com.ofss.fc.enumeration.MaintenanceType; import com.ofss.fc.infra.exception.FatalException; import com.ofss.fc.infra.exception.RunTimeException; import com.ofss.fc.infra.log.impl.MultiEntityLogger; import com.ofss.fc.service.response.TransactionStatus; /** * The TDAccountPayinApplicationService class exposes functions/services to perform the sample of composite operations. This extensibility sample services includes: opening account and deposit * @author Ofss */ public class ExtendedTermDepositApplicationService extends AbstractApplication implements IExtendedTermDepositApplicationService { /** * Extension point for the class. This is the factory implementation for the extension of this class. * Any extension-method call on this factory instance, internally triggers a call to corresponding

```
* extension methods of all the extension classes returned by the
ServiceExtensionFactory
*/
private transient IExtendedTermDepositApplicationServiceExtExecutor extension;
// This attribute holds the component name
private final String THIS COMPONENT NAME =
ExtendedTermDepositApplicationService.class.getName();
/**
\ast This is an instance variable and not a class variable (static or static final).
This is required to
* support multi-entity wide logging.
*/
private transient Logger logger =
MultiEntityLogger.getUniqueInstance().getLogger(THIS_COMPONENT_NAME);
/ Create instance of multi entity logger
private transient MultiEntityLogger formatter =
MultiEntityLogger.getUniqueInstance();
/**
* @param sessionContext
* @param termDepositAccountOpenDTO
 * @return TermDepositAccountResponse
* @throws FatalException
*/
public TDAccountPayinResponse openAccountWithPayin(SessionContext sessionContext,
TermDepositAccountOpenDTO termDepositAccountOpenDTO,
PayinTransactionDTO payinTransactionDTO,
AgentArrangementLinkageDTO agentArrangementLinkageDTO
) throws FatalException {
super.checkAccess("com.ofss.fc.app.td.service.composite.TDAccountPayinApplicationS
ervice.openAccountWithPayin", sessionContext, termDepositAccountOpenDTO,
pavinTransactionDTO,
agentArrangementLinkageDTO);
if (logger.isLoggable(Level.FINE)) {
logger.log(Level.FINE, formatter.formatMessage("Entered into
openAccountWithPayin(). Input : termDepositAccountOpenDTO %s ", THIS_COMPONENT_
NAME, termDepositAccountOpenDTO.toString()));
}
Interaction.begin(sessionContext);
TransactionStatus transactionStatus = fetchTransactionStatus();
TermDepositAccountResponse tdAccountResponse = null;
String newAccountId = null;
PayinResponse payinResponse = null;
TDAccountPayinResponse tdAccountPayinResponse = new TDAccountPayinResponse();
ITermDepositAccountApplicationService tdAccountApplicationService
= new TermDepositAccountApplicationService();
IDepositApplicationService tdDepositApplicationService= new
DepositApplicationService();
try {
    Interaction.markCurrentTask(TermDepositTaskConstants.TD ACCOUNT ATTRIBUTE);
    createTransactionContext(sessionContext, MaintenanceType.ADDITION);
    extension.preOpenAccountWithPayin(sessionContext, termDepositAccountOpenDTO,
payinTransactionDTO, agentArrangementLinkageDTO);
   termDepositAccountOpenDTO.setBankCode(sessionContext.getBankCode());
    if (logger.isLoggable(Level.FINE)) {
       logger.log(Level.FINE, formatter.formatMessage("Entered into
tdAccountApplicationService.openAccount().
           Input : termDepositAccountOpenDTO %s ",THIS_COMPONENT_NAME,
termDepositAccountOpenDTO.toString()));
    }
    tdAccountResponse = tdAccountApplicationService.openAccount(sessionContext,
```

```
termDepositAccountOpenDTO);
   if (logger.isLoggable(Level.FINE)) {
        logger.log(Level.FINE, formatter.formatMessage("Exiting from
tdAccountApplicationService.openAccount().
Input : termDepositAccountOpenDTO %s ", THIS_COMPONENT_NAME,
termDepositAccountOpenDTO.toString()));
  }
  if(tdAccountResponse!=null && tdAccountResponse.getAccountId()!=null &&
!Interaction.isTransactionFailure(transactionStatus)) {
      newAccountId = tdAccountResponse.getAccountId();
payinTransactionDTO.getAccountTransactionDTO().setAccountId(newAccountId);
      if (logger.isLoggable(Level.FINE)) {
Logger.log(Level.FINE, formatter.formatMessage("Entered into
tdDepositApplicationService.openDeposit().
Input : payinTransactionDTO %s ", THIS_COMPONENT_NAME,
termDepositAccountOpenDTO.toString()));
       }
       payinResponse = tdDepositApplicationService.openDeposit(sessionContext,
payinTransactionDTO, agentArrangementLinkageDTO);
      if (logger.isLoggable(Level.FINE)) {
logger.log(Level.FINE, formatter.formatMessage("Exiting from
tdDepositApplicationService.openDeposit().
Input : payinTransactionDTO %s ", THIS_COMPONENT_NAME,
termDepositAccountOpenDTO.toString()));
      }
      if (payinResponse != null) {
tdAccountPayinResponse.setAccountId(payinResponse.getAccountId());
tdAccountPayinResponse.setDepositId(payinResponse.getDepositId());
tdAccountPayinResponse.setDepositStatus(payinResponse.getDepositStatus());
tdAccountPayinResponse.setNetInterestRate(payinResponse.getNetInterestRate());
tdAccountPayinResponse.setAccountingEventItem (payinResponse.getAccountingEventItem
());
tdAccountPayinResponse.setMaintenanceType(payinResponse.getMaintenanceType());
tdAccountPayinResponse.setMaturityAmount(payinResponse.getMaturityAmount());
tdAccountPayinResponse.setProductCode(payinResponse.getProductCode());
tdAccountPayinResponse.setInterestStartDate(payinResponse.getInterestStartDate());
tdAccountPayinResponse.setValueDate(payinResponse.getValueDate());
tdAccountPayinResponse.setStatus(payinResponse.getStatus());
       }
  }
       extension.postOpenAccountWithPayin(sessionContext,
termDepositAccountOpenDTO, payinTransactionDTO, agentArrangementLinkageDTO);
        fillTransactionStatus(transactionStatus);
        tdAccountPayinResponse.setStatus(transactionStatus);
                 } catch (FatalException fatalException) {
logger.log(Level.SEVERE, formatter.formatMessage("FatalException from
openAccountWithPayin()"), fatalException);
 fillTransactionStatus(transactionStatus, fatalException);
                } catch (RunTimeException fcrException) {
             logger.log(Level.SEVERE, "RunTimeException from
openAccountWithPayin()", fcrException);
                fillTransactionStatus(transactionStatus, fcrException);
                } catch (Throwable throwable) {
logger.log(Level.SEVERE, "Throwable from openAccountWithPayin()", throwable);
                   fillTransactionStatus(transactionStatus, throwable);
                } finally {
                   Interaction.close();
               }
                  super.checkResponse(sessionContext, payinResponse);
   if (logger.isLoggable(Level.FINE)) {
```

18

ID Generation

OBP is shipped with the functionality of generation of the IDs in three ways that is, Automatic, Manual and Custom. These three configurations can be defined by the user as per their requirements:

- If the configuration type for the ID generation is set to automatic, the ID is generated as per the defined generation logic for the automated ID generation. You can set the pattern, sequence, weights and check digit modulo and modify the automatic generation logic.
- If the configuration type is set to manual then the ID will be input and it will be checked in the database if it is unique. For the ID, a certain range of serial numbers can be reserved in the range table by the custom developer and the teller can select it from amongst the ranges while doing the manual entry.
- In case the bank's requirement is to have the different ID generation process which can be written or modified, then the extensibility feature is provided in OBP. In this feature, customized ID generation logic can be written and can be plugged in the OBP application by creating the custom ID generation class and doing the required configurations in the database.

The configuration of the ID generation process is shown in the sequence diagram below where the generator is selected based on the set configuration type.



Figure 18–1 Configuration of ID Generation Process

From the implementation perspective, the following sections describe the change in configurations required for customizing the ID generation.

18.1 Database Setup

The configuration part of the ID generation requires the following components which need to be defined in the OBP application. The following tables are involved to store the generation logic details for ID generation:

 FLX_CS_ID_CONFIG_B: This is the main config table where the identifier is defined with the combination of the category and sub category columns. The type of generation logic is determined based on the configuration set in the CONFIG_ TYPE column of this table.

Column Name	Description
CATEGORY_ID	Represents the Category Example: Party, Origination, DDA and so on
SUB_CATEGORY_ID	Represents the Sub Category Example: PartyId, AccountNo and so on
PATTERN_TXT	Represents the pattern in which the ID is generated Example: SSSSSSSSC, NNNBBBBYYYYSSSSSS
CONFIG_TYP	Represents Generation type values are AUT for Automatic, MAN for Manual, CUS for Custom
GENERATOR_CLASS_NAME	Fully Qualified classname of ID generator for config type Custom
SEQ_VALUE	Running Serial Number
WEIGHT	Comma separated Weight for each character defined in the pattern text Example: '0,0,7,6,5,4,3,2', '3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
CHK_DIGIT_MODULO	Check digit modulo
CREATED_BY	Indicates the User who created the row

Column Name	Description
CREATION_DATE	Indicates the date and time of the creation of the row
LAST_UPDATED_BY	Indicates the User who last updated the row
LAST_UPDATE_DATE	Indicates the date and time of the last update of the row
OBJECT_VERSION_NUMBER	Indicates the version number, Used to implement optimistic locking
OBJECT_STATUS_FLAG	Status Flag Example: A

Table 18–1 (Cont.) FLX_CS_ID_CONFIG_B

• FLX_CS_ID_RANGE: This table is used to determine the range of the values which the ID can take.

Column Name	Description
RANGE_ID	Represents the identifier for the range definition
RANGE_NAME	Represents the name defined for the range Example: Party, DDA
RANGE_START	Defines the beginning value for the range
RANGE_CURRENT	Defines the current value for the range
RANGE_END	Defines the ending value for the range
CATEGORY_ID	Represents the Category defined in FLX_CS_ID_CONFIG_B
SUB_CATEGORY_ID	Represents the Sub Category defined in FLX_CS_ID_CONFIG

Table 18–2 FLX_CS_ID_RANGE

• FLX_CS_ID_USF: This table is used to determine the user selected fields for the ID generation logic.

Column Name	Description
USF_ID	Represents the identifier for the user selected fields
USF_NAME	Represents the name for the user selected fields
IS_FIXED_FLAG	Defines if the user selected fields are fixed
CATEGORY_ID	Represents the Category defined in FLX_CS_ID_CONFIG_B
SUB_CATEGORY_ID	Represents the Sub Category defined in FLX_CS_ID_CONFIG_B

Table 18–3 FLX_CS_ID_USF

18.1.1 Database Configuration

In case of existing ID generation logic in the database, end user can update the seed data scripts by modifying configuration type and other parameters (pattern, sequence, weight and check digit modulo). While in case of new type of ID generation logic, an insert sql can be added in the scripts of tables.

18.2 Automated ID Generation

For the configuration type as automatic, user needs to set the CONFIG_TYPE as "AUT" in the FLX_CS_ID_CONFIG_B table. The ID generation logic is determined based on the set values in the config table for the pattern, sequence, weight and check digit modulo. The three attributes 'sequence', 'weights' and 'check digit modulo' are primarily used for calculation of the check digit.

ID Generation with Sequence and Range

ID is picked using the database sequence. This is needed in the case where serial number is used as part of an ID. Database sequence is used to avoid deadlock while trying to update, a sequential value stored and retrieved as part of the configuration in-case where the application is multiple threaded. This might lead to 'gaps' in the sequence of ids generated, if an exception occurs in the Transaction. However, this suffices as the errors related to deadlocks are mitigated.

For the first call to derive the value, the sequence for the specific configuration pattern is created, with names as CATEGORYTYPE_SUBCATEGORYTYPE_SEQ. The creation of this sequence happens only once in the lifecycle of application deployment. For example, TD (category) and AccountId (sub-category), the sequence generated is TD_ACCOUNTID_SEQ. And, for the successive requests, the already created sequence is used for sequence generation.

ID Generation with Pattern Text

The pattern text is split and an array is created of the characters. In case of mask ID configuration's pattern, ID configuration's text patterns are split. If the value is found to contain the special character (out of range [65-90]), it will be appended as it is to generated ID. Following are the conditions of ID generation with pattern text:

- If the pattern value is not the special character and the ID value is 'S' that is, SerialNumber, then range is looked upon:
 - If the range is defined, the current position of the range is determined based on category and sub-category. If the current position value's length is greater than pattern length, then characters between [0-length of pattern] will be generated ID, else zeros are prefixed before current position value of range until it's size becomes pattern's length. For example, the pattern is 'SSSSSS' and the generated range gives the value as '2345' then the actual value will become '002345'.
 - If range is not defined, then next value from sequence category_subCategory_ SEQ is picked, it'll also be corrected to the size of pattern's length as mentioned in case of above example.
- If the pattern value contains 'C', that is, check digit. Check digit computation is done and then appended the computed value to the pre computed ID value. The input value, weight and check digit modulo are used for calculation of check-digit. The input value can be sequence ID or can be the ASCII value in case the inputs are characters. The weights will be comma separated string of the digits to be used for the calculation.
- If the pattern value contains 'R', related party identifier is used for that value.
- If the pattern value doesn't match any of the above character, the value is fetched from the pattern map for the pattern's ID and the length is adjusted to the pattern's attribute length. These pattern map characters need to be passed by the caller service for calculation.

For example, let us take the submissionId with the pattern as NNNYYYYBBSSSSS in the database.

File Edit View Navigate Run V	ersioning	Tools Help						
	0 - 0	·					la	sk
		. (200					J.	
Connections X Reports X	Sot NGPPT	ONL X EFLX_CS_ID_CONFI	G_B X			10.00		
🕂 - 🔞 Y 🖆	Columns	Data Constraints Grants Stat	stics Triggers Flashback Depe	endencies	Details F	Partitions Indexes SQL		
FLX_BATCH_JOB_GRP_BRՒ^	🏓 🚯	🛃 💥 🤍 🔍 Sort Filte	er:					▼ Actions
FLX_BATCH_JOB_GRP_CA1	8	CATEGOR SUB CATE	S DATTERN TXT	<u>.</u>	B GE	B B WEIGHT	CREATED B	CREATION DA
FLX_BATCH_JOB_SHELL_DI	1	DDA Account.Id	55555555C	AUT	(null)	00.0.7.6.5.4.3.2	11 SYSTELLER	08-02-13 06:4
FLX_BATCH_JOB_SHELL_D	2 1	Dantu DantuId	899999999	AUT	(mull)	01212121212	0 SVSTELLED	08-02-13 06:4
FLX_BATCH_JOB_SHELL_M	2 1	arcy Farcyra	00000000	AUT	(1011)		10 CUCTELLED	00-02-10 00.4
FLX_BATCH_JOB_SHELL_RI	5 2	arty Relatedrar	22222222	AUI	(null)	00, 3, 0, 1, 1, 2, 1	10 SISIELLER	08-02-13 06:4
FLX_CS_ID_CONFIG_B	4 1	D AccountId	SSSSSSSSC	AUT	(null)	00,0,7,6,5,4,3,2	11 SYSTELLER	08-02-13 06:4
THE FLX_CS_ID_RANGE	5 0	rigination Submission	Id NNNBBBBYYYYSSSSSSS	AUT	(null)	03,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,	10 SYSTELLER	08-02-13 06:4
	6 0	Drigination Application	Id NNNBBBBYYYYSSSSSSS	AUT	(null)	03,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,	10 SYSTELLER	08-02-13 06:4
	7	Single Record View	-		×	3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,	10 SYSTELLER	08-02-13 06:4
H EX EXT ACUNTVAR TRAT	8					3 ,8,1,1,1,1,1,1,1,1,1,1	10 SYSTELLER	08-02-13 06:4
FLX EXT BPAY DETAILSRE	9	00 0 0 00				. 3, 8, 1, 1, 2, 1	10 SYSTELLER	08-02-13 06:4
FLX EXT BPAY HEADERRE	10	00 0 0 0 00				0 3.8.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.	10 SYSTELLER	08-02-13 06:4
. FLX_EXT_BPAY_TRAILERRE	11	CATEGORY ID	au il		^	13.8.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1	10 SYSTELLER	08-02-13 06:4
. FLX_EXT_FILE_PARAMS	12		Origination			000765432	11 SVSTELLED	08-02-13 06:4
FLX_EXT_FILE_UPLOAD_HI	12	SUB_CATEGORY_ID	SubmissionId			00,0,1,0,0,4,0,2	10 GUGTELLED	00-02-13 00.4
FLX_EXT_FILE_UPLOAD_M	15				-	93,6,1,1	10 SISTELLER	08-02-13 06:4
FLX_EXT_FTPRM_DETAILSF	14	PATTERN_TXT	NNNBBBBYYYYSSSSSSS		1	03,8,1,1	10 SYSTELLER	08-02-13 06:4
FLX_EXT_FTPRM_HEADERF	15	CONFIG TYP	lu.m.			p 3, 8, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,	10 SYSTELLER	08-02-13 06:4
FLX_EXT_FTPRM_TRAILERF	16		AUT			01,2,1,2,1,2,1,2,1,2	0 SYSTELLER	08-02-13 06:4
FLX_EXT_FTPTP_DETAILSR	17	GENERATOR_CLASS_NAME				p 3, 8, 1, 1	10 SYSTELLER	08-02-13 06:4
	18					03,8,1,1,1,1,1,1,1,1,1,1	10 SYSTELLER	23-12-12 11:2
	19	SEQ_VALUE	0			0 3, 8, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,	10 SYSTELLER	08-02-13 06:4
	20	WEIGHT	3.8.1.1.1.1.1.1.1.1.1.1.1.1.1.1	.1.1.1.1.1		03.8.1.1	10 SYSTELLER	24-04-13 12:0
	21	CHK DICIT MODULO				03.8.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.	10 SYSTELLER	03-05-13 03:2
FLX EXT INDEBIT DETAIL	22	CHK_DIGI1_MODULO	10			1381111111111111111111111	10 SYSTELLED	15-05-13 10-5
FLX EXT INDEBIT HEADER	22	CREATED_BY	SYSTELLER				10 SISTELLER	15-05-15 10.5
- FLX_EXT_INPAYMNT_DETA	25					23,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,	IU SYSTELLER	15-05-13 10:5
. FLX_EXT_INPAYMNT_HEAD	24	CREATION_DATE	08-02-13 06:44:00.00000000	D PM		p 3, 8, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,	10 SYSTELLER	29-05-13 03:1
		LAST UPDATED BY						
FLX_EXT_JE_ENTRIES_FTR								
FLX_EXT_JE_ENTRIES_HDF		LAST_UPDATE_DATE			1			
FLX_EXT_LCM_ENTRIES_DE		OBJECT VEDCION NUMBER	L					
		ODJECT_VERSION_NUMBER	1					
FLX_EXT_LCM_ENTRIES_H		OBJECT_STATUS_FLAG	A		1			
HI FLX_EXT_NPACCT_PROVSN					~~~			
ELX_EXT_NPACCI_PROVSN		Help	Apply	Ca	ancel			

Figure 18–2 Automated ID Generation - Single Record View

The pattern hashmap 'value' will be populated and passed by the caller with the key value pair as pattern character as key and its corresponding value. As shown below, 'N' will contain name value, 'Y' will contain year value and 'B' will contain branch code.

Figure 18–3 Automated ID Generation - Generate Submission ID

/a - com.ofss.f	fc.module.origination/src/com/ofss/fc/app/origination/service/core/submission/SubmissionApplicationService.java - Eclipse 🗖 🗇	
Edit Source	e Refactor <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>Wi</u> ndow <u>H</u> elp	
	Quick Access	Þ
Submiss	ionApplicationService.java 🙁 👔 SubmissionIdGenerationService.java 🗢 🖻	
3010	nublic SubmissionApplicationService() / []	
307		
309⊕	* This method is used to generate a submission Identifier based on the header information which includes.	
△ 334⊕	public SubmissionCreationResponse generateSubmissionId(SessionContext sessionContext, HeaderDTO submissionHeaderDTO) throws FatalException {[]	
395		8
397⊕	* Generate submission id for the Submission, this method populates the value of each format mask based on the input	
4050	private SubmissionKey [generateSubmissionId(Submission submission] throws FatalException {	
406		
407	if (logger.isLoggable(Level.FINE)) {	
408	<pre>Logger.log(Level.FINE, formatter.formatMessage(THIS_COMPONENT_NAME + ".generateSubmissionId() Entry."));</pre>	
409		
410	HashMap(String, String) Values = new HashMap(String, String)();	
411	String branch = submission.getcommonuata(),getApplicationBranch() + 0000;	
412	String year = submission.getsubmissionmeader().getsubmissionCreationDate().getrear() + 0000 ;	
415	String type = submission.getsubmission.ype() + ;	
415	Stille nume - 500;	
415	values.put(SubmissionIdGenerationService.VEA, van)	
417	values nut(SubmissionIdGenerationService ReArch branch)	
418	values nut(SubmissionIdGenerationService_TVPE, type):	
419	SubmissionIdGenerationService = SubmissionIdGenerationService.aetInstance():	
420	String submissionId = <pre>service.generateSubmissionId(null1. values):</pre>	
421	Logger, log(Level.FINER, formatter.formatMessage("generateSubmissionId - submissionId = %s", submissionId));	
422	if (logger.isLoggable(Level.FINE)) {	
423	<pre>Logger.log(Level.FINE, formatter.formatMessage(THIS COMPONENT NAME + ".generateSubmissionId() Exit."));</pre>	
424		
425	return new SubmissionKey(submissionId);	
426		
427	-	

com.ofss.fc	module.orgination/src/com/otss/tc/domain/ongination/service/core/submission/id/generation/SubmissionIdGenerationService.java - Eclipse
it <u>S</u> ource	Refactor <u>N</u> avigate Search <u>Project Run <u>Mi</u>ndow <u>H</u>elp</u>
I R 🗅	🔌 🕸 ▾ 🗶 ▾ 🙆 🗁 🖉 ▾ 🕼 😰 🖉 ☜ 🗿 ☜ 🖗 ▾ 🖗 🗢 ▼ ⇒ ▾ 🛛 🖆
	Quick Access
Submissio	nApplicationService.java 🛛 SubmissionIdGenerationService.java 🛛 🗍 AutomaticIdGenerator.java
2⊕ * 0	copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
4	
5 pack	age com.otss.tc.domain.origination.service.core.submission.id.generation;
7⊕ impo	prt java.util.HashMap;[]
13	
15⊕ * 1 19 nubl	his class generates ID for <u>different entities</u> in Submission such as Submission, Application and Applicant.[]
20	
22⊕	* Constant for Submission_Id[
24	private static final String SUBMISSION_ID = "SubmissionId";
28	constant for origination_ private static final String (RIGINATION = "Origination":
30⊕	* Constant for Application_Id[
32	private static final String APPLICATION_ID = "ApplicationId";
36	" Constant for Applicant_Id
38⊕	* Constant for name[]
40	public static final String <u>NAME = "N"</u> ;
42⊕ 44	" constant for type]
46⊕	" Constant for branch
48	public static final String BRANCH = "B";
50⊕ 50	" Constant for year
52 54⊕	* Constant for channel
56	public static final String [HANNEL = "H ^T ;
57	
580	/** * Service to generate the submission id based on configurable format mask. The default format mask is
60	* *SUB*TBrcdYyyNnnnn
61	* the first three character is constant "SUB", following by one character that indicate the submission type
62	* (@link com.ofss.fc.enumeration.origination.SubmissionType} and combination of Branch Code + Year + Running Serial * Number
64	
65	* <u>@fcb.param</u> MI,String,submissionId, ,It is used for manual id generation, not applicable for Automatic Id
66	* generation
68	* specified range
69	* <u>@fcb.param</u> MI,HashMap,values,,The set of values used to generate the <u>submissio</u> id. The key is the pattern type
70	* (Name, Type, Branch, Year)
72	· @recurn ine Suumission iu.
730	public String generateSubmissionId(String submissionId, long rangeId, HashMap <string, string=""> values) throws FatalException {</string,>
74	
75	iquenerator generator = (iquenerator) AbstractueneratorFactory.getUniqueInstance().getIduenerator(<u>UKIGINATION, SUBMISSION_ID</u>); String tempId = null:
77	try (
78	<pre>tempId = generator.generateId(submissionId, rangeId, values);</pre>
79 80	<pre>} catch (FatalException fatalException) {</pre>
81	}
82	return tempId;
83	}
84	

Figure 18–4 Automated ID Generation - Submission ID Generation Service

The ID will be generated by the automatic generator with first three characters as name, next four digits as year, next three characters of branch and rest with generated sequence as per the mask pattern.

In case of without mask configuration's pattern. If range ID is -1, it means that there is no range defined for the mask configuration, it then picks up the range details with range ID based on the category and sub-category. The generated ID will become the current position of range. If range is not defined in the table, then the sequence needs to be defined and the value is picked based on that. The next value of the sequence will become the generated ID value.

18.3 Custom ID Generation

In case of configuration type as custom, user needs to set the CONFIG_TYPE as 'CUS' in the CONFIG_TYP column in the FLX_CS_ID_CONFIG_B table.

User can customize the ID generator by writing a new custom ID generator class which will need to extend the IdGenerator and write the abstract methods for the ID generation. This class needs to be mentioned in the GENERATOR_CLASS_NAME column of FLX_CS_ID_CONFIG_B table.



Figure 18–5 Custom ID Generation - Custom ID Generator

In case the user want to write the custom generation logic in a specific customized pattern definition, then user can do that by writing the custom constant class and the custom pattern class which can pick the defined pattern from the configuration object set in the PATTERN_TXT column of the FLX_CS_ID_CONFIG_B table of the database. The user will pass the values in the pattern hashmap which will then populate the pattern and generate the ID.



Figure 18–6 Custom ID Generation - Custom ID Generation Constants

Figure 18–7 Custom ID Generation - Custom Pattern Based Generator



<u>19</u>

Extensibility of Domain Objects - Dictionary Pattern

This chapter describes how consultants or other third parties can extend OBP domain by leveraging the dictionary design pattern to extend any Abstract Domain Object on which a maintenance screen and corresponding services are supported by product and are shipped for a release. This pattern provides true domain model extension capabilities by allowing addition of custom data fields to the underlying domain objects and the database tables mapped to them. Such capability alleviates an important limitation in the earlier approach of using User Defined Fields (UDF) to extend the OBP data model. In the UDF approach, the data model for the custom fields is separate from that of the domain objects itself and hence cannot be consumed in business policies or even rules as facts. The dictionary pattern enables using the custom data fields in the extensions, business rules (as facts) and custom business policies as the domain object load from the database retrieves the extended domain object and not just the product domain object.

The framework related changes to make such support available are supported from release 2.3 of the Oracle Banking Platform. These changes have been made across layers including the UI, JSON, Assembler, ORM and DB layer. The changes required to be made by consulting to support the persistence and usage of the extra attributes by extending the product domain object have been discussed in detail in the sections by taking common domain extensibility use cases as examples. The process in which data is transferred from the UI layer, to the host layer is mentioned briefly as points below:

- The proxy layer provides an extension point wherein the additional data fields on the screen can be populated as name value pairs and set in the input request.
- The custom attribute data gets passed through the JSON layer onto the middleware host as part of the application service invocation.
- These name value pairs are translated into the custom domain object which extends the base OBP domain object.
- The custom fields get persisted into the DB along with the domain object fields as part of ORM mapping.
- Exact opposite flow follows for inquiry services in which the data flows back via output response.





The dictionary data is passed in the request DTO and is therefore available as part of the pre and post application service extensions. The above process is described in detail in the sections below.

19.1 Customized Domain Object Attribute Placeholders

Data transfer object (DTO) is a design pattern used to transfer data between an external system and the application service. All the information may be wrapped in a single DTO containing all the details and passed as input request as well as returned as an output response. The client can then invoke accessor (or getter) methods on the DTO to get the individual attribute values from the Transfer Object. All request response classes in OBP application services are modelled as data transfer objects. These objects extend a base class DataTransferObject which holds an array of Dictionary object. The Dictionary encapsulates an array of NameValuePairDTO which is used to pass data of custom data fields or attributes from the UI layer to the host middleware. The following is mentioned as points below:

- All DTO classes should extend DomainObjectDTO class.
- The DomainObjectDTO class has been made to extend DataTransferObject class.
- This class has a single attribute which is an array of Dictionary class.
- Dictionary class has a single attribute which is an array of NameValuePairDTO

Using an array of name value pairs inside an array of dictionary allows for supporting two dimensional grid structures in the UI layer.

At present whenever any third party requires support for additional attributes in a Domain Object, the information regarding the corresponding Customized Domain Object name and attribute name-value pair is required to be populated as an array of NameValuePairDTO which in turn is set in the Dictionary class as the first and only element of the 'dictionaryArray' attribute of the DataTransferObject. This is shown in the following code extract.

Figure 19–2 Code Extract

- com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePairDTO1= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO(); nameValuePairDT01.setGenericName("com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate.CustomValue1");
- nameValuePairDT01.setValue("Y");
- com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePairDTO2= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO();
- nameValuePairDTO2.setGenericName("com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate.CustomValue2"); nameValuePairDTO2.setValue("Y");

com.ofss.fc.framework.domain.common.dto.NameValuePairDTO[] nameValuePairDTOArray= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO[2]; nameValuePairDTOArray[0]=nameValuePairDTO1;

- nameValuePairDTOArray[1]=nameValuePairDTO2;
- com.ofss.fc.framework.domain.common.dto.Dictionary dictionary= new com.ofss.fc.framework.domain.common.dto.Dictionary(); dictionary.setNameValuePairDTOArray(nameValuePairDTOArray);
- com.ofss.fc.framework.domain.common.dto.Dictionary[] dictionaryArray = new com.ofss.fc.framework.domain.common.dto.Dictionary[1];
- 13 dictionaryArray[0]=dictionary;

19.2 Customized Domain Object DTO Interceptor in UI Layer

All DTO classes should extend DomainObjectDTO in case maintenance fields are required.

For example, 'MessageDataAttributeDTO' Class which extends 'DomainObjectDTO' is used to transfer data between an external system and the application service and persist data for Domain Object 'MessageDataAttribute'.

'CustomizedMessageDataAttribute' is a subclass of this Customizable Maintenance Domain Object called 'MessageDataAttribute' which is extended by the partners or consulting teams to include and subsequently persist extra attributes along with those of 'MessageDataAttribute'.

This information can be mapped as input and output to the application services with the help of dictionaryArray attribute of MessageDataAttributeDTO inherited from DataTransferObject.

19.2.1 Interceptor Hook to Persist Customized Domain Object Attributes

This UI Layer Interceptor Hook is used during Create or Update mode to populate DataTransferObject with the dictionaryArray attributes from customized Screen Components to be persisted as the Customized Domain Object.

In the UI Layer, the ApplicationServiceProxyFacade is used to send the DataTransferObject on to the Host to be persisted. Before it does so, it uses the InterceptorFactory to instantiate the appropriate IProxyLayerInterceptor defined in the DictionaryInterceptor.properties corresponding to the key for this application service or task code. Thereafter it invokes the 'populateDictionaryArray' method of this IProxyLayerInterceptor to populate DataTransferObject with the dictionaryArray attributes from customized Screen Components. Thereafter, it sends the entire DataTransferObject on to the Host for persistence as the Customized Domain Object.

The following figure provides the details of Interceptor Hook to populate and persist Customized Domain Object.



Figure 19–3 Interceptor Hook to Persist Customized Domain Object

19.2.2 Interceptor Hook to Fetch Customized Domain Object Attributes

This UI Layer Interceptor Hook is used during read mode to extract the dictionaryArray attributes from the DataTransferObject and populate the customized Screen Components with the help of the screen view object.

In the UI Layer, the ApplicationServiceProxyFacade is used to receive the DataTransferObject from the Host. After it does so, it uses the InterceptorFactory to instantiate the appropriate IProxyLayerInterceptor defined in the DictionaryInterceptor.properties corresponding to the key for this application service or task code. Thereafter, it invokes the 'extractDictionaryArray' method of this IProxyLayerInterceptor to extract the dictionaryArray attributes from the DataTransferObject and populate the customized Screen Components with the help of the screen view object. Thereafter, it returns the entire DataTransferObject on to the Screen Backing Bean or Helper Class from where the proxy fetch call was invoked.

The following figure provides the details of Interceptor Hook to fetch Customized Domain Object and populate extra Screen Components.



Figure 19–4 Interceptor Hook to Fetch Customized Domain Object

InterceptorFactory instantiates the appropriate IProxyLayerInterceptor defined in the DictionaryInterceptor.properties corresponding to the key.

Examples of such key value pair is:-

com.ofss.fc.appx.ep.service.dispatch.message.service.client.proxy.MessageTemplateAp plicationServiceProxyFacade=com.ofss.fc.ui.taskflows.ep.messageTemplateUI.view.int erceptor.MessageTemplateUIInterceptor

com.ofss.fc.appx.party.service.contact.service.client.proxy.ContactPointApplicationServiceProxyFacade=com.ofss.fc.ui.view.party.contactPoint.interceptor.ContactPointUIInterceptor

19.3 Dictionary Data Transfer from UI to Host

The section describes the dictionary data transfer from UI to Host.

19.3.1 Customized Domain Object DTO Transfer from UI to Host

In UI server <ApplicationService>JSONClient constructs the JSON Object for <DomainObjectDTO> which includes the dictionaryArray of the DataTransferObject.

For example, in UI server MessageTemplateApplicationServiceJSONClient constructs the JSON Object for MessageTemplateDTO which includes MessageTemplateAttributeDTO and the dictionaryArray of DataTransferObject as shown below.

Figure 19–5 JSONClient constructs the JSON Object



<ApplicationService>JSONClient constructs the JSON Object for <DomainObjectDTO> which includes the dictionaryArray of the DataTransferObject

The above process uses AbstractJSONBindingStub class' serializeDictionaryArray to include 'genericName' and 'value' attributes of NameValuePairDTOArray which was inside dictionaryArray attribute of MessageTemplateAttributeDTO.



Figure 19–6 SerializeDictionaryArray to include GenericName and Value attributes

AbstractJSONBindingStub class's serializeDictionaryArray to include "genericName" and "value" attributes of NameValuePairDTOArray

In the Host Server <ApplicationService>JSONFacade extracts the 'DictionaryArray' attribute of JSON Object and sets it as <DomainObjectDTO>'s dictionaryArray attribute.

For example, in the Host Server, MessageTemplateApplicationServiceJSONFacade extracts the 'DictionaryArray' attribute of JSON Object and sets it as MessageDataAttributeDTO's dictionaryArray attribute.



Figure 19–7 Host Server JSONFacade extracts the attribute of JSON Object

In the Host Server <ApplicationService>JSONFacade extracts the "DictionaryArray" attribute of JSON Object and sets it as <DomainObjectDTO>'s dictionaryArray attribute

The above process uses AbstractJSONFacade's getDictionaryArray method that unmarshalls the 'genericName' and 'value' from JSON Object to get the dictionaryArray attribute.



Figure 19–8 AbstractJSONFacade's getDictionaryArray method

AbstractJSONFacade's getDictionaryArray method that unmarshalls the "genericName" and "value" from JSON Object to get the dictionaryArray attribute

19.3.2 Customized Domain Object DTO transfer from Host to UI

In the Host Server <ApplicationService>JSONFacade constructs the JSON Object for <DomainObjectDTO> and the dictionaryArray of DataTransferObject

MessageTemplateApplicationServiceJSONFacade's method serializeMessageDataAttributeDTOArray in Host Server constructs the JSON Object for MessageTemplateDTO which includes MessageTemplateAttributeDTO and the dictionaryArray of DataTransferObject as shown below:



Figure 19–9 Host Server JSONFacade constructs the JSON Object

In the Host Server <ApplicationService>JSONFacade constructs the JSON Object for <DomainObjectDTO> and the dictionaryArray of DataTransferObject

The above process uses AbstractJSONFacade's serializeDictionaryArray to include 'genericName' and 'value' attributes of NameValuePairDTOArray which was inside dictionaryArray attribute of MessageTemplateAttributeDTO.


Figure 19–10 AbstractJSONFacade's serializeDictionaryArray to include Generic Name and Value attributes

AbstractJSONFacade's serializeDictionaryArray to include "genericName" and "value" attributes of NameValuePairDTOArray

In the UI Server, <ApplicationService>JSONClient extracts the 'DictionaryArray' attribute of JSON Object and sets it as <DomainObjectDTO>DTO's dictionaryArray attribute.

In the UI Server, MessageTemplateApplicationServiceJSONClient extracts the 'DictionaryArray' attribute of JSON Object and sets it as MessageDataAttributeDTO's dictionaryArray attribute.



Figure 19–11 UI Server JSONClient extracts the DictionaryArray attribute

new 2		
1	private com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDT0[] getMessageDataAttributeDT0Array(JSONObject jsonObject, String	strNe 🔺
2	throws JSONException {	
3		
4	com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO[] 1MessageDataAttributeDTOs = null;	
5	JSONArray 1MessageDataAttributeDTOJSONArray = jsonObject.optJSONArray(strName);	
6	if (lMessageDataAttributeDTOJSONArray != null) {	
7	int numobjInputValArray = 1MessageDataAttributeDTOJSONArray.length();	
8	lMessageDataAttributeDTOs = new com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO[numobjInputValArray];	
9	JSONObject lobjInputValJSONObject = null;	
10	for (int icbjInputVal = 0; icbjInputVal < numobjInputValArray; icbjInputVal++) {	
11	com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO lstrName_MessageDataAttributeDTO = null;	
12	lobjInputValJSONObject = 1MessageDataAttributeDT0JSONArray.optJSONObject(iobjInputVal);	
13	if (lobjInputValJSONObject != null) {	
14	lstrName_MessageDataAttributeDTO = new com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO();	
15	<pre>String strMessageDataAttributeDTOType = lobjInputValJSONObject.optString("_type");</pre>	
16	com.ofss.fc.framework.domain.common.dto.Dictionary[] llstrName_MessageDataAttributeDTO_dictionaryArrays	
17	<pre>= getDictionaryArray(lobjInputValJSONObject, "DictionaryArray");</pre>	
18	lstrName_MessageDataAttributeDTO.setDictionaryArray(llstrName_MessageDataAttributeDTO_dictionaryArrays);	
19	3	
20	lMessageDataAttributeDTOs[iobjInputVal] = lstrName_MessageDataAttributeDTO;	
21	3	
22	3	
23	return 1MessageDataAttributeDTOs;	
24	3	
25		•
•		•
Normal text f	file nb char: 1636 Ln: 17 Col: 73 Sel: 0 Dos\Windows (ANSI	INS

In the UI Server, <ApplicationService>JSONClient extracts the "DictionaryArray" attribute of JSON Object and sets it as <DomainObjectDTO>DTO's dictionaryArray attribute

- 🗆 X

The above process uses AbstractJSONBindingStub's getDictionaryArray method that unmarshalls the 'genericName' and 'value' from JSON Object to get the dictionaryArray attribute.



Figure 19–12 AbstractJSONBindingStub's getDictionaryArray method

AbstractJSONBindingStub's getDictionaryArray method that unmarshalls the "genericName" and "value" from JSON Object

The provision of marshalling and un-marshalling of 'dictionaryArray' attribute of all DataTransferObjects has been included in the JSON layer for all application services.

19.4 Translating Dictionary Data into Custom Domain Object

This section describes the details of translating dictionary data into custom domain object.

19.4.1 Instantiation and Persistence of Custom Domain Objects

If a method has an input parameter that is a DataTransferObject, the first line of the method in the assembler will be of the form:

(populateDataTransferObjectDTOMap('Fully Qualified Name of this DataTransferObject>', dataTransferObject);

This method is defined in AbstractAssembler.java which newly instantiates referenceDataTransferObjectDTOMap if required and populates the map with the above entry.

This map is used as a set of globally available DataTransferObject's which can be retrieved by invoking another method defined in AbstractAssembler.java which is of the form:

```
retrieveDataTransferObjectDTOMapElement('<Fully Qualified Name of this
DataTransferObject >');
```

Whenever any AbstractDomainObject is instantiated, the Customized AbstractDomainObject should be instantiated instead of the original AbstractDomainObject wherever applicable.

The AbstractDomainObject is instantiated with the help of the below code fragment:

```
IAbstractDomainObject domainObject=null;
        trv {
            if (retrieveDataTransferObjectDTOMapElement("
<Fully Qualified Name of DataTransferObject from Naming Convention Rules
>").getDictionaryArray() == null) {
                domainObject = <Current Process Of Instantiation>;
            } else {
           domainObject=(IAbstractDomainObject)
                   getCustomizedDomainObject (
retrieveDataTransferObjectDTOMapElement (
                                               "<Fully Qualified Name of
DataTransferObject from Naming Convention Rules >"));
/******** In AbstractAssembler.java, we have defined the method
public IAbstractDomainObject getCustomizedDomainObject(DataTransferObject
dataTransferObjectDTO)
This method instantiates the Customized AbstractDomainObject based on the value of
the attribute "dictionaryArray" of the DataTransferObject passed as the only
parameter. The method also populates this customized domain object with the extra
attribute values also from the "dictionaryArray" attribute and finally returns
this instance of the Customized Domain Object.
********/
```

```
}
} catch (Exception e) {
    domainObject = <Current Process Of Instantiation>;
}
```

19.4.2 Fetching of Customized Domain Objects

If a method has an input parameter that is an IAbstractDomainObject, the first line of the method in the assembler will be of the form:

```
populateAbstractDomainObjectMap("<Fully_Qualified_Name_
IAbstractDomainObject>", abstractDomainObject);
```

This method is defined in AbstractAssembler.java which newly instantiates referenceAbstractDomainObjectMap if required and populates the map with the above entry.

This map is used as a set of globally available IAbstractDomainObject's which can be retrieved by invoking another method defined in AbstractAssembler.java which is of the form:

```
retrieveDataTransferObjectDTOMapElement("<Fully_Qualified_Name_
IAbstractDomainObject>");
```

Whenever any DataTransferObject is instantiated, we populate its 'dictionaryArray' attribute immediately after it's instantiation.

In AbstractAssembler.java, we have defined the method à

public Dictionary[] getDictionaryArray(IAbstractDomainObject obj)

This method creates and returns a dictionary array from the IAbstractDomainObject passed to it as input parameter.

Example of final piece of code:

Figure 19–13 Instantiation of DataTransferObjects



19.5 Customized Domain Object ORM Configuration

This section describes the details of customized domain object ORM configuration.

19.5.1 Case 1 - Non-Inheritance based mapping

Non-inheritance based mapping refers to those domain objects that are not mapped as a Hibernate Subclass or Union-Subclass or Joined-Subclass. Let us take the example of the class MessageDataAttribute. The fully qualified class name is

'com.ofss.fc.domain.ep.entity.dispatch.message.MessageDataAttribute'. This class has been mapped in ep.messagetemplate.hbm.xml.

Adding Discriminator column mapping in existing HBM file

Add the discriminator as:- <discriminator column="DOMAIN_OBJECT_EXTN" type="string"/>

For the purpose of identifying the extended domain object in the corresponding table, add a 'discriminator column' in the corresponding table and update the hibernate file. The name of the discriminator column used is DOMAIN_OBJECT_EXTN and the default discriminator value defined is 'CZ'

So any normal Create or Update operation will have a value 'CZ' for DOMAIN_ OBJECT_EXTN column.

1	🕑 Java - config_25Jun/orm/hibernate/hbm/ep/ep.messagetemplate.hbm.xml - Eclipse										
1	<u>F</u> ile <u>E</u> dit <u>S</u> ource <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp										
J		•	ତ୍ରି 🚔] 券 ▾ Q ▾ Q ▾ T 🖶 G ▾] 🕸 🖨 🖉 ▾] 注 શ] 🖞 ▾ 🏹 ▾ 🏷 🌣 ଦ ଦ ଦ જ								
F	Ţ[Cate	egoryListenerMDB. 🕼 RejectCodeAssembler. 🕼 AbstractAssembler.ja 👔 DDABankPolicyAssembl 🕼 AssemblerOperationCo 😢 ep.messagetemplate.h 🛛 "2 🦷 🖻								
		46 47 48	<pre><class cod_mess_tmpl_id"="" discriminator-value="CZ" dynamic-insert="false" dynamic-update="false" lazy="false" mutable="true" name="messageTemplateId" type="string"></class></pre>								
L		51	<pre><!-- key-property column= AllK_TYPE_name= attributelype_type= string /-->> /discriminator_column="OPMATW_OBJECT_FYTW" tune="string"/></pre>								
L		53	<pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>								
L		54	<pre><filter '#'="" (=""] ="" cod_attr_id="" cod_mess_tmpl_id="" condition="cod_attr_id '#' cod_mess_tmpLid in (SELECT rd.reference_dato_id FROM Fix_HE_REFERENCEDATA_B rd, FLX_HE_DATASET_E</pre></th></tr><tr><th>L</th><th></th><th>55</th><th><pre>sfilter condition=" ds,<="" flx_me_dataset_b="" flx_me_referencedata_b="" from="" in="" pre="" rd,="" rd.reference_data_id="" select="" =""></filter></pre>								
L		56	<pre>cfilter condition="cod_attr_id '#' cod_mess_tmpl_id in (SELECT rd.reference_data_id FROM FLX_ME_REFERENCEDATA_B rd, FLX_ME_DATASET_B ds,</pre>								
L		57	<pre>cfilter condition="cod_attr_id '#' cod_mess_tmpl_id in (SELECT rd.reference_data_id FROM FLX_ME_REFERENCEDATA_B rd, FLX_ME_DATASET_B ds,</pre>								
L		58	<pre><fiiter condition="cod_attr_id '#' cod_mess_tmpl_id in (SELECT rd.reference_data_id FROM FLX_ME_REFERENCEDATA_B rd, FLX_ME_DATASET_B ds,</pre></pre>								
L		59									
L		¢n.	4 D								
	(Design	Source								
ļ	•		🗎 hibernate-mapping/class/composite-id/#text Writable Smart Insert 53:1 Ioading http://www.hiapping-3.0.dtd 🔳 🐄 🖉 # 🕵 @ 😣 🖳 🗄 🔗								

Figure 19–14 Adding Discriminator Column Mapping in Existing HBM file

A new HBM file mapping to Customized Domain Object is added

The following figure explains adding a new HBM file mapping to Customized Domain Object.

Figure 19–15 HBM File Mapping to Customized Domain Object



For example a new file CustomizedMessageDataAttribute.hbm.xml is introduced to include the extra attributes added by consulting or any other third party along with

the discriminator value. This file will map to the new customized domain object and will be extending the existing Abstract Domain Object.

Adding new Java File corresponding to the Customized Domain Object

The following figure explains adding new Java file corresponding to the Customized Domain Object.

Figure 19–16 Adding New Java File to the Customized Domain Object



A Java File is added corresponding to the existing Abstract Domain Object. This will be extending the Abstract Domain Object that we are extending.

Adding extra columns along with the discriminator column to the domain object table

The following figure explains adding a new Java file corresponding to the Customized Domain Object.



Figure 19–17 Adding Extra Columns along with the Discriminator Column

The extra columns along with the discriminator column have to be added to the domain object table of this domain object.

In case of Creation or Updation of 'CustomizedMessageDataAttribute' instead of 'MessageDataAttribute' the new discriminator column 'DOMAIN_OBJECT_EXTN' has the value of 'FCMA' instead of 'CZ' and an additional value in columns 'CUSTOM_VALUE1' and 'CUSTOM_VALUE2' in table FLX_EP_MSG_ATTR_B.

In case of Creation or Updation of 'MessageDataAttribute' the new discriminator column 'DOMAIN_OBJECT_EXTN' has the value of 'CZ' and NULL values in columns 'CUSTOM_VALUE1' and 'CUSTOM_VALUE2' in table FLX_EP_MSG_ATTR_B.

19.5.2 Case 2 - Mapped as a Hibernate Subclass

The maintenance domain objects which are mapped as a Hibernate Subclass already have an existing discriminator. For the purpose of identifying the extended domain object in the same table, we shall be using the existing discriminator.

Let us take the example of 'com.ofss.fc.domain.party.entity.contact.Cellular'. This is mapped as a subclass in ContactPoint.hbm.xml.

A new HBM file mapping to Customized Domain Object is added

The following figure explains adding a new HBM file mapping to Customized Domain Object.



Figure 19–18 Adding a New HBM File Mapping to Customized Domain Object

A new file FirstCustomizedCellular.hbm.xml is introduced to include the extra attributes added by consulting or any other third party along with the discriminator value 'FCLR'. This file will map to the new customized domain object 'com.ofss.fc.domain.party.entity.contact.FirstCustomizedCellular' and will be extending the existing Abstract Domain Object which is 'com.ofss.fc.domain.party.entity.contact.Cellular'.

Adding new Java File corresponding to the Customized Domain Object

The following figure explains adding a new Java File corresponding to the Customized Domain Object.

Figure 19–19 Adding New Java File to Customized Domain Object



A Java File 'com.ofss.fc.domain.party.entity.contact.FirstCustomizedCellular' is added corresponding to the existing Abstract Domain Object. This will be extending the Abstract Domain Object that we are extending.

Adding Extra Columns to the Domain Object Table

The extra columns have to be added to the domain object table of this domain object.

In this case GRAPHICS_MODE column is added to FLX_PI_CONTACT_POINT table.

So in case of Creation or Updation of 'FirstCustomizedCellular' instead of 'Cellular' the existing discriminator column 'CONTACT_POINT_TYPE' has the value of 'FCLR' instead of 'CLR' and an additional value in column 'GRAPHICS_MODE' in table FLX_PI_CONTACT_POINT.

And in case of Creation or Updation of 'Cellular' the existing discriminator column 'CONTACT_POINT_TYPE' has the value of 'CLR' and NULL values in column 'GRAPHICS_MODE' in table FLX_PI_CONTACT_POINT.

19.5.3 Case 3 - Mapped as a Hibernate Union-Subclass or Joined-Subclass

Let us take the example of

'com.ofss.fc.domain.lcm.entity.limits.facility.proposedFacility.ProposedFacility'. This class has been mapped in Facility.hbm.xml as a union subclass.

Use the customized entity

'com.ofss.fc.cz.nab.domain.lcm.entity.limits.facility.proposedFacility.CustomizedPropo sedFacility' for the purpose of extensibility of this domain object.

Adding Discriminator in HBM file where base class has been mapped is not required

The existing Facility.hbm.xml file which contains the mapping for "com.ofss.fc.domain.lcm.entity.limits.facility.proposedFacility.ProposedFacility" is not required to be altered.

A new HBM file mapping to Customized Domain Object is added

The following figure explains adding a new HBM file mapped to new Customized Domain Object.

Figure 19–20 New HBM File Mapping



For example, a new file CustomizedProposedFacility.hbm.xml is introduced to include the extra attributes added by consulting or any other third party. This file will map to the new customized domain object and will be extending the existing Abstract Domain Object.

Adding new Java File corresponding to the Customized Domain Object

Figure 19–21 Adding New Java File

```
package com.ofss.fc.cz.nab.domain.lcm.entity.limits.facility.proposedFacility;
 import com.ofss.fc.domain.lcm.entity.limits.facility.proposedFacility.ProposedFacility;
public class CustomizedProposedFacility extends ProposedFacility {
     * Default serial version UID
     private static final long serialVersionUID = 1L;
     private String associatedConsumerLending;
     private String associatedBusinessLending;
     private String feeNegotiateApprovalCode;
     public String getFeeNegotiateApprovalCode() {
         return feeNegotiateApprovalCode;
     -}
     public void setFeeNegotiateApprovalCode(String feeNegotiateApprovalCode) {
         this.feeNegotiateApprovalCode = feeNegotiateApprovalCode;
     }
     public String getAssociatedConsumerLending() {
         return associatedConsumerLending;
     3
     public void setAssociatedConsumerLending(String associatedConsumerLending) {
         this.associatedConsumerLending = associatedConsumerLending;
     public String getAssociatedBusinessLending() {
         return associatedBusinessLending;
```

A Java File 'CustomizedProposedFacility.java' is added. This extends the Abstract Domain Object that we are extending.

Create a new table CZ_NAB_LM_PROPOSED_FACILITY similar to the Domain Object Table

We are extending that is, FLX_LM_PROPOSED_FACILITY_B and add the extra columns to the new table.

Figure 19–22 Create a New Table CZ_NAB_LM_PROPOSED_FACILITY

```
1 create table CZ_NAB_LM_PROPOSED_FACILITY as
2 select * from FLX_LM_PROPOSED_FACILITY_B where 1=2;
3
4
5 ALTER TABLE CZ_NAB_LM_PROPOSED_FACILITY ADD ASSOCIATED_CONSUMER_LENDING VARCHAR2(20)
6 /
7 ALTER TABLE CZ_NAB_LM_PROPOSED_FACILITY ADD ASSOCIATED_BUSINESS_LENDING VARCHAR2(20)
8 /
9
10 ALTER TABLE CZ_NAB_LM_PROPOSED_FACILITY add FEE_NEGOTIATE_APPROVAL_CODE VARCHAR2(50)
1 /
```

Adding Customized HQL Queries whenever the Domain Object is Referred

The following file has the attribute 'CustomizedHibernateQueriesConfig' to fire the Customized HQL if required: Preferences.xml.

The attribute is as follows:

<Preference name="CustomizedHibernateQueriesConfig"

- propertyFileName="com.ofss.fc.common.CustomizedHibernateQueriesConfig"
 syncTimeInterval="600000" />

The following files have also been changed to fire the Customized HQL if required.

com.ofss.fc.framework.domain@/com/ofss/fc/framework/repository/AbstractRepos itory.java

com.ofss.fc.common.jar@/src/com/ofss/fc/common/CustomizedHibernateQueriesC onfig.java

The following file has the attribute 'CustomizedHibernateQueriesConfigOverride' to fire the Customized HQL if required.

<lzn>/au/config/Preferences.xml

<Preference name="CustomizedHibernateQueriesConfigOverride"

PreferencesProvider="com.ofss.fc.infra.config.impl.JavaConstantsConfigProvider"
 parent=""

propertyFileName="com.ofss.fc.lz.au.common.CustomizedHibernateQueriesConfig"
 syncTimeInterval="600000"/>

Therefore, com.ofss.fc.lz.au.common.CustomizedHibernateQueriesConfig.java file needs to have the old HQL query name mapped to the customized HQL query name for this domain object.

Similarly, extensibility of domain objects mapped as joined-subclass can also be done.

19.5.4 Case 4 - Mapped as a Hibernate Component

This relates to only those component classes that implements IAbstractDomainObject and should be extensible.

The Java Class corresponding to this component class has to be extended and this new Java Class along with the additional attributes have to be mapped in the hibernate file.

The corresponding additional columns have to be added in the domain object table in question.

19.6 Extensibility using Dictionary in Origination Application

In this section, the Application Form screen (Fast path: OR097) of the Oracle Banking Platform is taken as an example.

19.6.1 ICustomDataHandler's as DictionaryArray Interceptor

The backing bean method of OR097 - Application Form 'com.ofss.fc.ui.taskflows.origination.application.applicationForm.view.backing.Applic ationForm.moveNext()' calls implementation of

com.ofss.fc.ui.taskflows.origination.application.common.handler.ICustomDataHandle r.

Implementation of

com.ofss.fc.ui.taskflows.origination.application.common.handler.ICustomDataHandle r can be configured in OriginationConfiguration.properties. Property name is **customDataHandler**

ApplicationFormHelper.getSubmissionInputDTO() will give the master DTO for the application form.

Figure 19–23 CustomDataHandler's as DictionaryArray Interceptor

ہ 😓	rack	le JDeveloper 11g Release 1 - View.jws : com.ofss.fc.ui.taskflows.origination.jpr : D:\UIWorkspace\View_03Dec	$\com.ofss.fc.ui.taskflows.origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofss\fc\ui\taskflows\origination\src\com\ofs\fc\ui\taskflows\origination\src\com\ofs\fc\ui\taskflows\origination\src\com\ofs\fc\ui\taskflows\origination\src\com\ofs\fc\ui\taskflows\origination\src\com\ofs\fc\ui\taskflows\origination\src\com\ofs\fc\ui\taskflows\origination\src\com\ofs\fc\ui\taskflows\origination\src\com\ofs\fc\ui\taskflows\origination\src\com\ofs\fc\taskflows\origination\src\com\ofs\fc\taskflows\origination\src\com\ofs\fc\taskflows\origination\src\com\ofs\fc\taskflows\origination\src\com\ofs\fc\taskflows\origination\src\com\ofs\fc\taskflows\origination\src\com\ofs\fc\taskflows\origination\src\com\ofs\fc\taskflows\origination\src\com\ofs\fc\taskflows\origination\src\com\ofs\fc\taskflows\origin\src\com\ofs\fc\taskflows\origin\src\com\ofs\fc\taskflows\origin\src\com\ofs\fc\taskflows\origin\src\com\ofs\fc\taskflows\origin\src\cou\ofs\fc\taskflows\origin\src\cow\ofs\staskflows\origin\src\cou\ofs\staskflows\origin\src\cow\ofs\staskflows\origin\src\cou\ofs\fc\taskflows\origin\src\cow\ofs\staskflows\origin\src\cou\ofs\staskflows\origin\src\ofs\staskflows\origin\src\ofs\staskflows\origin\src\ofs\staskflows\origin\src\ofs\staskflows\origin\staskflows\origin\src\ofs\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\staskflows\origin\stask$	\application\common\handler\ 💶 🗗 🗙					
<u>F</u> ile	File fdit View Application Refactor Search Navigate Build Bun Source Versigning Tools Window Help								
90	3	🗏 🗿 i 🧐 (*) 🕌 🛅 i 🔕 - 🔕 - i 🏤 🖆 🏯 - i 🕨 - 🌞 - 📕 - + 💷 i	98 JU 08 III> 18 😰 I 🛦	(ith - Search					
i ∰ As	lynd	dvonousApplicationFormProcessor.java 🛛 🔂 SynchronousApplicationFormProcessor.java 👘 🔝 applicationFormTe	emplate.jspx 🛛 🔯 ApplicationFormUtils.java 👘 🔂 ICustomDataHandler.java	🐻 DefaultDataHandler.java 👔 🕕 💌					
	• Fin	a 🕹 🚱 💥 🍻 🐗 i 🐘 🖨 🖓 🖓 🛤 i 🕽 🗰 🖓 🖓 🖉 🖓 👘 hi i 🔛 🐃							
1		Spublic class DefaultDataHandler implements ICustomDataHandler (public DefaultDataHandler() /							
	17	super();							
)							
- 1		public booleam onMovePrev(ApplicationFormHelper helper,							
		ApplicationTrainStepType currentTrainStep, ApplicationTrainStenType newtTrainSten) (
		approximately the motor and only (
		return falses							
·		}		•					
		public booleam onMoveNext(ApplicationFormHelper helper,							
		ApplicationTrainStepType currentTrainStep,							
		ApplicationTrainStepType nextTrainStep) (
		if ("FRODUCT_DETAILS".equalsIgnoreCase(currentTrainStep.toString())) (
•		ApplicantDataDTO applicantDataDTO = helper.getSubmissionInputDTO().get.	ApplicantDataList()[0];						
		com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePair	DT01= new com.ofss.fc.framework.domain.common.dto.NameValue	PairDTO();					
		<pre>nameValueFairDT01.setGenericName("com.ofss.fc.domain.origination.entit nameValueFairDT01.setValue("No");</pre>	y.core.applicant.CustomizedApplicant.CriminalRecord");	1					
		com.ofss.fc.framework.domain.common.dto.NameValuePairDT0[] nameValuePa	irDTOArray= new com.ofss.fc.framework.domain.common.dto.Name	eValuePairDTO[1];					
		nameValueFairDTOArray[0]=nameValuePairDTO1; com.ofss.fc.framework.domain.common.dto.Dictionary dictionary= new com	ofss.fc.framework.domain.common.dto.Dictionary();						
		dictionary.setNameValuePairDTOArray(nameValuePairDTOArray);							
		com.ofss.fc.framework.domain.common.dto.Dictionary[] dictionaryArray =	new com.ofss.fc.framework.domain.common.dto.Dictionary[1];	-					
		dictionaryArray[0]=dictionary;							
		applicantDataDTO.setDictionaryArray(dictionaryArray);							
)							
••)							
		mublic unid enlead/invlicationFormWalner balner							
	8	ApplicationTrainStepType currentTrainStep) (
		}							
-		public void onSave(ApplicationFormHelper helper,							
	8	ApplicationTrainStepType currentTrainStep) {							
		1							
1	6	B public void onSubmit(ApplicationFormHelper helper) (
		*							
Source	e la	public void onTrainStepClick(ApplicationFormHelper helper, Design History <							
D: JUEW	lorks	space (view_03Dec \com.ofss.fc.ui.taskflows.origination \src\com\ofss \fc\ui \taskflows \origination \application	Form \view \utils \Synchronous ApplicationFormProcessor .java Line 40 Column 1	Insert Unix/Mac: LP Debugging					
a St.	art	🐃 👩 🌾 🚯 🜆 🔊 🚨 斗 🔛 🖂	6 %	\$ ₽> 9m (7:57 PM					
				12/5/2013					

This hook should be used to populate the dictionary array of the concerned DTO at the correct stage of application form entry.

19.6.2 Create Customized Abstract Domain Object Class

A new Java File is added corresponding to the existing Abstract Domain Object. This extends the Abstract Domain Object that we are extending.



Figure 19–24 Create Customized Abstract Domain Object Class

19.6.3 Create Customized Abstract Domain Object Hibernate Mapping File

A new file hbm.xml is introduced to include the extra attributes added by consulting or any other third party along with the discriminator value. This file maps to the new customized domain object and extends the existing Abstract Domain Object.

Figure 19–25 Create Customized Abstract Domain Object Hibernate Mapping File



19.6.4 Create Customized Abstract Domain Object Attribute Columns

The extra columns have to be added to the domain object table of this domain object.

Figure 19–26 Create Customized Abstract Domain Object Attribute Columns



In case of Creation or Updation of 'CustomizedApplicant' instead of 'Applicant' the existing discriminator column 'DOMAIN_OBJECT_EXTN' has the value of 'CUST' instead of 'CZ' and an additional value in column 'CRIMINAL_RECORD' in table FLX_OR_APPLICANTS.

In case of Creation or Updation of 'Applicant' the existing discriminator column 'DOMAIN_OBJECT_EXTN' has the value of 'CZ' and NULL values in column 'CRIMINAL_RECORD' in table FLX_OR_APPLICANTS.

Similarly, other DomainObjectDTO's can have their dictionary arrays populated in the ICustomDataHandler class being used and the corresponding customized domain object will get persisted instead of the usual domain object.

19.7 Extensibility using Attributes of Various Supported Datatypes

Extensibility of maintenance domain objects now supports extended attributes with all data types that have a public constructor with a single argument of data-type "String".

This includes attributes of data-type"com.ofss.fc.datatype.Date" whose "toString()" method should be invoked to set its value in NameValuePairDTO array element of Dictionary array. The value set is of the format given in root.properties file.

Additionally extensibility of maintenance domain objects is now also supporting extended attributes with enumeration data types defined in "com.ofss.fc.enumeration" project.

Here is an example of extensibility of

"com.ofss.fc.domain.ep.entity.dispatch.message.MessageTemplate" using attributes of different supported datatypes.

The following customized class is created that contains the additional attributes.

```
Figure 19–27 Customized Message Template Class
```

```
package com.ofss.fc.domain.ep.entity.dispatch.message;
import com.ofss.fc.datatype.Date;
import com.ofss.fc.enumeration.ep.DestinationType;
public class CustomizedMessageTemplate extends MessageTemplate{
    private static final long serialVersionUID = 376283690240542791L;
    private Integer attributeInteger;
    private Boolean attributeBoolean;
    private String attributeString;
    private Date attributeDate;
    private DestinationType attributeEnum;
    public Integer getAttributeInteger() {
        return attributeInteger;
    }
    public void setAttributeInteger(Integer attributeInteger) {
        this.attributeInteger = attributeInteger;
    }
   public Boolean getAttributeBoolean() {
        return attributeBoolean;
    }
   public void setAttributeBoolean(Boolean attributeBoolean) {
        this.attributeBoolean = attributeBoolean;
    }
    public String getAttributeString() {
        return attributeString;
    }
    public void setAttributeString(String attributeString) {
        this.attributeString = attributeString;
    }
    public Date getAttributeDate() {
                74 4 5 6 6 5
```

The following extra columns have been added in the domain object table "flx_ep_msg_tmpl_b".

Figure 19–28 Domain Object Table

	Name	Туре		Nullable	Default	Storage	Comments
۲	COD_TMPL_ID	VARCHAR2(100)	۳				Indicates unique message template id
	DESTINATION_TYPE	VARCHAR2(20)	٠	v			Indicates destination type like SMS,Email
	MSG_TMPL_NAME	VARCHAR2(100)	٠	V			Indicates message template name
	MSG_TMPL_DESC	VARCHAR2(100)	٠	V			Indicates message template description
	TXT_MSG_TMPL	CLOB	٠	v			Indicates message template buffer
	CREATED_BY	VARCHAR2(64)	٠	V			Indicates the creator
	CREATION_DATE	DATE	٠	v			Indicates the creation Date
	LAST_UPDATED_BY	VARCHAR2(64)	٠	V			Indicates the approver of the transaction
	LAST_UPDATE_DATE	DATE	٠	v			Indicates the last updated date
	OBJECT_VERSION_NUMBER	NUMBER(9)	٠	v			Indicates the version number. Defaults to 1 for new instances.
	OBJECT_STATUS	VARCHAR2(5)	٠	v			Indicates current status of the entity.
	TXT_SUBJECT_TMPL	CLOB	•	V			Indicates message for subject Buffer
Г	DOMAIN_OBJECT_EXTN	VARCHAR2(100)	٠	V			
Г	CUST_INTEGER	NUMBER(9)	٠	v			
T	CUST_BOOLEAN	VARCHAR2(5)	٠	v			
T	CUST_DATE	DATE	٠	~			
T	CUST_STRING	VARCHAR2(100)	•	~			
	CUST_ENUM	VARCHAR2(100)	٠	V			
*				N N			

The following hibernate file maps the customized class attributes with the table columns.

Figure 19–29 Hibernate File

```
<?xml version="1.0" encoding="UTF-9" standalone="no"?>
<!-- Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved. --><!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate H
</pre>
<hipernate-mapping auto-import="false" default-access="property" default-cascade="none" default-lazy="true">
    <subclass discriminator-value="CDST"</pre>
                              dynamic-insert="true"
                              dynamic-update="true"
                              uynamic-apuale= true
lary="fialse"
name="com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate"
extends="com.ofss.fc.domain.ep.entity.dispatch.message.NessageTemplate"
select-before-update="false">
                 select-berore-upuate folse
<property name="attributeInteger"
column="CUST_INTEGER"
generated="never"
lazy="folse"
optimistic-lock="true"
                optimistic-lock="true"
type="int"
unique="false"/>
<property name="attributeBoolean"
column="CDST_BOOLEAN"
generated="never"
lazy="false"
optimistic-lock="true"
two="heel co"
                 optimistic-lock= true
type="boolean"
unique="false"/>
column="distributeString"
column="Cist_STRING"
                                       generated="never"
lazy="false"
optimistic-lock="true"
                 optimistic-lock="tru
types"string"
uniques"false"/>
<property name="attributeDate"
column="dist_DATE"
generated="never"
laze="false"
ortimistic-lock="tru
                                        optimistic-lock="true"
                                       type="cam.ofss.fc.datatype.Date"
unique="false"/>
                  <property name="attributeEnum
column="CUST_ENUM"</pre>
                                       generated="never"
lazy="false"
optimistic-lock="true"
                                       type="DestinationType"
unique="false"/>
  </subclass>
```

</hibernate-mapping>

The following JUnit test case has been used to test a "create" operation.

Figure 19–30 JUnit Test Case

```
try {
    deleteMessageTemplate(testCase);

           com.ofss.fc.framework.domain.common.dto.Dictionary[] dictionaryArray= new com.ofss.fc.framework.domain.common.dto.Dictionary[];
com.ofss.fc.framework.domain.common.dto.Dictionary dictionaryObject = new com.ofss.fc.framework.domain.common.dto.Dictionary();
            com.ofss.fc.framework.domain.common.dto.NameValuePairDTO[] nameValuePairDTOArray= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO[5];
            com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePairDTO0= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO();
            nameValuePairDTO0.setGenericName("com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedNessageTemplate.AttributeInteger");
            nameValuePairDTO0.setValue("100");
            nameValuePairDTOArray[0]=nameValuePairDTO0;
           cam.ofss.fc.fnamework.damain.common.dto.NameValuePairDTO nameValuePairDTO1= new cam.ofss.fc.fnamework.damain.common.dto.NameValuePairDTO();
nameValuePairDTO1.setGenerieName("com.ofss.fc.damain.ep.entity.dispatch.message.CustomizedNessageTemplate.AttributeBoolean");
nameValuePairDTO1.setValue("false");
nameValuePairDTOArray[1]=nameValuePairDTO1;
            com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePairDTO2= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO();
            nameValuePairDT02.setGenericName("com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedNessageTemplate.AttributeString");
nameValuePairDT02.setValue("ABCDEFR");
            nameValuePairDTOArray[2]=nameValuePairDTO2;
            com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePairDTO3= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO();
            nameValuePairDTO3.setGenericName("com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate.AttributeDate");
            Date newDate = new Date();
nameValuePairDTO3.setValue(newDate.toString());
            nameValuePairDTOArray[3]=nameValuePairDTO3;
           cam.ofss.fc.framework.damain.common.dto.NameValuePairDTO nameValuePairDTO4= new cam.ofss.fc.framework.damain.common.dto.NameValuePairDTO();
nameValuePairDTO4.setSemeriRName("com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedNessageTemplate.AttributeEnum");
nameValuePairDTO4.setSuble(com.ofss.fc.enumeration.ep.DestinationType.EM4IL.getEnumValue());
nameValuePairDTOArray[4]=nameValuePairDTO4;
            dictionaryObject.setNameValuePairDTOArray(nameValuePairDTOArray);
            dictionaryArray[0]=dictionaryObject;
messageTemplateDTO.setDictionaryArray(dictionaryArray);
            TransactionStatus result = applicationService.addMessageTemplate(sessionContext, messageTemplateDTO);
assertEquals(result.getErrorCode(), FAPIErrorConstants.MID_SUCCESS);
dumpTransactionStatus("MessageTemplateApplicationService", "testAddMessageTemplate", result);
     } catch (FatalException e) {
    dumpFatalException("HessageTemplateApplicationService", "testAddHessageTemplate", e);
    fai("Indexpected failure from " + THIS_COMPONENT_UNAGE + ".testAddHessageTemplate");
}
```

The above JUnit runs to add the following record in the table.

Figure 19–31 JUnit Adds Table Record

			_
	Row 1	Fields	Γ
►	COD_TMPL_ID	Junit_Message	1
	DESTINATION_TYPE		
	MSG_TMPL_NAME	Junit message template 🔤	
	MSG_TMPL_DESC	Message template description via junit test cas	
	TXT_MSG_TMPL	<clob> ····</clob>	
	CREATED_BY	ofssuser	1
	CREATION_DATE	7/8/2014 6:40:34 PM	1
	LAST_UPDATED_BY	ofssuser	1
	LAST_UPDATE_DATE	7/8/2014 6:40:34 PM	1
	OBJECT_VERSION_NUMBER	1	1
	OBJECT_STATUS	A	1
	TXT SUBJECT TMPI		
	DOMAIN_OBJECT_EXTN	CUST	1
	CUST_INTEGER	100	1
	CUST_BOOLEAN	0	1
	CUST_DATE	7/8/2014 6:40:24 PM	1
	CUST_STRING	ABCDEFR ···	1
	CUST_ENUM	EMAIL	1

Similarly, a JUnit is run to do "fetch" operation. This fetches the customized record whose dictionary array values have been shown below.





Deployment Guideline

This chapter explains the deployment guidelines.

20.1 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.

20.2 Database Objects

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

20.3 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 20–1 Extensibility Deployment



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.



Acci	ount 👻 Back Office 👻 CASA 👻	Channel 🔻	Collection 🔻	LCM -	Loan 🔻	Origination 🔻	Party 🔻	Payment And	Collection 🔻	Securi	** 🚖	★ → Fasi Path	t [
	LCM <u>0</u> 22												
Pe	erfection Capture												
								🚇 <u>R</u> ea	ad + Cr <u>e</u> ate	🗎 Prir	t √ 0 <u></u>	🤄 🤣 C <u>l</u> ear	🛛 Cancel
11													
P	erfect Charge												
	Collateral Perfection Details												
	Charge Registration Number			_			*	Charge Status	Proposed				_
	Execution Date		12h				Title Doci	iments Status			•		
	* Charge Registration Date		100			Charg	e Registra	tion Required	Yes				
	* Date of Stamping		10				Stam	ping Required	Yes				
	* Registration Amount	\$0.00		Al	JD								
1	* Stamping Amount	\$0.00		Al	JD								=
Г	End Date Changed		120				Pr	ocessed Date	[20		1	
	Amended Date		120					Amended By					
	Removed Date		20					Removed By					
	Token						Secure	d Party Group					
	Change Number							Origination					
	Giving Of Notice							Transitional					
	Earlier Registration Number	[
												ОК	
	View Document Details												
	View 👻 🛃 Detach												
Ir	ndex Type Index value												
		<u></u>											

21.1 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 21–2 Localization Implementation Architectural Change

The Package structure for the implementation is shown below:





21.2 Customizing UI Layer

This section explains the customization of UI layer.

21.2.1 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:

Figure 21–4 Customization of the JDeveloper





Figure 21–5 Customization Context in Customization Developer Role

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.

File Edit View Application Refactor Search Navigate Build Run Versioning Tools Window Help								
The Tax The Thissen wester Terral Handare Tana Han Academic Toola Handar Heb								
📄 🗁 🗐 🗇 (1) 🗏 🗎 🛅 (1) 🗸 - (2) - (2								
😰 Application 🗴 😋 9PM Pro X 🕼 X 🔤 🗠 CustomizationLayerYalues.xml X 🖤 🖆 ConstantizationLayerYalues.xml X 🖤 🎬 Component Palet	te ×							
Al Pages								
Projects 🖸 🖏 🖓 🕆 💯 - Business Components								
Com. cons. fs. i.e. au.ui.model.lcm MDS Configuration MDS Configuration MDS Configuration	(VMI)							
Controler Controler Controler Controler	(Inc)							
Conconfiss.fct.au.ui.view.lom	tustion (VMI.)							
B C C C C C C C C C C C C C C C C C C C	rucuori (xmic)							
The controls - Customization Classes	a							
and address for using the second seco	(nttp://xmins.oracie							
com.ofss.fc.lz.au.ui.optioncc.OptionCC	ntig (http://xmins.o							
Application Resources OP BC42Config (http://www.application.executi	://xmins.oracle.con							
Energia Connections	tp://xmlns.oracle.cc							
Descriptors	://xmlns.oracle.com							
Configure Device Two D	n (http://xmlns.orad							
application.xm Configure Lesion interfuences application.xm workers application.xm	http://xmlns.oracle							
with the second se	config (http://xmlns							
jos-config.xml with the second seco	://xmlns.oracle.com							
September 2015 Septem	(http://xmlns.oracle							
a- DF META-INF	usage (http://xmlns							
antespace (http://www.antespace.com/antes	://xmlns.oracle.com							
a connections.xml	in (http://xmlps.ora							
	>							
Contraction and a second	or ×							
	P 👸 Find ,							
b Data Controls								
Becent Coned Files								
T CustomizationLayerValues.sml - Structure ×								
B-C								
Cust-layers Messages - Ion X Smulatons X 200 numeritation X View - Customization Context X								
Q cust-layer								
B - O custager =								
Customation layers und are only meant to a RARNING: Project "com.ofss.fc.lz.au.ui.view.lcm.jpr" Al Tolever Name Value								
The second secon	-							

Figure 21–6 Configure Design Time Customization layer

Figure 21–7 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.

Figure 21–8 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 21–9 MDS Configuration

🍲 Oracle JDeveloper 11g Release 1 - View.jws : com.ofss.fc.ui.vie	w.jpr : D:\R2BOracle\LZN\vie	w\.adf\META-INF\adf-config.xml	_ 0 <mark></mark> ^
<u>File Edit View Application Refactor Search Naviga</u>	te <u>B</u> uild <u>R</u> un Versi <u>o</u> ni	ing <u>T</u> ools <u>W</u> indow <u>H</u> elp	
- 🔒 : - 🔾 - 🖸 : 🏥 🐰 🐫 19 (* : 3) - 1 🏯 -	🚵 🎎 🚢 🛥 - 🔈 -	🅸 - 📕 - + 💷 🖽 JI, 💷 II- II. 👚 I 🛆 I 🏣 🗐 🧐 🏉 🗺	💏 🕶 Search
Application × 🖓 Database Navig × 🛞 × 🎧 × 💷	🐼 adf-config.xml 🗡	•	Component Palette ×
🔁 View 💌 🐨 🔹			All Pages
🗢 Projects 💽 🖓 🗸 📴 🕶 🗌	Business Components		m (
e	MDS Configuration	 Note that additional configuration can be edited manually in the source. 	() #rdata-section (XMI.)
com.ofss.fc.ui.view	Controller	🗆 Customization Configuration: Match Path = "/" 🛛 🐥 💥	#comment (XML)
E-Contorss. IC.12. ad. di. taskilows	View	Use the following editor for cases where the customization configuration man 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/a
		com.ofss.fc.lz.au.ui.optioncc.OptionCC	BC4JConfig (http://xmlns.oracle.com/bc4j/confi
		4	Cache-config (http://xmlns.oracle.com/mds/con
		⊻	Cust-config (http://xmins.oracle.com/mds/config
			event-notification (http://xmins.orade.com/md:
		Configure Design Time Customization Layer Values	mds-component-config (http://xmins.oracle.com/mus/c
			mds-component-comp (http://xmins.oracle.com/mds/confic oracle.com/mds/confic
✓ Application Resources			> metadata-store (http://xmlns.oracle.com/mds/c
			metadata-store-usage (http://xmlns.oracle.com
E Descriptors			namespace (http://xmlns.oracle.com/mds/config
META-INF			persistence-config (http://xmlns.oracle.com/md
cwallet.sso			type-config (http://xmlns.oracle.com/mds/confi
jazn-data.xml			untyped-dependency-config (http://xmlns.orac
weblogic-application.xml			
adf-config.xml			
connections.xml			-
services			
MDS DI			
		~	
b Data Gardada	Overview Source History <	> • • •	
Data controls Q Y Recently Opened Files	Elog × ● Breakpoints ×	Smart Data × 113Data × 113Watches × 113ADF Data × 🕼 × 🖃 × 🗐	
(<<		Actions ▼	
Structure × Stack × ADF Structure × .	at weblogic	.servlet.internal.WebAppServletContext\$ServletInvocationAction.wrapRu	
	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 21–10 Manually Add entries



21.2.2 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.

21.2.3 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 21–11 MAR Creation



Step 2

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



Figure 21–12 MAR Creation - Application Properties

Step 3 Select Deployment and click New.



Figure 21–13 MAR Creation - Create Deployment Profile

Step 4 Select the MAR File option.

State JDeveloper 11g Release 1 - View.jws : com.o	fss.fcuiviewjpr	_ 0 _ X
File Edit View Application Refactor Search	n Navigate Build Run Versioning Tools Window Help	
	· 🎄 · 🎄 🖄 🛳 · 🕨 · 🕸 · 🔲 · + 💷 🖬 🛄 🛄 🕼 🕼 🏦 (🛦) 🎲 🕘 🏈 🏈 🖂	(Search
Application X 🖾 Database Navig X 🛞 X 🖓	x 。	Component Palette X
Application × Database Navig × × Vew Projects Projects Projects	Application Properties - D:R280racle/LZNview/View.jws Application Properties - D:R280racle/LZNview/View.jws Depkoyment Custom Cust	Component Palette X
	Help OK Cancel	

Figure 21–14 MAR Creation - MAR File Selection

Step 5

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

Stracke JDeveloper 11g Release 1 - View.jws : com.o	vfssfcuiview.jpr	_ 0 X
File Edit View Application Refactor Search	h Navinate Ruild Run Versioning Tools Window Help	
File Edit Yiew Application Refactor Search Refactor Refactor Refactor Refactor Refactor Refactor Refactor Refactor Refactor Refactor Refactor Refactor Refactor Refactor R	h Navigate Build Run Versioning Tools Window Help	Component Palette x
B Gonthmi B Gonthmi	Opplyment Applicat Create Deployment Profile Custom Clk OK to create your new deployment profile and immediately open it to see its configuration. Advice Type: Report WS Pole Bane: Immediately open it to see its configuration. Bane: Immediately open its configuration. Bane: Immediately open	No Components

Figure 21–15 MAR Creation - Enter Details

Step 6

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


Figure 21–16 MAR Creation - ADF Library Customization

Step 7

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 21–17 MAR Creation - Edit File



Step 9

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



Figure 21–18 MAR Creation - Application Assembly

21.3 Source Maintenance and Build

This section describes the source maintenance and build details.

21.3.1 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.

For page definition	
File name (with path)	adfmsrc/com/ofss/fc/ui/view/lcm/collaterals/collateralPerfectionCapture/pageDefn/CollateralPerfectionCapturePageDef.xml
Meta-data file name (with path)	$com\ofss\fc\ui\view\collaterals\collateralPerfectionCapture\pageDefn\mdss\ys\cust\option\LZ\CollateralPerfectionCapturePageDef.xml.xml$
For Screens	
File name (with path)	com/ofss/fc/ui/view/lcm/collaterals/collateralPerfectionCapture/form/Collateral PerfectionCapture.jsff
Meta-data file name (with path)	$com\ofss\fc\ui\view\collaterals\collateralPerfectionCapture\form\mdssys\cust\option\LZ\CollateralPerfectionCapture.jsff.xml$

Table 21–1 Path Structure

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.

21.3.2 .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

21.3.3 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.

21.4 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.

Figure 21–19 Package Deployment



OCH Integration

This chapter describes how additional information can be added to an Oracle Customer Hub (henceforth mentioned as OCH) publish request. Publishing additional information can be required base on the client requirements, and hence OBP Integration adapters and assemblers need to be extended for such additional informations. Integration adapters are used for gathering data related to a customer, which is further used by assemblers to map OBP DTO to AIA Enterprise Business Objects (henceforth mentioned as EBOs).

OBP OCH integration involves the following steps:

- 1. Fetching all the data related to customer depending on the use case
- 2. Conversion of OBP DTO to AIA EBOs
- 3. Posting the EBO to AIA queue using Asynch JMS protocol

Integration adapters are invoked from the post hook of application service extensions. After the successful execution of the use case, adapters further call Integration assemblers for conversion of DTO to EBO.

After conversion, adapters post the serialized EBO request to AIA queue using Integration strategy, which is fetched on the basis of use case.

A few examples of Integration strategies are as follows:

- AsyncFireForgetIntegrationStrategyJMS: It is used in use cases where a response is not expected from OCH. Integration use cases involving creation/updation of customer information use this strategy.
- SyncIntegrationStrategy: It is used where a response is required from OCH. Uses cases, like Party Search or Party Deduplication where customer information is fetched from OCH, use this strategy.

A few examples of Integration adapters are:

- UpdatepartyAdapter: It is used for populating customer information.
- **ChangeAccountTitleAdapter**: It is used in use cases where customer's account information is to be published to OCH.

A few examples of Integration assemblers are:

- UpdatePartyAssembler: It is invoked from UpdatepartyAdapter and maps customer information to EBO attributes.
- **CreateAccountAssember**: It is invoked from ChangeAccountTitleAdapter and maps customer's account information to respective EBO attribute.

22.1 Integration Adapter Interface

OBP framework contains an interface, IIntegrationAdapter which provides two basic methods for OCH integration.

These two methods must be implemented by any adapter implementing the interface and use them for publishing data to OCH. Signature of these two methods are:

void update(SessionContext context, DomainObjectDTO dto, BaseResponse response)
throws FatalException;
Object updateWithResponse(SessionContext context, DomainObjectDTO dto,

BaseResponse response) throws FatalException;

Update() method is used in the use cases where response it not expected from OCH.

UpdateWithResponse() method is used when the data is required from OCH.

Figure 22–1 Integration Adapter Interface



22.2 Abstract Integration Adapter Class

OBP framework has an abstract class AbstractIntegrationAdapter which provides methods for common data, such as audit information or session context etc. This abstract class implements IIntegrationAdapter interface.

All adapters must extend AbstractIntegrationAdapter and implement the two methods defined in the IIntegrationAdapter interface.

Figure 22–2 Abstract Integration Adapter Class

```
public abstract class AbstractIntegrationAdapter implements IIntegrationAdapter {
    protected SessionContext sessionContext;
    protected String serviceId;
    private static final String ALL_SERVICES = "ALL";
     * Constructor that validates the service to be integrated.
    public AbstractIntegrationAdapter(SessionContext sessionContext, String serviceId) throws ConfigurationInitializationException {
        this.sessionContext = sessionContext;
        this.serviceId = serviceId;
boolean isAllowed = isIntegrationAllowed(sessionContext.getChannel(), serviceId);
        if ( !isAllowed) {
            throw new ConfigurationInitializationException(InfraErrorConstants.INTEGRATION_NOT_CONFIGURED);
        }
    }
    @Override
    public abstract void update(SessionContext context, DomainObjectDTO dto, BaseResponse response) throws FatalException;
    @Override
    public abstract Object updateWithResponse(SessionContext context, DomainObjectDTO dto, BaseResponse response) throws FatalException;
     * @return the sessionContext
    public SessionContext getSessionContext() {
        return sessionContext;
    }
    protected DomainObjectDTO populateCreateAuditInformation(SessionContext sessionContext, DomainObjectDTO dto) {
        dto.setCreatedBy(sessionContext.getUserId());
        dto.setLastUpdatedBy(sessionContext.getUserId());
        return dto;
    }
    protected DomainObjectDTO populateUpdatedAuditInformation(SessionContext sessionContext, DomainObjectDTO dto) {
        dto.setLastUpdatedBy(sessionContext.getUserId());
        return dto;
    }
```

22.3 Sample Integration Adapter

The following figure is a sample adapter for customer information:

Figure 22–3 Sample Integration Adapter



22.4 Integration Abstract Assembler

OBP framework has as abstract class AbstractAssembler which provides design for DTO to EBO conversion. These methods are used while mapping DTO to EBO and vice versa.

Signature of methods are:

```
public abstract T toCanonicalModel(D dto) throws FatalException;
public abstract D fromCanonicalModel(T domainObject) throws FatalException;
toCanonicalModel() is used when DTO is to be converted to EBO and
fromCanonicalModel() in the other case.
```

Figure 22–4 Integration Abstract Assembler

public abstract class AbstractAssembler<T extends ICanonicalModel, D extends DomainObjectDTO> {



All the assemblers must implement these two methods for conversion of DTO to EBO and vice versa.

Assemblers also populate the header of the request which is posted to the queue.

22.5 Sample Assembler

A sample assembler which extends AbstractAssembler should be like:

}

Figure 22–5 Sample Assembler

public class SampleAssember extends AbstractAssembler<SyncCustomerPartyListEBMType, IntegrationPartyOnBoardingDTO> {

```
@Override
public SyncCustomerPartyListEBMType toCanonicalModel(IntegrationPartyOnBoardingDTO dto) throws FatalException {
    //Populate OCH EBO using OBP DTO
   SyncCustomerPartyListEBMType syncCustomerPartyListEBMType = new SyncCustomerPartyListEBMType();
    List<SyncCustomerPartyListDataAreaType> syncCustomerPartyListDataAreaTypes = new ArrayList<SyncCustomerPartyListDataAreaType>();
   SyncCustomerPartyListDataAreaType dataArea = new SyncCustomerPartyListDataAreaType();
   //call to populate details using utility
   dataArea.setSyncCustomerPartyList(PartyAssemblerUtility.CustomerPartyData(dto));
    dataArea.setSync(new SyncType());
    syncCustomerPartyListDataAreaTypes.add(dataArea);
    syncCustomerPartyListEBMType.getDataArea().addAll(syncCustomerPartyListDataAreaTypes);
    //call to populate request header
    syncCustomerPartyListEBMType.setEBMHeader(PartyAssemblerUtility.createUpsert());
    syncCustomerPartyListEBMType.setLanguageCode("English");
    return syncCustomerPartyListEBMType;
@Override
```

public IntegrationPartyOnBoardingDTO fromCanonicalModel(SyncCustomerPartyListEBMType domainObject) throws FatalException {

```
// Populate OBP Entity using OCH EBO
IntegrationPartyOnBoardingDTO integrationPartyOnBoardingDTO = new IntegrationPartyOnBoardingDTO();
PartyOnBoardingDTO partyOnBoardingDTO = new PartyOnBoardingDTO();
//fetching value of party type
String partyTypeStr = domainObject.getDataArea().get(0).getSyncCustomerPartyList().getTypeCode().getValue();
PartyType partyType = (PartyType) EnumerationHelper.getInstance().fromValue(PartyType.class, partyTypeStr);
//setting party type in OBP DTO
partyOnBoardingDTO.setPartyType(partyType);
integrationPartyOnBoardingDTO.setPartyOnBoardingDTO(partyOnBoardingDTO);
return integrationPartyOnBoardingDTO;
```

User can extend assemblers to add more DTO to EBO mapping.

Note: EBOs are generated from AIA wsdl, and can be extended to add extra fields in the custom tag using the standard AIA extension framework. For each newly added field, customization developer must set that field in the assembler.

Appendix

The detailed list of adapters which can be used for extending and customizing the product is present in the Oracle Banking Platform Extensibility Guide - Adapter Usage Details.