Oracle® Banking Enterprise Product Manufacturing

Host Extensibility Guide

Release 2.7.0.0.0

F11754-01

November 2018

ORACLE®

Oracle Banking Enterprise Product Manufacturing Host Extensibility Guide, Release 2.7.0.0.0

F11754-01

Copyright © 2017, 2018, 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 **26**

Audience 26

Documentation Accessibility 26

Related Documents 26

Conventions 26

1 About This Guide **29**

1.1 Sections Not Applicable for Oracle Banking Enterprise Product Manufacturing 29

1.2 Sections Not Applicable for Oracle Banking Enterprise Originations 29

2 Objective and Scope **31**

2.1 Overview 31

2.2 Objective and Scope 31

2.2.1 Extensibility Objective 31

2.3 Complementary Artefacts 31

2.4 Out of Scope 32

3 Overview of Use Cases **33**

3.1 Extensibility Use Cases 33

3.1.1 Extending Service Execution 33

3.1.2 OBP Application Adapters 34

3.1.3 Extending Business Policy 34

3.1.4 User Defined Fields 35

3.1.5 Batch Framework Extension 35

3.1.6 Uploaded File Processing 36

3.1.7 Alert Extension 37

3.1.8 Create New Reports Using BI Publisher 38

3.1.9 Security Customization 39
3.1.10 Loan Schedule Computation Algorithm 41
3.1.11 Facts and Business Rules 41
3.1.12 Composite Application Service 42
3.1.13 ID Generation 43
3.1.14 OCH Integration 43
4 Extending Service Executions 45
4.1 Service Extension – Extending the "app" Layer 45
4.1.1 Application Service Extension Interface 46
4.1.2 Default Application Service Extension 47
4.1.3 Application Service Extension Executor 48
4.1.4 Extension Configuration 50
4.1.5 Application Service Extension Using Groovy 51
4.2 Extended Application Service Extension – Extending the "appx" Layer 52
4.2.1 Extended Application Service Extension Interface 54
4.2.2 Default Implementation of Appx Extension 55
4.2.3 Configuration 56
4.2.4 Extended Application Service Extension Executor 57
4.2.5 Application Service "appx" Extension using Groovy 59
4.3 End-to-End Example of an Extension 61
4.4 Support for Middleware Specific Tasks and Application service 65
4.4.1 Pre and Post Middleware Specific Transaction Tasks Overview 65
4.4.2 Sample Configuration 66
4.4.3 Custom Application Service 69
5 OBP Proxy Extension 71
6 OBP Application Adapters 75

6.1 Adapter Implementation Architecture 75
6.1.1 Package Diagram 75
6.1.2 Adapter Mechanism Class Diagram 77
6.1.3 Adapter Mechanism Sequence Diagram 77
6.2 Examples of Adapter Implementation 78
6.2.1 Example 1 – EventProcessingAdapter 78
6.2.2 Example 2 – DispatchAdapter 80
6.2.3 Example 3 - Adapter Implementation Using Groovy 81
6.3 Customizing Existing Adapters 83
6.3.1 Custom Adapter Example 1 – DispatchAdapter 83
6.3.2 Custom Adapter Example 2 – PartyKYCCheckAdapter 84
7 Business Policy Extension 89
7.1 Base Implementation of Business Policy 89
7.2 Extending Business Policy 90
7.3 Configuration 91
7.4 Extensions Using Groovy 91
8 Batch Framework Extensions 93
8.1 Typical Business Day in OBP 93
8.2 Overview of Categories 94
8.2.1 Beginning of Day (BOD) 94
8.2.2 Cut-off 94
8.2.3 End of Day (EOD) 95
8.2.4 Internal EOD 95
8.2.5 Statement Generation 95
8.2.6 Customer Communication 95
8.3 Batch Framework Architecture 95

8.3.1 Static View 96
8.3.2 Dynamic View 97
8.4 Batch Framework Components 99
8.4.1 Category Components 99
8.4.2 Shell Components 100
8.4.3 Stream Components 101
8.4.4 Database Components 103
8.5 Batch Configuration 103
8.5.1 Creation of New Category 103
8.5.2 Creation of Bean Based Shell 106
8.5.3 Creation of Procedure Based Shell 111
8.5.4 Population of Other Parameters 113
8.6 Batch Execution 115
9 Uploaded File Data Processing 117
9.1 Configuration 118
9.1.1 Database Tables and Setup 119
9.1.2 File Handlers 122
9.1.3 Record Handlers for Both Header and Details 123
9.1.4 DTO and Keys Classes for Both Header and Details 124
9.1.5 XFF File Definition XML 126
9.2 Processing 129
9.2.1 API Calls in the Handlers 129
9.2.2 Processing Adapter 130
9.3 Outcome 131
9.4 Failure/Exception Handling 132
10 Alerts Extension 133

10.1 Transaction as an Activity 133
10.1.1 Activity Record 133
10.1.2 Attaching Events to Activity 134
10.1.3 Event Record 134
10.1.4 Activity Event Mapping Record 135
10.1.5 Activity Log DTO 136
10.1.6 Alert Metadata Generation 136
10.1.7 Alert Message Template Maintenance 139
10.1.8 Alert Maintenance 140
10.2 Alert Subscription 141
10.2.1 Transaction API Changes 142
10.3 Alert Processing Steps 144
10.4 Alert Dispatch Mechanism 147
10.5 Adding New Alerts 150
10.5.1 New Alert Example 151
10.5.2 Testing New Alert 152
10.6 Support For Derived Facts 153
11 Creating New Reports Using BI Publisher 159
11.1 Data Objects for the Report 159
11.2 Catalog Folder 162
11.3 Data Source 163
11.4 Data Model 163
11.5 XML View of Report 166
11.6 Layout of the Report 167
11.7 View Report in BIP 168
11.8 OBP Batch Report Configuration - Define the Batch Reports 169

11.9 OBP Batch Report Configuration - Define the Batch Report Shell 170
11.10 OBP Batch Report Configuration - Define the Batch Report Shell Dependencies 170
11.11 OBP Batch Report Configuration 171
11.11.1 Batch Report Generation for a Branch Group Code 171
11.11.2 Batch Report Generation Status 171
11.11.3 Batch Report Generation Path 172
11.12 OBP Adhoc Report Configuration 172
11.12.1 Define the Adhoc Reports 172
11.12.2 Define the Adhoc Report Parameters 173
11.12.3 Define the Adhoc Reports to be listed in Screen 173
11.12.4 Adding Screen Tab for Report Module 173
11.13 Adhoc Report Generation – Screen 7775 174
11.14 Adhoc Report Viewing – Screen 7779 175
12 Security Customizations 179
12.1 OPSS Access Policies / Matrix Auth – Adding Attributes 181
12.1.1 Steps 181
12.1.1.1 Example of Matrix_auth conditional rule 184
12.2 OAAM Fraud Assertions – Adding Attributes 187
12.2.1 Steps 187
12.3 Security Validators 189
12.3.1 Customer Validators 190
12.3.2 Account Validators 190
12.3.3 Business Unit Validators 190
12.4 Customizing User Search 191
12.4.1 Steps 191
12.5 Customizing One-Time-Password (OTP) Processing Logic 191

12.5.1 Steps 191
12.6 Customizing Role Evaluation 191
12.6.1 Steps 192
12.7 Customizing Limits Exclusions 192
12.7.1 Steps 192
12.8 Customizing Business Rules 192
12.8.1 Steps to Update the Business Rules by Browser 192
12.8.2 Steps to Update the Business Rules in JDeveloper 202
13 Loan Schedule Computation Algorithm 207
13.1 Adding a New Algorithm 207
13.2 Consuming Third Party Schedules 210
14 Facts and Rules Configuration 211
14.1 Facts 211
14.1.1 Type of Facts 211
14.1.2 Facts Vocabulary 212
14.1.3 Generation of Facts using Eclipse Plug-in 213
14.1.4 Object Facts 233
14.2 Business Rules 236
14.2.1 Rules Engine 236
14.2.2 Rules Creation by Guided Rule Editor 236
14.2.3 Rules Creation By Decision Table 237
14.2.4 Rules Storage 238
14.2.5 Rules Deployment 239
14.2.6 Rules Versioning 239
14.3 Rules Configuration in Modules 239
14.3.1 Generic Rules Configuration 240

14.4 Rules Migration 243
14.4.1 Rules Configured for Modules 243
15 Composite Application Service 247
15.1 Composite Application Service Architecture 247
15.2 Multiple APIs in Single Module 248
16 ID Generation 255
16.1 Database Setup 256
16.1.1 Database Configuration 257
16.2 Automated ID Generation 257
16.3 Custom ID Generation 260
17 Extensibility of Domain Objects using Flex Fields 263
17.1 Flex Field - Provisioning 263
17.1.1 How to know Maximum Flex Fields Provisioned for Entity? 263
17.1.2 Increase Maximum Flex Fields Provisioned for Entity (Optional Step) 264
17.2 Flex Field - Utilization 264
17.2.1 Maintain Flex Field Metadata using Seed Data Configuration (Fast Path: OPA006) Page 264
17.3 Runtime Storage and Retrieval of Flex Field Attribute values 268
17.4 Flex Field - Fact support 269
17.5 Flex Field – Validation Support 271
17.6 Flex Field – Usage Instructions 280
18 Extensibility of Domain Objects - Dictionary Pattern 283
18.1 Customized Domain Object Attribute Placeholders 284
18.2 Customized Domain Object DTO Interceptor in UI Layer 285
18.2.1 Interceptor Hook to Persist Customized Domain Object Attributes 285
18.2.2 Interceptor Hook to Fetch Customized Domain Object Attributes 286
18.3 Dictionary Data Transfer from UI to Host 287

18.3.1 Customized Domain Object DTO Transfer from UI to Host 287
18.3.2 Customized Domain Object DTO transfer from Host to UI 291
18.4 Translating Dictionary Data into Custom Domain Object 295
18.4.1 Instantiation and Persistence of Custom Domain Objects 295
18.4.2 Fetching of Customized Domain Objects 296
18.4.3 Defining of Customized Domain Objects 297
18.5 Customized Domain Object ORM Configuration 298
18.5.1 Case 1 - Non-Inheritance based mapping 298
18.5.2 Case 2 - Mapped as ORM Subclass 301
18.5.3 Case 3 - Mapped as ORM Union-Subclass or Joined-Subclass 302
18.5.4 Case 4 - Mapped as ORM Component 305
18.6 Extensibility using Dictionary in Origination Application 305
18.6.1 ICustomDataHandler's as DictionaryArray Interceptor 305
18.6.2 Create Customized Abstract Domain Object Class 306
18.6.3 Create Customized Abstract Domain Object ORM Mapping File 307
18.6.4 Create Customized Abstract Domain Object Attribute Columns 307
18.7 Extensibility using Attributes of Various Supported Datatypes 308
18.8 Customized Domain Object having Collection of Objects as Attributes 313
18.9 Limitation to Extensibility using Dictionary Pattern 316
19 OCH Integration 319
19.1 Integration Adapter Interface 319
19.2 Abstract Integration Adapter Class 320
19.3 Sample Integration Adapter 321
19.4 Integration Abstract Assembler 322
19.5 Sample Assembler 323
20 Algorithm Extensions 325

20.1 Overview 325
20.2 Algorithm Spots 325
20.3 Algorithm Components 327
20.4 List of Algorithm Spots 337

List of Figures

Figure 3–1 Extending Service Execution 33
Figure 3–2 OBP Application Adapters 34
Figure 3–3 Extending Business Policy 35
Figure 3–4 Batch Framework Extension 36
Figure 3–5 Upload File Processing 37
Figure 3–6 Alerts Extension 38
Figure 3–7 Creating New Reports 39
Figure 3–8 Security Customization 40
Figure 3–9 Loan Schedule Computation Algorithm 41
Figure 3–10 Facts and Business Rules 42
Figure 3–11 Composite Application Service 42
Figure 3–12 ID Generation 43
Figure 3–13 OCH Integration 44
Figure 4–1 Standard Set of Framework Method Calls 46
Figure 4–2 Extension Hook for Document Type Application Service 47
Figure 4–3 Default Application Service Extension 48
Figure 4–4 Application Service Extension Executor 49
Figure 4–5 Extension Factory Hook for Document Type Application Service 49
Figure 4–6 Factory Implementation of Extension Hook for Document Type Application Service 50
Figure 4–7 Application Service Extension Using Groovy 51
Figure 4–8 PROP_ID and CATEGORY_ID 52
Figure 4–9 SUMMARY_TEXT 52
Figure 4–10 Add Groovy Library to Classpath 52
Figure 4–11 Extended Application Service Extension 53

Figure 4–12 Extended Application Service Extension - Post and Pre Hook 54
Figure 4–13 Extension Hook for Document Type Application Service Spi Ext 55
Figure 4–14 Default Implementation of Appx Extension 56
Figure 4–15 Extended Application Service Extension Executor 57
Figure 4–16 Extension Factory Hook for Document Type Application Service Spi Ext 58
Figure 4–17 Factory Implementation of Extension Hook for Document Type Application Service Spi Ext 59
Figure 4–18 Application Service Appx Extension using Groovy 60
Figure 4–19 PROP_ID and CATEGORY_ID 60
Figure 4–20 SUMMARY_TEXT 60
Figure 4–21 Add Groovy Library to Classpath 60
Figure 4–22 Maintenance of Document Types 61
Figure 4–23 Document Type Application Service Spi Ext - Appx Layer 62
Figure 4–24 Doc Type Application Service Spi Ext - Appx Layer 63
Figure 4–25 Document Type Application Service Spi Ext - App Layer 64
Figure 4–26 Doc Type Application Service Spi Ext - App Layer 65
Figure 4–27 Pre and Post Middleware Specific Transaction Tasks Overview 66
Figure 4–28 FLX_FW_MW_TASKS 67
Figure 4–29 FLX_FW_MW_TASKS_DTO_DEFN 67
Figure 4–30 FLX_FW_MW_TASKS_DTO_MAP 68
Figure 4–31 FLX_MD_SERVICE_ATTR 68
Figure 4–32 FLX_MD_GEN_ATTR_LEGACY_B 69
Figure 4–33 Custom Application Service 69
Figure 6–1 Package Diagram 76
Figure 6–2 Adapter Mechanism Class Diagram 77
Figure 6–3 Adapter Mechanism Sequence Diagram 78

Figure 6–4 Adapter Implementation Using Groovy 81
Figure 6–5 Credit Card Adapter Implementation Using Groovy 82
Figure 6–6 Modify AdapterFactories.properties for GroovyCred-itCardAdapterFactory 82
Figure 6–7 Modify Preferences.xml for GroovyCreditCardAdapterFactory 82
Figure 6–8 Add Groovy Library to Classpath 83
Figure 6–9 Party KYC Status Check Adapter Interface 85
Figure 6–10 Default Implementation of I Party KYC Check Adapter Interface 85
Figure 6–11 KYC Adapter Factory with Mocking Support 86
Figure 7–1 Business Policy Extension 89
Figure 7–2 validate() method in AbstractBusinessPolicy.java 90
Figure 7–3 validatePolicy() in creditCardBusinessPolicy.java 90
Figure 7–4 Add a preference for custom business policy in preferences.xml 91
Figure 7–5 Extensions using Groovy 92
Figure 8–1 Business Day in OBP 94
Figure 8–2 Batch Framework Architecture - Static View 97
Figure 8–3 Dynamic View Sequence Diagram 98
Figure 8–4 State Diagram of a Shell 99
Figure 8–5 Creation of New Category 106
Figure 8–6 Population of Other Parameters 113
Figure 8–7 Population of Other Parameters - General Tab 113
Figure 8–8 Population of Other Parameters - Connection Pool 114
Figure 8–9 Population of Other Parameters - Set IS_DB_RAC 114
Figure 8–10 Population of Other Parameters - Specify Data 115
Figure 8–11 Batch Execution 115
Figure 9–1 Uploaded Data File Processing Framework 118
Figure 9–2 File Handlers 123

Figure 9–3 Record Handlers for Both Header and Details 124
Figure 9–4 DTO and Keys Classes for Both Header and Details - HeaderRecDTOKey 125
Figure 9–5 DTO and Keys Classes for Both Header and Details - AbstractDTORec 126
Figure 9–6 XXF File Definition XML 128
Figure 9–7 API Calls in Adapters 130
Figure 9–8 Processing Adapter 131
Figure 10–1 Sample script for Activity Record 134
Figure 10–2 Sample script for Event Record 135
Figure 10–3 Activity Event Mapping Record 135
Figure 10–4 Activity Log DTO 136
Figure 10–5 Metadata Generation 137
Figure 10–6 Service Data Attribute Generation 138
Figure 10–7 Alert Message Template Maintenance 140
Figure 10–8 Alert Maintenance 141
Figure 10–9 Alert Subscription 142
Figure 10–10 Transaction API Changes - Service Call 142
Figure 10–11 Transaction API Changes - Conditional Evaluation 143
Figure 10–12 Transaction API Changes - persistActivityLog(..) 143
Figure 10–13 Transaction API Changes - Activity Log 143
Figure 10–14 Transaction API Changes - Register Activity 144
Figure 10–15 Alert Processing Steps 145
Figure 10–16 Event Processing Status Type 146
Figure 10–17 Batch Alerts 147
Figure 10–18 Alert Dispatch Mechanism 148
Figure 10–19 Alert Dispatch Mechanism - Dispatcher Factory 149

Figure 10–20 Alert Dispatch Mechanism - Destination 150
Figure 10–21 Alert.Party.FirstName 154
Figure 10–22 Facts in Alerts Framework 154
Figure 10–23 Alert.Party.PartyId 154
Figure 10–24 Alert.Party.Prefix and Alert.Party.LastName 155
Figure 10–25 Message Template (Fast Path: AL03) 155
Figure 10–26 Placeholder for Derived Facts 156
Figure 10–27 Alert Maintenance (Fast Path: AL04) 156
Figure 10–28 Alert Maintenance - Map the New Message Template Placeholders 157
Figure 10–29 Alert Maintenance - Facts List 157
Figure 10–30 Alert Maintenance - Mapping Completed 158
Figure 10–31 Alert Mail on Mobile Number Update in Contact Point screen 158
Figure 11–1 Creating New Reports 159
Figure 11–2 Global Temporary Table 160
Figure 11–3 Report Record Type 160
Figure 11–4 Report Table Type 161
Figure 11–5 Report DML Function 161
Figure 11–6 Report DDL Function 162
Figure 11–7 Catalog Folder 162
Figure 11–8 Data Source 163
Figure 11–9 Data Model 164
Figure 11–10 Data Set 164
Figure 11–11 Group Fields 165
Figure 11–12 XML Structure and Labels 165
Figure 11–13 XML Code 166
Figure 11–14 Add Input Parameters 166

Figure 11–15 XML View of Report 167
Figure 11–16 Layout of the Report - Create Layout 168
Figure 11–17 Layout of the Report - Batch Job Results 168
Figure 11–18 View Report in BIP 169
Figure 11–19 Batch Report Generation for a Branch Group Code 171
Figure 11–20 Batch Report Generation Path 172
Figure 11–21 Adhoc Report Generation - Report Request 174
Figure 11–22 Adhoc Report Generation - Report Generated 175
Figure 11–23 Advice Report 176
Figure 11–24 View Generated Adhoc Report 177
Figure 12–1 Security Customizations Interface 180
Figure 12–2 Security Use Case with Access Checks and Assertions 181
Figure 12–3 Add Attributes Name for Service ID 182
Figure 12–4 Add Service Attributes in SM500 183
Figure 12–5 Constraint Attribute Config with Associated Adapter 183
Figure 12–6 Select Resource Type and Add Service ID 185
Figure 12–7 Add PolicyTable Details 186
Figure 12–8 Add Condition 187
Figure 12–9 Add or Modify Fraud Rules in OAAM - Data Tab 188
Figure 12–10 Add or Modify Fraud Rules in OAAM - Conditions Tab 189
Figure 12–11 Log in to SOA Composer Application screen 193
Figure 12–12 Search Rules file 194
Figure 12–13 Search Rules file -View 195
Figure 12–14 Composite to Configure Routing Rules 196
Figure 12–15 Stages of Approval 197
Figure 12–16 Add pattern-Submission Payload Type 198

Figure 12–17 Add New Test 199
Figure 12–18 Select Fact with Appropriate Value 200
Figure 12–19 Create Participant From Group 201
Figure 12–20 Validate Rules File 202
Figure 12–21 Expand Business Rules 203
Figure 12–22 Create New Rule 204
Figure 12–23 Existing Rule 205
Figure 12–24 Change the approval group 206
Figure 13–1 Add New Algorithm 207
Figure 13–2 Create New Installment 208
Figure 14–1 Select Window Preferences 214
Figure 14–2 Window Preferences - OBP Plugin Development 215
Figure 14–3 Enter the Preferences Fact values 216
Figure 14–4 Fact Properties - aggregateCodeFilePath 217
Figure 14–5 Fact Properties - sourceFilePath 218
Figure 14–6 Start Host Server 219
Figure 14–7 Select Open Perspective value 220
Figure 14–8 Fact Explorer 221
Figure 14–9 Fact Vocabulary 222
Figure 14–10 Domain Category 223
Figure 14–11 Fact Groups 224
Figure 14–12 Facts 225
Figure 14–13 Business Definition Tab 226
Figure 14–14 Value Definition Tab 226
Figure 14–15 Enum Definition Tab 227
Figure 14–16 Aggregrate Definition Tab 228

Figure 14–17 Aggregate File Tab 229
Figure 14–18 Creating New Fact - Add 230
Figure 14–19 Creating New Fact - Fact Business Definition 231
Figure 14–20 Creating New Fact - Domain Group 232
Figure 14–21 Saving New Fact 232
Figure 14–22 Saving New Fact - Fact Added 233
Figure 14–23 Designate Class as Object Fact 234
Figure 14–24 Object Fact in UI 235
Figure 14–25 Generic Rule Configuration 241
Figure 14–26 Rule Author - Decision Table 242
Figure 14–27 Rule Author - Expression Builder 243
Figure 15–1 Composite Application Service Architecture 248
Figure 16–1 Configuration of ID Generation Process 255
Figure 16–2 Automated ID Generation - Single Record View 258
Figure 16–3 Automated ID Generation - Generate Submission ID 259
Figure 16–4 Automated ID Generation - Submission ID Generation Service 259
Figure 16–5 Custom ID Generation - Custom ID Generator 260
Figure 16–6 Custom ID Generation - Custom ID Generation Constants 261
Figure 16–7 Custom ID Generation - Custom Pattern Based Generator 262
Figure 17–1 OPA006 - Error 265
Figure 17–2 OPA006 - Duplicate Attribute 265
Figure 17–3 OPA006 - Enumeration Type 266
Figure 17–4 OPA006 - Consecutive Attributes Names 266
Figure 17–5 OPA006- Invalid Input for Attribute Data Type. 267
Figure 17–6 Attribute Data Type as Date 267
Figure 17–7 Runtime Storage Saved Data 268

Figure 17–8 Runtime Storage Retrieved Data 269
Figure 17–9 Flex Field - Fact support Saved Data 269
Figure 17–10 Flex Field - Fact support Data in Table 270
Figure 17–11 Flex Field - Fact support Seed 270
Figure 17–12 Flex Field - Fact support After Seed 270
Figure 17–13 Flex Field - Validation Support - Metadata 276
Figure 17–14 Flex Field - Validation Support - Mandatory Validation - Input 276
Figure 17–15 Flex Field - Validation Support - Mandatory Validation -Output 277
Figure 17–16 Flex Field - Validation Support - Min/Max Lenght Validation Input 277
Figure 17–17 Flex Field - Validation Support - Min/Max Lenght Validation Output 278
Figure 17–18 Flex Field - Validation Support - Pattern (Regex) Validation Input 278
Figure 17–19 Flex Field - Validation Support - Pattern (Regex) Validation Output 279
Figure 17–20 Flex Field - Validation Support - Enum Validation Input 279
Figure 17–21 Flex Field - Validation Support - Enum Validation Output 279
Figure 17–22 Flex Field - Validation Support - Date Validation Input 280
Figure 17–23 Flex Field - Validation Support - Date Validation Output 280
Figure 18–1 Extensibility of Domain Objects - Framework 284
Figure 18–2 Code Extract 285
Figure 18–3 Interceptor Hook to Persist Customized Domain Object 286
Figure 18–4 Interceptor Hook to Fetch Customized Domain Object 287
Figure 18–5 JSONClient constructs the JSON Object 288
Figure 18–6 SerializeDictionaryArray to include GenericName and Value attributes 289
Figure 18–7 Host Server JSONFacade extracts the attribute of JSON Object 290
Figure 18–8 AbstractJSONFacade's getDictionaryArray method 291
Figure 18–9 Host Server JSONFacade constructs the JSON Object 292

Figure 18–10 AbstractJSONFacade's serializeDictionaryArray to include Generic Name and Value attributes 293
Figure 18–11 UI Server JSONClient extracts the DictionaryArray attribute 294
Figure 18–12 AbstractJSONBindingStub's getDictionaryArray method 295
Figure 18–13 Instantiation of DataTransferObjects 297
Figure 18–14 Adding Discriminator Column Mapping in Existing ORM file 299
Figure 18–15 ORM File Mapping to Customized Domain Object 299
Figure 18–16 Adding New Java File to the Customized Domain Object 300
Figure 18–17 Adding Extra Columns along with the Discriminator Column 300
Figure 18–18 Adding a New ORM File Mapping to Customized Domain Object 301
Figure 18–19 Adding New Java File to Customized Domain Object 302
Figure 18–20 New ORM File Mapping 303
Figure 18–21 Adding New Java File 303
Figure 18–22 Create a New Table CZ_NAB_LM_PROPOSED_FACILITY 304
Figure 18–23 CustomDataHandler's as DictionaryArray Interceptor 306
Figure 18–24 Create Customized Abstract Domain Object Class 307
Figure 18–25 Create Customized Abstract Domain Object ORM Mapping File 307
Figure 18–26 Create Customized Abstract Domain Object Attribute Columns 307
Figure 18–27 Customized Message Template Class 309
Figure 18–28 Domain Object Table 310
Figure 18–29 ORM File 310
Figure 18–30 JUnit Test Case 311
Figure 18–31 JUnit Adds Table Record 311
Figure 18–32 Dictionary Array Values 312
Figure 18–33 Customized Domain Object having collection of Objects as Attributes 313
Figure 18–34 Member Attributes of Customized Domain Object 314

Figure 18–35 Dictionary Array Elements 314
Figure 18–36 Customized Domain Object constructed by AbstractAssembler 315
Figure 18–37 Dictionary Array returned by AbstractAssembler 316
Figure 19–1 Integration Adapter Interface 320
Figure 19–2 Abstract Integration Adapter Class 321
Figure 19–3 Sample Integration Adapter 322
Figure 19–4 Integration Abstract Assembler 323
Figure 19–5 Sample Assembler 324
Figure 20–1 Algorithm Spot Interface 325
Figure 20–2 Example: Algorithm Spot Interface 326
Figure 20–3 Example: Adding New Algorithm Spot 327
Figure 20–4 Example: Data Validation Algorithm Component 328
Figure 20–5 Example: CharAdhocDateValidation 329
Figure 20–6 New Algorithm Implementation 330
Figure 20–7 Adhoc Characteristic Date Validation 330
Figure 20–8 Algorithm Spot Interface Methods 331
Figure 20–9 Algorithm Spot Interface Methods (continued) 332
Figure 20–10 Generated Artifacts 333
Figure 20–11 Generated Artifacts 334
Figure 20–12 Create Algorithm Type 335
Figure 20–13 Attach Algorithm 336

List of Tables

Table 6–1 Components of Adapter Implementation 75
Table 8–1 Database Server Components 103
Table 8–2 FLX_BATCH_JOB_CATEGORY_MASTER 104
Table 8–3 FLX_BATCH_JOB_GRP_CATEGORY 104
Table 8–4 FLX_BATCH_JOB_CATEGORY_DEPEND 105
Table 8–5 FLX_BATCH_JOB_SHELL_MASTER 107
Table 8–6 FLX_BATCH_JOB_SHELL_DTLS 108
Table 8–7 FLX_BATCH_JOB_SHELL_DEPEND 109
Table 8–8 Driver Table 109
Table 8–9 Actions Table 110
Table 9–1 FLX_EXT_FILE_UPLOAD_MAST 119
Table 9–2 Mandatory Fields in Record Tables 120
Table 9–3 FLX_EXT_FILE_PARAMS 120
Table 9–4 FLX_BATCH_JOB_SHELL_DTLS 121
Table 9–5 XXF File Definition XML 127
Table 9–6 Process Status 132
Table 10–1 FLX_EP_ACT_B 133
Table 10–2 FLX_EP_EVT_B 134
Table 10–3 FLX_EP_ACT_EVT_B 135
Table 10–4 Key Fields in FLX_MD_SERVICE_ATTR 138
Table 14–1 Example of a Decision Table 237
Table 14–2 Actions 238
Table 14–3 Conditions 238
Table 14–4 Rules Versioning 239

Table 14–5 Details of Configured Rules in Modules 243
Table 15–1 Java Classes 248
Table 16–1 FLX_CS_ID_CONFIG_B 256
Table 16–2 FLX_CS_ID_RANGE 256
Table 16–3 FLX_CS_ID_USF 257
Table 17–1 metadata configuration details for validations 271

Preface

This guide explains customization and extension of Oracle Banking Enterprise Product Manufacturing.

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

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

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/us/corporate/accessibility/index.html.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/us/corporate/accessibility/support/index.html#info or visit http://www.oracle.com/us/corporate/accessibility/support/index.html#trs if you are hearing impaired.

Related Documents

For more information, see the following documentation:

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

Conventions

The following text conventions are used in this document:

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

1 About This Guide

This guide is applicable for the following products:

- Oracle Banking Platform
- Oracle Banking Enterprise Product Manufacturing
- Oracle Banking Enterprise Originations

References to Oracle Banking Platform or OBP in this guide apply to all the above mentioned products. The chapters and sections that are not applicable for any of the products are listed in this chapter.

1.1 Sections Not Applicable for Oracle Banking Enterprise Product Manufacturing

The following chapters and sections in this guide are not applicable for Oracle Banking Enterprise Product Manufacturing.

- Section 3.1.7 Alert Extension
- Section 3.1.10 Loan Schedule Computation Algorithm
- Section 3.1.14 OCH Integration
- Section 6.2.2 Example 2 – DispatchAdapter
- Section 10.3 Alert Processing Steps
- Chapter 13 Loan Schedule Computation Algorithm
- Chapter 11 Creating New Reports Using BI Publisher
- Chapter 19 OCH Integration
- Chapter 20 Algorithm Extensions

1.2 Sections Not Applicable for Oracle Banking Enterprise Originations

The following chapters and sections in this guide are not applicable for Oracle Banking Enterprise Originations.

- Section 3.1.10 Loan Schedule Computation Algorithm
- Chapter 13 Loan Schedule Computation Algorithm
- Chapter 20 Algorithm Extensions

2 Objective and Scope

This chapter defines the objective and scope of this document.

2.1 Overview

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

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

2.2 Objective and Scope

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

2.2.1 Extensibility Objective

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

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

2.3 Complementary Artefacts

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

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

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

2.4 Out of Scope

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

3 Overview of Use Cases

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

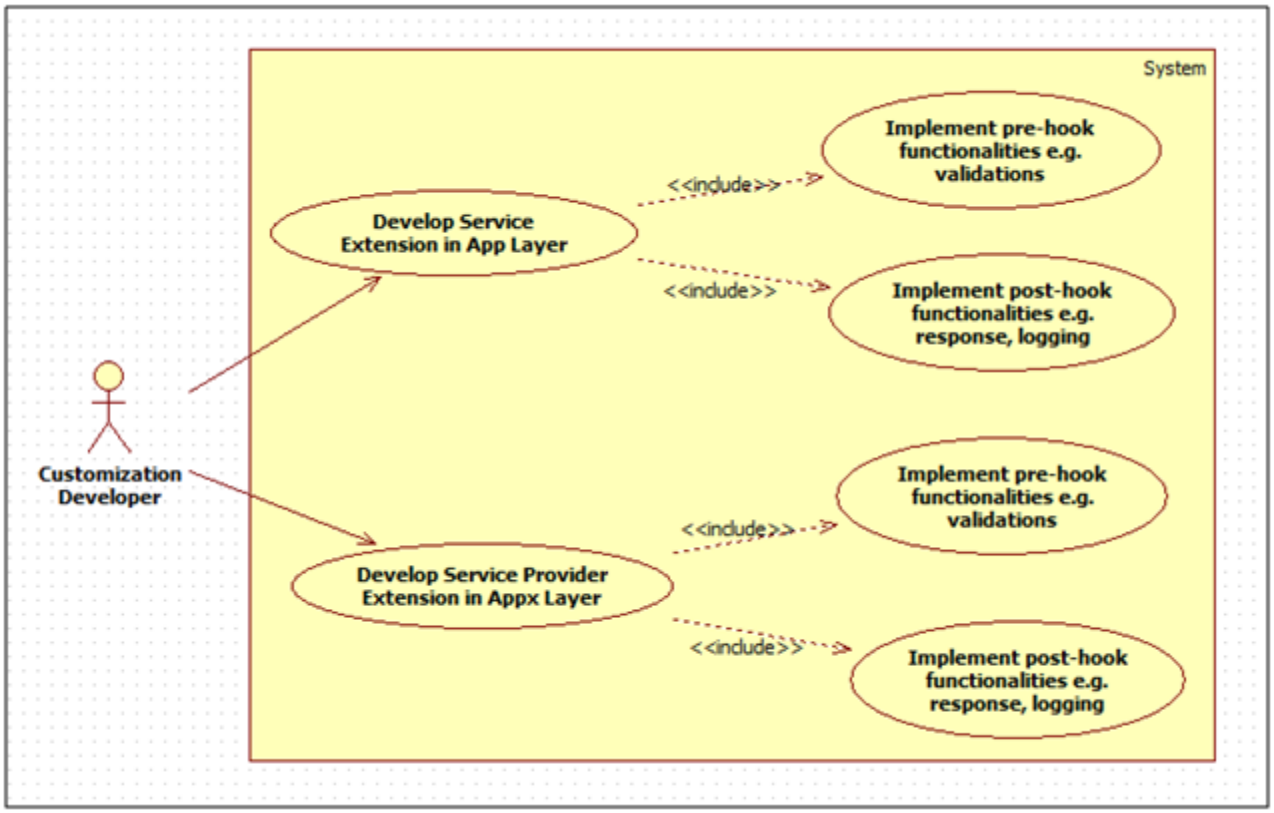
3.1 Extensibility Use Cases

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

3.1.1 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.

Figure 3–1 Extending Service Execution



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.

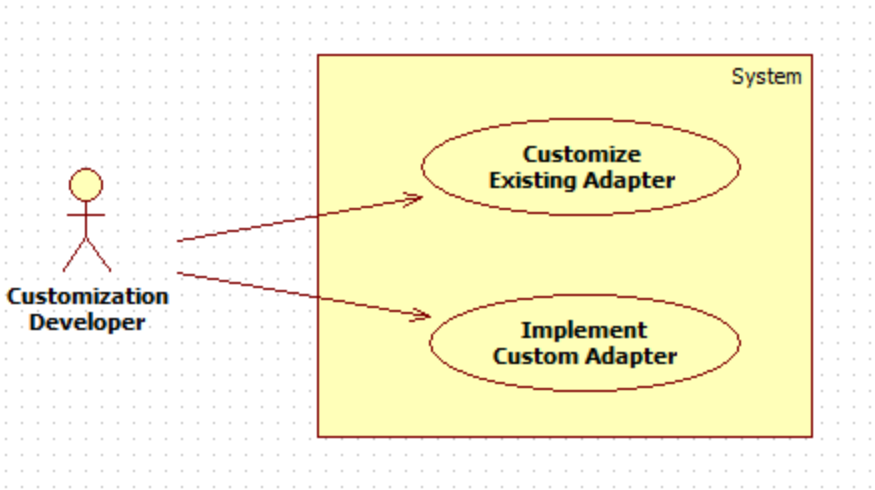
Service Provider Extension

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.

3.1.2 OBP Application Adapters

In OBP, adapters are used for helping two different modules or systems to communicate with each other. It helps the consuming side adapt 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 3–2 OBP Application Adapters



3.1.3 Extending Business Policy

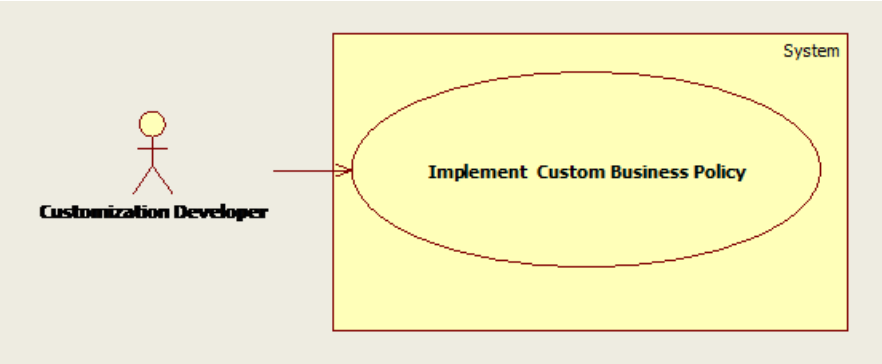
In OBP, business policies are used for common business validations. For instance, credit card number validation to check whether or not the credit card number entered by user complies with the specified format or exists in the record. Business policy implementation strategy is based on factory design pattern and implements a common business policy factory class for each module. All the business policy factory classes

extend to AbstractBusinessPolicyFactory Class. AbstractBusinessPolicyFactory Class returns the BusinessPolicy class instance which extends to AbstractBusinessPolicy class. Application service invokes the validate() method in AbstractBusinessPolicy class which in turn invokes validatePolicy() method in the BusinessPolicy class.

Custom BusinessPolicies are implemented in OBP by configuring preferences in the preferences.xml file. In this file a preference for customBusinessPolicy is defined which represents a query to the database. For customization, create an entry in the Flx_or_config_all_b table with preference name and businessPolicy code.

When application service invokes the createPolicyInstance() method of the BusinessPolicyFactory class, this class invokes a getPolicyInstance() method of the AbstractBusinessPolicy class which looks for any custom businessPolicy class in the database and returns the custom class if it gets one. Otherwise it returns null, and a new instance of base BusinessPolicy class is created and returned to the invoking application service.

Figure 3–3 Extending Business Policy



3.1.4 User Defined Fields

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.

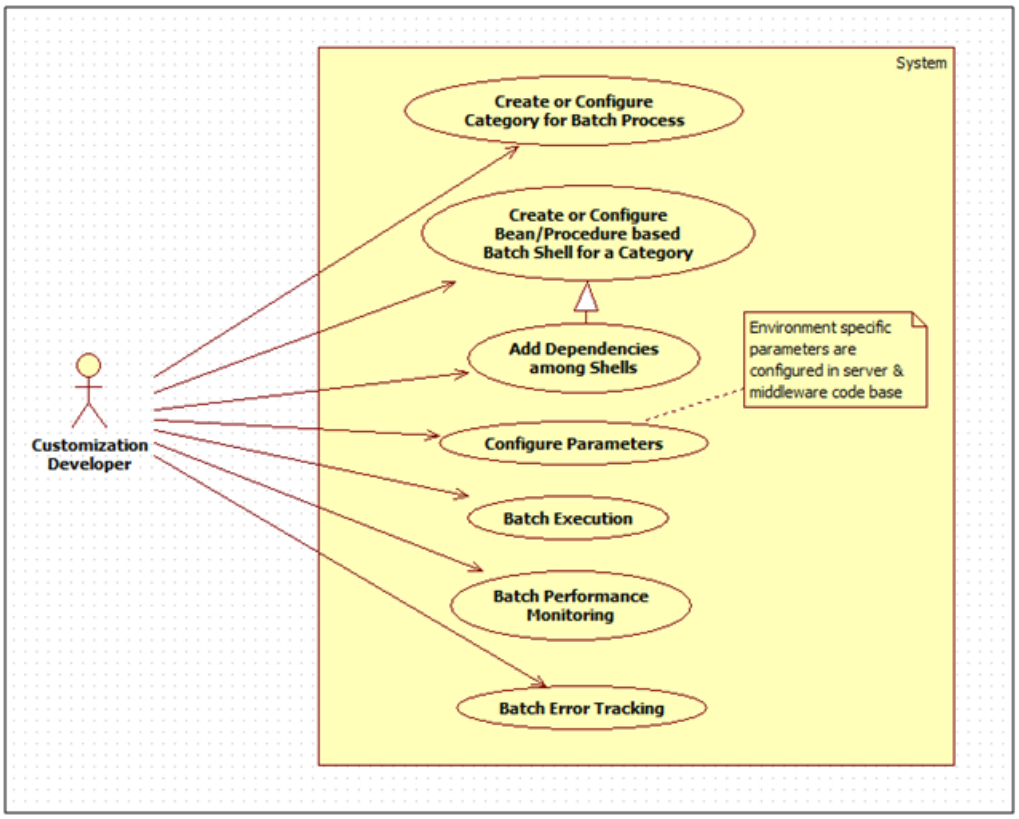
3.1.5 Batch Framework Extension

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.

Figure 3–4 Batch Framework Extension

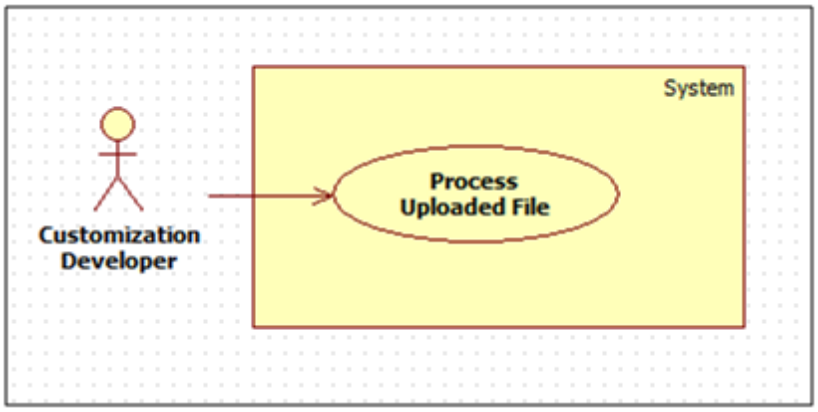


3.1.6 Uploaded File Processing

File processing is an independent process and is done separately after file upload. Every upload provides a unique file ID 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 3–5 Upload File Processing

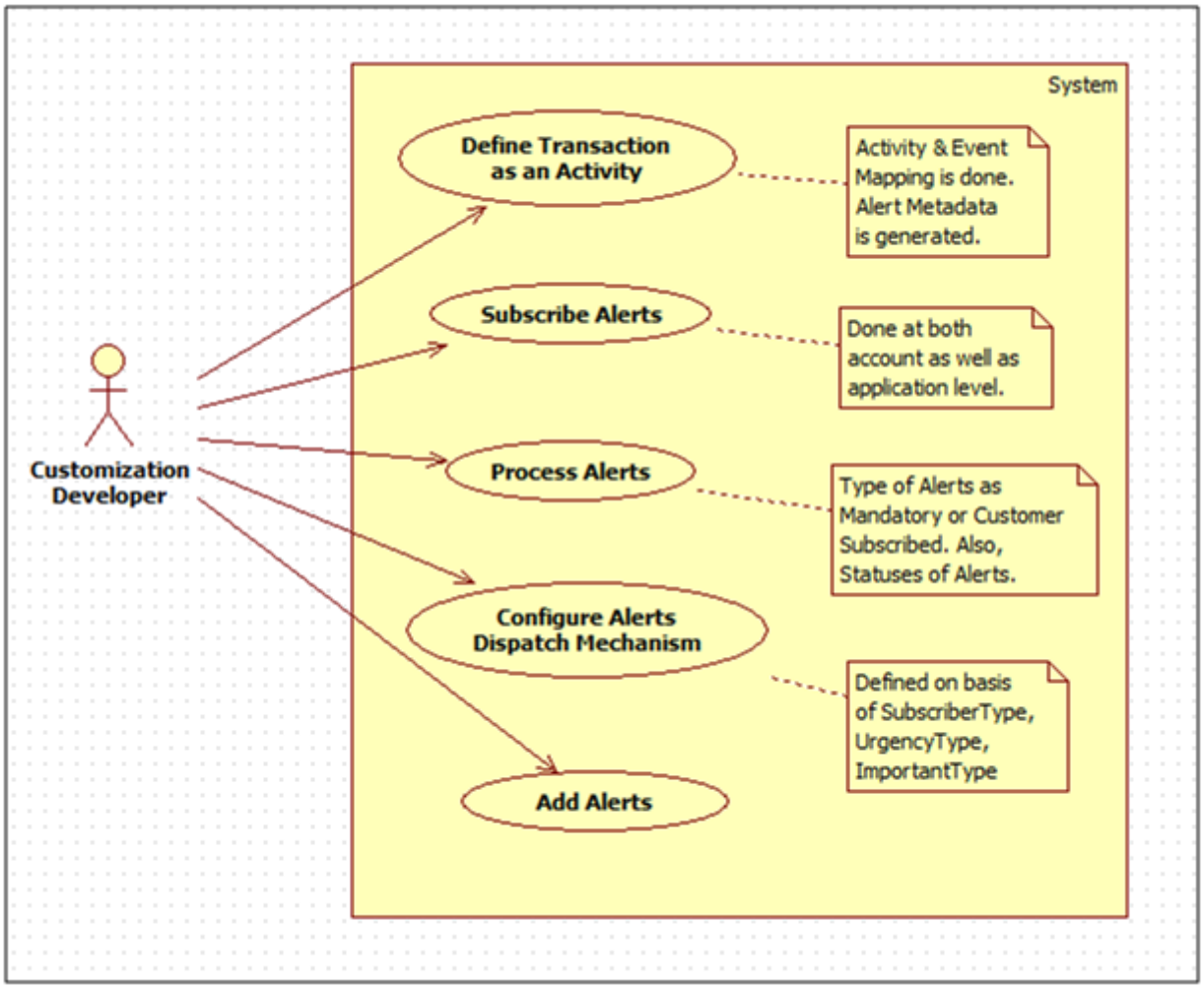


3.1.7 Alert Extension

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.

Figure 3–6 Alerts Extension

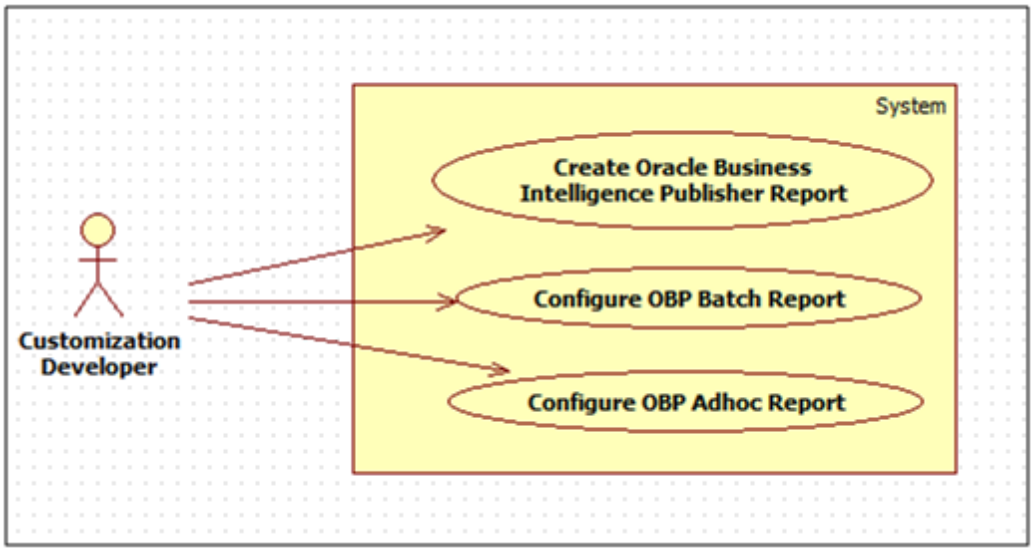


3.1.8 Create New Reports Using BI Publisher

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.

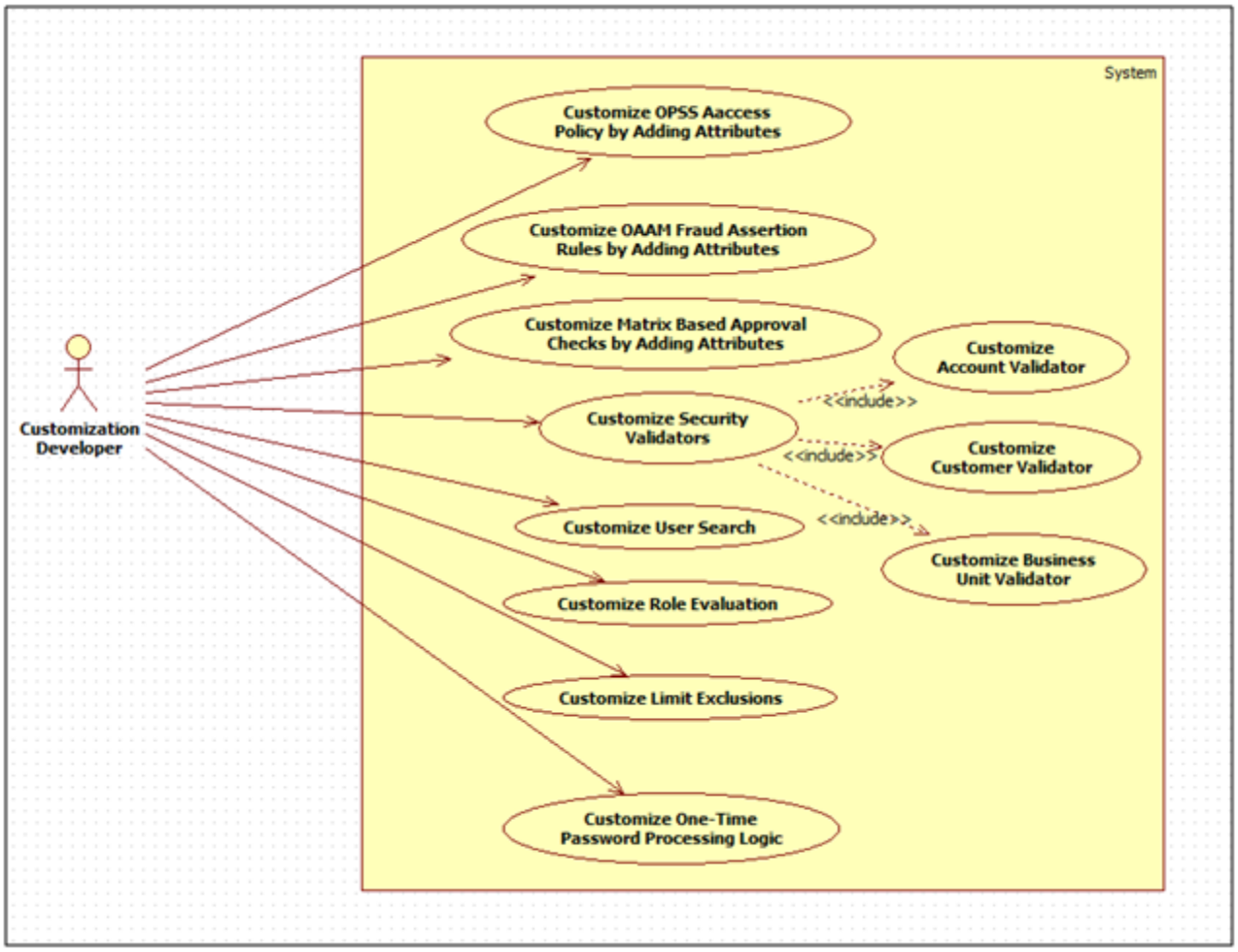
Figure 3–7 Creating New Reports



3.1.9 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.

Figure 3–8 Security Customization



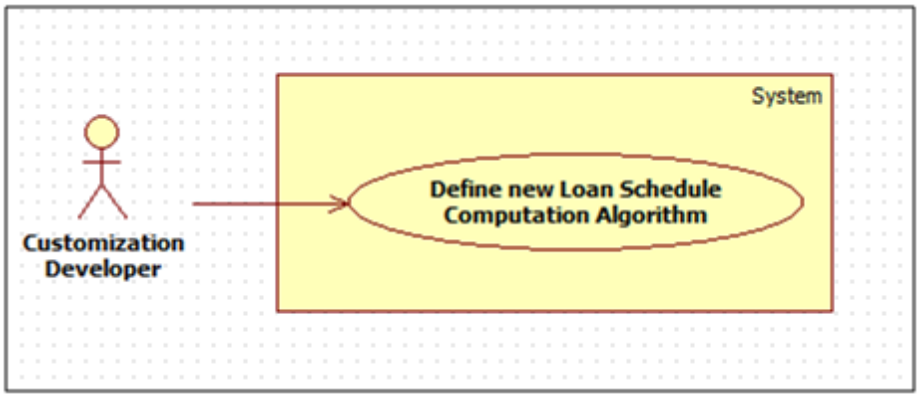
OBP application therefore provides functionality where there is a provision for customizing the security attributes or functions. For example:

- 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

3.1.10 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.

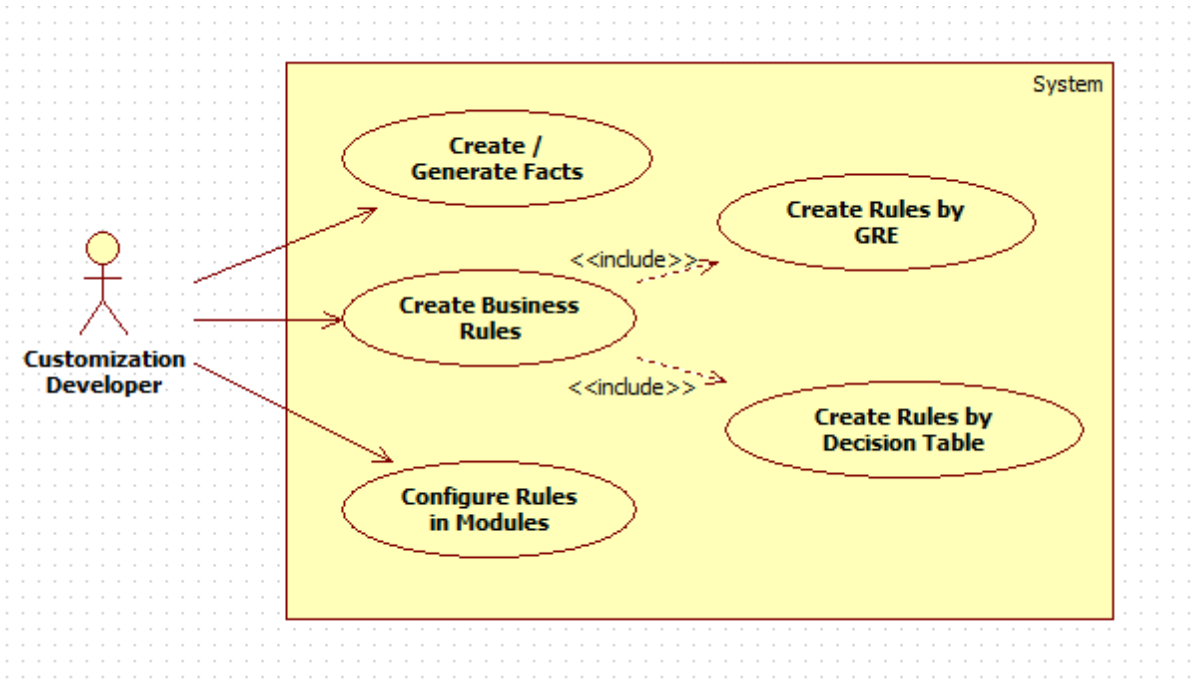
Figure 3–9 Loan Schedule Computation Algorithm



3.1.11 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 provide agility, it also provides the bonus of cost reduced development.

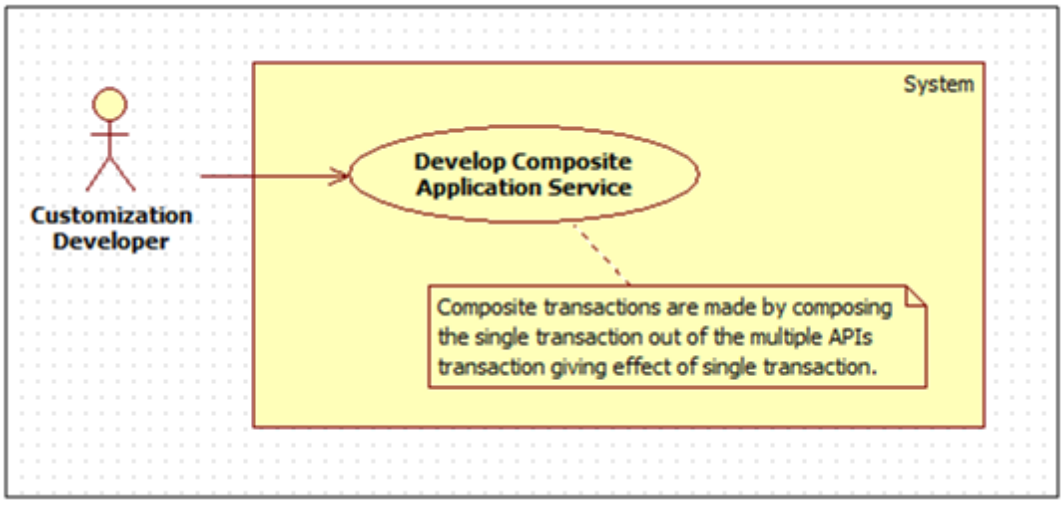
Figure 3–10 Facts and Business Rules



3.1.12 Composite Application Service

OBP provides the extensibility feature by which user 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.

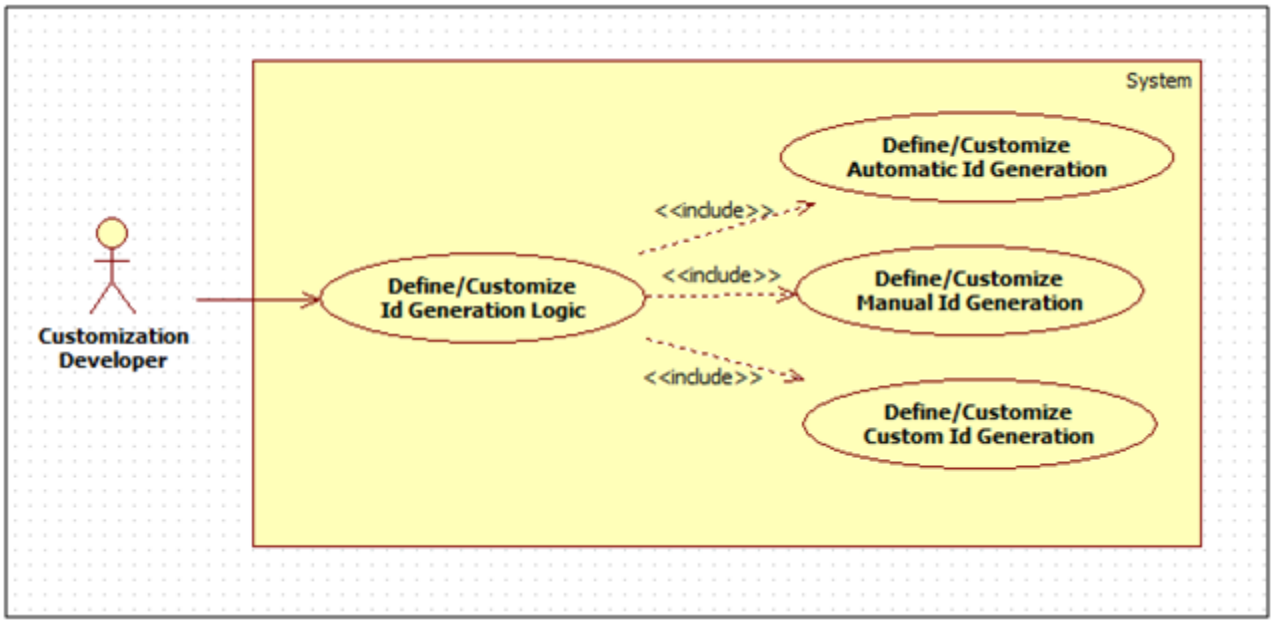
Figure 3–11 Composite Application Service



3.1.13 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 user as per their requirements and IDs can be generated accordingly.

Figure 3–12 ID Generation

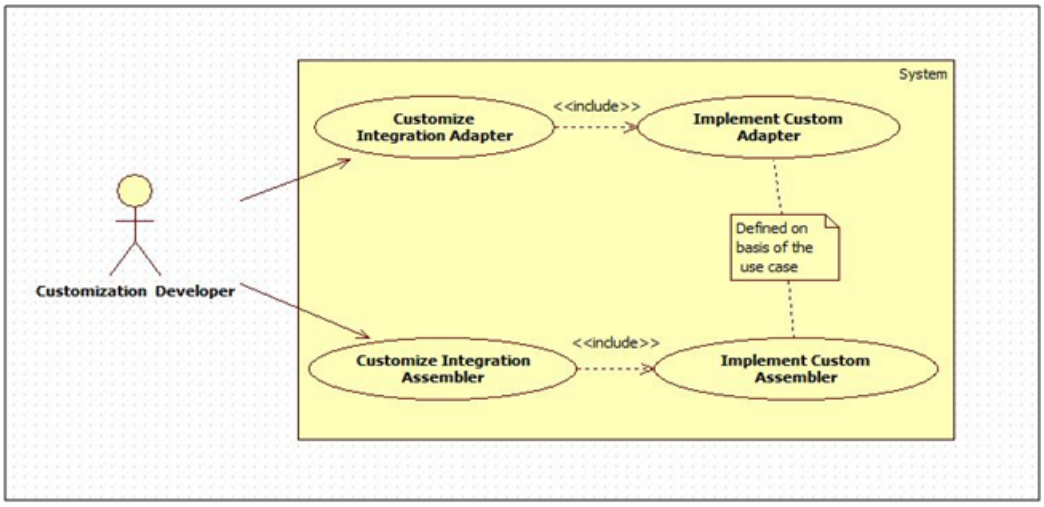


3.1.14 OCH Integration

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 3–13 OCH Integration



4 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. A few examples in which the application service extensions in the form of pre and post hook could be required are mentioned below.

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 layer – referred to as the “app” layer extension.
- Extended Application Service – referred to as the “appx” layer extension.

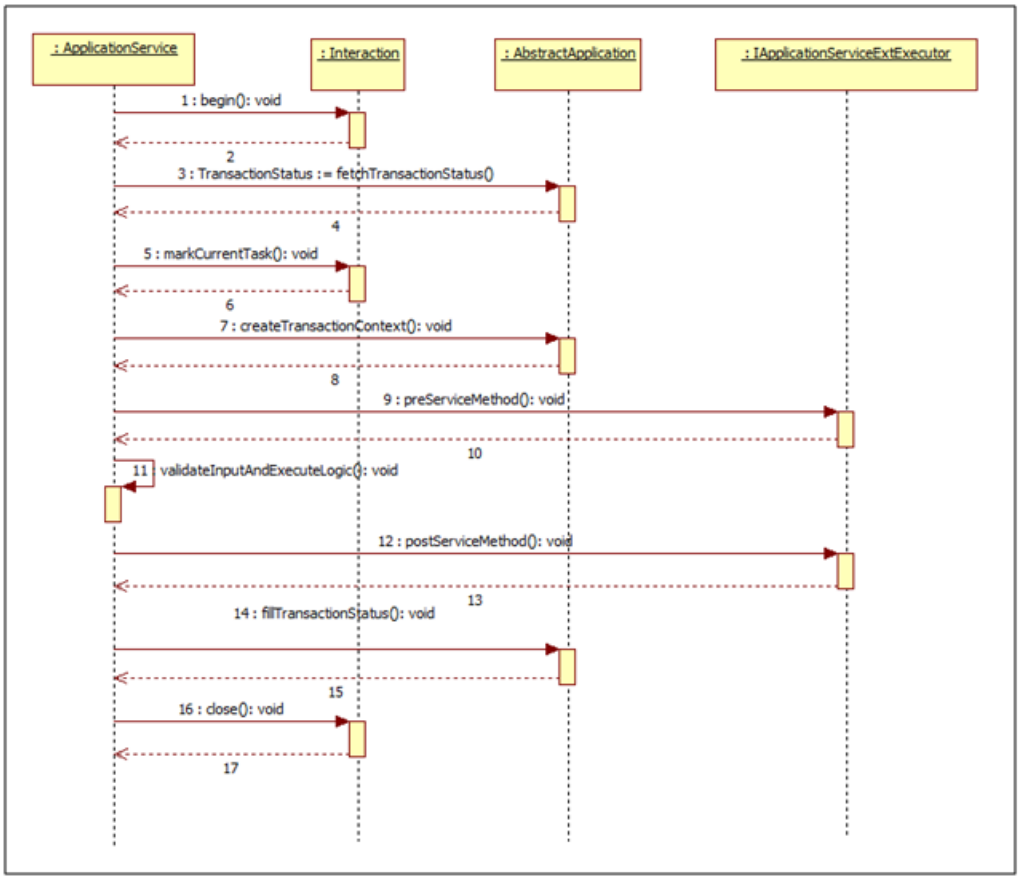
There are few differences in the extensions of the app and appx layer:

- 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.

4.1 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 4–1 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.

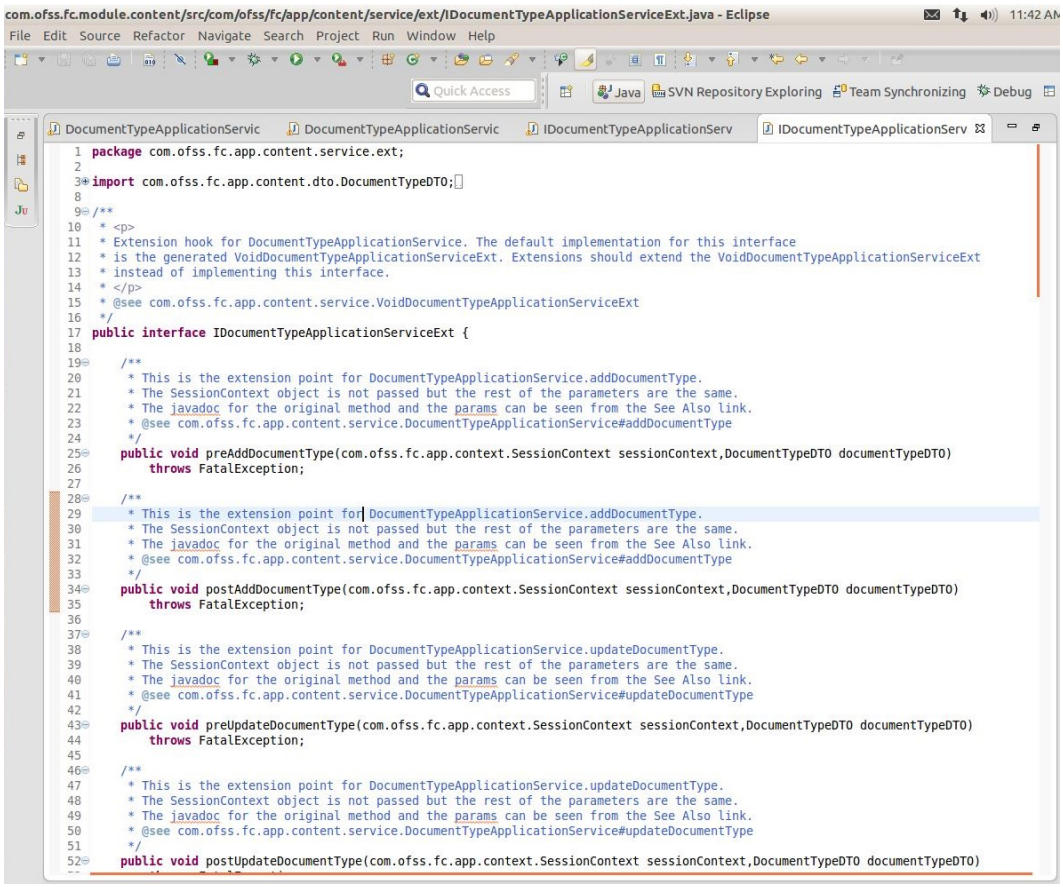
4.1.1 Application Service Extension Interface

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 4–2 Extension Hook for Document Type Application Service

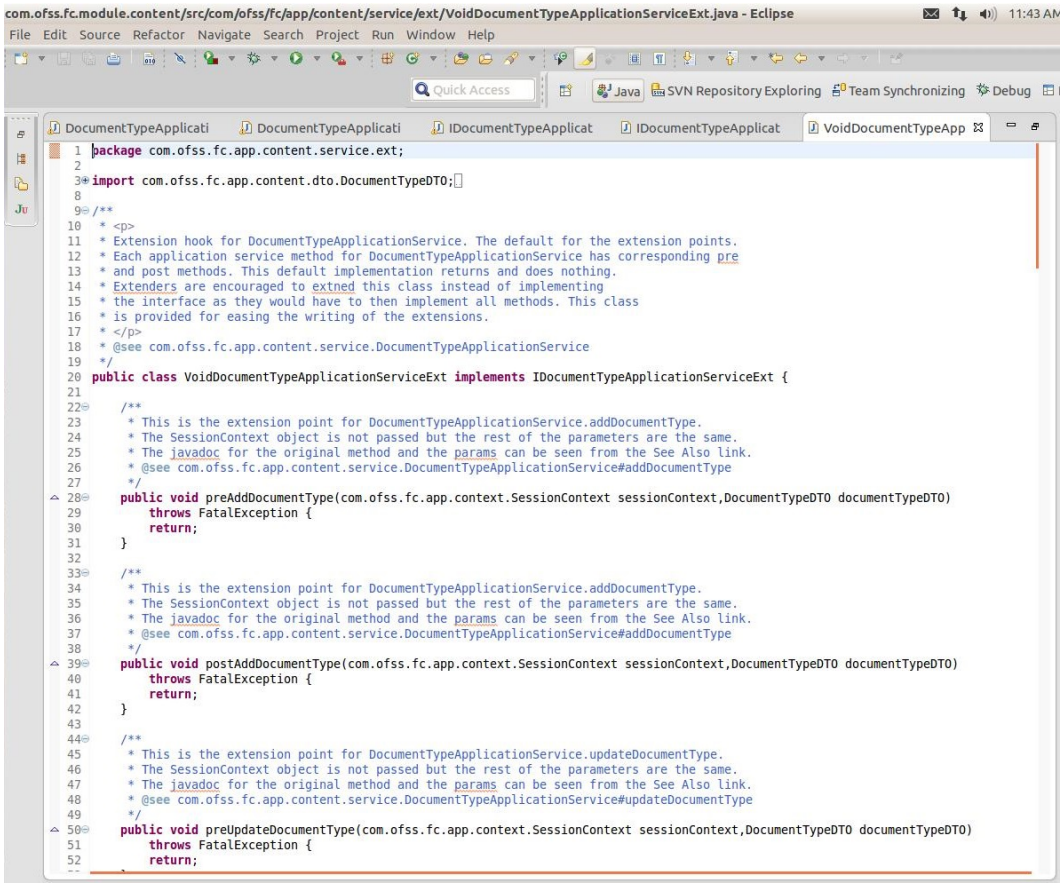


4.1.2 Default Application Service Extension

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 4–3 Default Application Service Extension



4.1.3 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 4–4 Application Service Extension Executor

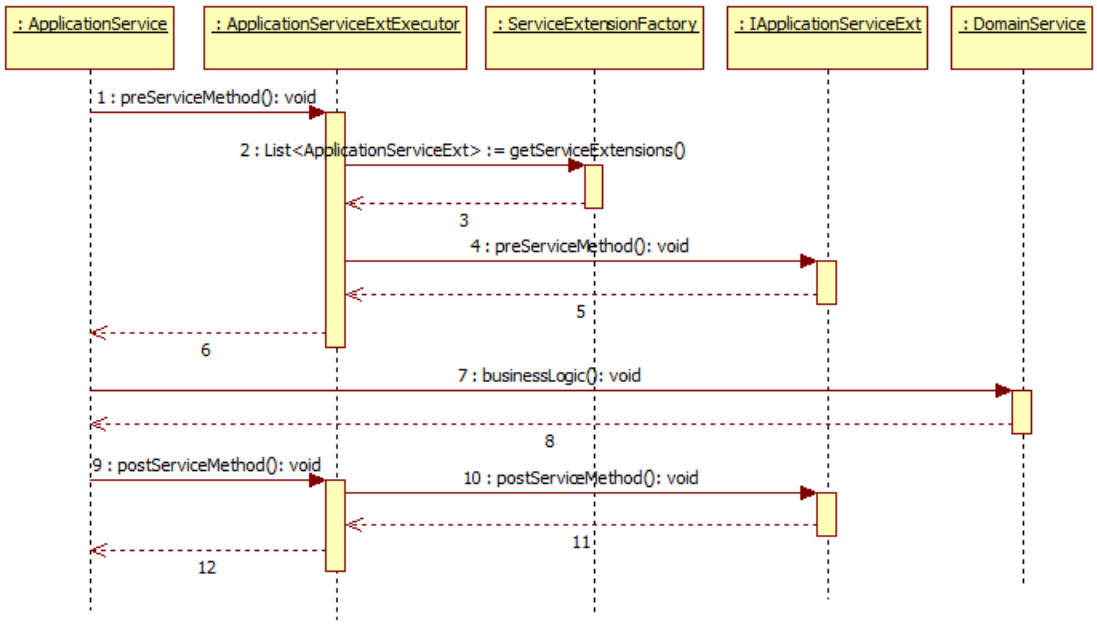


Figure 4–5 Extension Factory Hook for Document Type Application Service

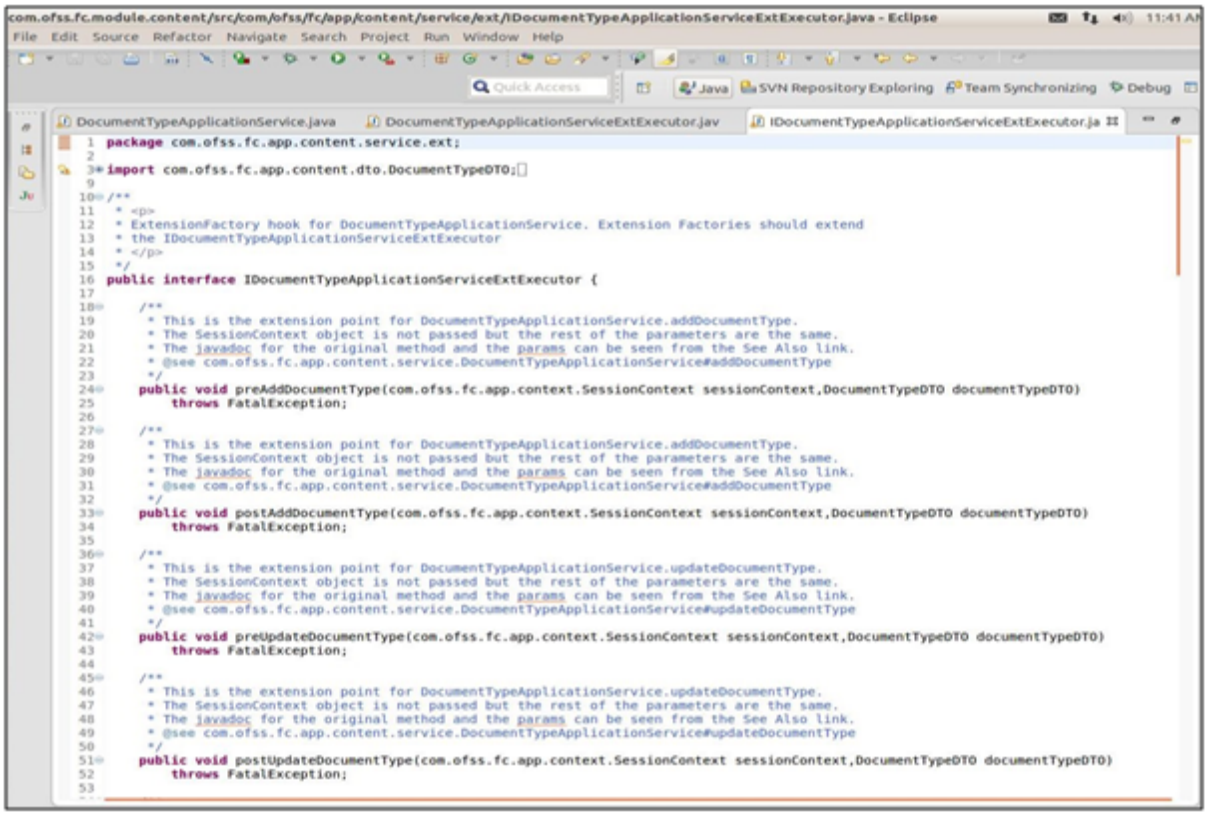
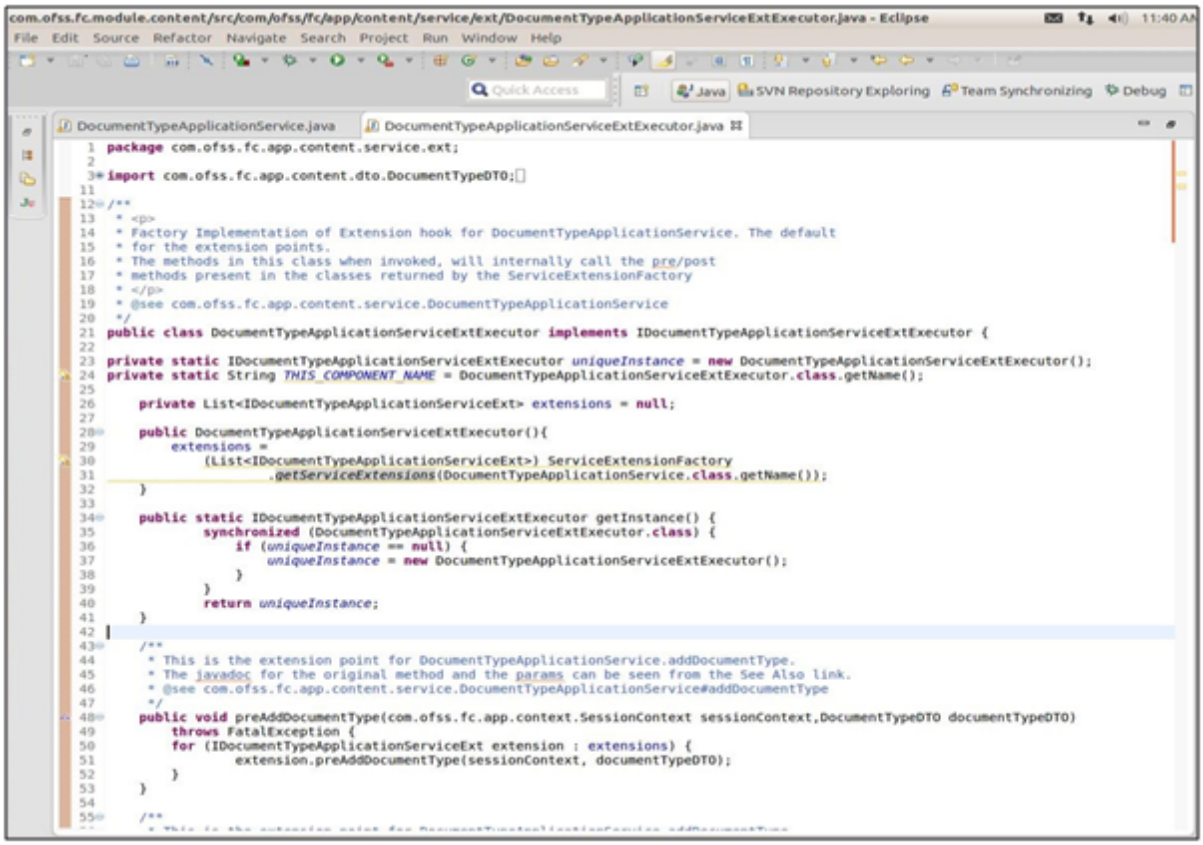


Figure 4–6 Factory Implementation of Extension Hook for Document Type Application Service



4.1.4 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.DocumentTypeApplicationService
Ext

Multiple extension configuration

com.ofss.fc.app.content.service.DocumentTypeApplicationService=
```

```
com.ofss.fc.app.content.service.ext.in.DocumentTypeApplicationServ
iceExtension,
com.ofss.fc.app.content.service.ext.in.mum.DocumentTypeApplication
ServiceExtension,
com.ofss.fc.app.content.service.ext.in.mum.ExtendedDocumentTypeApp
licationService,
com.ofss.fc.app.content.service.ext.in.blr.DocumentTypeApplication
ServiceExtension,
com.ofss.fc.app.content.service.ext.in.blr.ExtendedDocumentTypeApp
licationService
```

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.

4.1.5 Application Service Extension Using Groovy

Application service extension can be implemented using Groovy. The sample code and steps for service extension implementation using groovy is as follows:

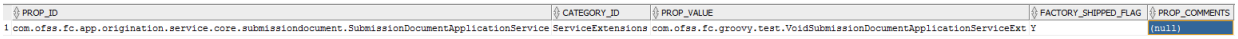
Service extension groovy implementation class 'VoidSubmissionDocumentApplicationServiceExt' implementing product service extension interface 'com.ofss.fc.app.origination.service.core.submissiondocument.ext.ISubmissionDocumentApplicationServiceExt.

Figure 4–7 Application Service Extension Using Groovy

```
4
5  package com.ofss.fc.module.originationGroovy
6
7  import com.ofss.fc.app.context.SessionContext
8  import com.ofss.fc.app.origination.dto.core.document.DocumentReferenceInputDTO
9  import com.ofss.fc.app.origination.service.core.submissiondocument.ext.ISubmissionDocumentApplicationServiceExt
10 import com.ofss.fc.common.OriginationConstants
11 import com.ofss.fc.datatype.Date
12 import com.ofss.fc.enumeration.origination.OfferDocReferenceType
13 import com.ofss.fc.infra.exception.FatalException
14 /**
15  * <p>
16  * Groovy Extension hook for SubmissionDocumentApplicationService. The customization for the extension points.
17  * Each application service method for SubmissionDocumentApplicationService has corresponding pre
18  * and post methods. Whenever the Service Extensions are overriden with Groovy Extensions, the call will
19  * go to corresponding pre/post groovy extensions method and will execute the implemented logic.
20  * </p>
21  * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService
22  */
23 public class VoidSubmissionDocumentApplicationServiceExt implements ISubmissionDocumentApplicationServiceExt {
24
25     /**
26      * This is the extension point for SubmissionDocumentApplicationService.createDocumentChecklist.
27      * The SessionContext object is not passed but the rest of the parameters are the same.
28      * The javadoc for the original method and the params can be seen from the See Also link.
29      * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService#createDocumentChecklist
30      */
31     public void preCreateDocumentChecklist(SessionContext sessionContext, DocumentReferenceInputDTO documentReferenceInputDTO) throws FatalException {
32     }
33     /**
34      * This is the extension point for SubmissionDocumentApplicationService.createDocumentChecklist.
35      * The SessionContext object is not passed but the rest of the parameters are the same.
36      * The javadoc for the original method and the params can be seen from the See Also link.
37      * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService#createDocumentChecklist
38      */
39
40     public void postCreateDocumentChecklist(SessionContext sessionContext, DocumentReferenceInputDTO documentReferenceInputDTO) throws FatalException {
41     }
42
```

Provide the fully qualified name of the above groovy implementation in flx_fw_config_all_b against the corresponding service extension prop_id and category_id.

Figure 4–8 PROP_ID and CATEGORY_ID



	PROP_ID	CATEGORY_ID	PROP_VALUE	FACTORY_SHIPPED_FLAG	PROP_COMMENTS
1	com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService	ServiceExtensions	com.ofss.fc.groovy.test.VoidSubmissionDocumentApplicationServiceExt	Y	(null)

Figure 4–9 SUMMARY_TEXT

	SUMMARY_TEXT	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATED_DATE
1	com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService com.ofss.fc.groovy.test.VoidSubmissionDocumentApplicationServiceExt ServiceExtensions	ofssuser	24-NOV-15 03.07.11.000000000 PM	ofssuser	24-NOV-15 03.07.11.(

Package the above implementation and add in custom library which the application is referring to and add the groovy li in the classpath of the server which will be taken care by deployment team.

Figure 4–10 Add Groovy Library to Classpath

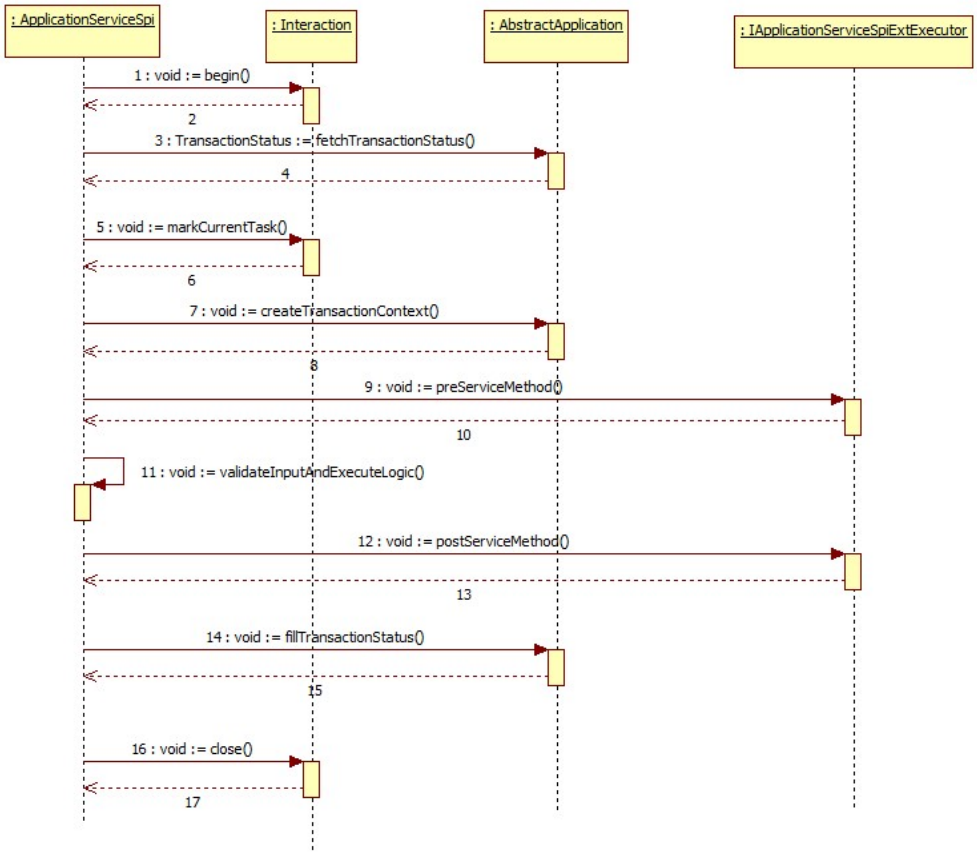
setDomainEnv.sh

```
465
466  if [ "${PRE_CLASSPATH}" != "" ] ; then
467     PRE_CLASSPATH="/scratch/app/product/fmw/obpinstall/obp/obp.thirdparty.app.domain/APP-INF/lib/groovy-all-2.3.10.jar${CLASSPATHSEP}${PRE_CLASSPATH}"
468     export PRE_CLASSPATH
469  else
470     PRE_CLASSPATH="/scratch/app/product/fmw/obpinstall/obp/obp.thirdparty.app.domain/APP-INF/lib/groovy-all-2.3.10.jar"
471     export PRE_CLASSPATH
472  fi
```

4.2 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 custom fields (that is, Dictionary pattern). 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.

Figure 4–11 Extended Application Service Extension

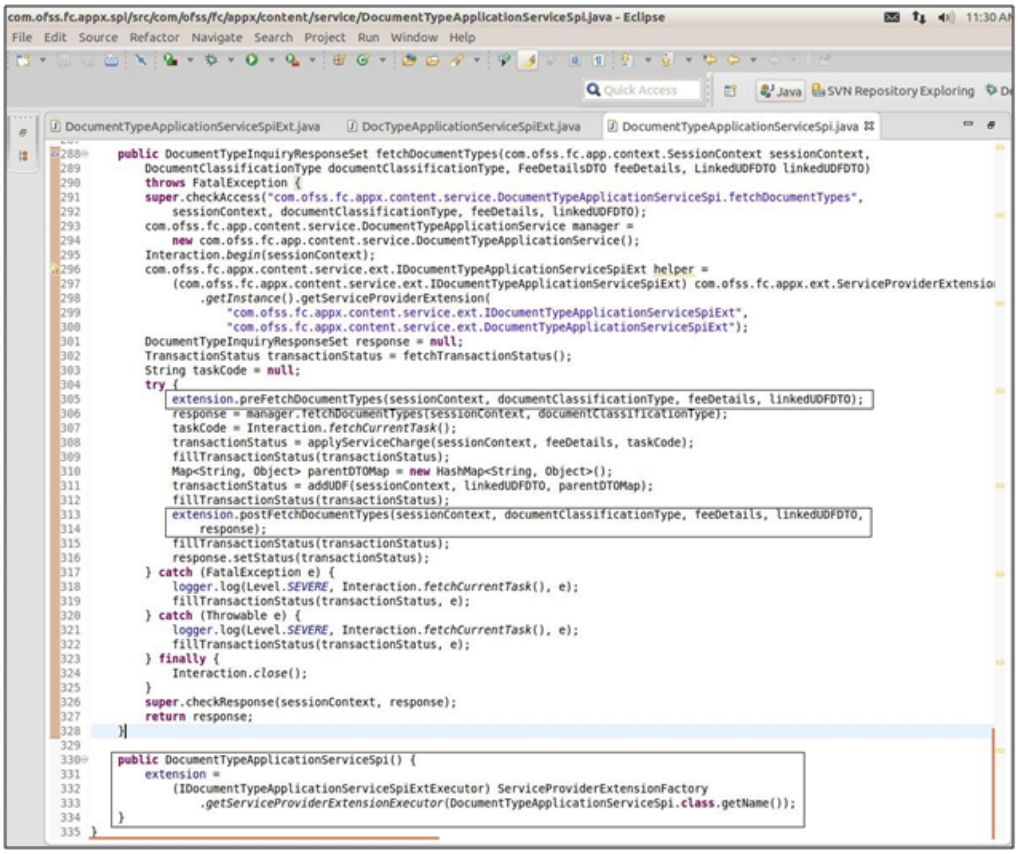


The pre hook is provided before the invocation of actual application service call inside the extended application service layer. At this step, the extended host application core framework is aware of the executing service with respect to the authenticated subject or the user who has posted the transaction 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. This is required to support extended features like approval related processing and to complete execution before marking the service execution status as successful 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 the approval view input 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 4–12 Extended Application Service Extension - Post and Pre Hook



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

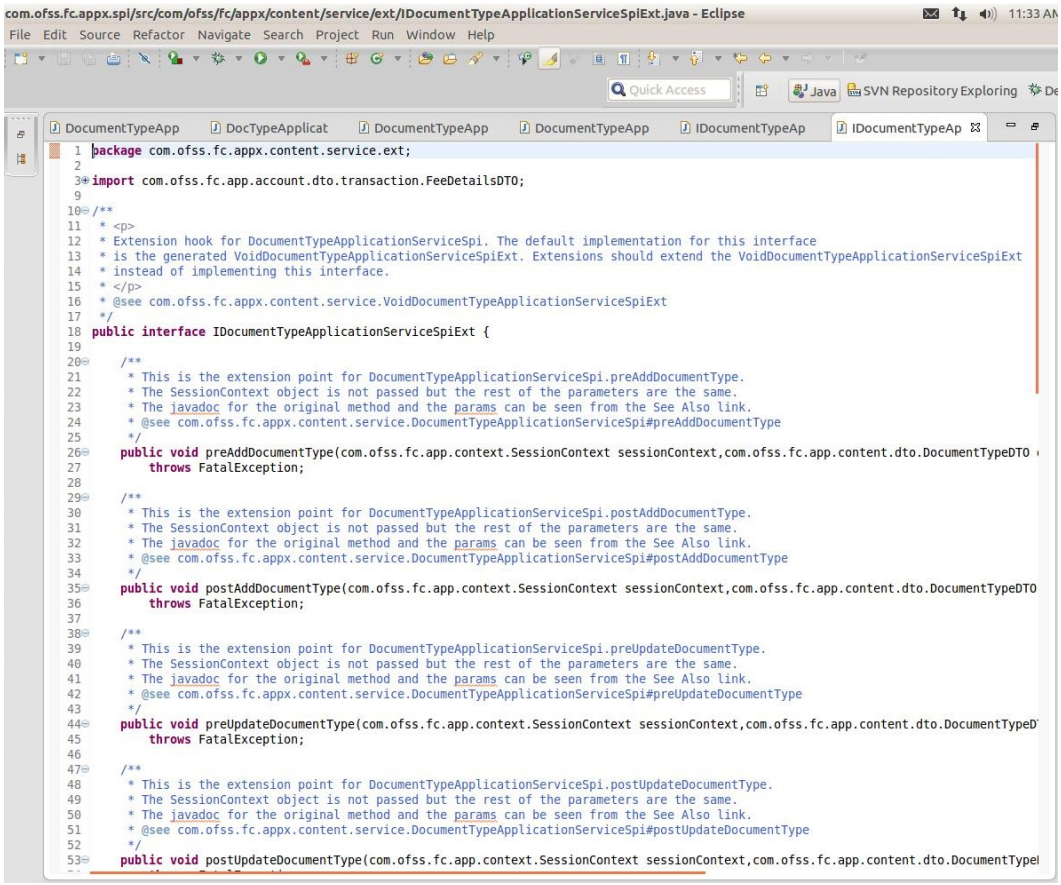
4.2.1 Extended Application Service Extension Interface

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 4–13 Extension Hook for Document Type Application Service Spi Ext

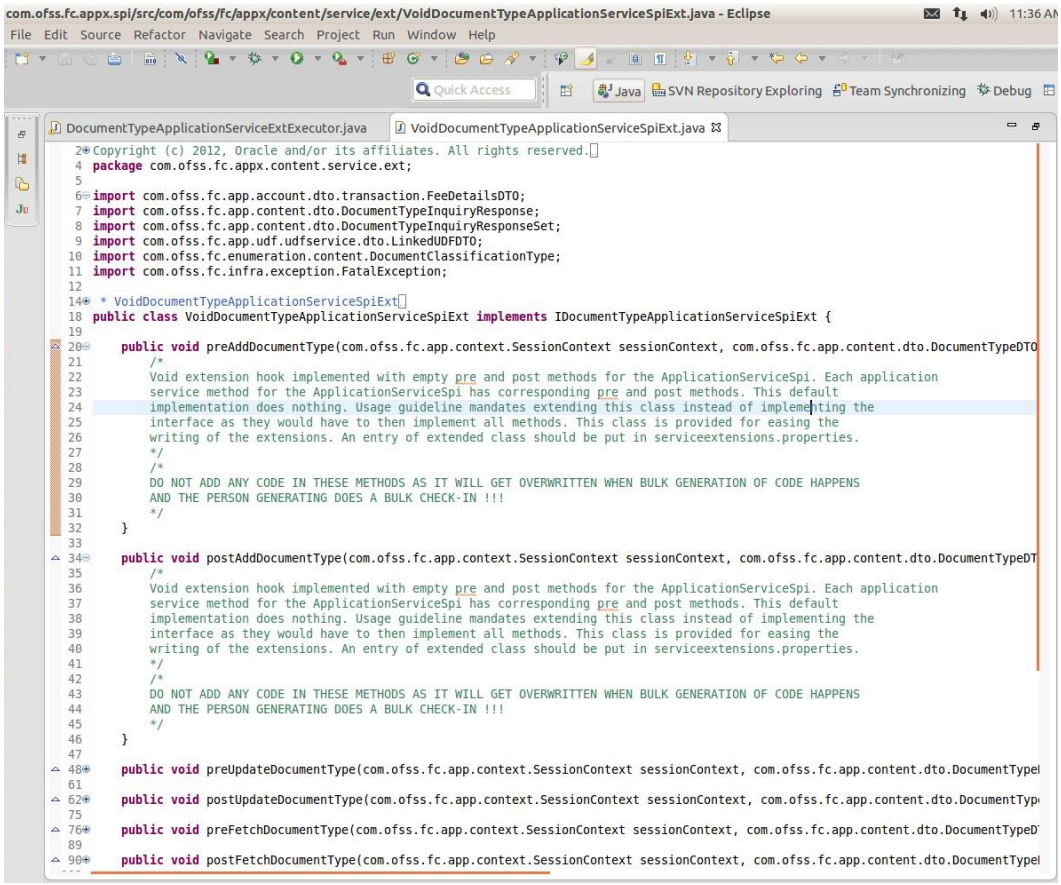


4.2.2 Default Implementation of Appx Extension

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 4–14 Default Implementation of Appx Extension



4.2.3 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.DocumentTypeApplicationServic
eSpi=
com.ofss.fc.appx.content.service.ext.DocumentTypeApplicationServic
eSpiExt
Multiple extension configuration
com.ofss.fc.appx.content.service.ext.DocumentTypeApplicationServic
eSpi=
```

```
com.ofss.fc.appx.content.service.ext.in.DocumentTypeApplicationSer
viceExt,
com.ofss.fc.appx.content.service.ext.in.mum.DocumentTypeApplicatio
nServiceExt,
com.ofss.fc.appx.content.service.ext.in.mum.ExtendedDocumentTypeAp
plicationService,
com.ofss.fc.appx.content.service.ext.in.blr.DocumentTypeApplicatio
nServiceExt,
com.ofss.fc.appx.content.service.ext.in.blr.ExtendedDocumentTypeAp
plicationService
```

4.2.4 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.

Figure 4–15 Extended Application Service Extension Executor

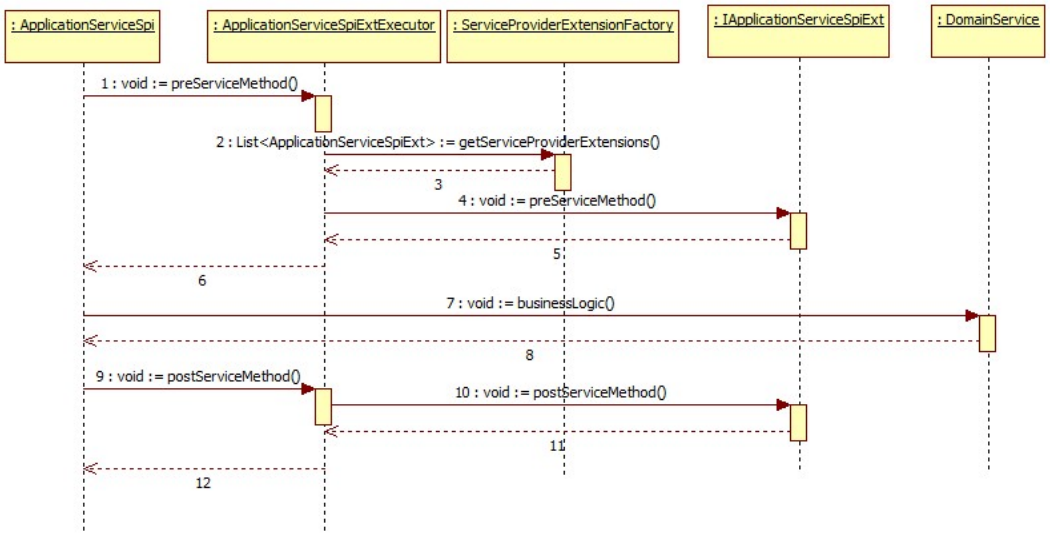


Figure 4–16 Extension Factory Hook for Document Type Application Service Spi Ext

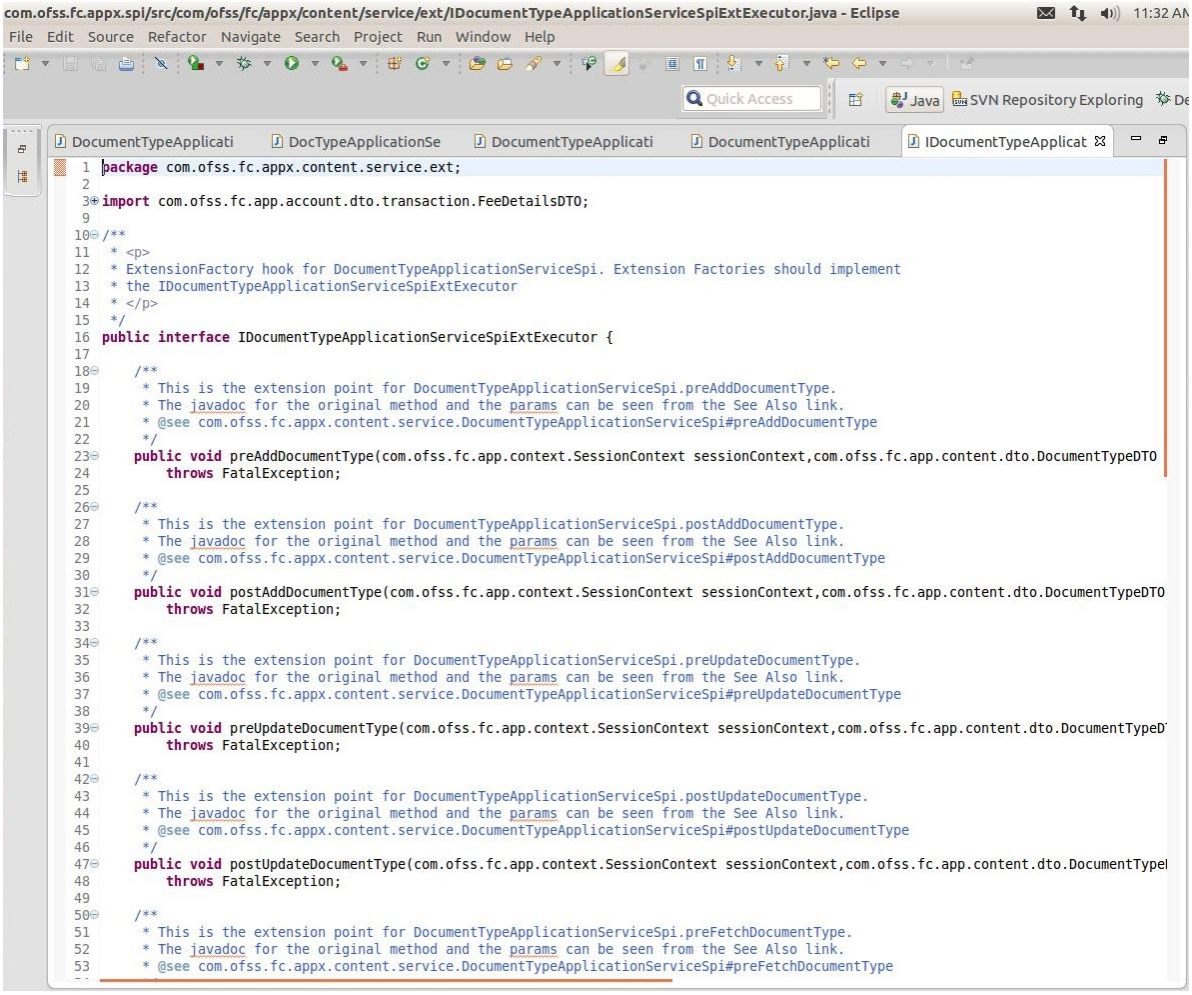
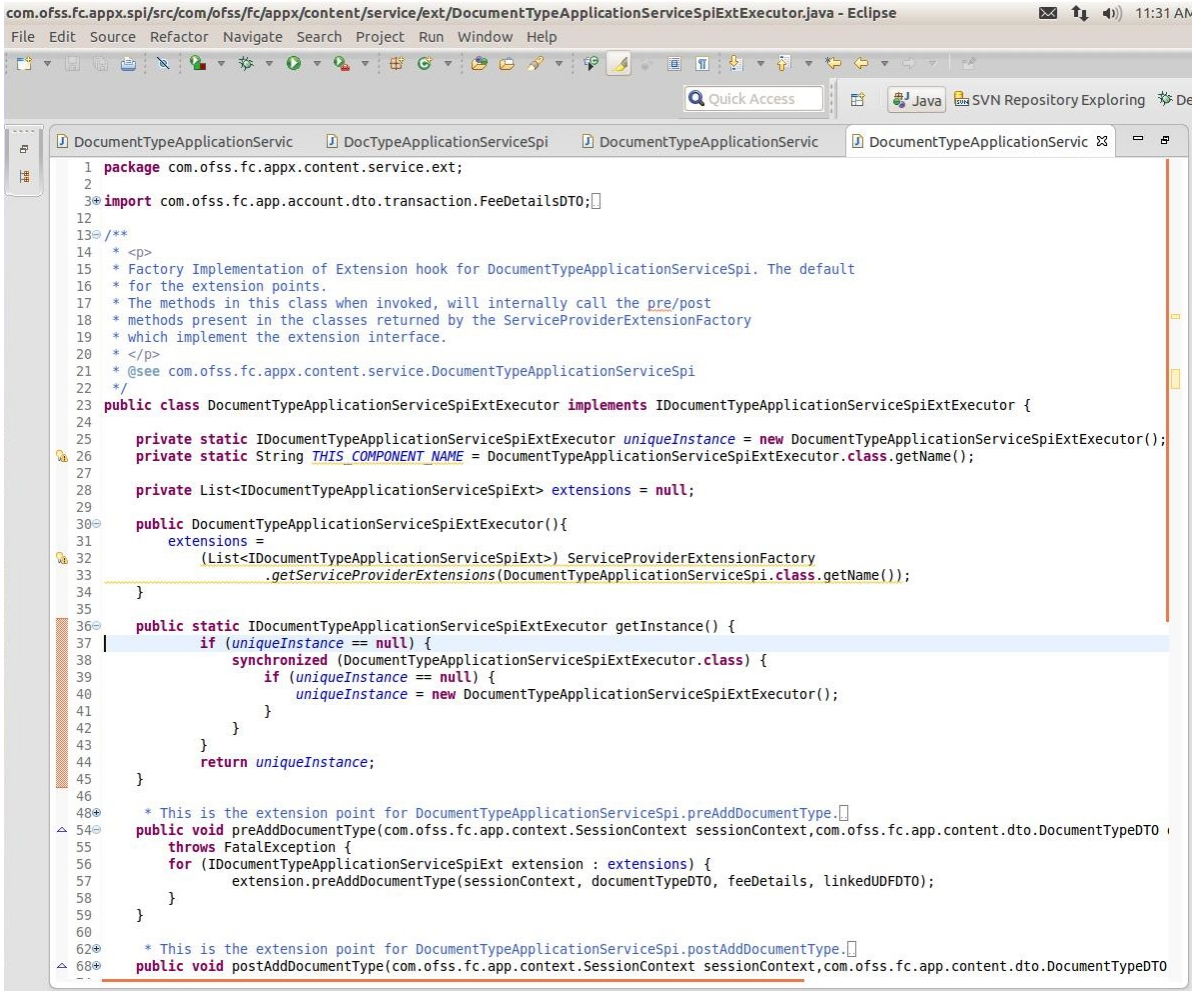


Figure 4–17 Factory Implementation of Extension Hook for Document Type Application Service Spi Ext



4.2.5 Application Service "appx" Extension using Groovy

Application service extension can be implemented using Groovy. The sample code and steps for service extension implementation using groovy is as follows:

Service extension groovy implementation class 'VoidSubmissionDocumentApplicationServiceExt' implementing product service extension interface 'com.ofss.fc.app.origination.service.core.submissiondocument.ext.ISubmissionDocumentApplicationServiceExt.

Figure 4–18 Application Service Appx Extension using Groovy

```
4
5  package com.ofss.fc.module.originationGroovy
6
7  import com.ofss.fc.app.context.SessionContext
8  import com.ofss.fc.app.origination.dto.core.document.DocumentReferenceInputDTO
9  import com.ofss.fc.app.origination.service.core.submissiondocument.ext.ISubmissionDocumentApplicationServiceExt
10 import com.ofss.fc.common.OriginationConstants
11 import com.ofss.fc.datatype.Date
12 import com.ofss.fc.enumeration.origination.OfferDocReferenceType
13 import com.ofss.fc.infra.exception.FatalException
14 /**
15  * <p>
16  * Groovy Extension hook for SubmissionDocumentApplicationService. The customization for the extension points.
17  * Each application service method for SubmissionDocumentApplicationService has corresponding pre
18  * and post methods. Whenever the Service Extensions are overriden with Groovy Extensions, the call will
19  * go to corresponding pre/post groovy extensions method and will execute the implemented logic.
20  * </p>
21  * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService
22  */
23 public class VoidSubmissionDocumentApplicationServiceExt implements ISubmissionDocumentApplicationServiceExt {
24
25     /**
26      * This is the extension point for SubmissionDocumentApplicationService.createDocumentChecklist.
27      * The SessionContext object is not passed but the rest of the parameters are the same.
28      * The javadoc for the original method and the params can be seen from the See Also link.
29      * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService#createDocumentChecklist
30      */
31     public void preCreateDocumentChecklist(SessionContext sessionContext, DocumentReferenceInputDTO documentReferenceInputDTO) throws FatalException {
32     }
33     /**
34      * This is the extension point for SubmissionDocumentApplicationService.createDocumentChecklist.
35      * The SessionContext object is not passed but the rest of the parameters are the same.
36      * The javadoc for the original method and the params can be seen from the See Also link.
37      * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService#createDocumentChecklist
38      */
39
40     public void postCreateDocumentChecklist(SessionContext sessionContext, DocumentReferenceInputDTO documentReferenceInputDTO) throws FatalException {
41     }
42
```

Provide the fully qualified name of the above groovy implementation in flx_fw_config_all_b against the corresponding service extension prop_id and category_id.

Figure 4–19 PROP_ID and CATEGORY_ID

	PROP_ID	CATEGORY_ID	PROP_VALUE	FACTORY_SHIPPED_FLAG	PROP_COMMENTS
1	com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService	ServiceExtensions	com.ofss.fc.groovy.test.VoidSubmissionDocumentApplicationServiceExt	Y	(null)

Figure 4–20 SUMMARY_TEXT

	SUMMARY_TEXT	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATED_DATE
1	com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService com.ofss.fc.groovy.test.VoidSubmissionDocumentApplicationServiceExt ServiceExtensions	ofssuser	24-NOV-15 03.07.11.000000000 PM	ofssuser	24-NOV-15 03.07.11.(

Package the above implementation and add in custom library which the application is referring to and add the groovy li in the classpath of the server which will be taken care by deployment team.

Figure 4–21 Add Groovy Library to Classpath

setDomainEnv.sh

```
465
466 if [ "${PRE_CLASSPATH}" != "" ] ; then
467    PRE_CLASSPATH="/scratch/app/product/fmw/obpinstall/obp/obp.thirdparty.app.domain/APP-INF/lib/groovy-all-2.3.10.jar${CLASSPATHSEP}${PRE_CLASSPATH}"
468    export PRE_CLASSPATH
469 else
470    PRE_CLASSPATH="/scratch/app/product/fmw/obpinstall/obp/obp.thirdparty.app.domain/APP-INF/lib/groovy-all-2.3.10.jar"
471    export PRE_CLASSPATH
472 fi
```

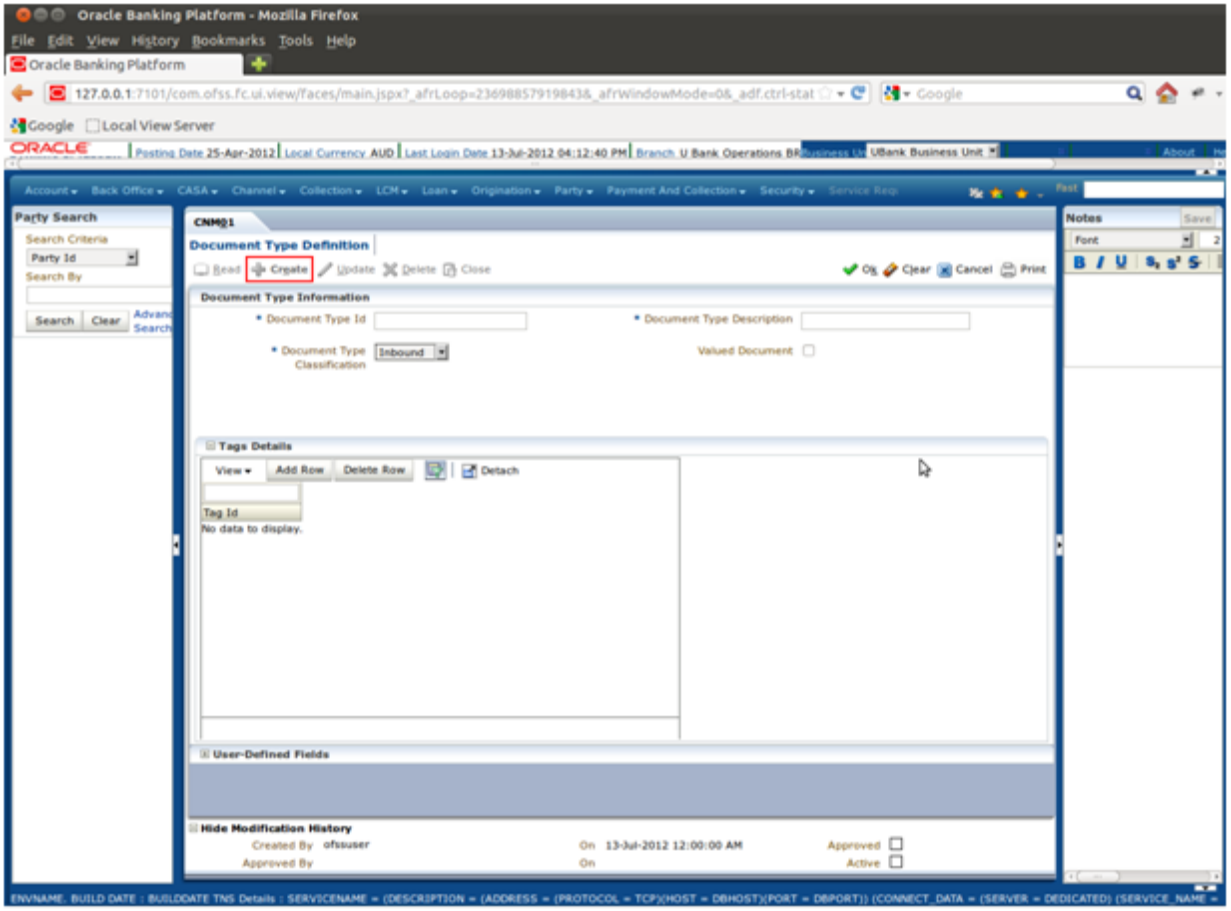
4.3 End-to-End Example of an Extension

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 4–22 Maintenance of Document Types



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.

Figure 4–23 Document Type Application Service Spi Ext - Appx Layer

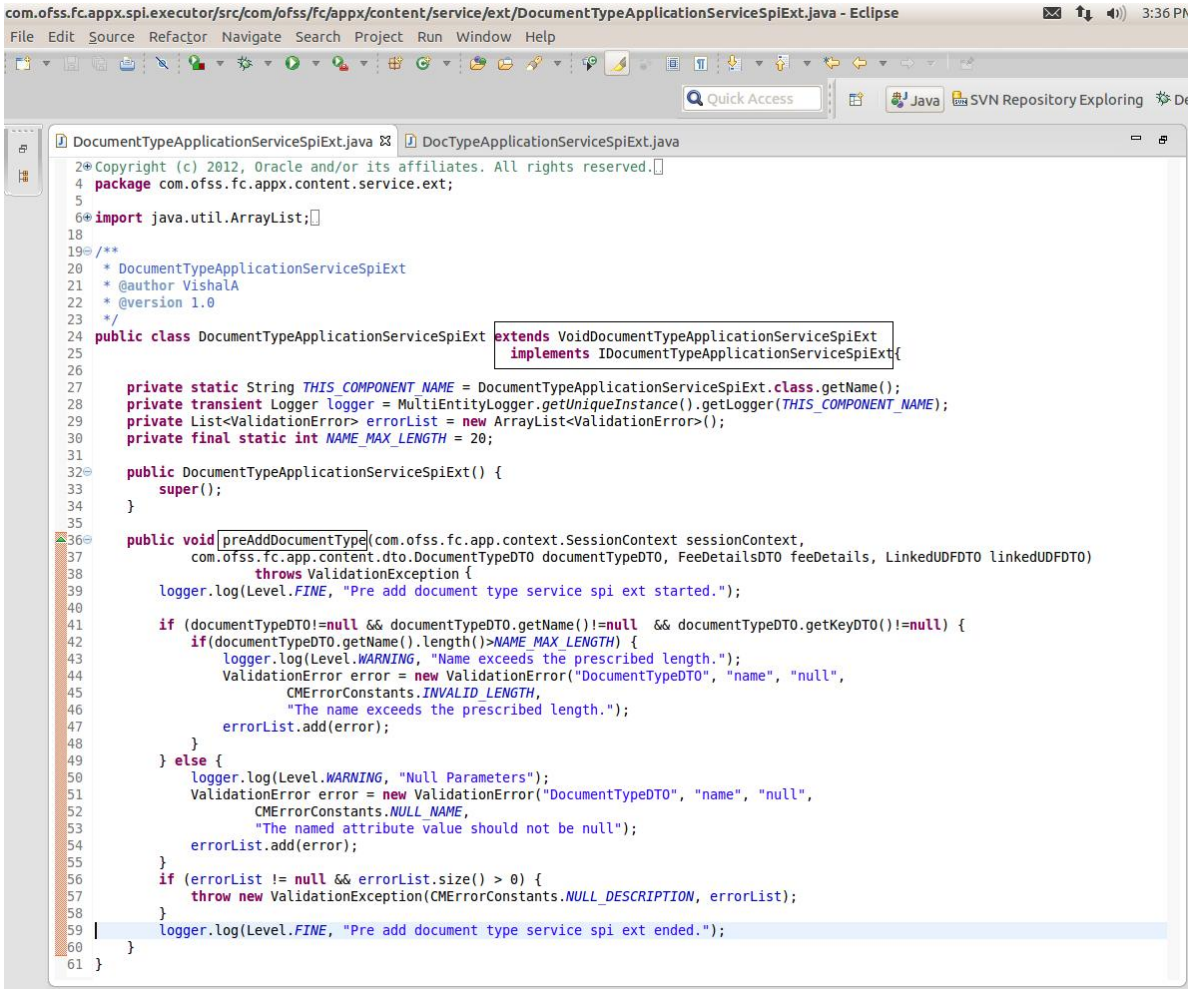
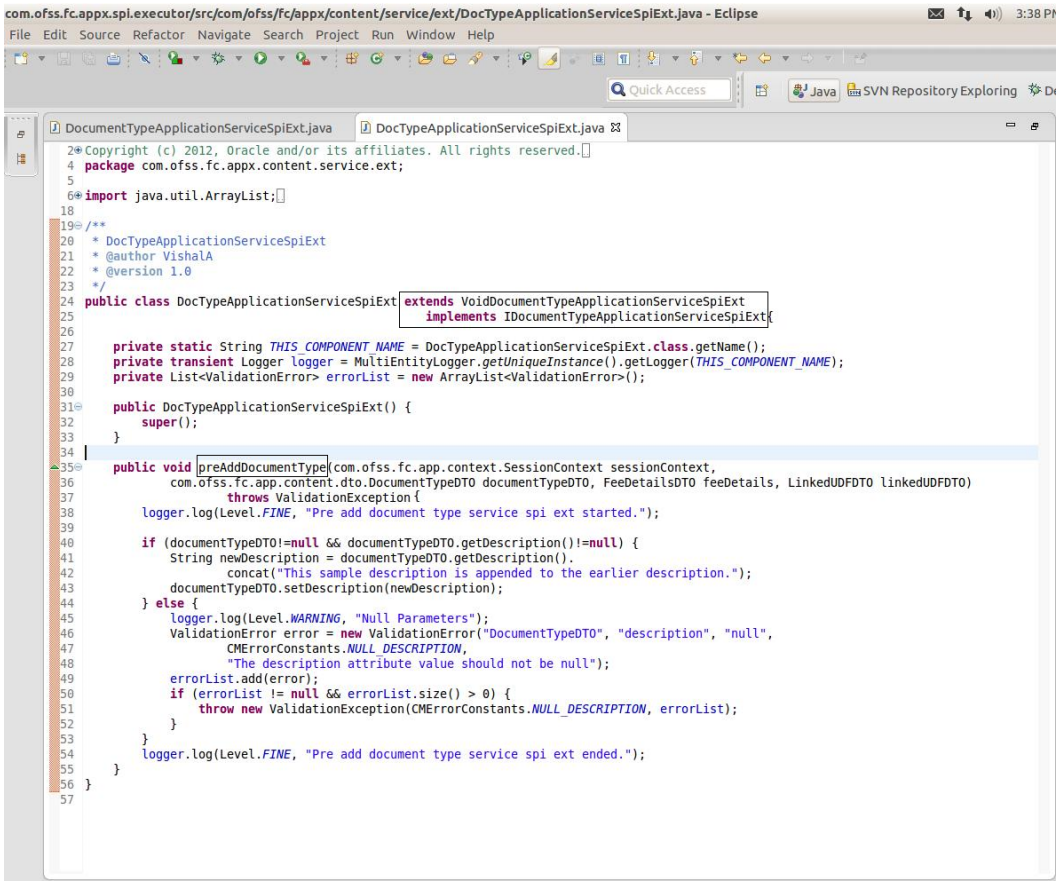


Figure 4–24 Doc Type Application Service Spi Ext - Appx Layer



In this example, we have added multiple extensions to the service of the app layer through the extension executor. We have 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.

Figure 4–25 Document Type Application Service Spi Ext - App Layer

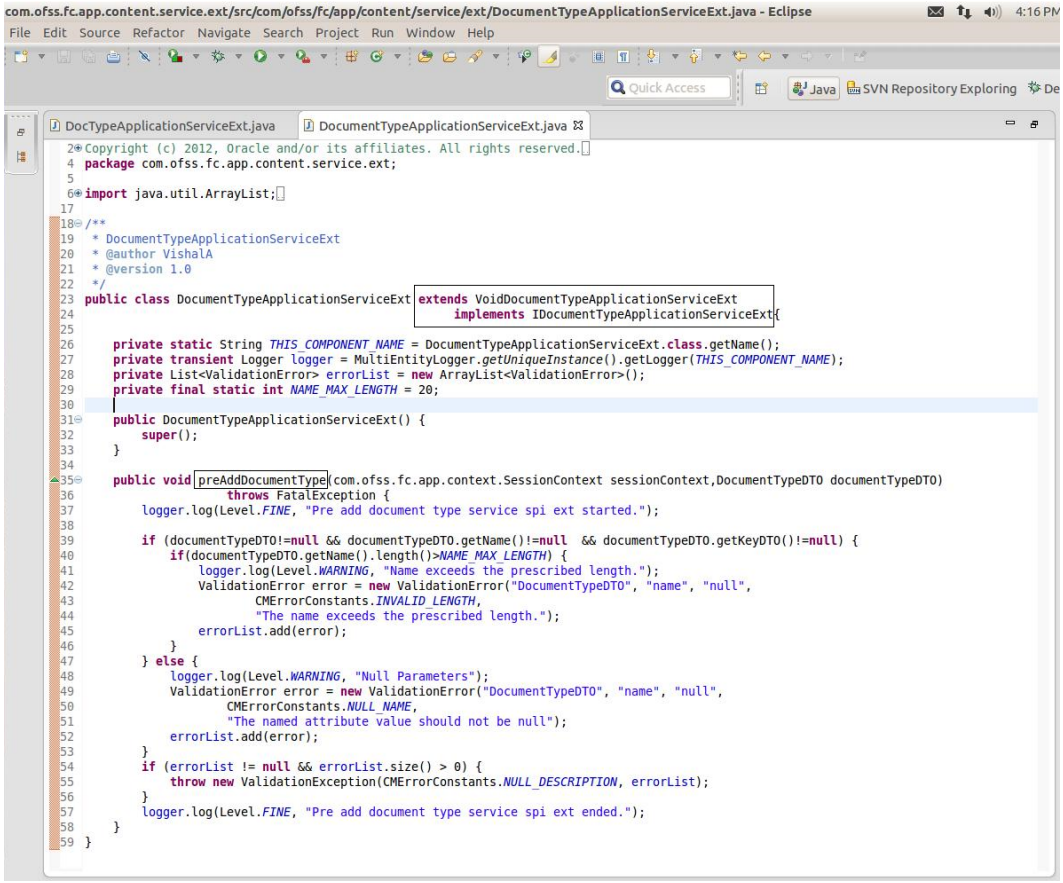
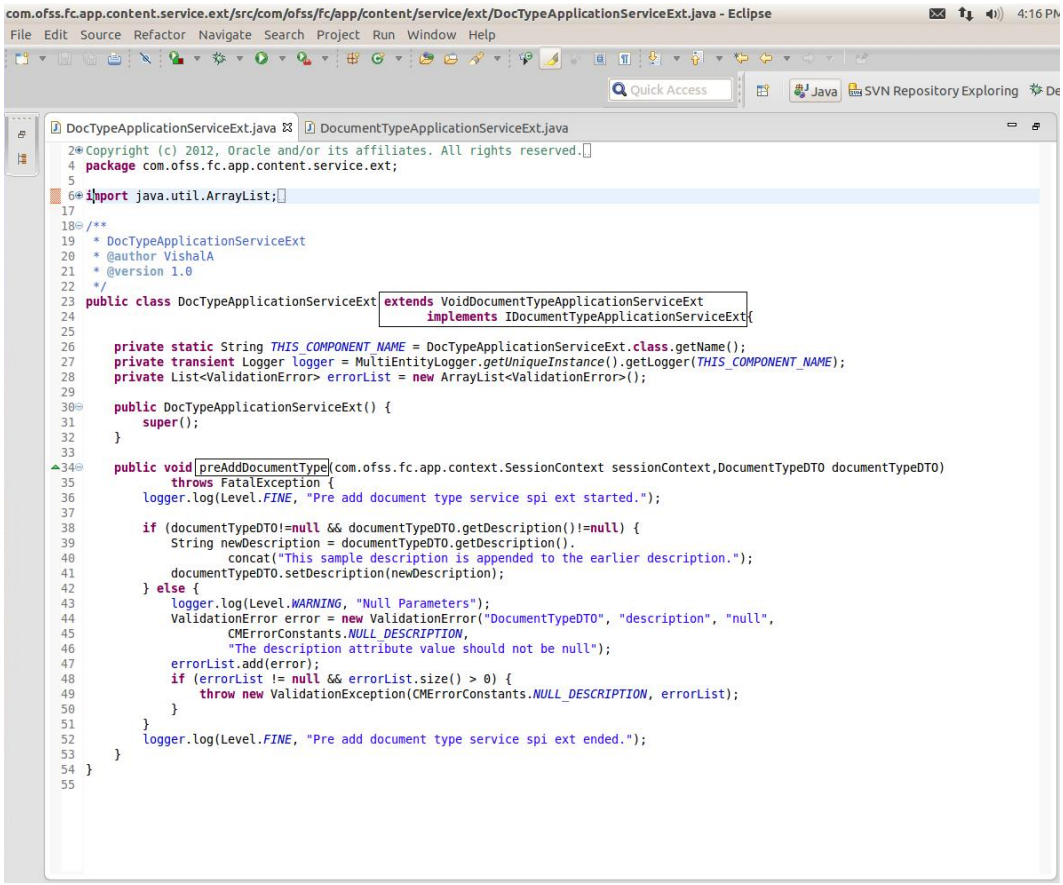


Figure 4–26 Doc Type Application Service Spi Ext - App Layer



4.4 Support for Middleware Specific Tasks and Application service

In case of OBP middleware implementation, SPI layer has ability to perform tasks before and after execution of application service. Also, you can have customized implementation of application service.

Following are the advantages of this feature:

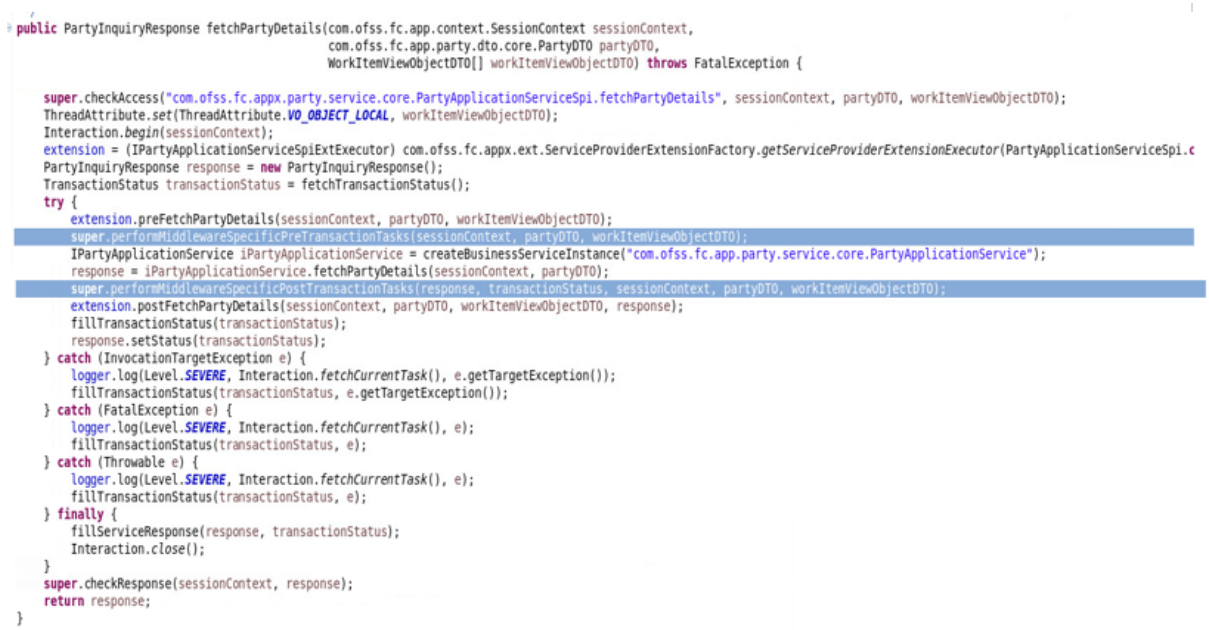
1. OBP signatures and Spi content will be same across all sites irrespective of OBP-middleware or Product processor implementation.
2. No appreciable change is required when the bank migrates from OBP Middleware to a full-fledged OBP product processor implementation.
3. OBP Middleware signatures are self-sufficient to address integrations to non-OBP core servicing systems and there is no need for wrapper consulting Spi class to be created.

4.4.1 Pre and Post Middleware Specific Transaction Tasks Overview

- Methods ‘performMiddlewareSpecificPreTransactionTasks’ and ‘performMiddlewareSpecificPostTransactionTasks’ is available in every spi to execute tasks.

- Pre tasks generally includes request enrichment, pre transaction auditing, business policy validations, post tasks generally includes alert processing, notification to external system.
- For example, in HDFC bank, in fund transfer transactions referenceNumber field is defaulted in pre processing if request comes from net banking.
- Tasks will be performed only in case of middleware implementation.
- Response enrichment: Response fields can be populated via metadata mapping.
- Example: “com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails” method look like this.

Figure 4–27 Pre and Post Middleware Specific Transaction Tasks Overview



```
public PartyInquiryResponse fetchPartyDetails(com.ofss.fc.app.context.SessionContext sessionContext,
                                              com.ofss.fc.app.party.dto.core.PartyDTO partyDTO,
                                              WorkItemViewObjectDTO[] workItemViewObjectDTO) throws FatalException {

    super.checkAccess("com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails", sessionContext, partyDTO, workItemViewObjectDTO);
    ThreadAttribute.set(ThreadAttribute.VO_OBJECT_LOCAL, workItemViewObjectDTO);
    Interaction.begin(sessionContext);
    extension = (IPartyApplicationServiceSpiExtExecutor) com.ofss.fc.appx.ext.ServiceProviderExtensionFactory.getServiceProviderExtensionExecutor(PartyApplicationServiceSpi.c
    PartyInquiryResponse response = new PartyInquiryResponse();
    TransactionStatus transactionStatus = fetchTransactionStatus();
    try {
        extension.preFetchPartyDetails(sessionContext, partyDTO, workItemViewObjectDTO);
        super.performMiddlewareSpecificPreTransactionTasks(sessionContext, partyDTO, workItemViewObjectDTO);
        IPartyApplicationService iPartyApplicationService = createBusinessServiceInstance("com.ofss.fc.app.party.service.core.PartyApplicationService");
        response = iPartyApplicationService.fetchPartyDetails(sessionContext, partyDTO);
        super.performMiddlewareSpecificPostTransactionTasks(response, transactionStatus, sessionContext, partyDTO, workItemViewObjectDTO);
        extension.postFetchPartyDetails(sessionContext, partyDTO, workItemViewObjectDTO, response);
        fillTransactionStatus(transactionStatus);
        response.setStatus(transactionStatus);
    } catch (InvocationTargetException e) {
        logger.log(Level.SEVERE, Interaction.fetchCurrentTask(), e.getTargetException());
        fillTransactionStatus(transactionStatus, e.getTargetException());
    } catch (FatalException e) {
        logger.log(Level.SEVERE, Interaction.fetchCurrentTask(), e);
        fillTransactionStatus(transactionStatus, e);
    } catch (Throwable e) {
        logger.log(Level.SEVERE, Interaction.fetchCurrentTask(), e);
        fillTransactionStatus(transactionStatus, e);
    } finally {
        fillServiceResponse(response, transactionStatus);
        Interaction.close();
    }
    super.checkResponse(sessionContext, response);
    return response;
}
```

4.4.2 Sample Configuration

Middleware task configuration is based on channel and service Id. The DB tables associated with the execution steps are:

- **FLX_FW_MW_TASKS:** This table is used to make entries for middleware specific task based on channel and service id.

  Sample entry for ‘com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi. fetchPartyDetails' will look like, where PartyDeatilsAdapter is having several methods to perform tasks like pre business policy, post business policy, pre and post processing.

Figure 4–28 FLX_FW_MW_TASKS

	CHANNEL_ID	APP_SERVICE_NAME	METHOD_NAME	CATEGORY_ID	EXECUTION_ORDER	CALL_ATTR_ID
1	IB	com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi	fetchPartyDetails	PreProcessing	1	PARTY_DETAILS_PreProcessing
2	IB	com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi	fetchPartyDetails	PreBusinessPolicy	1	PARTY_DEATILS_PreBusinessPolicy
3	IB	com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi	fetchPartyDetails	PostProcessing	1	PARTY_DETAILS_PostProcessing
4	IB	com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi	fetchPartyDetails	PostBusinessPolicy	1	PARTY_DETAILS_PostBusinessPolicy

ADAPTER_FACTORY_CONSTANT	ADAPTER_NAME	ADAPTER_METHOD	DTO_CLASS	IS_ENABLED	IGNORE_EXCEPTION
PARTY_ADAPTER_FACTORY	PartyDetailsAdapter	fetchPartyDetailsPreProcessing	com.ofss.fc.app.party.dto.core.PartyDTO	Y	N
PARTY_ADAPTER_FACTORY	PartyDetailsAdapter	fetchPartyDetailsPreBusinessPolicy	com.ofss.fc.app.party.dto.core.PartyDTO	Y	N
PARTY_ADAPTER_FACTORY	PartyDetailsAdapter	fetchPartyDetailsPostProcessing	com.ofss.fc.app.party.dto.core.PartyDTO	Y	N
PARTY_ADAPTER_FACTORY	PartyDetailsAdapter	fetchPartyDetailsPostBusinessPolicy	com.ofss.fc.app.party.dto.core.PartyDTO	Y	N

- **FLX_FW_MW_TASKS_DTO_DEFN:** This table is used to make entires for DTO class and DTO fields for response enrichment purpose.

  Sample entry for service name 'com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails' will look like, where PartyInquiryResponse is response type and AdapterResponsibilityChainResponse is type of holder dto for response enrichment.

Figure 4–29 FLX_FW_MW_TASKS_DTO_DEFN

	DTO_CLASS	FIELD_NAME
1	com.ofss.fc.framework.domain.adapter.AdapterResponsibilityChainResponse	postExecutionResponse1
2	com.ofss.fc.framework.domain.adapter.AdapterResponsibilityChainResponse	preExecutionResponse1
3	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	partyType
4	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	individualDTO
5	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	organizationDTO
6	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	bankingEntityDTO
7	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	trustDTO
8	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	comments
9	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	dateofOnboarding
10	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	roleSpecificDetail
11	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	partyWarningIndicatorDTO
12	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	employmentHistoryDTO
13	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	applicantMarketingConsentDTO
14	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	blacklisted

- **FLX_FW_MW_TASKS_DTO_MAP:** This table is used to establish mapping between flw_fw_mw_tasks_dto_defn columns(dto class and dto field) and task entry defined in flx_fw_mw_tasks column (call_attr_id).

  Sample entry for service id 'com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails (service name + service method name) task entry and response enrich dto field mappings, where field name postExecutionResponse is having mapping with PartyDetailsAdapter method fetchPartyDetailsPostProcesing through cod_attr_id PARTY_DATAILS_PreProcessing.

Figure 4–30 FLX_FW_MW_TASKS_DTO_MAP

SERVICE_ID	CATEGORY_ID	DTO_CLASS	FIELD_NAME	COD_ATTR_ID
com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails	PostProcessing	com.ofss.fc.framework.domain.adapter.AdapterResponsibilityChainResponse	postExecutionResponse1	PARTY_DETAILS_PostProcessing
com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails	PreProcessing	com.ofss.fc.framework.domain.adapter.AdapterResponsibilityChainResponse	preExecutionResponse1	PARTY_DETAILS_PreProcessing
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	partyType	PartyType
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	individualDTO	IndividualDTO
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	organisationDTO	OrganisationDTO
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	bankingEntityDTO	BankingEntityDTO
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	trustDTO	TrustDTO
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	comments	Comments
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	dateofOnboarding	DateofOnboarding
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	roleSpecificDetail	RoleSpecificDetail
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	partyWarningIndicatorDTO	PartyWarningIndicatorDTO
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	employmentHistoryDTO	EmploymentHistoryDTO
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	applicantMarketingConsentDTO	ApplicantMarketingConsentDTO
DEFAULT	ResponseDTO	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	blacklisted	Blacklisted

- **FLX_MD_SERVICE_ATTR:** This table is used to keep entry for source and destination dto for response enrichment purpose through column entry ref_field_defn_id.

  Sample entry for service id 'com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails', where enriched dto fields through adapter method having cod_attr_id like RESP_ENRICH.X.

Figure 4–31 FLX_MD_SERVICE_ATTR

	COD_SERVICE_ATTR_ID	TYP_DATA_AVAIL	TYP_DATA_SRC	COD_ATTR_ID
1	com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails.RESP_ENRICH.PartyWarningIndicatorDTO.DTO	INDIRECT	OUTPUT	RESP_ENRICH.PartyWarningIndicatorDTO
2	com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails.RESP_ENRICH.DateofOnboarding.DTO	INDIRECT	OUTPUT	RESP_ENRICH.DateofOnboarding

COD_SERVICE_ID	PARAMETER_NAME	REF_ENT_DEFN_ID	KEY_SERVICE_ATTR_ID	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE	OBJECT_VERSION_NUMBER
com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails	(null)	(null)	(null)	SETUP	30-MAY-17	SETUP	30-MAY-17	1
com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails	(null)	(null)	(null)	SETUP	30-MAY-17	SETUP	30-MAY-17	1

OBJECT_STATUS	REF_FIELD_DEFN_ID
A	com.ofss.fc.framework.domain.adapter.AdapterResponsibilityChainResponse.PreExecutionResponse1,com.ofss.fc.app.party.dto.core.PartyInquiryResponse.PartyWarningIndicatorDTO
A	com.ofss.fc.framework.domain.adapter.AdapterResponsibilityChainResponse.PostExecutionResponse1,com.ofss.fc.app.party.dto.core.PartyInquiryResponse.DateofOnboarding

- **FLX_MD_GEN_ATTR_LEGACY_B:** This table contains all the attributes for metadata, while making entry for attribute which has to enriched you can append responseenrich in cod_constraint_attr_id so you can differentiate between actual service attributes entry and response enriched entry.

  Sample entry for PartyInquireResponse fields with their respective data type.

Figure 4–32 FLX_MD_GEN_ATTR_LEGACY_B

	COD_CONSTRAINT_ATTR_ID	TXT_CONSTRAINT_ATTR_NAME	DATA_TYPE	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE	OBJECT_VERSION_N...	OBJECT_STATUS
294	PartyType	PartyType	com.ofss.fc.enumeration.party.PartyType	SETUP	30-MAY-17 12.48.38.000000000 PM	(null)	(null)	1	A
295	Blacklisted	Blacklisted	java.lang.Boolean	SETUP	30-MAY-17 12.48.38.000000000 PM	(null)	(null)	1	A
296	IndividualDTO	IndividualDTO	com.ofss.fc.app.party.dto.individual.IndividualDTO	SETUP	30-MAY-17 02.02.29.000000000 PM	(null)	(null)	1	A
297	OrganizationDTO	OrganizationDTO	com.ofss.fc.app.party.dto.organization.OrganizationDTO	SETUP	30-MAY-17 02.02.29.000000000 PM	(null)	(null)	1	A
298	BankingEntityDTO	BankingEntityDTO	com.ofss.fc.app.party.dto.organization.BankingEntityDTO	SETUP	30-MAY-17 02.02.29.000000000 PM	(null)	(null)	1	A
299	TrustDTO	TrustDTO	com.ofss.fc.app.party.dto.trust.TrustDTO	SETUP	30-MAY-17 02.02.29.000000000 PM	(null)	(null)	1	A
300	Comments	Comments	com.ofss.fc.app.party.dto.core.CommentDTO	SETUP	30-MAY-17 02.02.29.000000000 PM	(null)	(null)	1	A
301	RESP_ENRICH.DateofOnboar...	DateofOnboarding	com.ofss.fc.datatype.Date	SETUP	30-MAY-17 02.02.29.000000000 PM	(null)	(null)	1	A
302	RoleSpecificDetail	RoleSpecificDetail	com.ofss.fc.app.party.dto.core.RoleSpecificDetailDTO	SETUP	30-MAY-17 02.02.29.000000000 PM	(null)	(null)	1	A
303	RESP_ENRICH.PartyWarning...	PartyWarningIndicatorDTO	com.ofss.fc.app.party.dto.core.PartyWarningIndicatorDTO	SETUP	30-MAY-17 02.02.29.000000000 PM	(null)	(null)	1	A
304	EmploymentHistoryDTO	EmploymentHistoryDTO	com.ofss.fc.app.party.dto.individual.EmploymentHistoryDTO	SETUP	30-MAY-17 02.02.29.000000000 PM	(null)	(null)	1	A
305	ApplicantMarketingConsen...	ApplicantMarketingConsen...	com.ofss.fc.app.party.dto.core.MarketingConsentDTO	SETUP	30-MAY-17 02.02.29.000000000 PM	(null)	(null)	1	A

4.4.3 Custom Application Service

- In SPI method createBusinessServiceInstance is used to get customized instance of application service.
- Custom Application Service name is maintained 'CustomEntities' preferences.
- For example com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails can call com.ofss.cz.hdfc.app.party.service.core.PartyApplicationService.fetchPartyDetails.

Figure 4–33 Custom Application Service

```
private IPartyApplicationService createBusinessServiceInstance(String businessServiceName) throws InvocationTargetException {

    IPartyApplicationService partyApplicationService = null;
    Object customApplicationServiceInstance = getCustomBusinessServiceInstance(businessServiceName);
    if (customApplicationServiceInstance != null) {
        partyApplicationService = (IPartyApplicationService) customApplicationServiceInstance;
    } else {
        partyApplicationService = new PartyApplicationService();
    }
    return partyApplicationService;
}
```

5 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.service.json.client.Refer
enceObjectApplicationServiceJSONClient jsonStub = new
com.ofss.fc.app.me.service.referencedata.service.json.client.Refer
enceObjectApplicationServiceJSONClient(jsonServiceUrl);
returnObj = jsonStub.addReferenceObject(sessionContext,
referenceObjectDTO);
} else if ("LOCAL".equals(protocol) && "APP".equals
(hostApplicationLayer)) {
try {
Object[] args = new Object[] { sessionContext, referenceObjectDTO
};
String stringToCompleteClassName =
"com.ofss.fc.app.me.service.referencedata.ReferenceObjectApplicati
onService";
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.Refer
enceObjectApplicationServiceJSONClient jsonStub = new
com.ofss.fc.app.me.service.referencedata.service.json.client.Refer
enceObjectApplicationServiceJSONClient(jsonServiceUrl);
returnObj = jsonStub.addReferenceObject(sessionContext,
referenceObjectDTO);
} else if ("LOCAL".equals(protocol) && "APP".equals
(hostApplicationLayer)) {
try {
Object[] args = new Object[] { sessionContext, referenceObjectDTO
};
String stringToCompleteClassName =
"com.ofss.fc.app.me.service.referencedata.ReferenceObjectApplicati
onService";
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;
}
```

6 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.

6.1 Adapter Implementation Architecture

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

6.1.1 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.

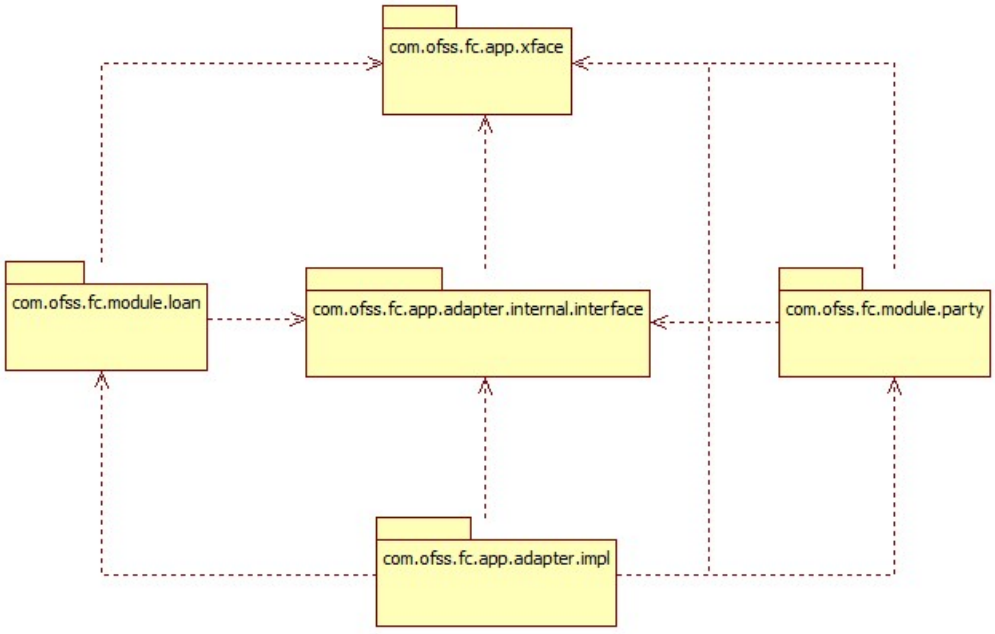
Table 6–1 Components of Adapter Implementation

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.adapter.internal.interface	Package contains adapter interfaces for	com.ofss.fc.app.adapter.ep.IEventProcessingAdapter Abstract Factory com.ofss.fc.app.adapter.AdapterFactory

Sr.	Project Name	Description	Example
		all modules and the abstract factory implementation (i.e. factory of adapter factories).	
3	com.ofss.fc.app.adapter.impl	This project has the **implementation** of adapter interfaces and corresponding adapter factories.	com.ofss.fc.app.adapter.ep.**impl**.EventProcessingAdapter com.ofss.fc.app.adapter.ep.**impl**.EventProcessingAdapterFactory

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:

Figure 6–1 Package Diagram



The dependencies among the packages as shown in the diagram 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.

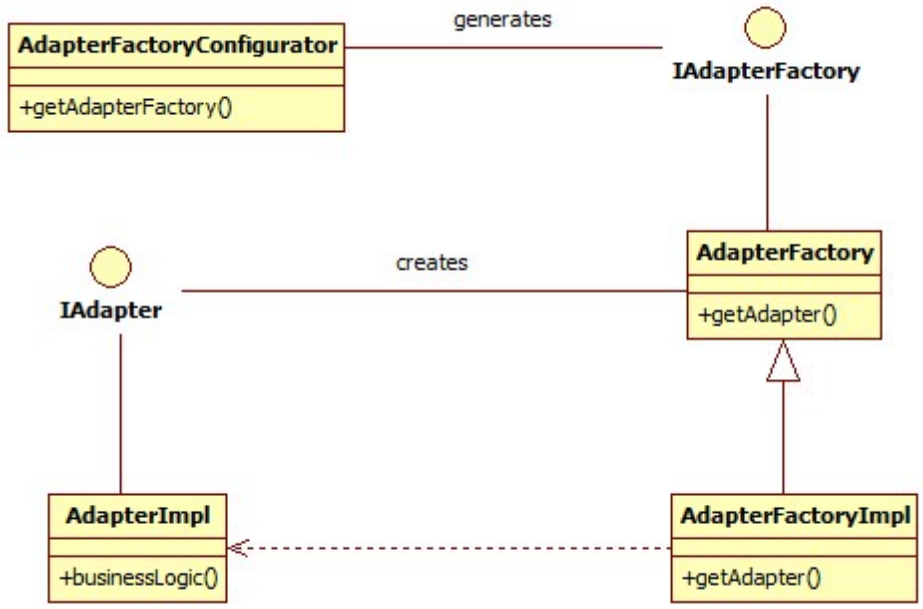
6.1.2 Adapter Mechanism Class Diagram

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.

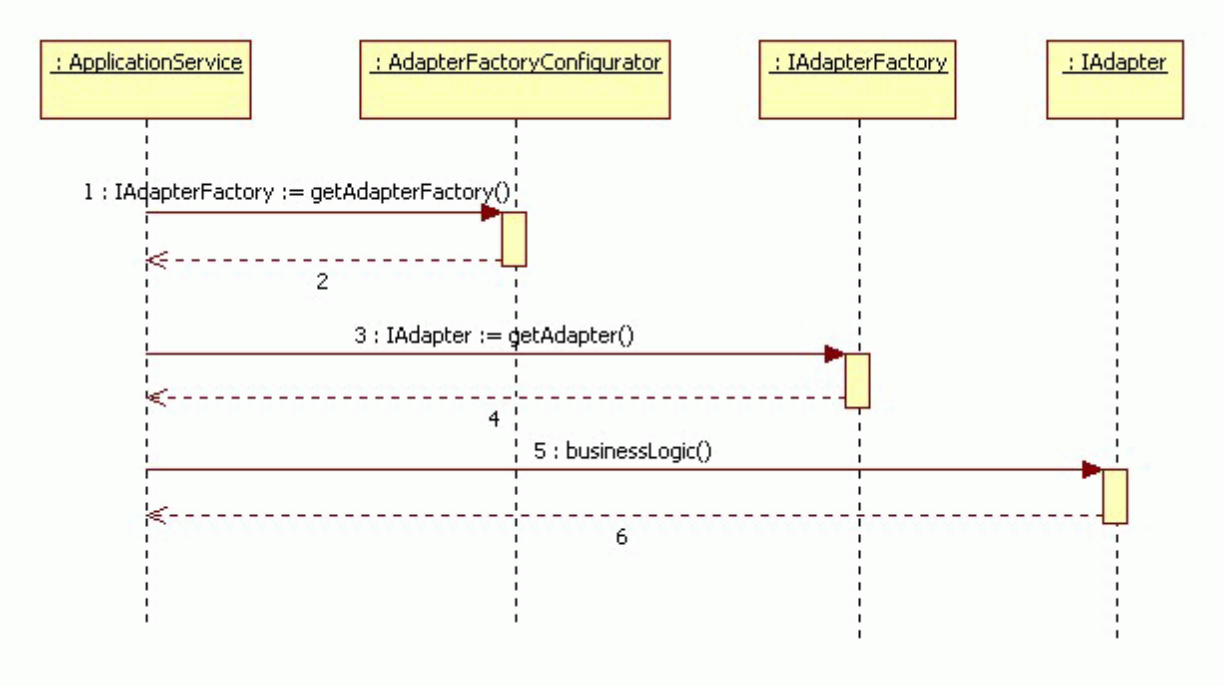
Figure 6–2 Adapter Mechanism Class Diagram



6.1.3 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 6–3 Adapter Mechanism Sequence Diagram



6.2 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:

6.2.1 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.EventProcessingAdapterF
actory
```

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.fetchSessionContext(), key, activityDataId);
}
```

6.2.2 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.

```
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);
}
```

6.2.3 Example 3 - Adapter Implementation Using Groovy

Groovy adapter implementation acts as a wrapper on the product. Adapter implementation in OBP is used to make service call from one module to another module.

Existing product adapter will be overridden by the new custom made adapter for Groovy. This new Groovy adapter would contain groovy implementation methods which might call groovy files internally to perform desired functionality.

For example, for CreditCardAdapter, the following steps would have to be followed for implementation of a custom Groovy Adapter.

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

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

Figure 6–4 Adapter Implementation Using Groovy

```
import com.ofss.fc.app.adapter.AdapterFactory

/**
 * This class represents GroovyCreditCardAdapterFactory.This factory class creates a GroovyCreditCardAdapter Object.
 *
 * @author sambedita.nayak@oracle.com
 */
class GroovyCreditCardAdapterFactory extends AdapterFactory{

    private static final String THIS_COMPONENT_NAME = GroovyCreditCardAdapterFactory.class.getName();
    private static GroovyCreditCardAdapterFactory instance;
    private static boolean isMockEnabled=false;

    /**
     * Return the empty new instance of the class in case it is not present<br>
     * else return the instance that is already present<br>
     * This method internally synchronizes on the class monitor to ensure that<br>
     * only one caller can create the instance at any given point in time thereby<br>
     * ensuring that this class is present as an singleton instance inside the JVM.<br>
     *
     */
    public static GroovyCreditCardAdapterFactory getInstance() {

        synchronized (GroovyCreditCardAdapterFactory.class) {
            if (instance == null) {
                instance = new GroovyCreditCardAdapterFactory();
            }
        }
        return instance;
    }
    /**
     * This method will create the instance of GroovyCreditCardAdapter and return the same.if mocking is enabled, the method would return a mocked <br>
     * instance of the adapter.
     *
     * @param adapterClass
     * @return ICreditCardAdapter
     */
    public ICreditCardAdapter getAdapter(String adapterClass) {

        ICreditCardAdapter adapter = null;
        adapterClass = "GroovyCreditCardAdapter";
        if (adapterClass.equals('GroovyCreditCardAdapter')) {
            if ( !isMockEnabled) {
                adapter = new GroovyCreditCardAdapter();
                return adapter;
            } else {
                return adapter;
            }
```

OBP gives an adapter implementation for CreditCard. The adapter implements to the interface shown below. The interface method *inquireCreditCardDetailsForCardNumber* would be overridden by the customization developer while providing the actual implementation of the desired functionality.

Figure 6–5 Credit Card Adapter Implementation Using Groovy

```
import com.ofss.fc.app.adapter.card.ICreditCardAdapter;


/**
 *This class represents GroovyCreditCardAdapter.This factory class creates a GroovyCreditCardAdapter Object.
 *This class contains the interaction services to be used from credit card (cc) module to Groovy credit card.
 *
 *@author sambenay
 */

class GroovyCreditCardAdapter implements ICreditCardAdapter{

    /**
     *
     * This method is used to call the fetchCreditCardDetailsForCardNumber method of class GroovyCreditCardAppService in order to fetch<br>
     * the details of credit card.
     * @param SessionContext
     *            ,sessionContext, , Session Context which must contain the session context information such as user id,
     *            branch, branch code, channel etc.
     * @param String ,cardNumber
     * @return {@link com.ofss.fc.app.card.dto.credit.account.CreditCardDetailsDTO} This class represents DTO class which has details about the credit card
     * @throws FatalException
     */

    @Override
    public CreditCardDetailsDTO inquireCreditCardDetailsForCardNumber(SessionContext sessionContext, String cardNumber) throws FatalException {
        GroovyCreditCardAppService groovyCreditCardAppService = new GroovyCreditCardAppService();

        return groovyCreditCardAppService.fetchCreditCardDetailsForCardNumber();
    }

    public CreditCardBasicDetailsDTO inquireBasicCreditCardDetailsForCardNumber(SessionContext sessionContext, String cardNumber) throws FatalException {

    public CreditCardDetailsDTO inquireCreditCardDetailsForCardReferenceNumber(SessionContext sessionContext, String cardReferenceNumber) throws FatalException {

    public CreditCardBasicDetailsDTO inquireBasicCreditCardDetailsForCardReferenceNumber(SessionContext sessionContext, String cardReferenceNumber)

    public CreditCardDetailsDTO openCreditCardAccount(SessionContext sessionContext, CreditCardAccountInputDTO creditCardAccountInputDTO) throws FatalException {

    public CreditCardDetailsDTO performPostOpenCreditCardAccountOperation(SessionContext sessionContext, CreditCardAccountInputDTO creditCardAccountInputDTO)

    public CreditCardDetailsDTO amendCardLimit(SessionContext sessionContext, CreditCardAccountInputDTO creditCardAccountInputDTO) throws FatalException {

    public CreditCardDetailsDTO performPostAmendCardLimitOperation(SessionContext sessionContext, CreditCardAccountInputDTO creditCardAccountInputDTO)

    public AddOnCardHolderDTO linkAddOnCardHolder(SessionContext sessionContext, AddOnCardHolderDTO addOnCardHolderDTO) throws FatalException {

    public TransactionStatus updateBundleBenefits(SessionContext sessionContext, CreditCardAccountInputDTO creditCardAccountInputDTO) throws FatalException {

    public ExternalCardDetailsDTO inquireExternalCardDetails(SessionContext sessionContext, ExternalCardInquiryRequestDTO externalCardInquiryRequestDTO)
}
```

Assume the same are named as *GroovyCreditCardAdapter* which conforms to the interface of the product Credit Card adapter and *GroovyCreditCardAdapterFactory* 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.

The entry in *AdapterFactories.properties* corresponding to the *CreditCardAdapterFactory* would have to be modified to instantiate and return the *GroovyCreditCardAdapterFactory*. In preferences.xml, the custom *GroovyCreditCardAdapterFactory* has overridden the *AdapterFactories*.

Figure 6–6 Modify AdapterFactories.properties for GroovyCreditCardAdapterFactory

```
<Preference name="AdapterFactories" overriddenBy="GroovyCreditCardAdapterFactory"
    PreferencesProvider="com.ofss.fc.infra.config.impl.JavaConstantsConfigProvider"
    parent="" propertyFileName="com.ofss.fc.common.AdapterFactoriesConfig"
    syncTimeInterval="60000" />
```

In preferences.xml, the following has been defined for the Custom *GroovyCreditCardAdapterFactory*.

Figure 6–7 Modify Preferences.xml for GroovyCreditCardAdapterFactory

```
<Preference name="GroovyCreditCardAdapterFactory"
    PreferencesProvider="com.ofss.fc.infra.config.impl.DBBasedPropertyProvider"
    parent="jdbcpreference"
    propertyFileName="select prop_id, prop_value from flx_fw_config_all_b where category_id='GroovyCreditCardAdapterFactory'"
    syncTimeInterval="600000" />
```

Insert a record in table flx_fw_config_all_b to identify a Customized Domain Object in the following manner, where the fully qualified name of the groovy adapter factory can be specified.

```
Insert into FLX_FW_CONFIG_ALL_B(CATEGORY_ID,PROP_ID,PROP_
VALUE,PROP_COMMENTS,OBJECT_VERSION_NUMBER,CREATED_BY,CREATION_
DATE,LAST_UPDATED_BY,LAST_UPDATED_DATE,OBJECT_STATUS_FLAG,FACTORY_
SHIPPED_FLAG) values
('GroovyAdapterFactory','Groovy','com.ofss.fc.groovy.origination.G
roovyCreditCardAdapterFactory','Class for deriving
groovy',1,'ofssuser',SYSDATE,'ofssuser',SYSDATE,'A','Y');
```

The implementation should be packaged and added as part of custom library which the application is referring to and the groovy library will be added in the classpath of the server as below, which will be taken care by deployment team.

Figure 6–8 Add Groovy Library to Classpath

```
if [ "${PRE_CLASSPATH}" != "" ] ; then
   PRE_CLASSPATH="/scratch/app/product/fmw/obpinstall/obp/obp.thirdparty.app.domain/APP-INF/lib/groovy-all-2.3.10.jar${CLASSPATHSEP}${PRE_CLASSPATH}"
   export PRE_CLASSPATH
else
   PRE_CLASSPATH="/scratch/app/product/fmw/obpinstall/obp/obp.thirdparty.app.domain/APP-INF/lib/groovy-all-2.3.10.jar"
   export PRE_CLASSPATH
fi
```

6.3 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.

6.3.1 Custom Adapter Example 1 – DispatchAdapter

The example of DispatchAdapter is further explained here on how to customize the same. This is followed up by an example of customizing a party KYC status check adapter for further clarity and reference.

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 Dispatch Adapter 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 the 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.CustomDispatchAdapterFact
ory
```

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.

6.3.2 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 6–9 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 6–10 Default Implementation of I Party KYC Check Adapter 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 KYCHistoryDTO performOnlineKYCCheck(KYCHistoryDTO kycCheckDTO) throws FatalException {
        if (logger.isLoggable(Level.FINE)) {
            logger.log(Level.FINE, formatter.formatMessage("Entered performOnlineKYCCheck."));
        }
        kycCheckDTO.getAutomaticKYCDetails().setKycStatus(KYCStatus.CONFIRMED);
        kycCheckDTO.getAutomaticKYCDetails().setKycProcessStage(KYCProcessStage.Complete);
        kycCheckDTO.getAutomaticKYCDetails().setKycComments("KYC Staus maintained by Party");
        String bankCode = (String) FCRThreadAttribute.get(FCRThreadAttribute.USER_BANK);
        Date postingDate = new CoreService().fetchBankDates(bankCode).getCurrentDate();
        kycCheckDTO.getAutomaticKYCDetails().setKycDate(postingDate);
        if (logger.isLoggable(Level.FINE)) {
            logger.log(Level.FINE, formatter.formatMessage("Exit performOnlineKYCCheck with KYCStatus:UNCONFIRMED and KYCProcessStage:Pending "));
        }
        return kycCheckDTO;
    }
}
```

```
… 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 6–11 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 ( !isMockEnabled) {
            return new PartyKYCCheckAdapter();
        } else {
            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 KYCDetailsDTO();
                        automaticKYCDetails.setKycStatus(KYCStatus.CONFIRMED);
                        automaticKYCDetails.setKycProcessStage(KYCProcessStage.Complete);
                        automaticKYCDetails.setKycComments("KYC Staus 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;
        }
    } 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.class)));
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.

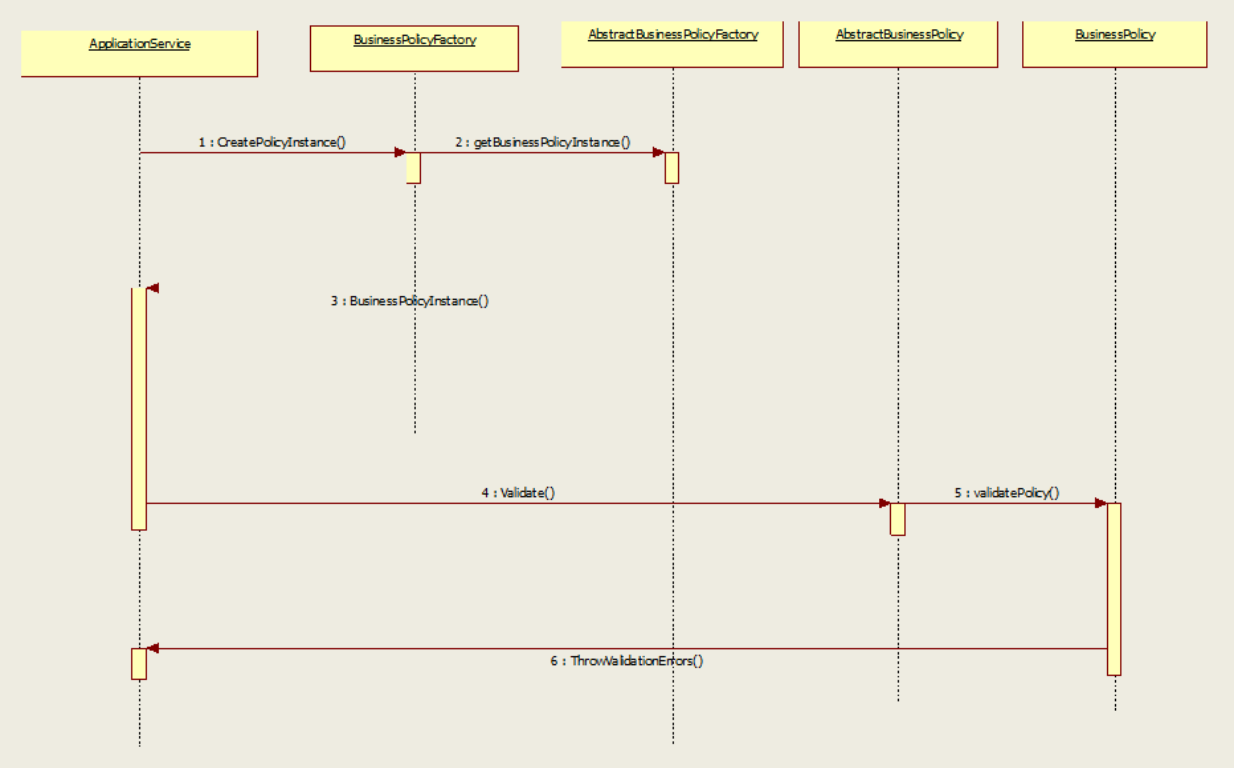
```
Original entry
PARTY_KYC_ADAPTER_
FACTORY=com.ofss.fc.app.adapter.impl.party.PartyKYCCheckAdapterFac
tory
Changed entry
PARTY_KYC_ADAPTER_FACTORY=
com.ofss.fc.app.adapter.impl.party.CustomPartyKYCCheckAdapterFacto
ry
```

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.

7 Business Policy Extension

This chapter describes how custom business policies are implemented in OBP for overriding business validations. Business policy extensions are useful in overriding or extending the existing validations.

Figure 7–1 Business Policy Extension



The sequence diagram above shows a generic view of base implementation of business policy. Wherever business validations are required, application service invokes createPolicyInstance() methods in the business policy factory of the corresponding module. This business policy factory extends to AbstractBusinessPolicyFactory class which is maintained at framework level. CreatePolicyInstance() method in the business policy factory class invokes getBusinessPolicyInstance() method to look for any custom business policy class present in the database. If there is no custom class present, it creates an instance of base business policy class and return it to the invoking application service. Then application service invokes the validate() method in AbstractBusinessPolicy class which in turn invokes validatepolicy() method implemented in base business policy class. All the validation logic is written in this method and it throws validation error if any of the validation conditions fails.

7.1 Base Implementation of Business Policy

The sequence diagram, Figure 7–1, shows a generic view of base implementation of business policy.

For more clarification let's take an example of creditCardDetailsBusinessPolicy implementation. Following are the code snippets of different key methods:

- validate() method in AbstractBusinessPolicy.java

Figure 7–2 validate() method in AbstractBusinessPolicy.java

```
public abstract class AbstractBusinessPolicy {

    /**
     * This attribute indicates validationErrors. it will hold all the validation errors in the list.
     */
    protected List<ValidationError> validationErrors;
    /**
     * This attribute represents the error code for the policy.
     */
    private String policyErrorCode;

    /**
     * This represents the validate method for the policy. This methods needs to be over-riden in every policy.
     */
    public abstract void validatePolicy();

    /**
     * This method needs to be invoked by every caller of the business policy. This method would invoke the validate
     * policy method of the policy and then throw exception in case the policy is not validated for the data provided to
     * the policy.
     */
    public final void validate() {

        validatePolicy();
        checkForError();
    }

    /**
     * This method needs to be invoked by every caller of the business policy. This method would invoke the validate
     * policy method of the policy and then throw exception in case the policy is not validated for the data provided to
     * the policy.
     */
    public final void validate(String policyErrorCode) {

        setPolicyErrorCode(policyErrorCode);
        validatePolicy();
        checkForError();
    }
```

- validatePolicy() in creditCardBusinessPolicy.java

Figure 7–3 validatePolicy() in creditCardBusinessPolicy.java

```
public CreditCardDetailsBusinessPolicy(CreditCardDetailsBusinessPolicyDTO creditCardBusinessPolicyDTO) {

    this.creditCardBusinessPolicyDTO = creditCardBusinessPolicyDTO;
}

/**
 * Validate Policy. Includes,<br>
 * <ul>
 * <li>Validate Card Details</li>
 * </ul>
 * @param sessionContext
 * @param creditCardDetailsDTO
 * @exception
 * <ul><li>{@link CardErrorConstants#CARD_NUMBER_NOT_PRIMARY}</li>
 * <li>{@link CardErrorConstants#CARD_NOT_ACTIVE}</li>
 * <li>{@link CardErrorConstants#BLOCKED_FOR_LIMIT_CHANGE_ADD_ON_CARD}</li>
 * <li>{@link CardErrorConstants#CARD_STATUS_CLOSED}</li>
 * </ul>
 */
@Override
public void validatePolicy() {

    if (this.creditCardBusinessPolicyDTO.getCreditCardDetailsDTO() != null) {
        validateCardDetails(this.creditCardBusinessPolicyDTO.getSessionContext(), this.creditCardBusinessPolicyDTO.getCreditCardDetailsDTO());
    }
    if (this.creditCardBusinessPolicyDTO.getCreditCardBasicDetailsDTO() != null) {
        validateCardDetails(this.creditCardBusinessPolicyDTO.getSessionContext(), this.creditCardBusinessPolicyDTO.getCreditCardBasicDetailsDTO());
    }
}
```

7.2 Extending Business Policy

Custom implementation of business policy is achieved by defining a preference for customBusinessPolicy in preferences.xml which represents a query to the FLX_FW_CONFIG_ALL_B table in the database. To

override a base business policy, class name of the custom business policy with the policy code is inserted into the table. As a guideline, the custom business class should extend the product base business policy, to inherit the product base implementation. Base code already handles the fetching of custom class, if any, from the table. If customization of a policy is not required then query returns null and base business policy is implemented.

7.3 Configuration

For custom business policy implementation following configuration steps are required to be followed:

1. Add a preference for custom business policy in preferences.xml.

   Figure 7–4 Add a preference for custom business policy in preferences.xml

   ```
   <Preference name="CustomBusinessPolicyPreferences"
       PreferencesProvider="com.ofss.fc.infra.config.impl.DBBasedPropertyProvider"
       parent="jdbcpreference"
       propertyFileName="select prop_id, prop_value from flx_fw_config_all_b where category_id = 'CustomBusinessPolicy'"
       syncTimeInterval="600000" />
   ```

2. Add an entry in FLX_FW_CONFIG_ALL_B table in database with custom class name and policy code.

   ```
   INSERT INTO FLX_FW_CONFIG_ALL_B (PROP_ID,CATEGORY_ID,PROP_
   VALUE,FACTORY_SHIPPED_FLAG,PROP_COMMENTS,SUMMARY_TEXT,CREATED_
   BY,CREATION_DATE,LAST_UPDATED_BY,LAST_UPDATED_DATE,OBJECT_
   STATUS_FLAG,OBJECT_VERSION_NUMBER) VALUES ('FC_CC_BP_
   001','CustomBusinessPolicy','com.ofss.fc.module.originationGr
   oovy.CreditCardDetailsBusinessPolicyGroovy','Y','This is
   accessed from
   AbstractBusinessPolicyFactory.getCustomBusinessPolicyNameTDS'
   ,'','ofssuser',to_date('09/05/2016 11:25:30', 'dd/mm/rrrr
   hh:mi:ss'),'ofssuser',to_date('09/05/2016 11:25:30',
   'dd/mm/rrrr hh:mi:ss'),'A',1);
   ```

7.4 Extensions Using Groovy

Groovy is a lightweight, dynamically typed object-oriented programming language. It has got similarities with java and can run on jvm platform. Groovy class provides the functionalities for interacting with a java program so can be efficiently used as extensions for customization purpose.

In addition to the configurations mentioned above, add the groovy-all-2.3.10.jar in the classpath of weblogic server in setDomain.sh file, which will be done by deployment team. No other specific configuration is needed.

Following is the snippet of a groovy custom business policy class implemented for creditCardDetails validations:

Figure 7–5 Extensions using Groovy

```
package com.ofss.fc.module.originationGroovy
import com.ofss.fc.app.card.service.credit.policy.CreditCardDetailsBusinessPolicy

 public class CreditCardDetailsBusinessPolicyGroovy extends AbstractBusinessPolicy{
     private static final String THIS_COMPONENT_NAME = CreditCardDetailsBusinessPolicyGroovy.class.getName()



    private CreditCardDetailsBusinessPolicyDTO creditCardBusinessPolicyDTO= null




    public CreditCardDetailsBusinessPolicyGroovy(){
        super();
    }

    public CreditCardDetailsBusinessPolicyGroovy(CreditCardDetailsBusinessPolicyDTO creditCardBusinessPolicyDTO) {
                    super(creditCardBusinessPolicyDTO);
                this.creditCardBusinessPolicyDTO = creditCardBusinessPolicyDTO
            }

@Override
  public void   validatePolicy(){

// validation logic goes here..
  }
  public CreditCardDetailsBusinessPolicyDTO getCreditCardBusinessPolicyDTO() {
      return creditCardBusinessPolicyDTO;
  }


  public void setCreditCardBusinessPolicyDTO(CreditCardDetailsBusinessPolicyDTO creditCardBusinessPolicyDTO) {
      this.creditCardBusinessPolicyDTO = creditCardBusinessPolicyDTO;
  }


}
```

8 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:

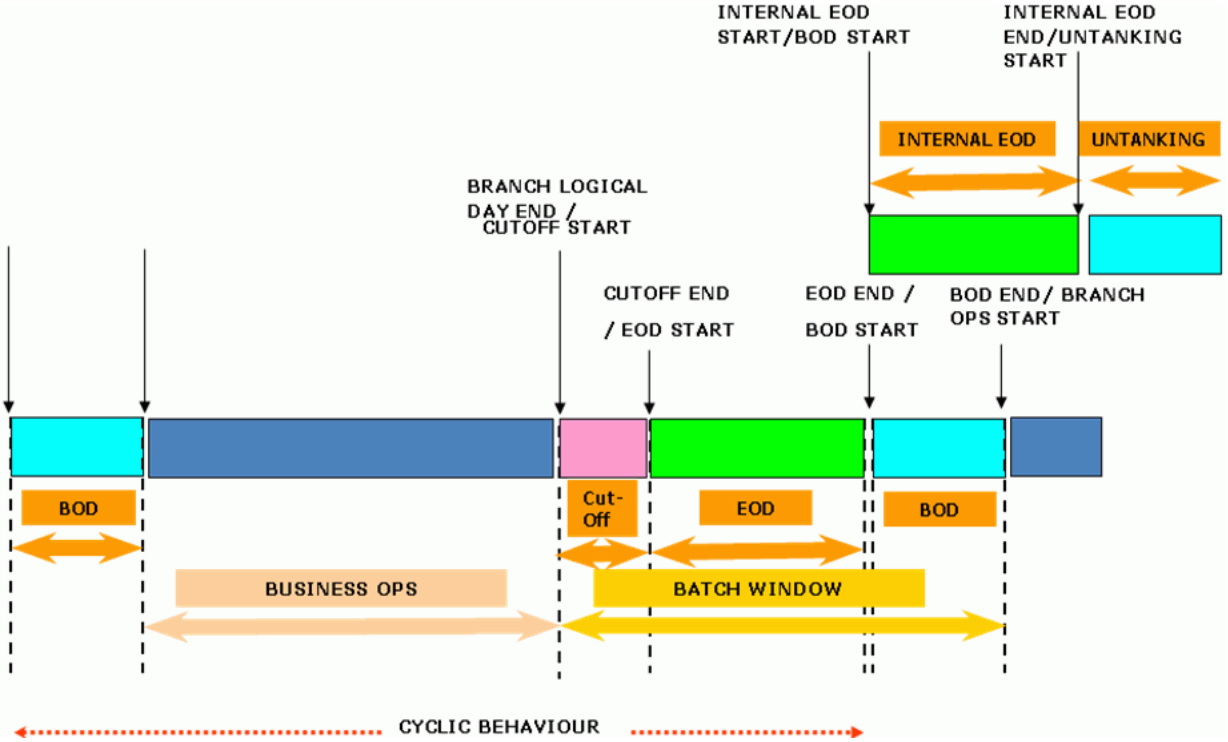
- Beginning of Day (BOD)
- Cut-off
- End of Day (EOD)
- Internal EOD
- Statement Generation
- Customer Communication

Additional categories can also be configured as per the requirement.

8.1 Typical Business Day in OBP

The following graphic describes a typical business day in OBP:

Figure 8–1 Business Day in OBP



8.2 Overview of Categories

This topic describes the categories in OBP Batch Processes.

8.2.1 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

8.2.2 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.

8.2.3 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

8.2.4 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

8.2.5 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

8.2.6 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

8.3 Batch Framework Architecture

This section describes the architecture of the Batch Framework.

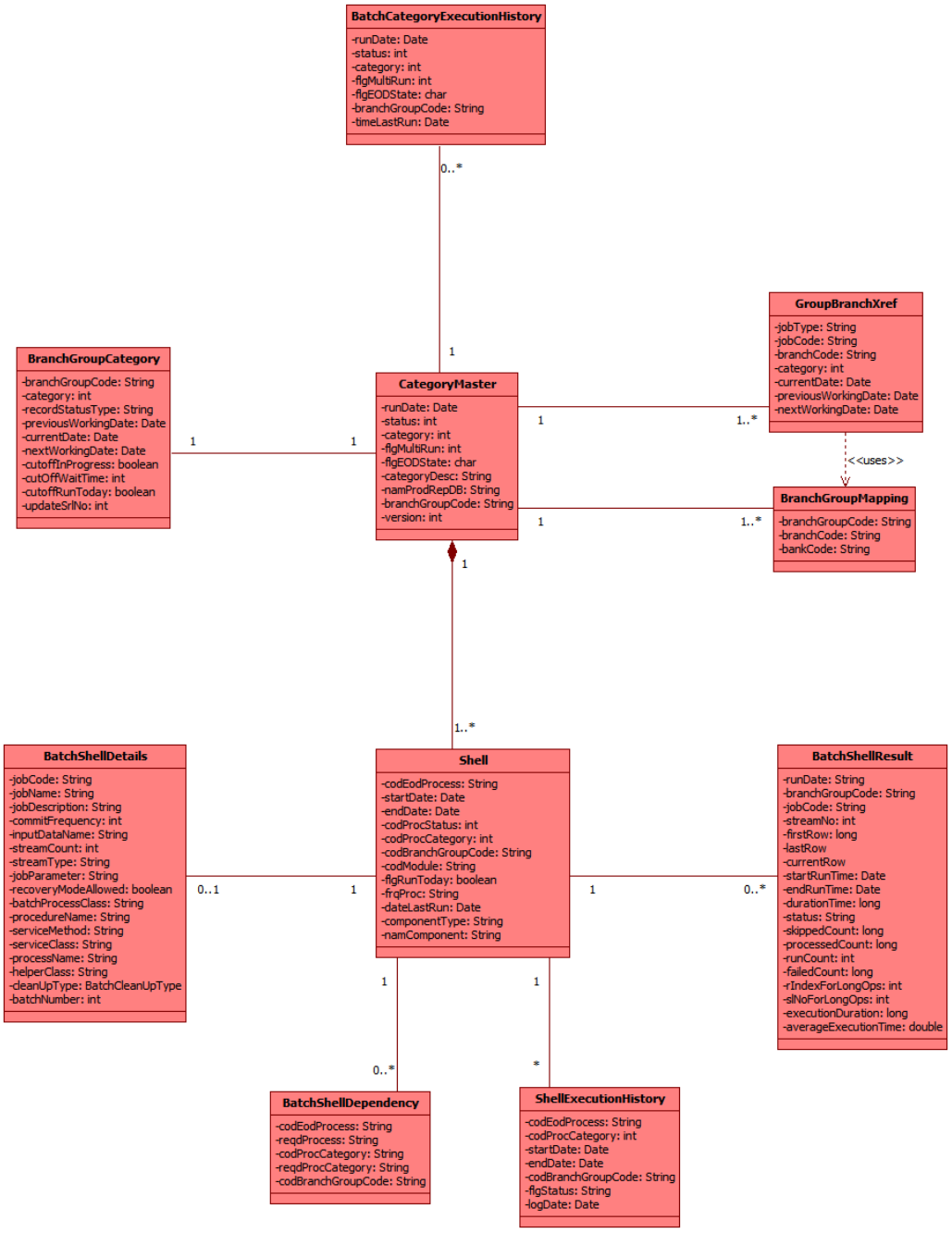
8.3.1 Static View

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.

Figure 8–2 Batch Framework Architecture - Static View



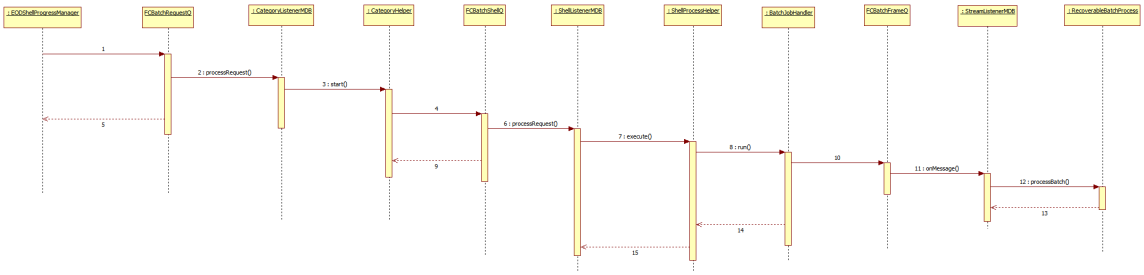
8.3.2 Dynamic View

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.

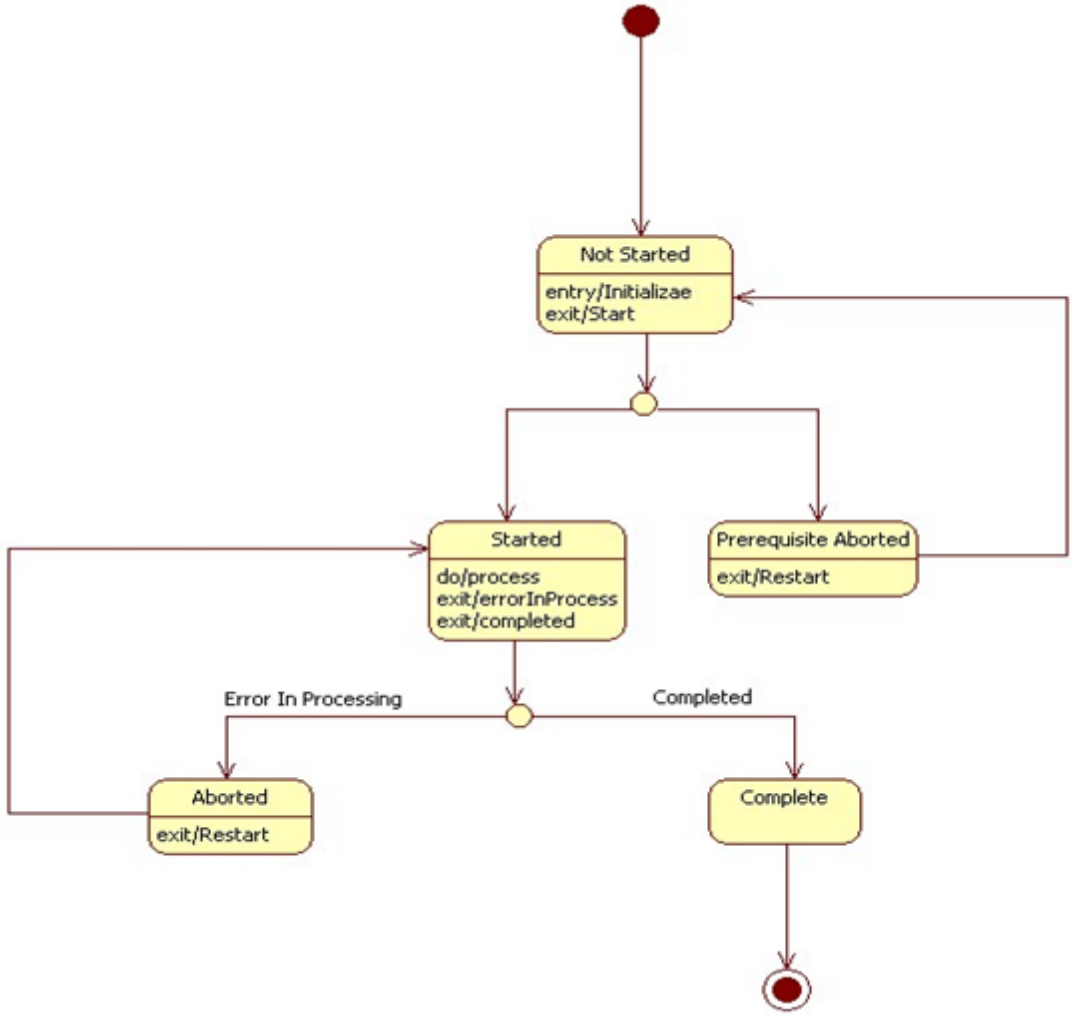
Figure 8–3 Dynamic View Sequence Diagram



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 8–4 State Diagram of a Shell



8.4 Batch Framework Components

This section describes the batch framework components.

8.4.1 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.

8.4.2 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.

8.4.3 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.

For Example:

```
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.

8.4.4 Database Components

The Database Server houses the following components:

Table 8–1 Database Server 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>	This driver table contains the batch execution details for the particular action.
flx_<module>_actions_ b	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.

8.5 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.

8.5.1 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.

   Table 8–2 FLX_BATCH_JOB_CATEGORY_MASTER

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.

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 8–3 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.

Table 8–4 FLX_BATCH_JOB_CATEGORY_DEPEND

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.

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.

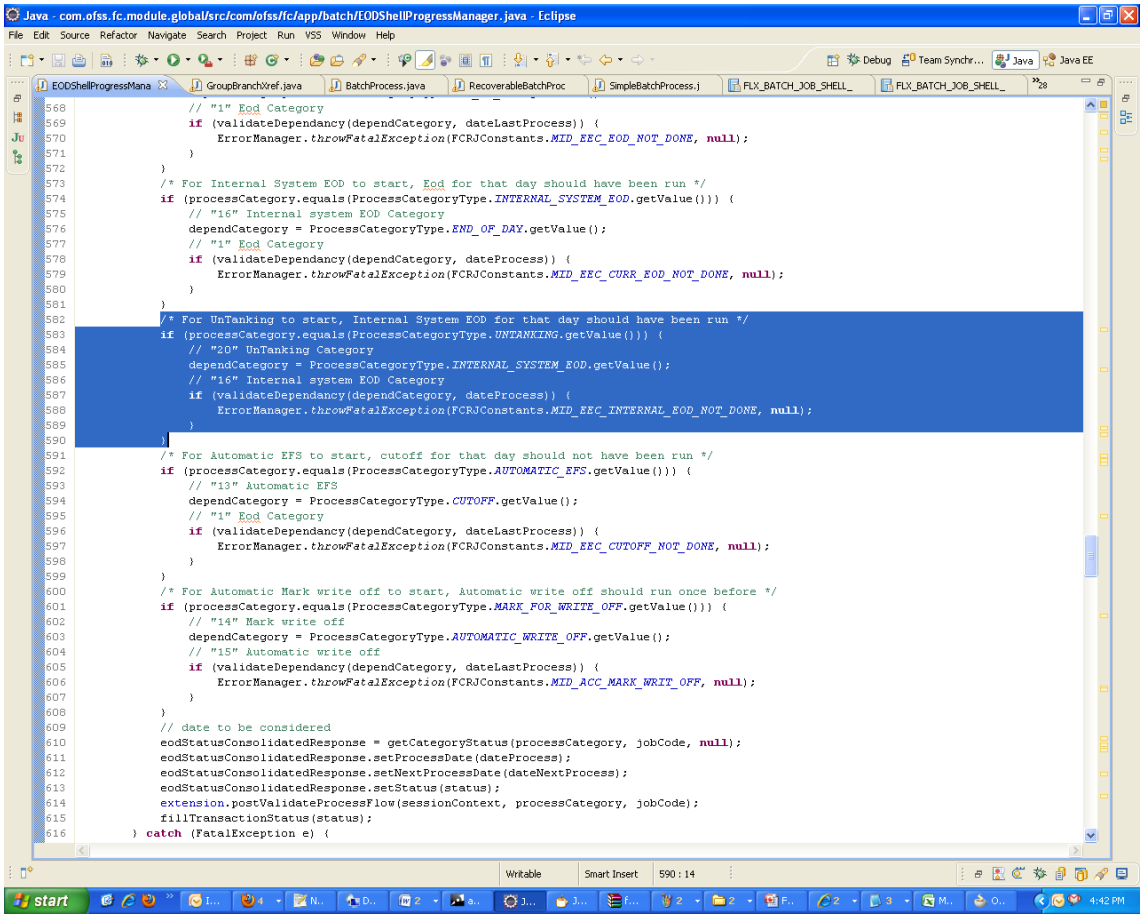
6. Add category code in the property file:

   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 8–5 Creation of New Category



8.5.2 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**.

Table 8–5 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.
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). com.ofss.fc.bh.batch.BatchFrameworkShellHelper - java bean based shell com.ofss.fc.bh.batch.BatchReportShellBean - procedure based shell for reports 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.

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

   This table contains the following parameters;

Table 8–6 FLX_BATCH_JOB_SHELL_DTLS

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.
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.

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

Table 8–7 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:

Table 8–8 Driver Table

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.
ERROR_DESC	Attribute specifies error description. This will populated by the batch framework in case the record aborts.
ACTION_ SEQUENCE	In case of service action as ActionSetProcessor, the batch execution is done through the executor framework defined in the action table of the

Column	Description
(Optional)	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>	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.

Note

DATE_RUN, SEQ, BRANCH_GROUP_CODE columns are part of the Unique key. For example, 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 8–9 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. For example, 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-mappings.cfg** file for the new hbm entities created for BatchData. For example, account-mappings.cfg.xml
10. 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.

8.5.3 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.
2. 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);
```

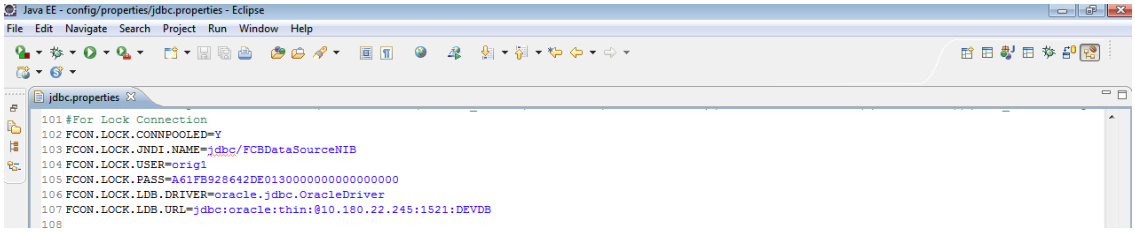
```
RETURN 95;
END;/
```

8.5.4 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 8–6 Population of Other Parameters



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

Figure 8–7 Population of Other Parameters - General Tab

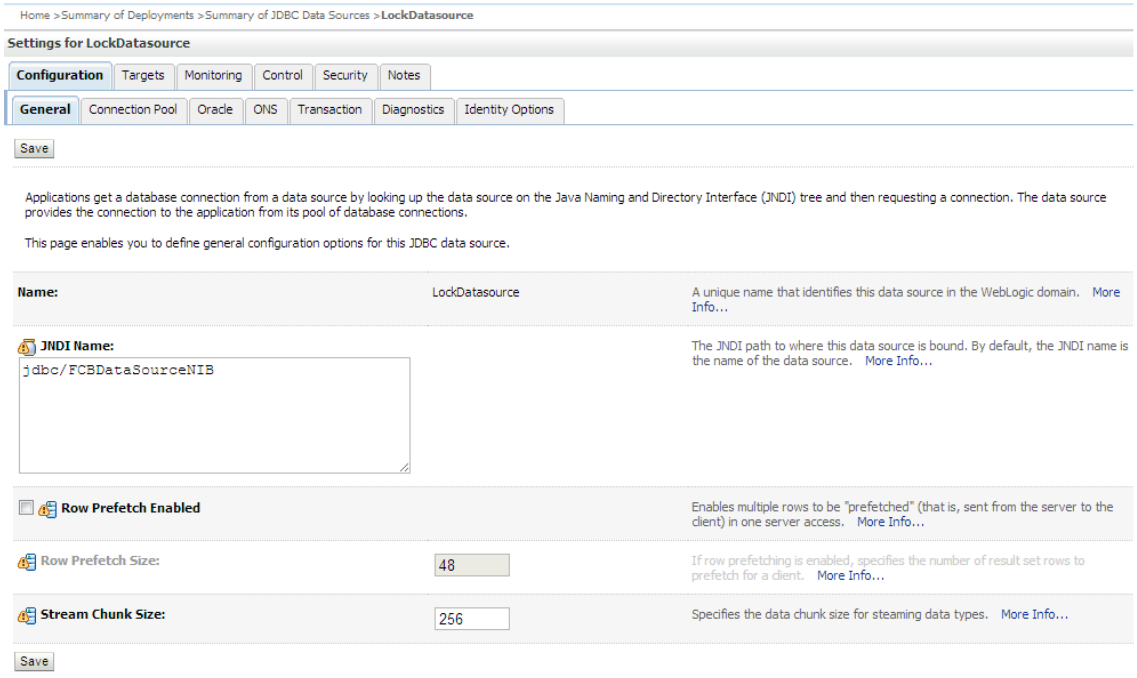
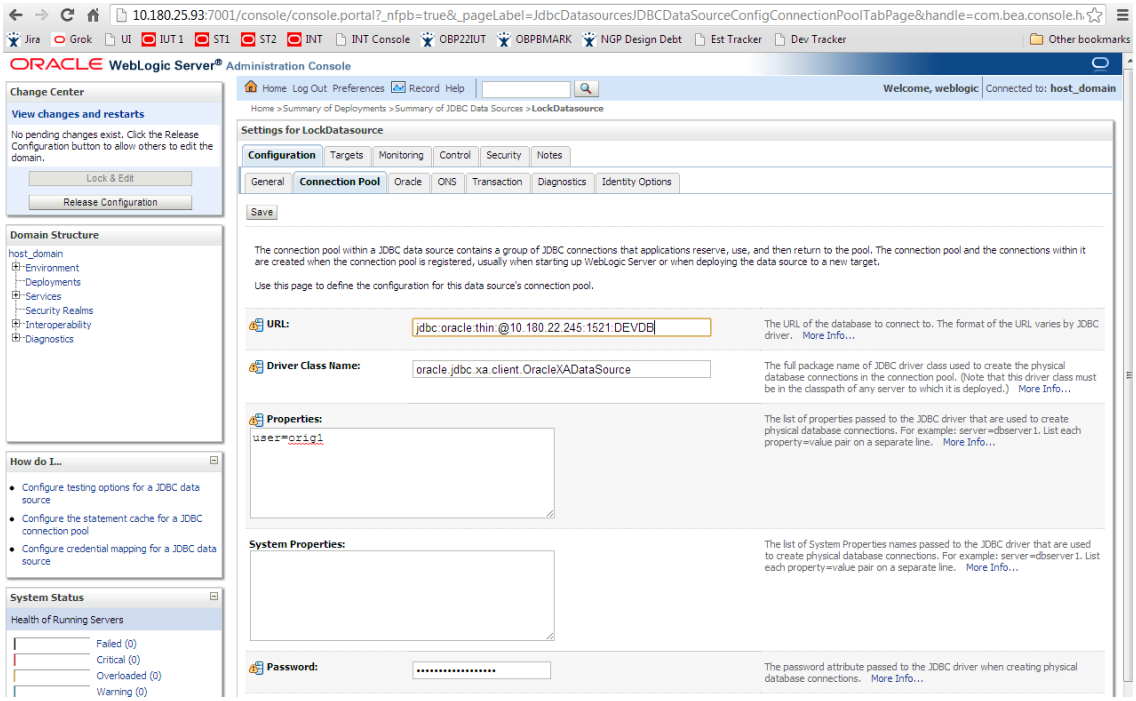


Figure 8–8 Population of Other Parameters - Connection Pool



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 8–9 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 8–10 Population of Other Parameters - Specify Data

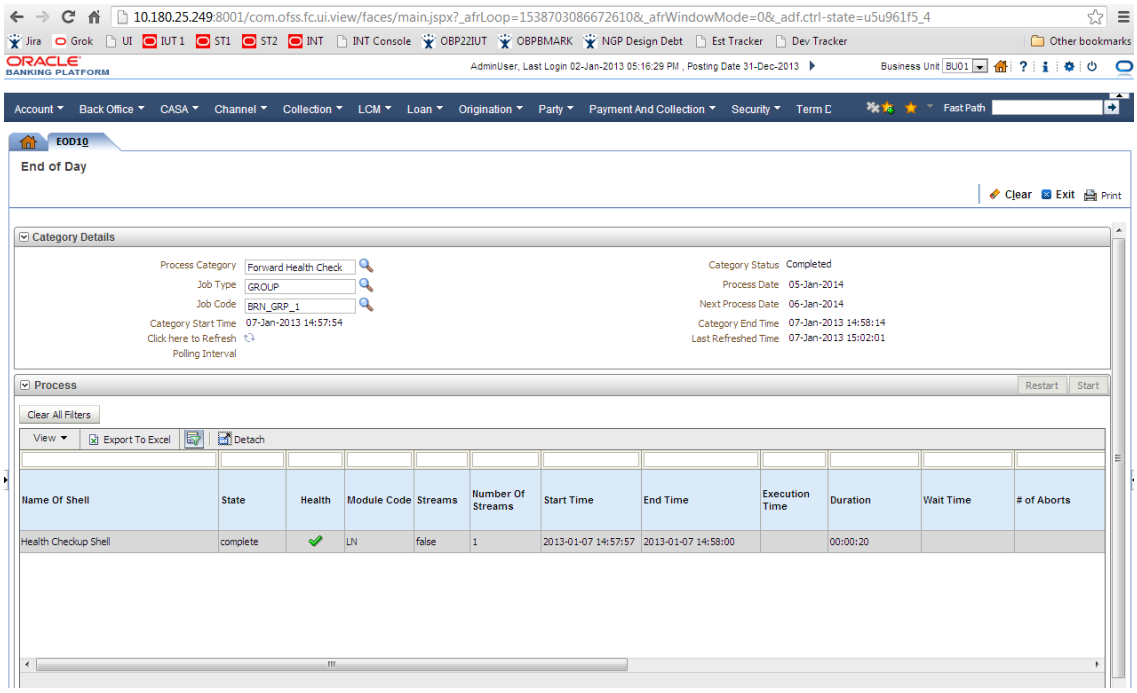
```
#Used in Clustered env for pinning connection to stream
FCON.BATCH1.CONNPOOLED=Y
FCON.BATCH1.JNDI.NAME=jdbc/FCBDataSourceN1
FCON.LOCK.USER=orig1
FCON.LOCK.PASS=A61FB928642DE0130000000000000000
FCON.LOCK.LDB.DRIVER=oracle.jdbc.OracleDriver
FCON.LOCK.LDB.URL=jdbc:oracle:thin:@10.180.22.245:1521:DEVDB
#Used in Clustered env for pinning connection to stream
FCON.BATCH2.CONNPOOLED=Y
FCON.BATCH2.JNDI.NAME=jdbc/FCBDataSourceN2
FCON.LOCK.USER=orig1
FCON.LOCK.PASS=A61FB928642DE0130000000000000000
FCON.LOCK.LDB.DRIVER=oracle.jdbc.OracleDriver
FCON.LOCK.LDB.URL=jdbc:oracle:thin:@10.180.22.245:1521:DEVDB

```

8.6 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 8–11 Batch Execution



9 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.

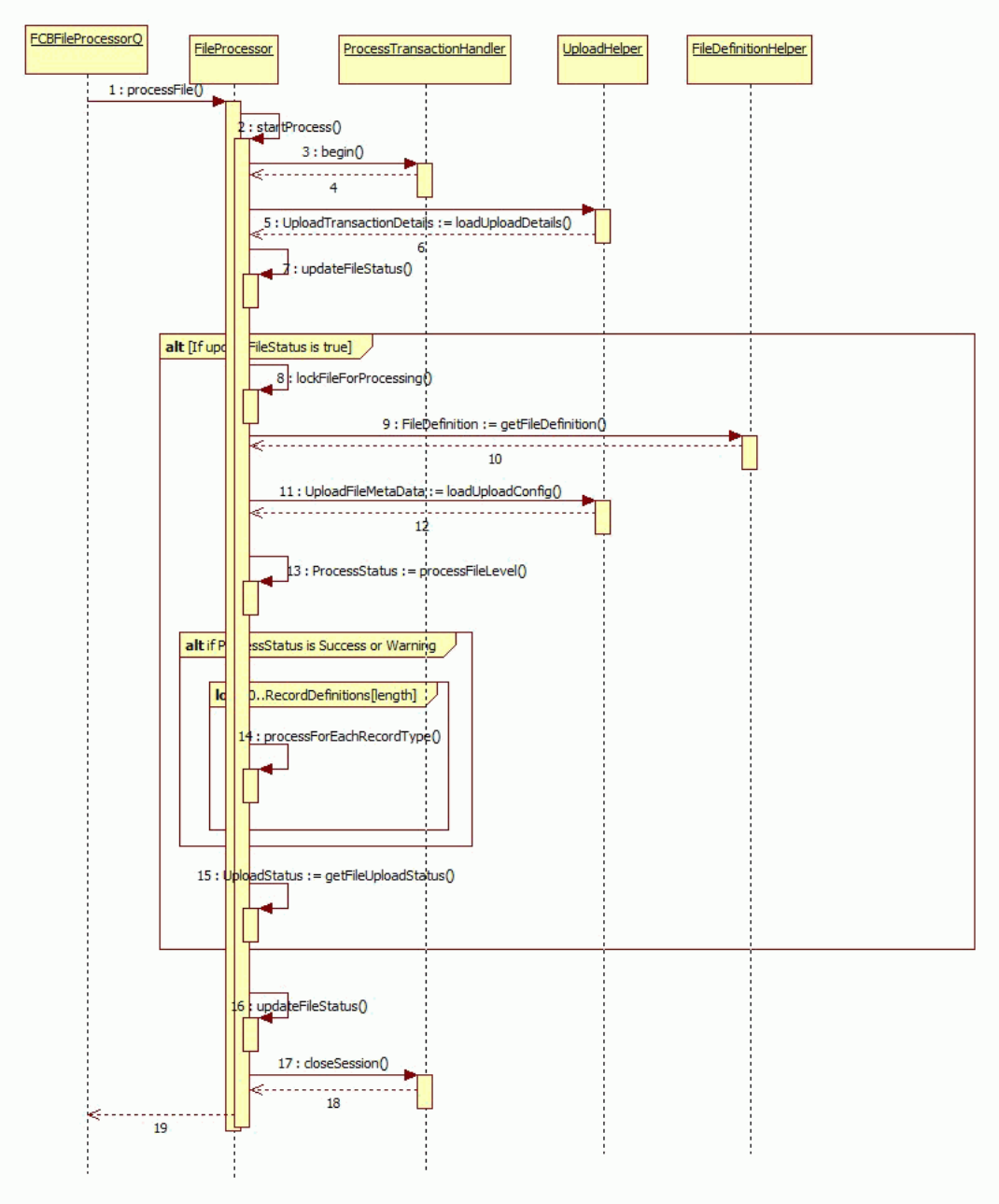
In OBP, file upload and file processing are two independent processes where the upload of file is done as one process and the processing on the uploaded data is done as another process. Every upload provides a unique file ID for the uploaded file. The processing is then done for each uploaded file and 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 9–1 Uploaded Data File Processing Framework



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

9.1 Configuration

The configuration part of the uploaded file processing requires definition of the following components that needs to be defined for the processing on the uploaded file.

9.1.1 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

Table 9–1 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 recordType is successfully processed.

- FLX_EXT_<<Process>>_HEADERRECDTO e.g. FLX_EXT_SALCREDIT_HEADERRECDTO
- FLX_EXT_<<Process>>_DETAILRECDTO e.g. FLX_EXT_SALCREDIT_DETAILRECDTO

The file Id 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.

Table 9–2 Mandatory Fields in 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.

- 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.

Table 9–3 FLX_EXT_FILE_PARAMS

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

Column Name	Description
	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.
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.

- 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 9–4 FLX_BATCH_JOB_SHELL_DTLS

Columns	Description	Sample Values
COD_SHELL	A unique code for batch shell. For example, 'upld_batch_shell_ <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

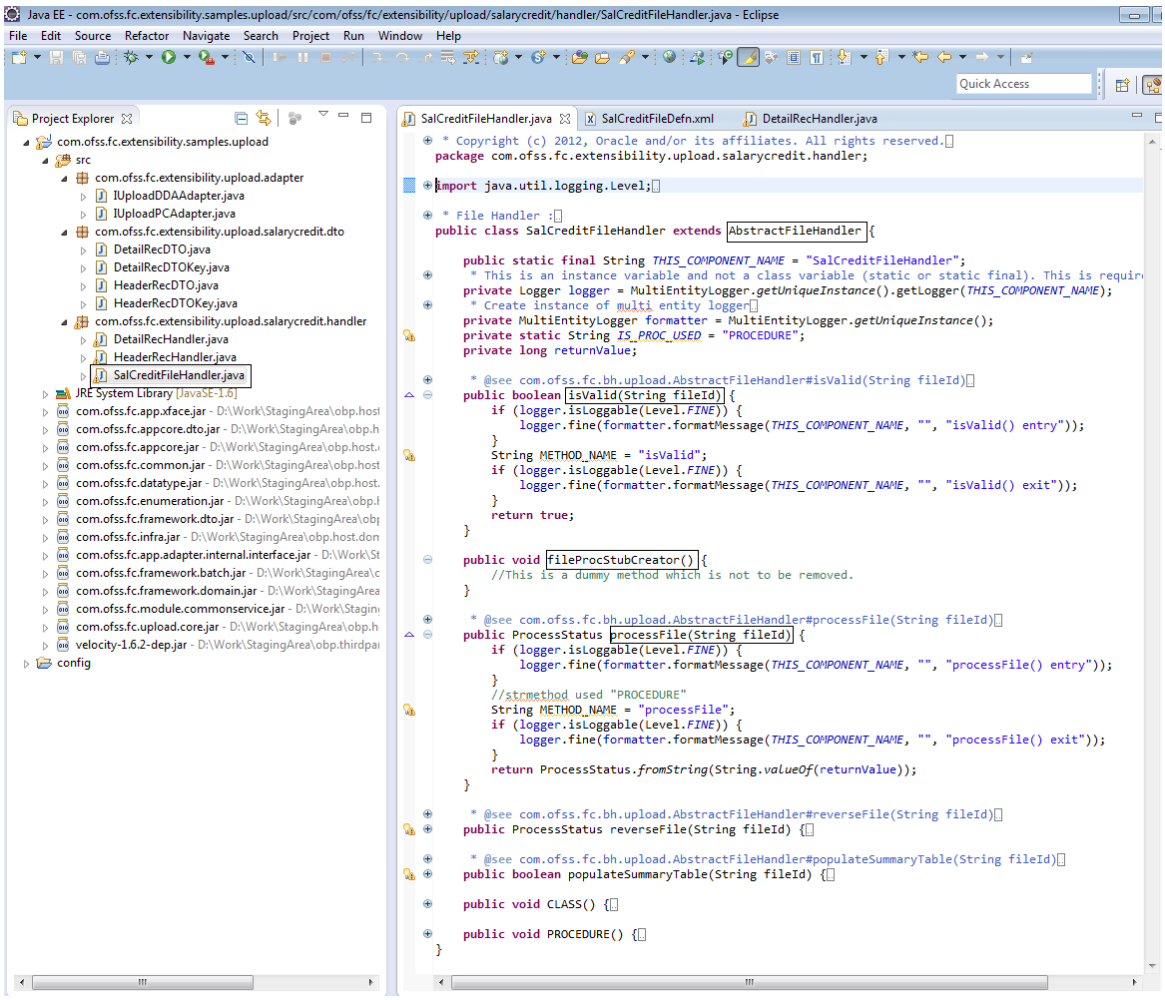
Columns	Description	Sample Values
STREAM_ COUNT	No of streams for the batch shell. Applicable only for StreamType as 'S' or 'R'	2
INPUT_DRV_ NAME	Fully classified class name mapped to the driver table	com.ofss.fc.entity.upload.AbstractRecordDTO
INPUT_SHELL_ PARAM	Name for the shell parameter	AbstractRecordDTO
SERVICE_ CLASS_NAME	Fully classified class name - starting point of the business logic execution	com.ofss.fc.upload.processor.batch.BatchRecordProcessor
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
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.GEFUBatchJobHelper
BATCH_NO	Batch number for the shell	1

9.1.2 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 9–2 File Handlers

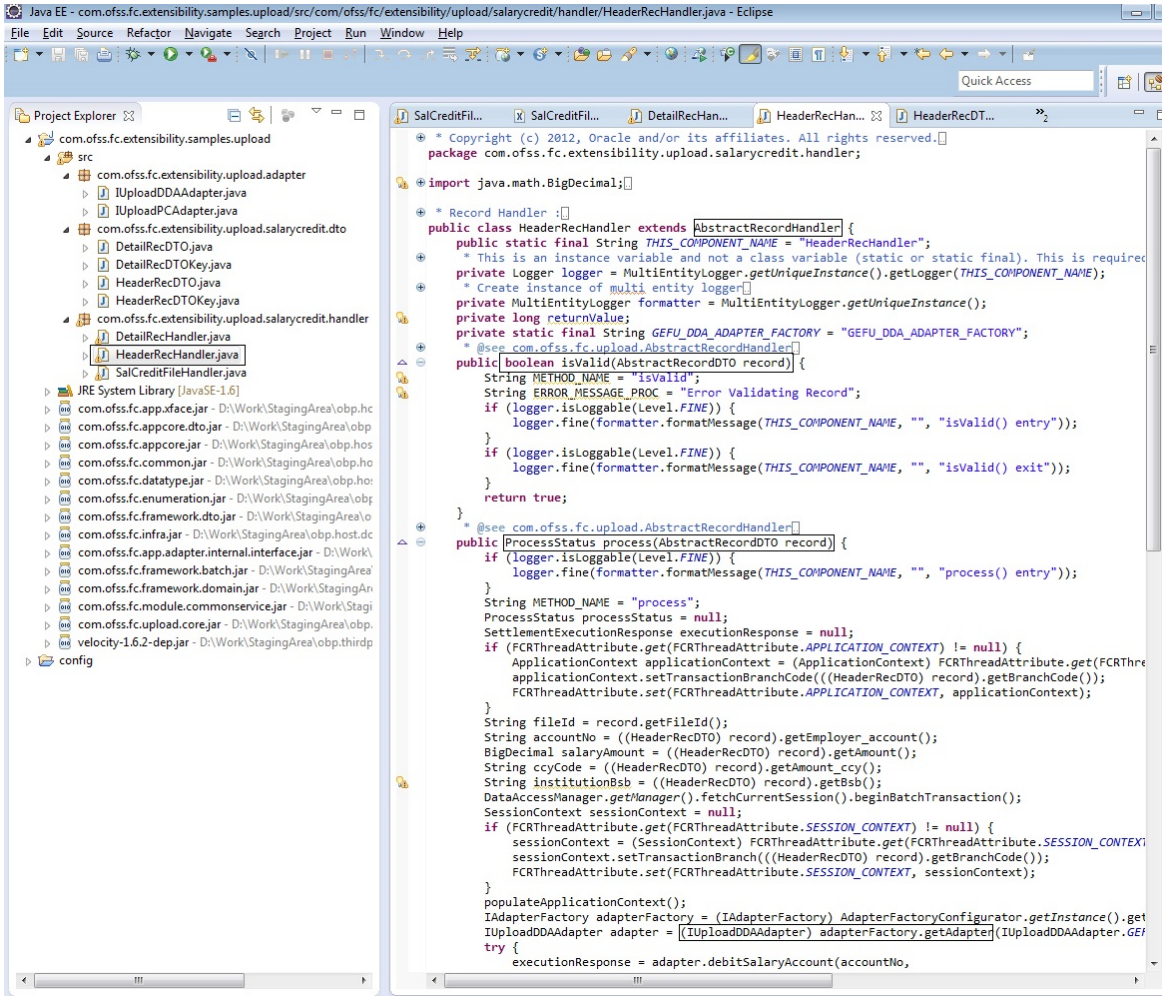


9.1.3 Record Handlers for Both Header and Details

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.

Figure 9–3 Record Handlers for Both Header and Details



9.1.4 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 9–4 DTO and Keys Classes for Both Header and Details - HeaderRecDTOKey

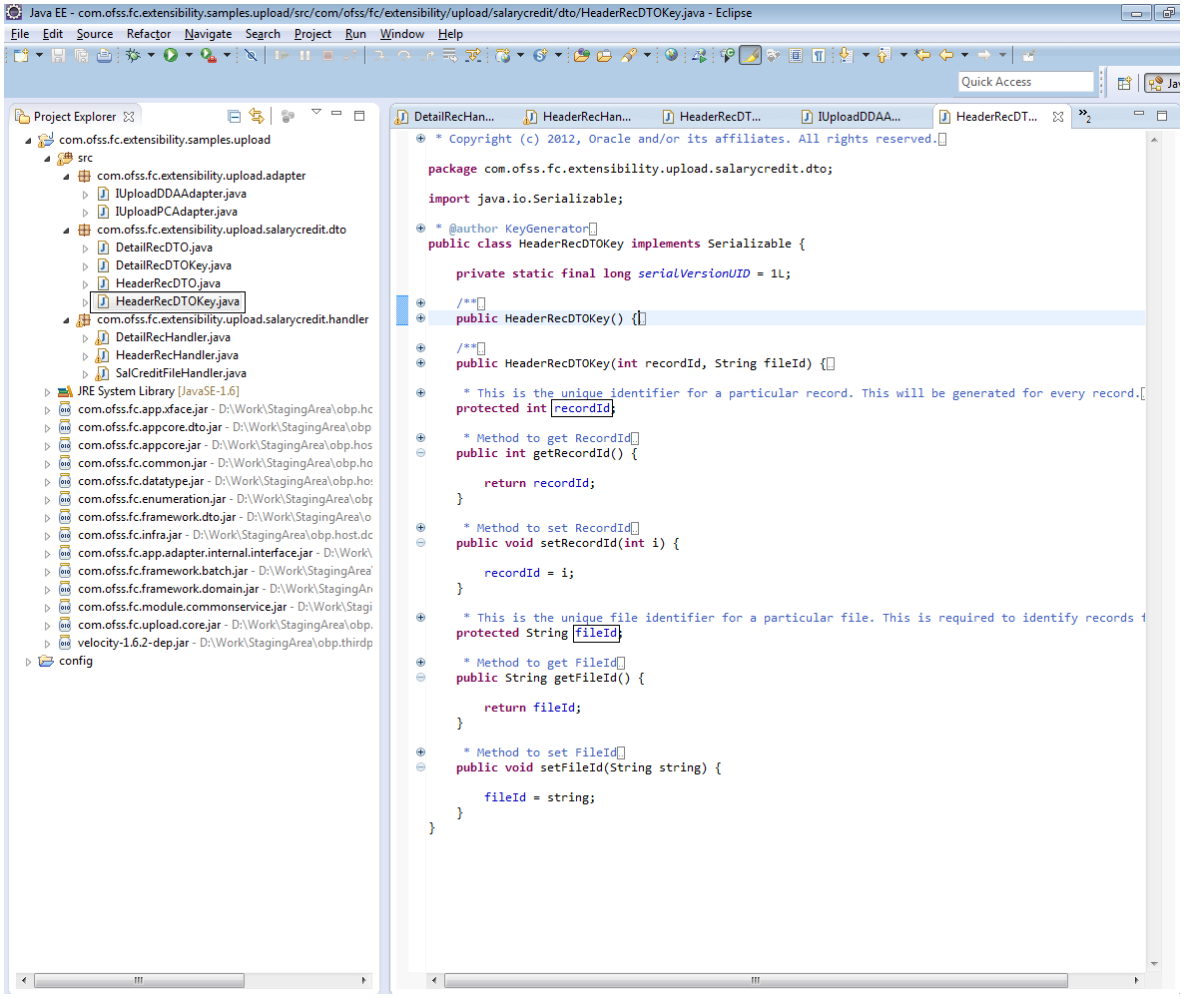
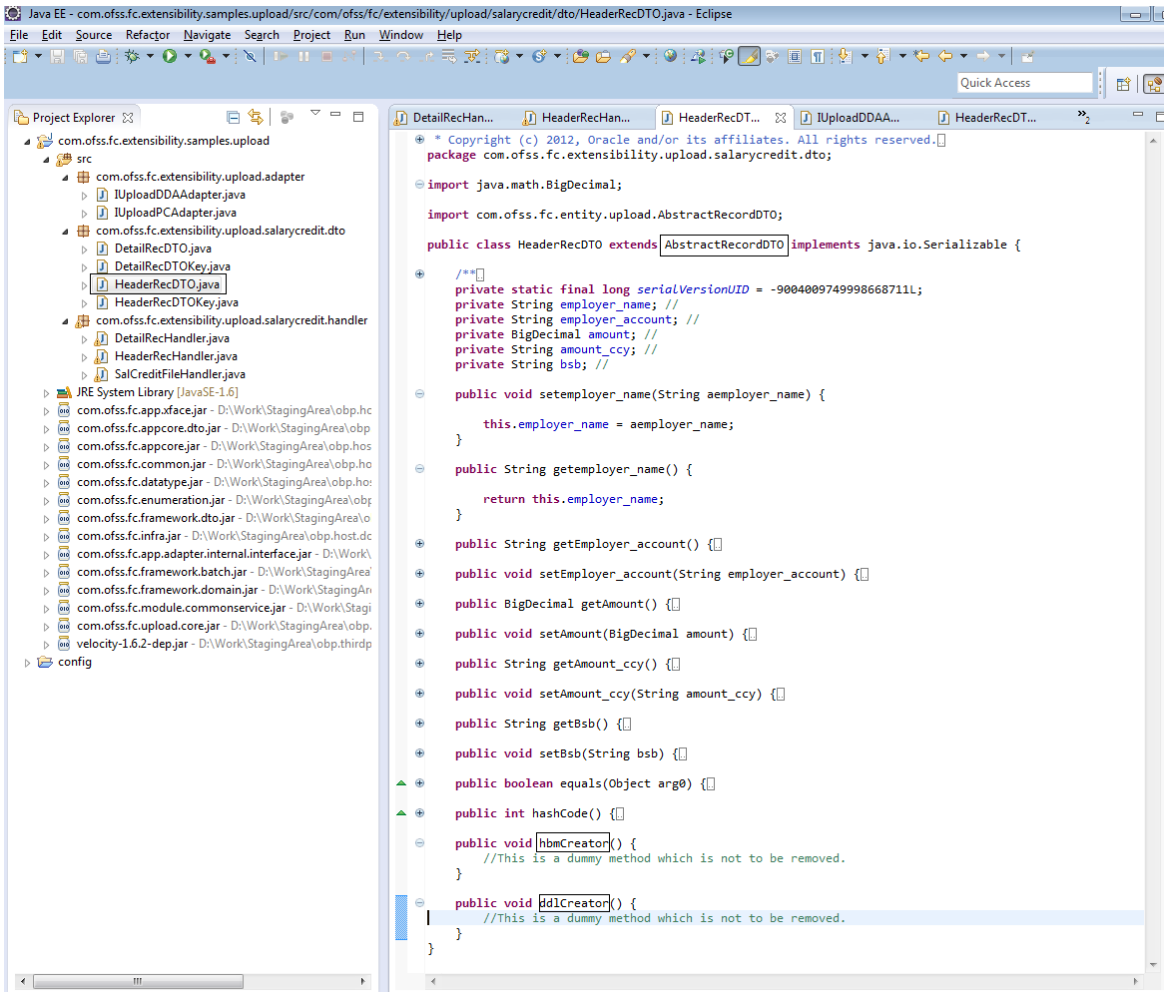


Figure 9–5 DTO and Keys Classes for Both Header and Details - AbstractDTORec



9.1.5 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:

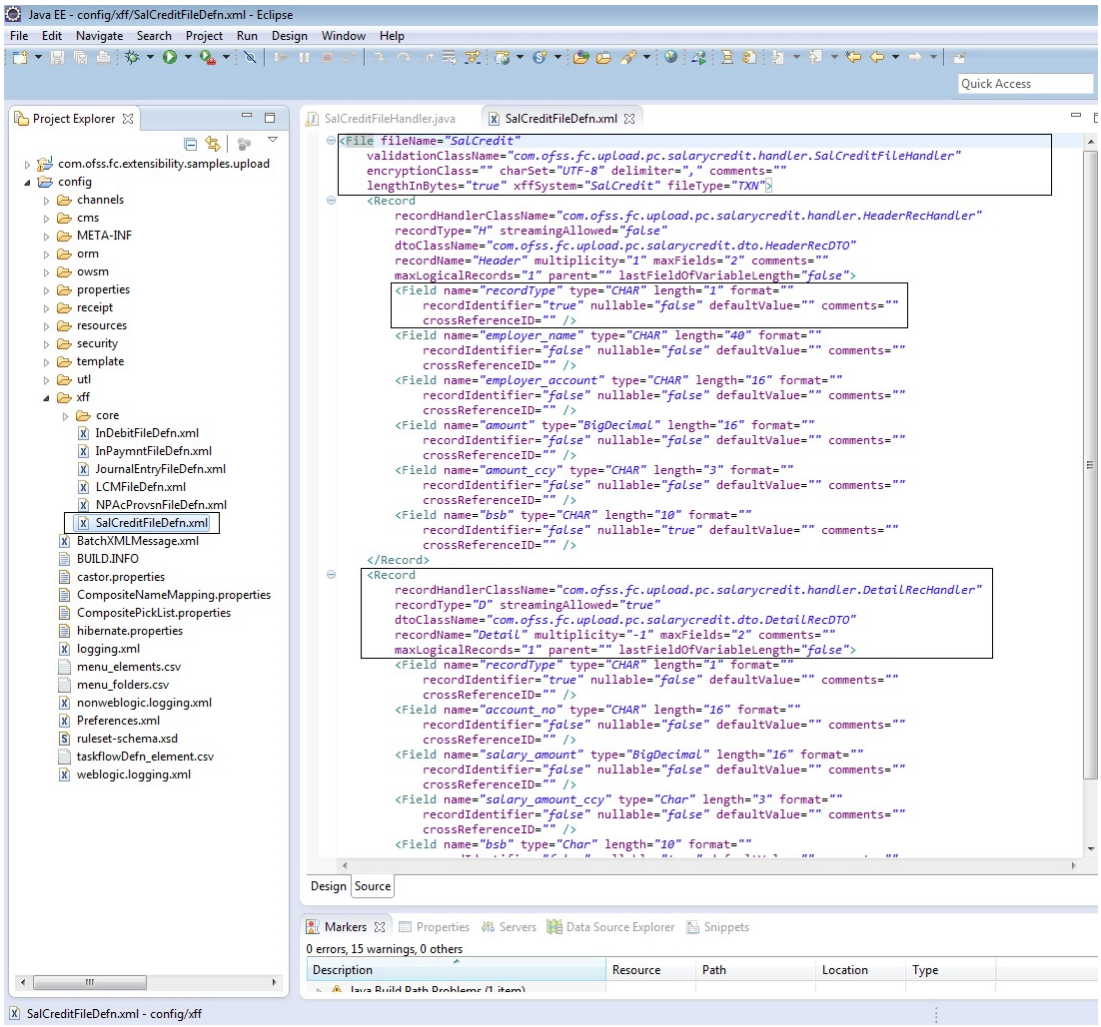
- One 'File' element can have many 'Record' elements, depending upon the number of recordType present for this upload.
- One 'Record' element can have many 'Field' elements, depending upon the number of fields present for this recordType of upload.

Table 9–5 XXF File Definition XML

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.

Elements	Attributes	Description
	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.
	defaultValue	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.

Figure 9–6 XXF File Definition XML



9.2 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.

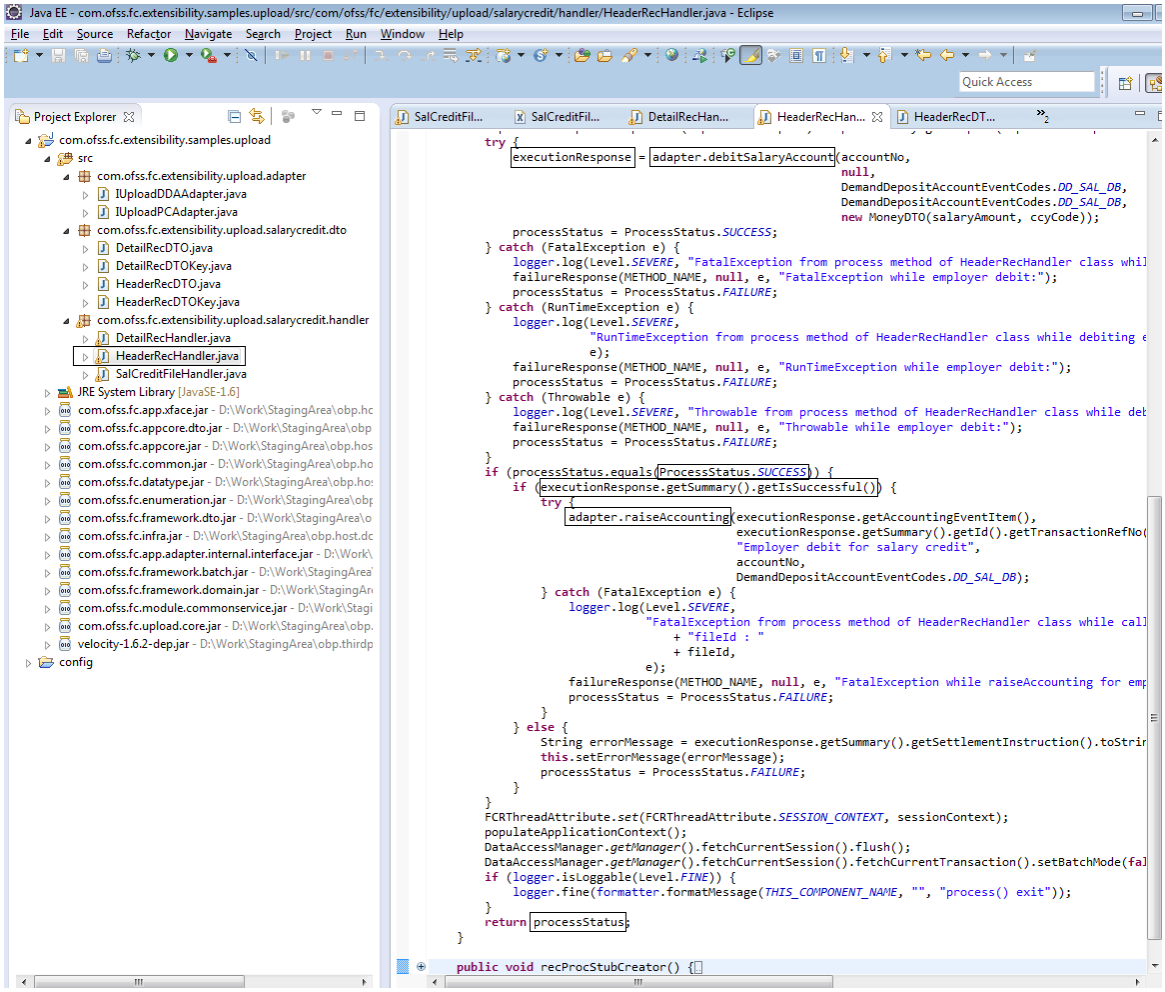
As per the process, the headers are processed first and then the details records. Each and every record is processed individually. As soon as a file is picked for processing, its status is changed to InProgress so that the same file is not picked by any other process for processing. Individual records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

9.2.1 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.

```
<Response1>=Adapter1.<method call>(<method parameters>)
If(<Validation on Response1>) {
<Response2>=Adapter2.<method call>(<method parameters containing
Response1>) }
Example:
executionResponse = adapter.debitSalaryAccount()
if(executionResponse.getSummary().getIsSuccessful()) {
adapter.raiseAccounting(); }
```

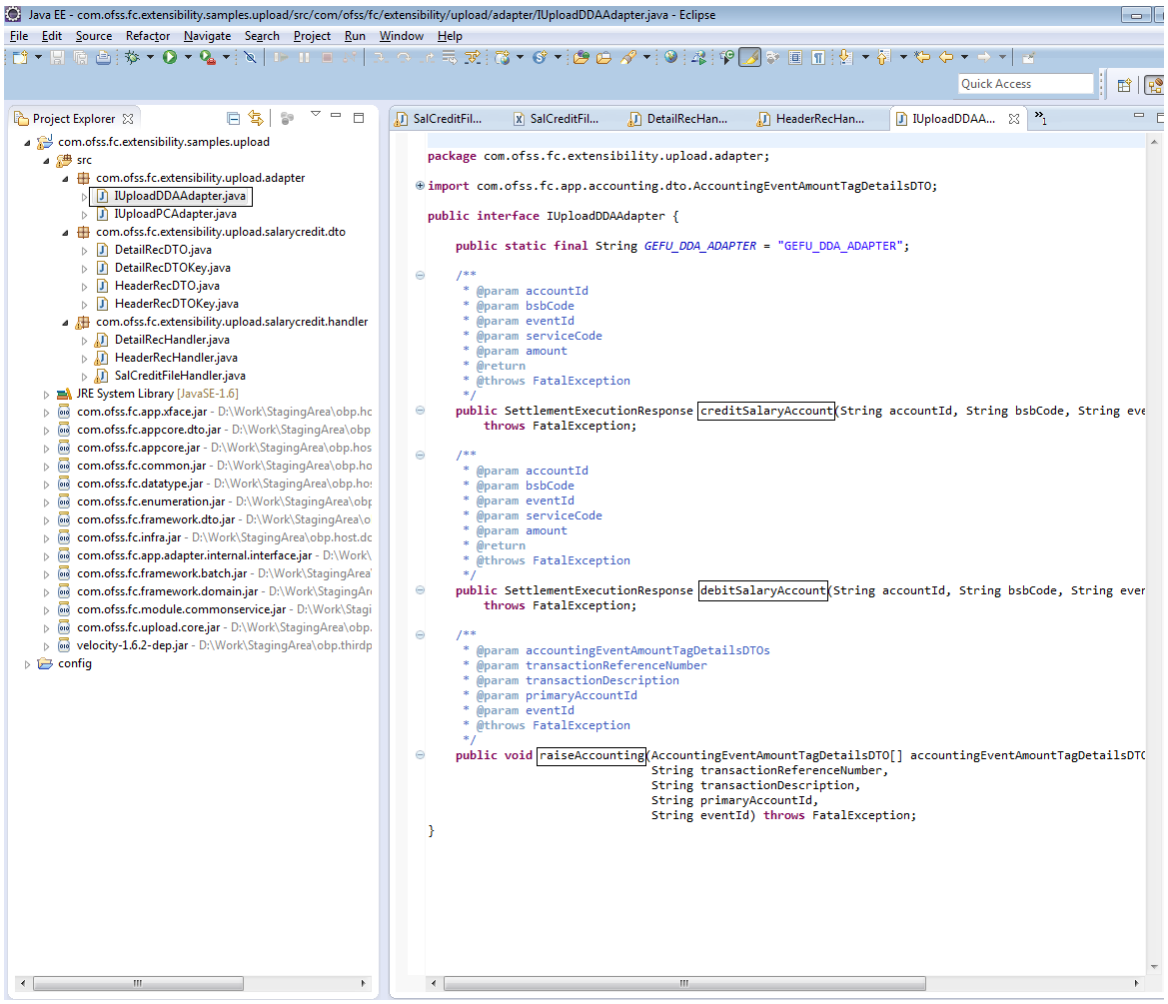
Figure 9–7 API Calls in Adapters



9.2.2 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 9–8 Processing Adapter



9.3 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.

Table 9–6 Process Status

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.

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.

9.4 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;
```

10 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.

10.1 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.

10.1.1 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:

Table 10–1 FLX_EP_ACT_B

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. ContactPointApplicationService.updateContactPoint.dndInfo'
TXT_ACT_ NAME	Activity name	'ContactPointApplicationService.updateContactPoint.dndInfo'
TXT_ACT_ DESC	Meaningful description of the activity	'DND Info Change'

Column Name	Use	Example
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'

Sample script for Activity Record:

Figure 10–1 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, OBJECT_VERSION_NUMBER, OBJECT_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');
```

10.1.2 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.

10.1.3 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:

Table 10–2 FLX_EP_EVT_B

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'

Column Name	Use	Example
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

Sample script for Event Record:

Figure 10–2 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);
```

10.1.4 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:

Table 10–3 FLX_EP_ACT_EVT_B

Column Name	Use	Example
COD_ACT_ID	The unique activity id as specified in the activity table	'com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint.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'

Sample script for Activity Event Mapping Record:

Figure 10–3 Activity Event Mapping Record

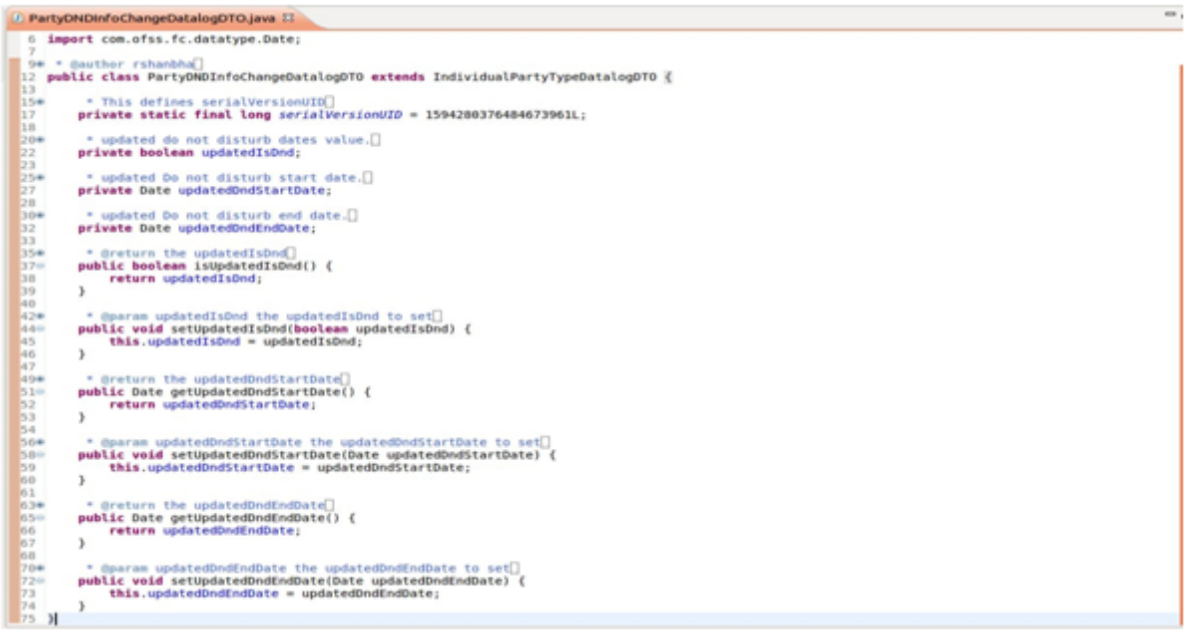
```
--for insertion of activity - event mapping
DELETE FROM FLX_EP_ACT_EVT_B WHERE COD_ACT_ID =
'com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint.dndInfo' AND COD_EVENT_ID = 'PI_UPD_DND_INFO';
INSERT INTO FLX_EP_ACT_EVT_B (COD_ACT_ID, COD_EVENT_ID, TXT_ACT_EVT_DESC,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

10.1.5 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 10–4 Activity Log DTO



10.1.6 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.

1. Once you have plugin you need to set properties in preferences in windows tab for Service Publisher, Service Deployer and Workspace Path.
2. Go to your DTO class and right-click that class and click the following : *Oracle Banking Platform -> Generate DTO Metadata*.

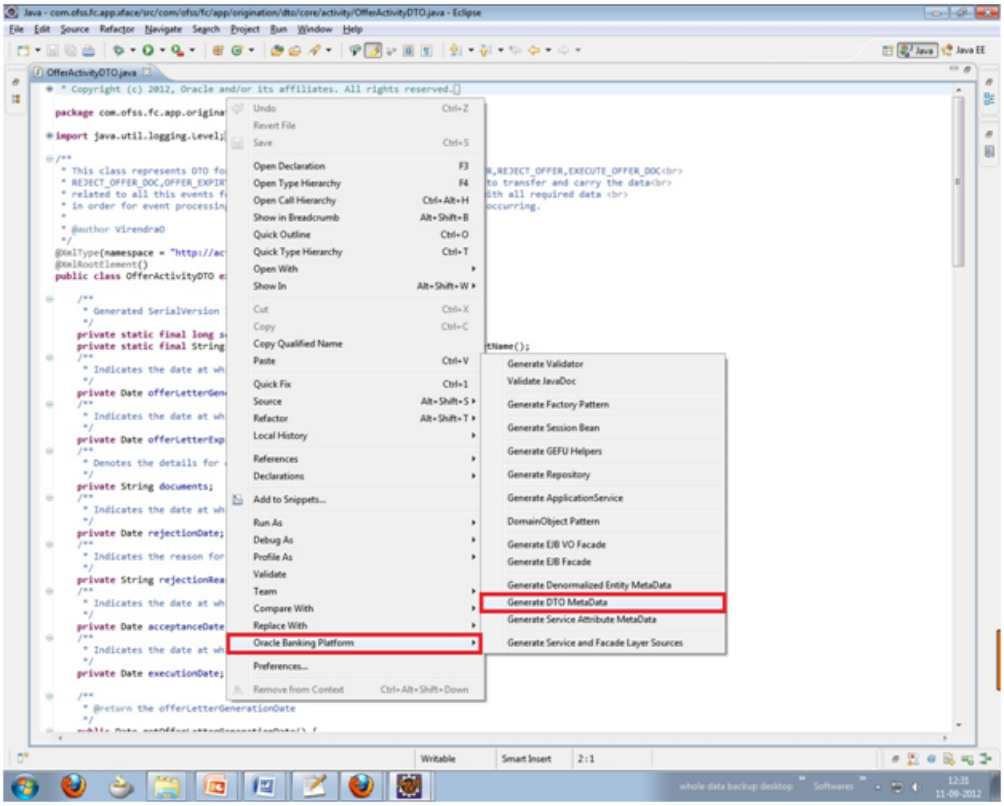
   This will generate the insert scripts for following two tables:

   - 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 10–5 Metadata Generation



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.

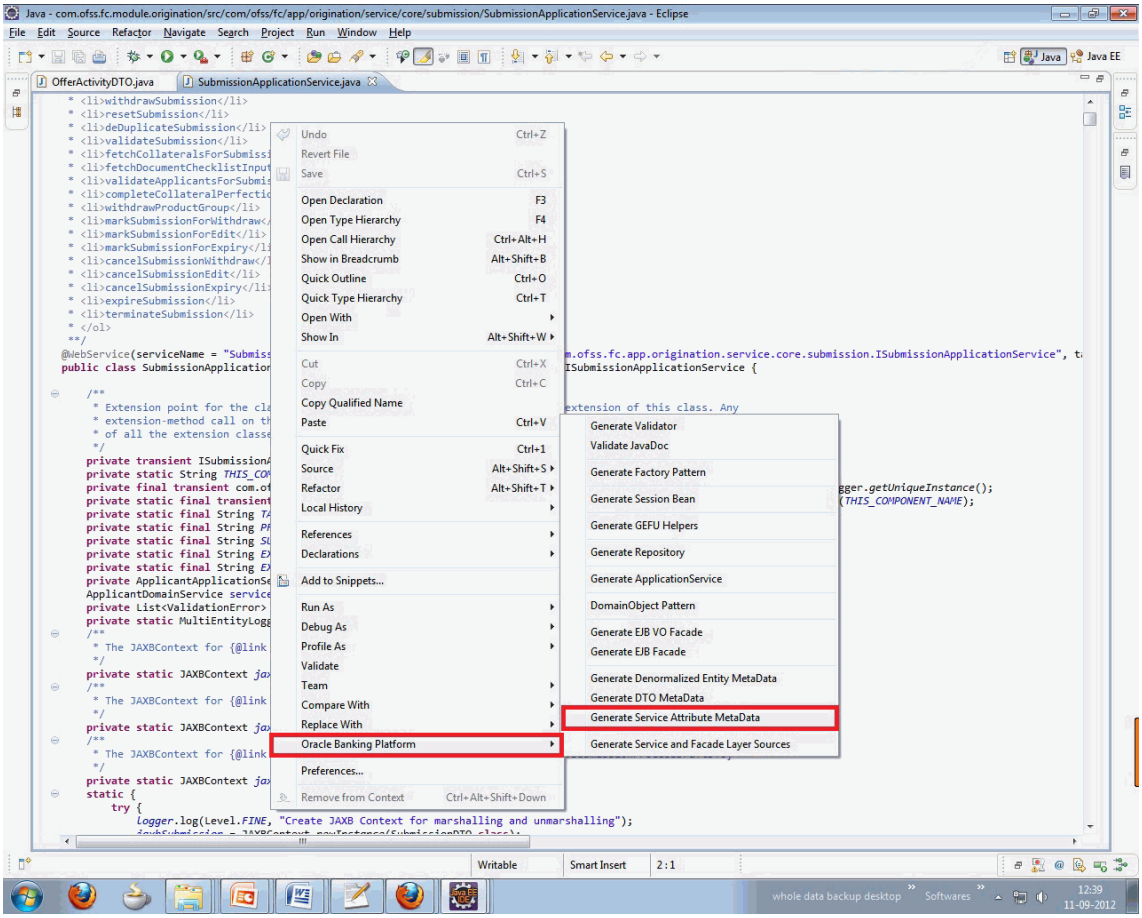
1. Right-click that service and select *Oracle Banking Platform -> Generate Service Attribute Metadata*.
2. In this case also insert scripts will be generate in same location as metadata attributes.

   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 10–6 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 10–4 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. AccountAddressLinkageApplicationService.createAccountAddressLinkage. Alert.Party.Address.City.DTO
TYP_DATA_SRC	Indicates the Data Source(entity/input/DTO) for the Attribute of the Resource
COD_ATTR_ID	This field indicates the Fact Code. For example, Alert.Party.Address.City

Column	Description
COD_SERVICE_ID	This field indicates the Activity ID. For example, com.ofss.fc.app.account.service.accountaddresslinkage.AccountAddressLinkageApplicationService.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()

10.1.7 Alert Message Template Maintenance

User will maintain template format and template ID to be used for the alerts definition.

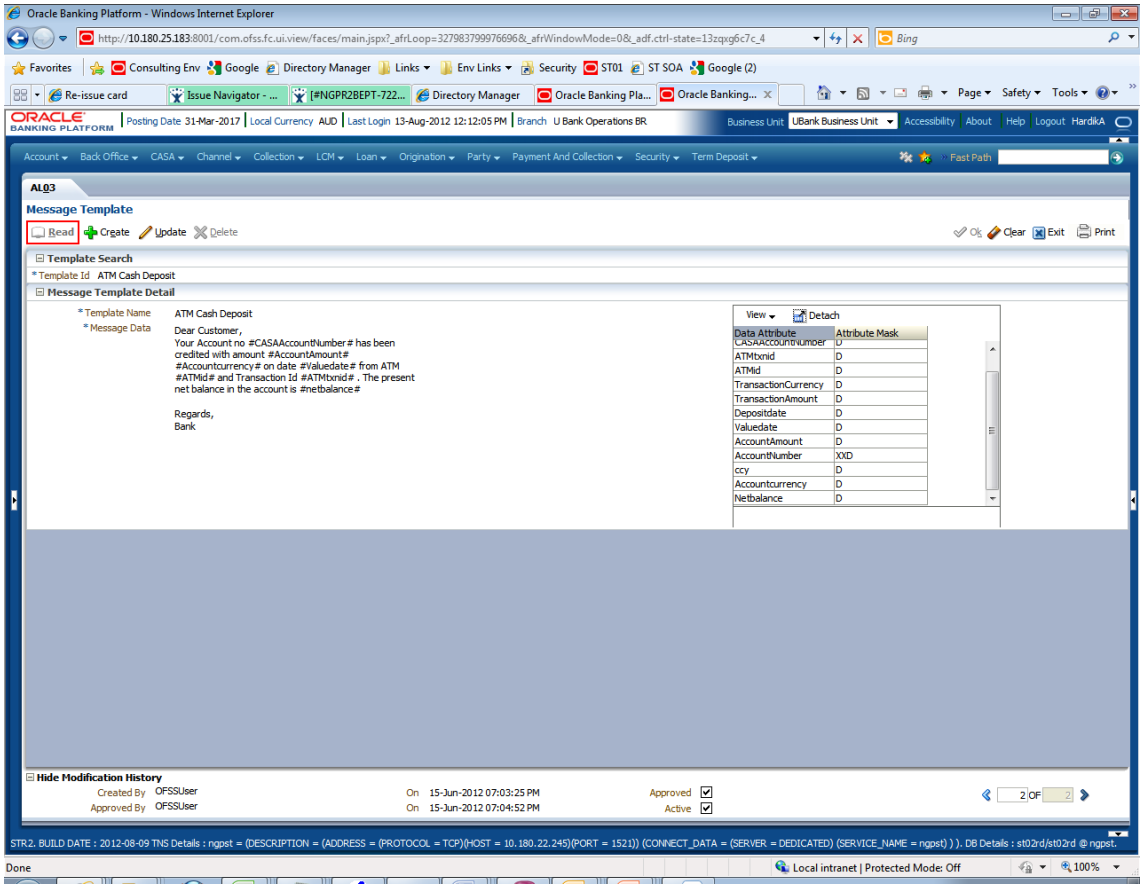
These messages need to be defined only if the same template is going to be used for multiple events. Else there is a provision to define the message template during the definition of the alert itself.

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.

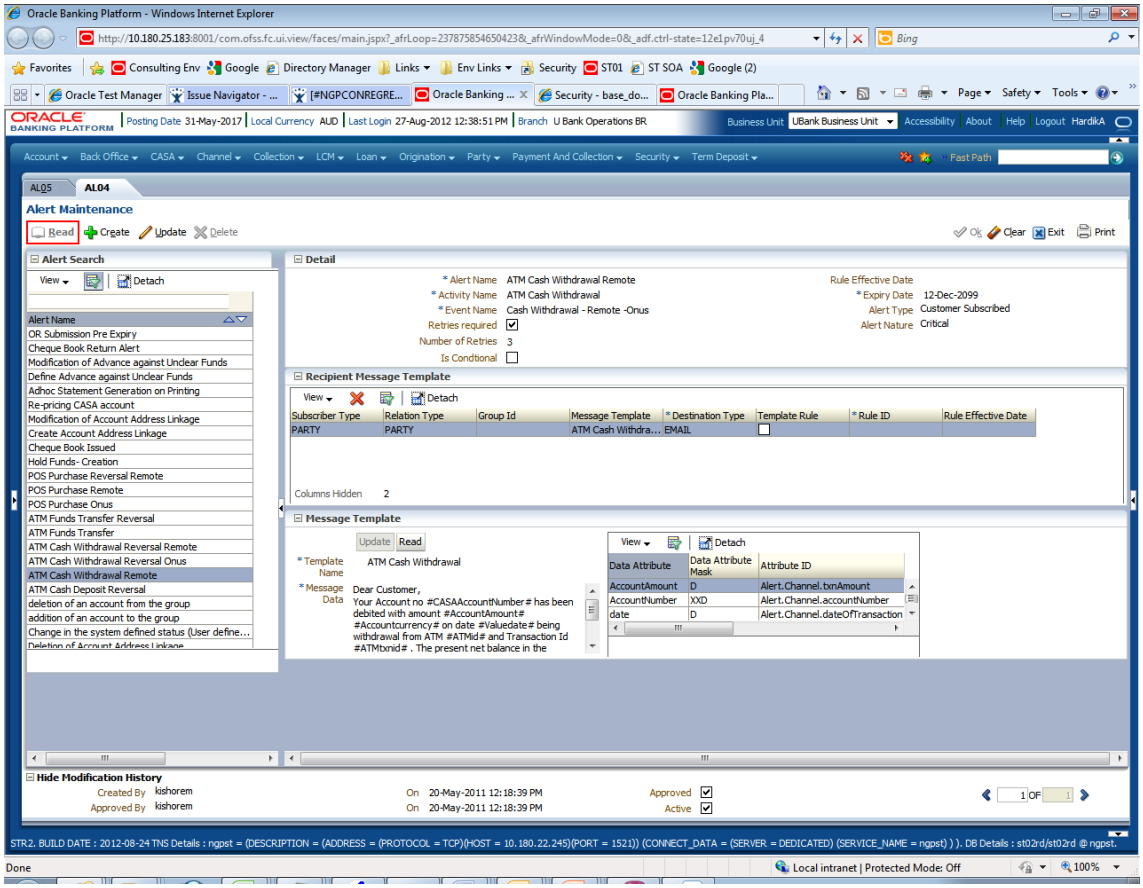
Figure 10–7 Alert Message Template Maintenance



10.1.8 Alert Maintenance

Given below is the Alert Maintenance screen.

Figure 10–8 Alert Maintenance



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".

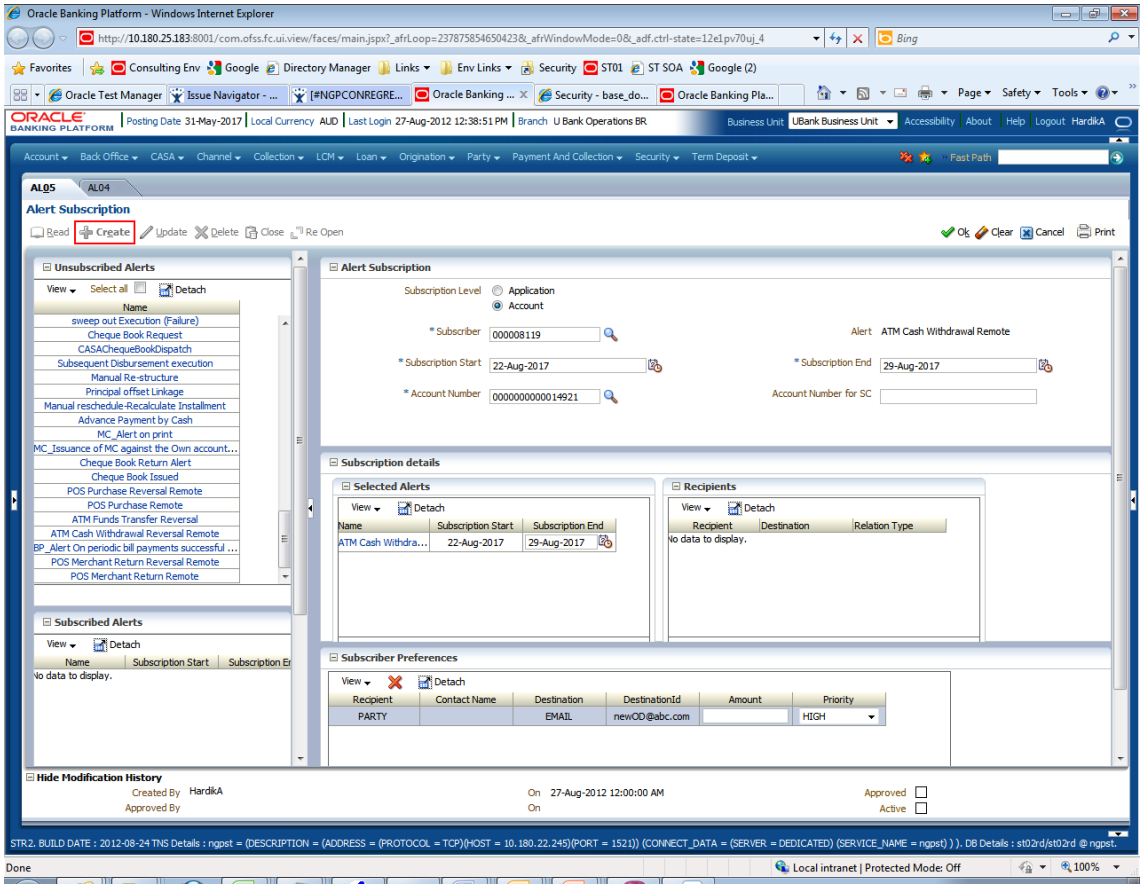
Now, we need to link a Recipient Message Template/s with this alert. For this we drag recipients from the Recipient Panel on to the Recipient Message Template Panel. In this setup, we define the kind of recipient and link this to predefined Message Template and Destination Types. The entry for this goes 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"

10.2 Alert Subscription

Subscription can be done for alerts at **account level** or at **application level** (called as subscription level).

Figure 10–9 Alert Subscription



10.2.1 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 10–10 Transaction API Changes - Service Call

```
public TransactionStatus updateContactPoint(SessionContext sessionContext, ContactPointDTO dto) throws FatalException {

    super.checkAccess("com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint", sessionContext, dto);
    if (logger.isLoggable(Level.FINE)) {
        logger.log(Level.FINE,
                MultiEntityLogger.getUniqueInstance()
                                 .formatMessage("Entered method updateContactPoint with partyId as '%s', contact point type as '%s' an
                                                dto.getPartyId(),
                                                dto.getContactPoint() == null ? "null" : dto.getContactPoint().toString(),
                                                dto.getPreferenceType() == null ? "null" : dto.getPreferenceType().toString()));
    }
    Interaction.begin(sessionContext);
    TransactionStatus transactionStatus = fetchTransactionStatus();
    try {
        Interaction.markCurrentTask(PartyTaskConstants.CONTACT_PREFERENCE);
        createTransactionContext(sessionContext, MaintenanceType.MODIFICATION);
```

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 10–11 Transaction API Changes - Conditional Evaluation

```
} else if(dto.isDnd() != telephoneNumberExisting.isDnd()
        || (dto.getDndStartDate() != null && !dto.getDndStartDate().equals(telephoneNumberExisting.getDndStartDate()))
        || (dto.getDndEndDate() != null && !dto.getDndEndDate().equals(telephoneNumberExisting.getDndEndDate()))) {
    if (logger.isLoggable(Level.FINE)) {
        logger.log(Level.FINE, MultiEntityLogger.getUniqueInstance()
                                        .formatMessage("Registering activity for alerts on change of dnd details for party '%s'",
                                                dto.getPartyId()));
    }
    partyAlertHelper.persistActivityLog(CONTACTPOINT_UPDATECONTACTPOINT_DNDINFO,
                                        dto,
                                        sessionContext,
                                        partyName.getFullName(),
                                        EVENTCODE_DNDINFO_CHANGE);
}
```

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 10–12 Transaction API Changes - persistActivityLog(..)

```
/**
 * This method logs/registers the activity log DTO
 *
 * @fcb.param MI,String,activityId, The ActivityId for which we need to log the data for the further processing of
 *           alerts.
 * @fcb.param MI,Object,object, holds the new data, sent as DTO's.
 * @fcb.param MI,SessionContext,sessionContext, holds the session data, used for creating the ApplicationContext.
 * @fcb.param MI,String,partyName, holds the Party Name.
 * @fcb.param MI,String,eventId, holds the event Id.
 * @throws FatalException
 */
public void persistActivityLog(String activityId, Object object, SessionContext sessionContext, String partyName, String eventId)
        throws FatalException {

    IActivityLog activityLog = populateActivityLog(activityId, object, partyName);
    if (activityLog != null) {
        com.ofss.fc.app.adapter.IAdapterFactory adapterFactory = AdapterFactoryConfigurator.getInstance()
                                                                                .getAdapterFactory(ModuleConstant.EVENT_PROCESSING);
        IEventProcessingAdapter adapter = (IEventProcessingAdapter) adapterFactory.getAdapter(EventProcessingAdapterConstant.MODULE_TO_ACTIVITY);
        adapter.registerActivityAndGenerateEvent(createApplicationContext(sessionContext), activityId, eventId, new Date(), activityLog);
    } else {
        if (logger.isLoggable(Level.FINE)) {
            logger.log(Level.FINE, "PartyAlertHelper.persistActivityLog  :  ActivityLog is null");
        }
    }
}
```

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 10–13 Transaction API Changes - Activity Log

```
private IActivityLog populateActivityLogForDNDInfoChange(Object object, String partyName) {

    ContactPointDTO contactPointDTO = (ContactPointDTO) object;
    PartyDNDInfoChangeDatalogDTO activityLog = new PartyDNDInfoChangeDatalogDTO();
    activityLog.setCustomerId(contactPointDTO.getPartyId());
    activityLog.setPartyId(contactPointDTO.getPartyId());
    activityLog.setFullName(partyName);
    activityLog.setUpdatedIsDnd(contactPointDTO.isDnd());
    activityLog.setUpdatedDndStartDate(contactPointDTO.getDndStartDate());
    activityLog.setUpdatedDndEndDate(contactPointDTO.getDndEndDate());
    activityLog.setCriticalNotification(true);
    return activityLog;
}
```

Figure 10–14 Transaction API Changes - Register Activity

```
/**
 * Used to register an Activity with an associated Event
 *
 * @param activityID
 * @param eventID
 * @param eventProcessingDate
 * @param activityLog
 * @return
 * @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,
                                                                                              activityEventKeyDTO,
                                                                                              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.

10.3 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.

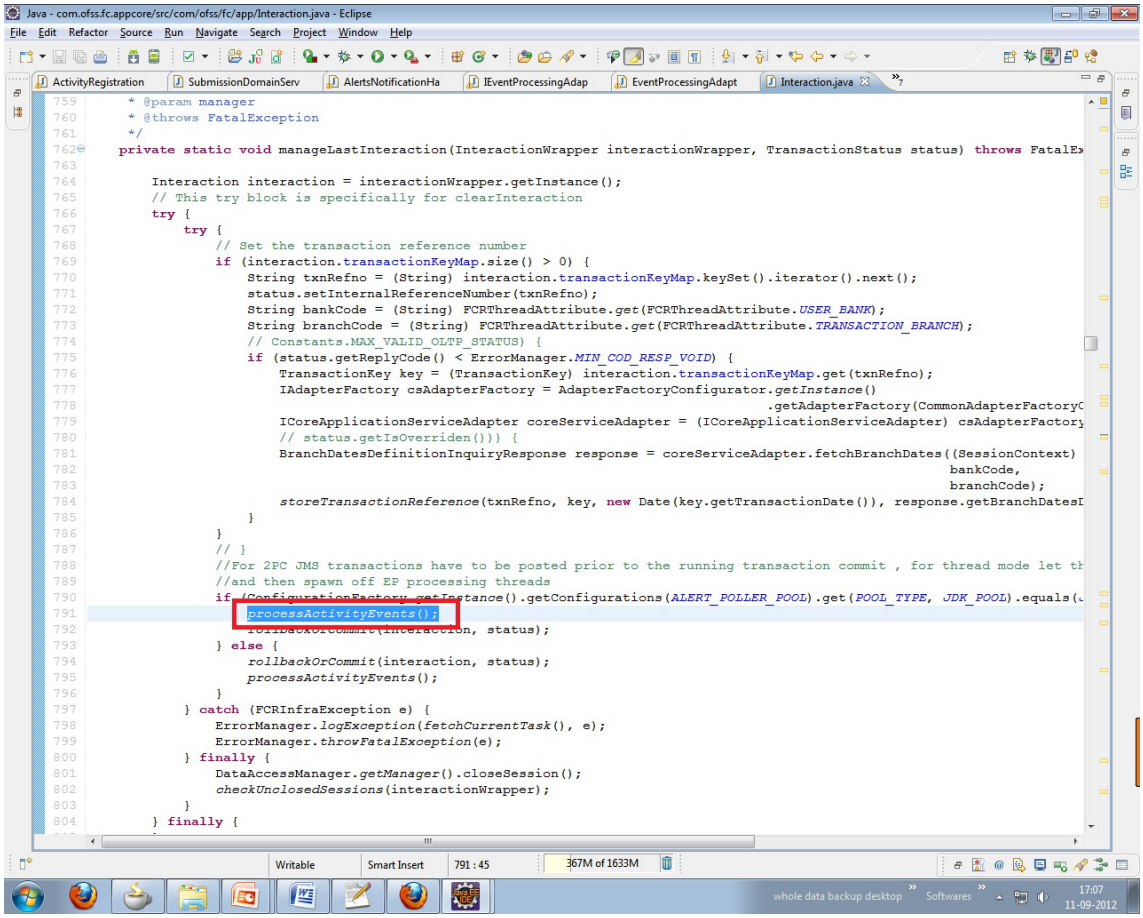
During this activity the entries are made in table FLX_EP_ACT_LOG_B and FLX_EP_EVT_LOG_B with appropriate comments depending on type of Alerts whether it is Mandatory (M) or Customer Subscribed (S).

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 10–15 Alert Processing Steps



EventProcessStatusType

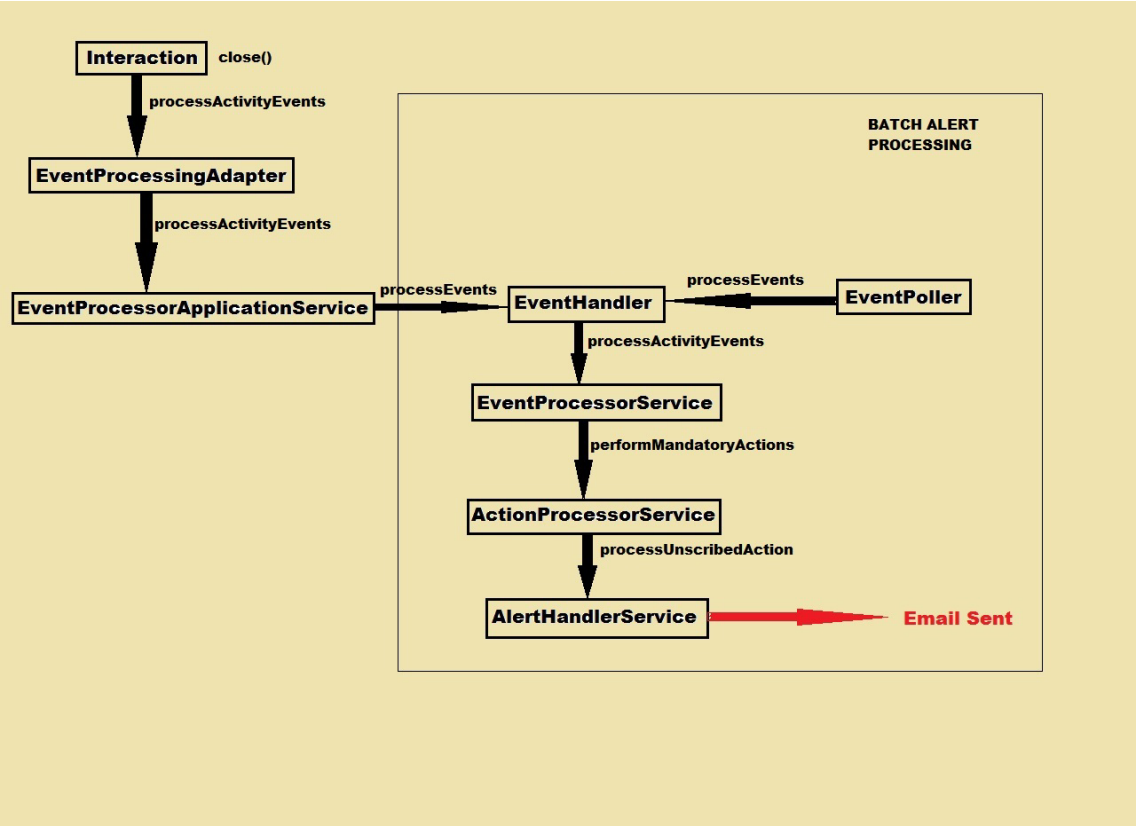
This shows status of event throughout cycle of event processing from Registration of event to Dispatch of Alert. (It is maintained in FLX_EP_EVT_LOG_B table as "flg_process_stat").

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 10–16 Event Processing Status Type

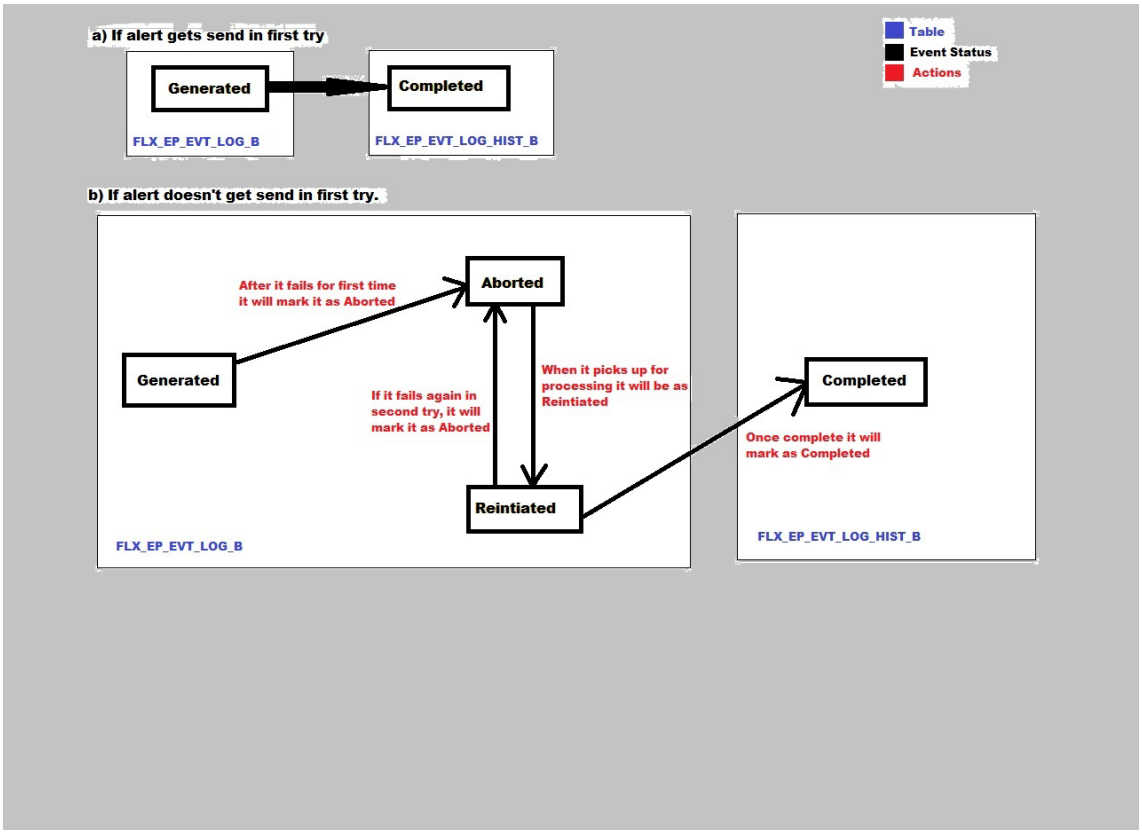


JMS (Java Messaging Service) is used for dispatch of alerts.

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.

Figure 10–17 Batch Alerts



10.4 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 10–18 Alert Dispatch Mechanism

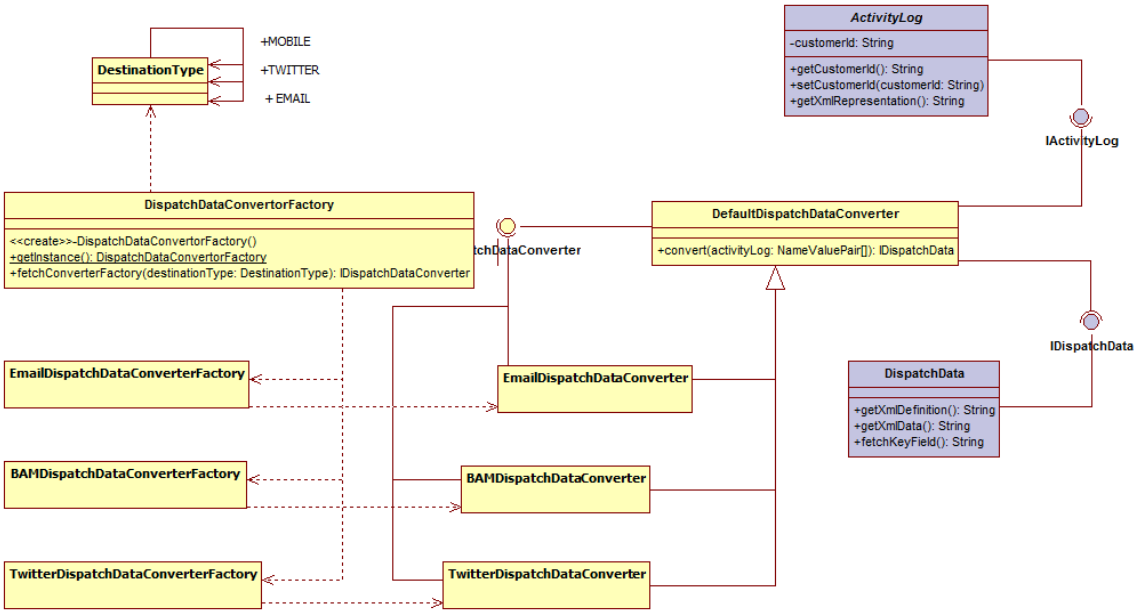


Figure 10–19 Alert Dispatch Mechanism - Dispatcher Factory

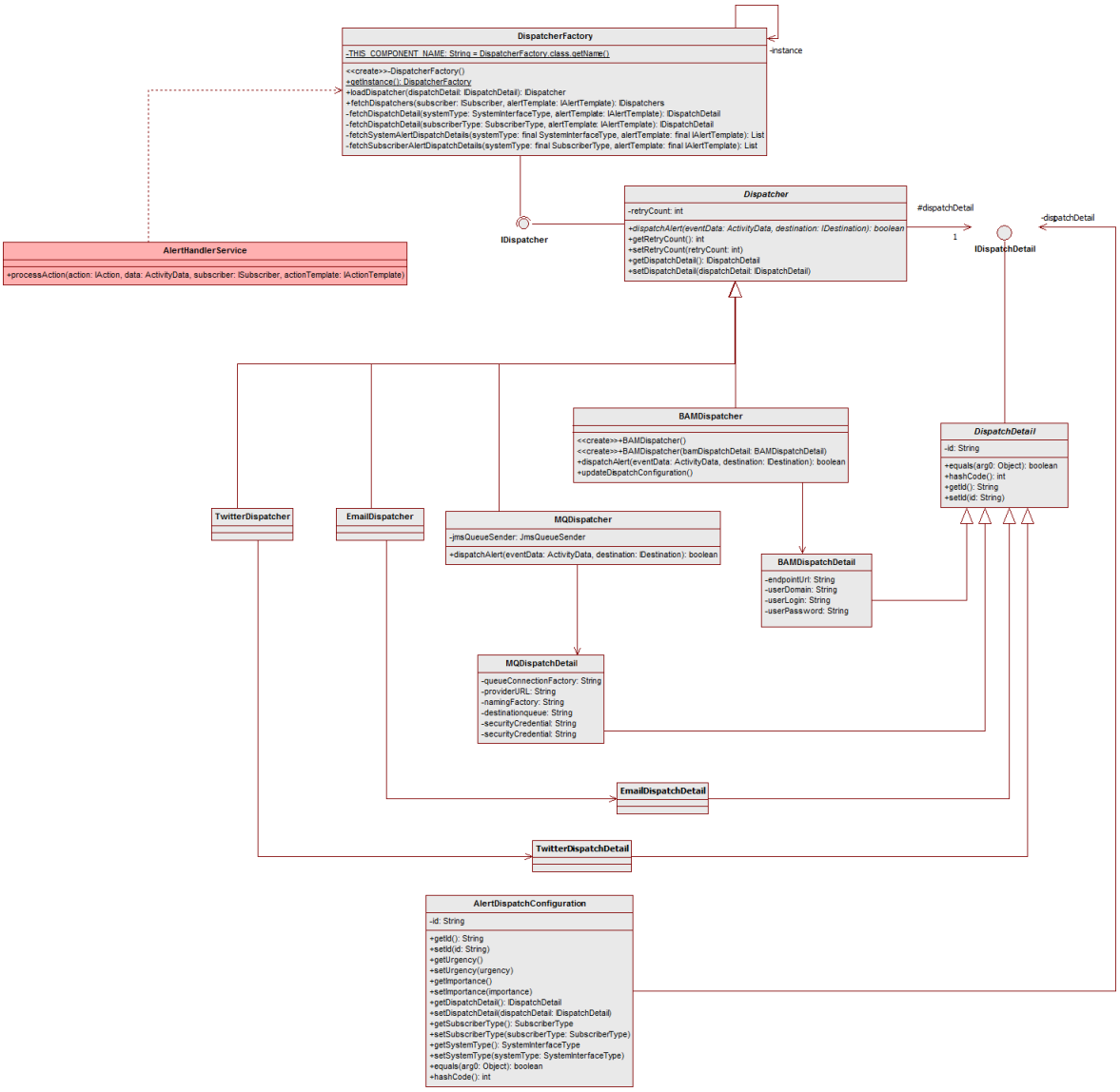
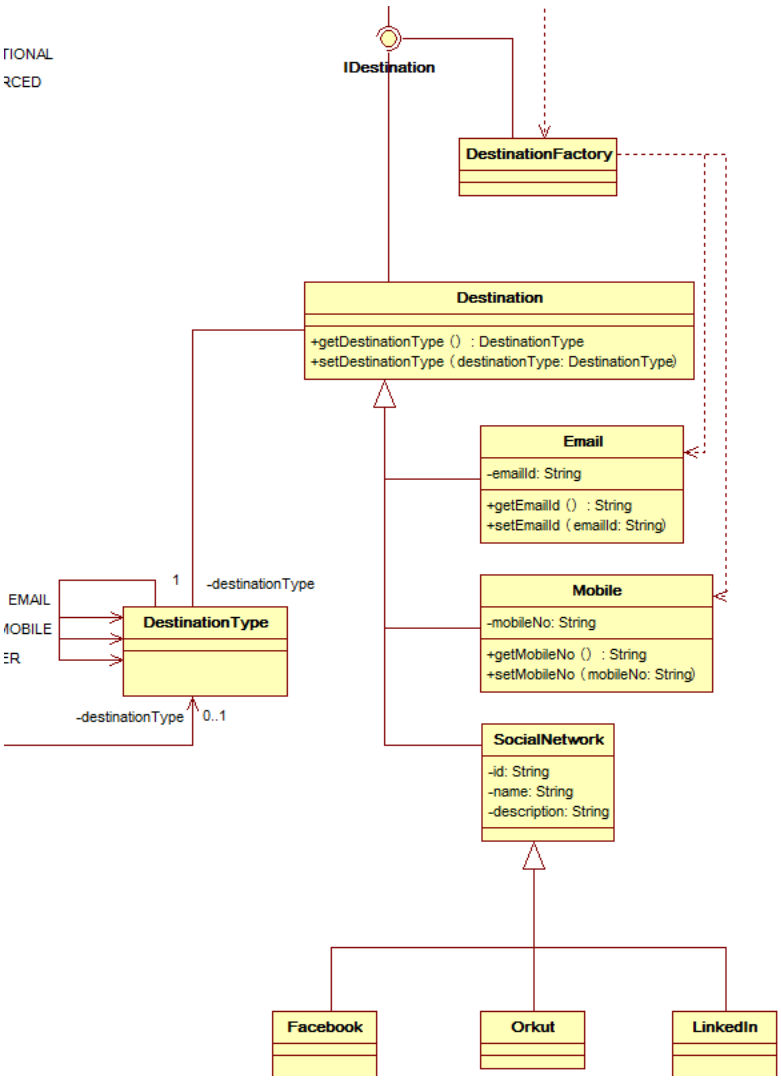


Figure 10–20 Alert Dispatch Mechanism - Destination



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.

10.5 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.

10.5.1 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,activityEventKeyDTO,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.

Eg.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;
}
```

10.5.2 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");
try{
ActivityRegistrationResponse response
=
activityRegistrationApplicationService.registerActivityAndGenerate
Event(
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.

10.6 Support For Derived Facts

Alerts are generated by assigning values to Facts that are mapped to the Alert Message Template placeholders.

These values are derived from the ActivityLog attributes based on the seed data that maintains the mapping information between the ActivityLog attributes and the Facts.

In Facts Module there is a provision to co-relate different Facts and derive the value of one Fact based on the value of the related Fact. This is done by maintaining the relationship in certain Fact tables.

The same support for Derived Facts has been included in Alerts framework.

For example, to add Party First Name information to an Alert this Fact has to be defined.

The following inserts are used to create this Fact with the name Alert.Party.FirstName.

Figure 10–21 Alert.Party.FirstName

```
Insert into flx_fa_facts_b
(FACT_CODE,FACT_NAME,FACT_DESC,DOMAIN_CODE,DOMAIN_CATEGORY_CODE,VALUE_TYPE,BUSINESS_TYPE_CODE,FACT_VALUE_REGEX,FACT_VAL_ERR_MSG,RETRIEVAL_KEY,CREATED_
BY,CREATION_DATE,LAST_UPDATED_BY,LAST_UPDATE_DATE,OBJECT_VERSION_NUMBER,LAST_UPDATE_LOGIN,FACT_CLASS,SHORT_NAME,DOMAIN_OBJECT_EXTN)
values ('Alert.Party.FirstName','Alert Party FirstName', 'FirstName', 'Banking', 'Alert', 'Open', 'Alphanumeric', null, null,
'Alert.Party.FirstName', 'manojpalk',to_timestamp('02-MAR-11 12.20.18.199000000 PM','DD-MON-RR HH.MI.SS.FF AM'),'manojpalk',to_timestamp('02-MAR-11
12.20.18.199000000 PM','DD-MON-RR HH.MI.SS.FF AM'),1,null,'NonFinancial','Alert.Party.FirstName','CZ');

Insert into flx_fa_group_xref (FACT_CODE, GROUP_CODE, CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATE_DATE, OBJECT_VERSION_NUMBER,
LAST_UPDATE_LOGIN) values ('Alert.Party.FirstName','Alert',null,null,null,null,null,null);
```

In Alerts framework, the facts that are available by default are:

Figure 10–22 Facts in Alerts Framework

```
1  Alert.MultiEntity.LegalEntity.Code denoting Legal Entity Code
2  Alert.MultiEntity.LegalEntity.Name denoting LegalEntity Name
3  Alert.MultiEntity.MarketEntity.Code denoting Market Entity Code
4  Alert.MultiEntity.MarketEntity.Name denoting Market Entity Name
5  Alert.MultiEntity.BusinessUnit.Code denoting Business Unit Code
6  Alert.MultiEntity.BusinessUnit.Name denoting Business Unit Name
7  Alert.Submission.SourcingEntityType denoting Sourcing Entity Type
8  Alert.Submission.SourcingEntityId denoting Sourcing Entity Id
9  Alert.Party.PartyId denoting the Party Id
```

In addition to these Facts all the Facts that have been mapped with the Service Attributes of the Activity log for the Activity Id of the Alert are available to the Alerts Framework for usage.

Facts that can be derived from any of the above Facts can be added to this list.

To relate and derive value of Alert.Party.FirstName with the help of available Fact Alert.Party.PartyId, the relationship information and value derivation logic must be maintained in the Facts tables.

Figure 10–23 Alert.Party.PartyId

```
Insert into flx_fa_value_bindings (FACT_CODE,PARAM_NAME,DATA_TYPE_CODE,PARAM_DESC,VARIABLE_BASED_FACT,LITERAL_FACT_VALUE,BINDING_TYPE)
values ('Alert.Party.FirstName','partyId',null,null,'Alert.Party.PartyId',null,'Variable');

Insert into flx_fa_value_datasources
(FACT_CODE,DATA_SOURCE_CODE,JDBC_DERIVATION_QUERY,HQL_DERIVATION_QUERY,JAVA_DERIVATION_CLASS,DSN_CODE,DB_FUNCTION_NAME)
values ('Alert.Party.FirstName','HQL',null,'SELECT a.firstName FROM com.ofss.fc.domain.party.entity.individual.IndividualName a WHERE
a.id.partyNameType=com.ofss.fc.enumeration.PartyNameType.Legal and a.id.partyId = :partyId',null,null,null);
```

FLX_FA_VALUE_BINDINGS defines the relationship and FLX_FA_VALUE_DATASOURCES defines the data derivation logic.

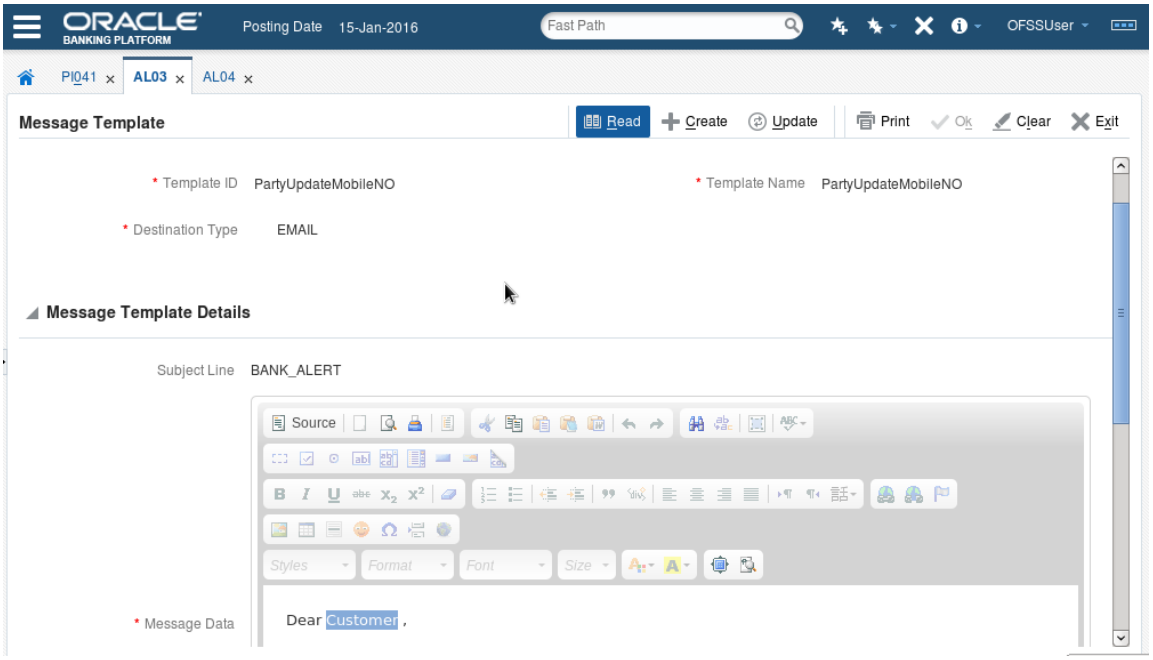
Similarly, additional derived Facts: Alert.Party.Prefix and Alert.Party.LastName can be maintained.

Figure 10–24 Alert.Party.Prefix and Alert.Party.LastName

```
For 'Alert.Party.Prefix':-

Insert into flx_fa_value_bindings (FACT_CODE,PARAM_NAME,DATA_TYPE_CODE,PARAM_DESC,VARIABLE_BASED_FACT,LITERAL_FACT_VALUE,BINDING_TYPE)
values ('Alert.Party.Prefix','partyId',null,null,'Alert.Party.PartyId',null,'Variable');

Insert into flx_fa_value_datasources
(FACT_CODE,DATA_SOURCE_CODE,JDBC_DERIVATION_QUERY,HQL_DERIVATION_QUERY,JAVA_DERIVATION_CLASS,DSN_CODE,DB_FUNCTION_NAME)
values ('Alert.Party.Prefix','HQL',null,'SELECT b.name from com.ofss.fc.domain.party.entity.global.Prefix b where b.prefixKey.id in (SELECT
a.prefixPrimary FROM com.ofss.fc.domain.party.entity.identity.PartyName a WHERE a.id.partyNameType=com.ofss.fc.enumeration.PartyNameType.Legal  and
a.id.partyId = :partyId)',null,null,null);

For 'Alert.Party.LastName':-

Insert into flx_fa_value_bindings (FACT_CODE,PARAM_NAME,DATA_TYPE_CODE,PARAM_DESC,VARIABLE_BASED_FACT,LITERAL_FACT_VALUE,BINDING_TYPE)
values ('Alert.Party.LastName','partyId',null,null,'Alert.Party.PartyId',null,'Variable');

Insert into flx_fa_value_datasources (FACT_CODE, DATA_SOURCE_CODE, JDBC_DERIVATION_QUERY, HQL_DERIVATION_QUERY, JAVA_DERIVATION_CLASS, DSN_CODE,
DB_FUNCTION_NAME) values ('Alert.Party.LastName','HQL',null,'SELECT a.lastName FROM com.ofss.fc.domain.party.entity.individual.IndividualName a WHERE
a.id.partyNameType=com.ofss.fc.enumeration.PartyNameType.Legal  and a.id.partyId = :partyId',null,null,null);
```

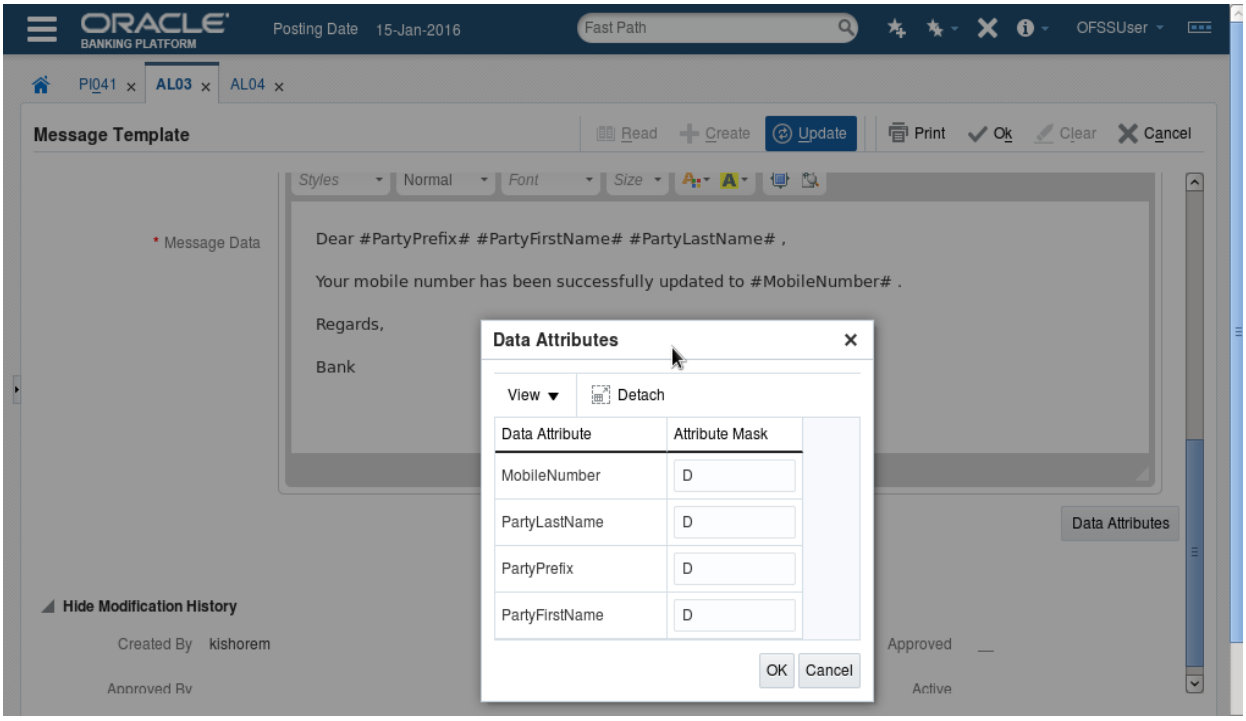
Use and test the maintenance and generation of Alerts using Derived Facts.

Figure 10–25 Message Template (Fast Path: AL03)



First, alter the existing Alert Message Template using the placeholder for the derived facts.

Figure 10–26 Placeholder for Derived Facts



Next, map the new Message Template placeholders in Alert Maintenance screen with the Derived Facts, which will also appear in the drop down of the Facts that are available to the Alerts Framework.

Figure 10–27 Alert Maintenance (Fast Path: AL04)

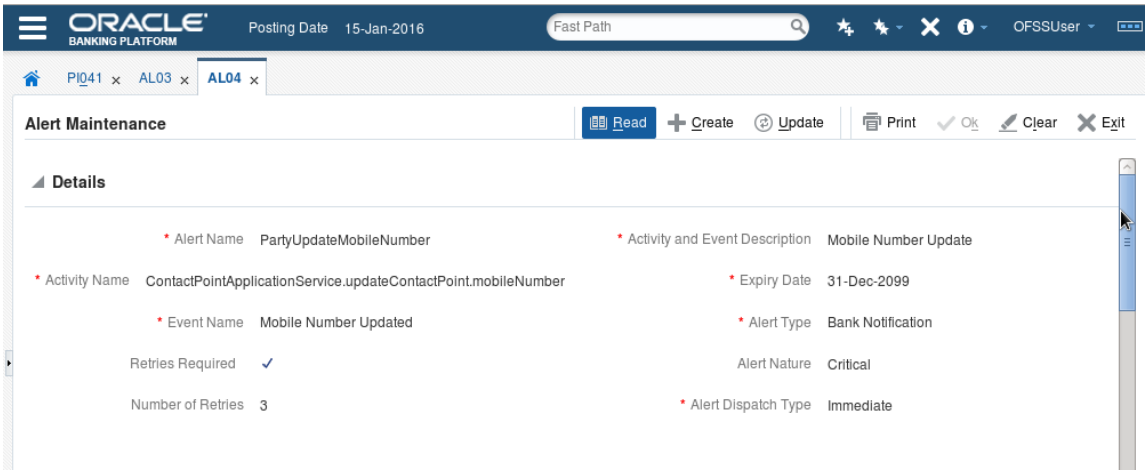


Figure 10–28 Alert Maintenance - Map the New Message Template Placeholders

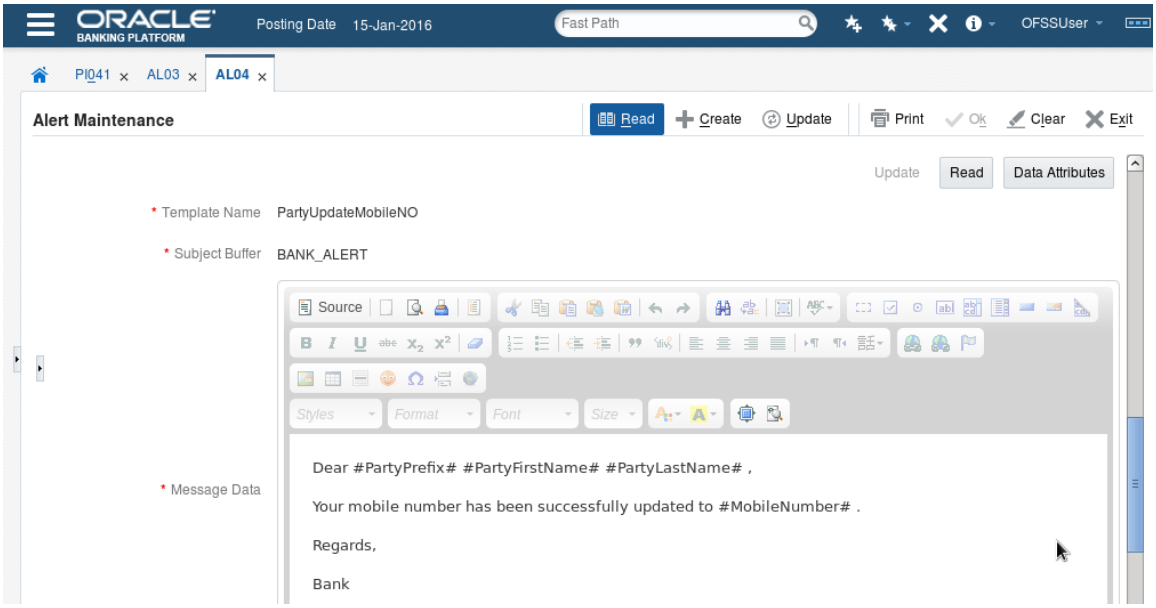


Figure 10–29 Alert Maintenance - Facts List

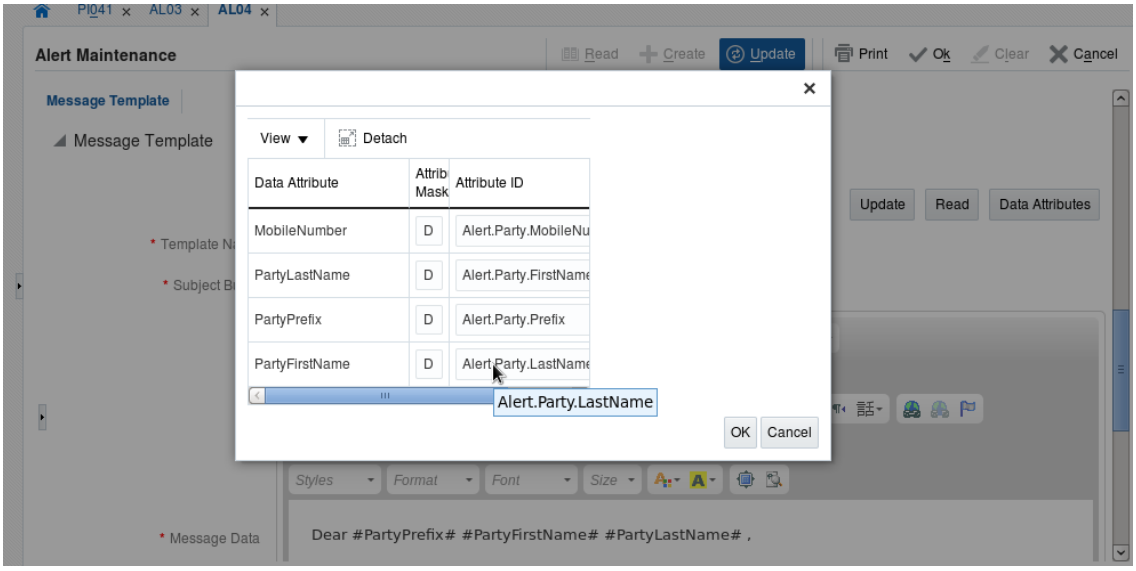
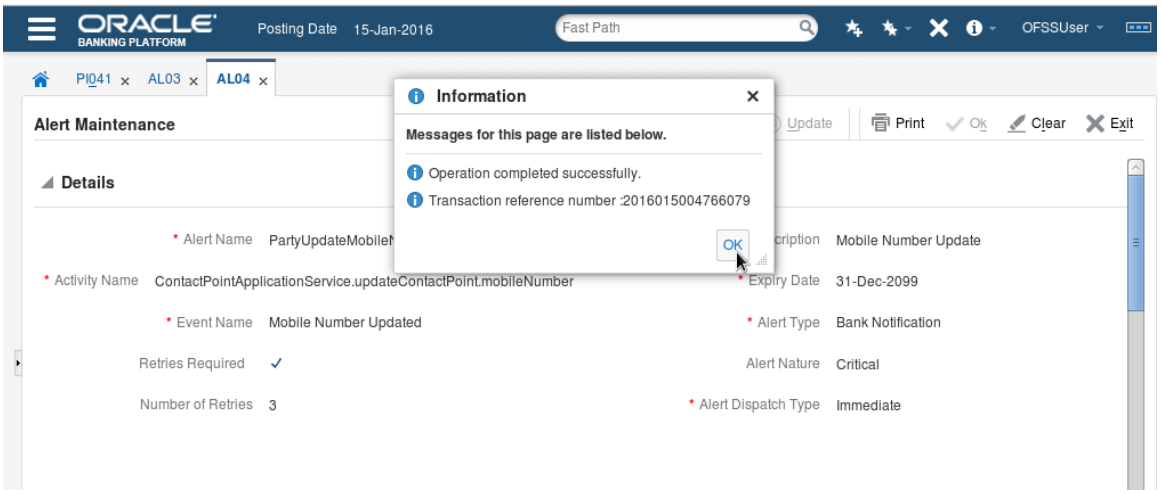
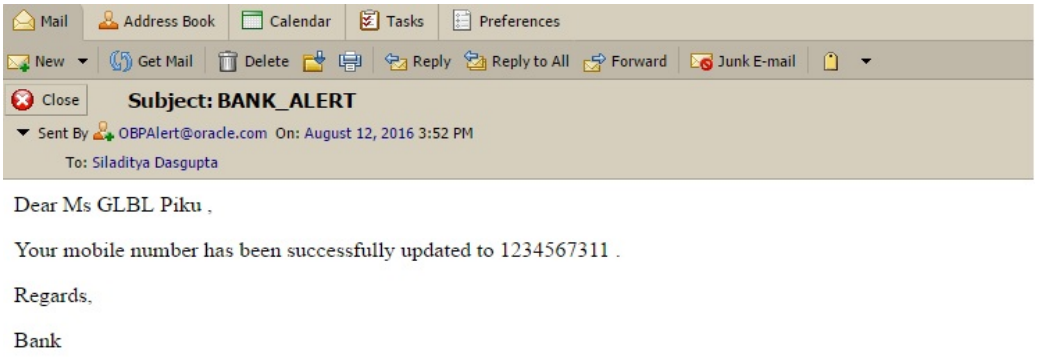


Figure 10–30 Alert Maintenance - Mapping Completed



Next, perform a Mobile Number updation from the Contact Point screen. This triggers the Alert that was altered earlier and the following mail is received.

Figure 10–31 Alert Mail on Mobile Number Update in Contact Point screen

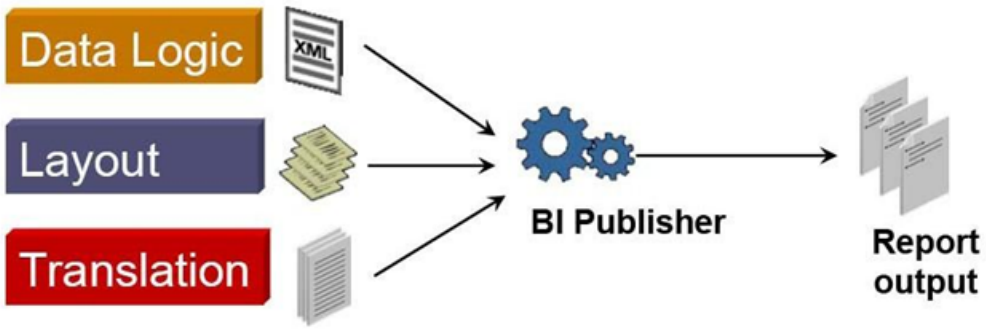


The Alerts Framework has been able to substitute the place holders of the Message Template with the Fact values derived from Derived Fact derivation logic in Facts Framework.

11 Creating New Reports Using BI Publisher

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.

Figure 11–1 Creating New 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.

11.1 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 11–2 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 11–3 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 11–4 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 11–5 Report DML Function

```
-- DML function for the report
CREATE OR REPLACE FUNCTION AP_DML_PI007(var_l_session_id  IN NUMBER,
                                        var_bank_code     IN VARCHAR2,
                                        var_cod_shell     IN VARCHAR2)
RETURN NUMBER AS

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 := '%';
  ELSE
    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_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_GROUP_CODE, BJSR.EXECUTION_DURATION, var_l_session_id
  FROM
    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
  WHERE
    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_DML_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 11–6 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

v_ret                  REP_TAB_PI007;
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(SQLCODE, '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.

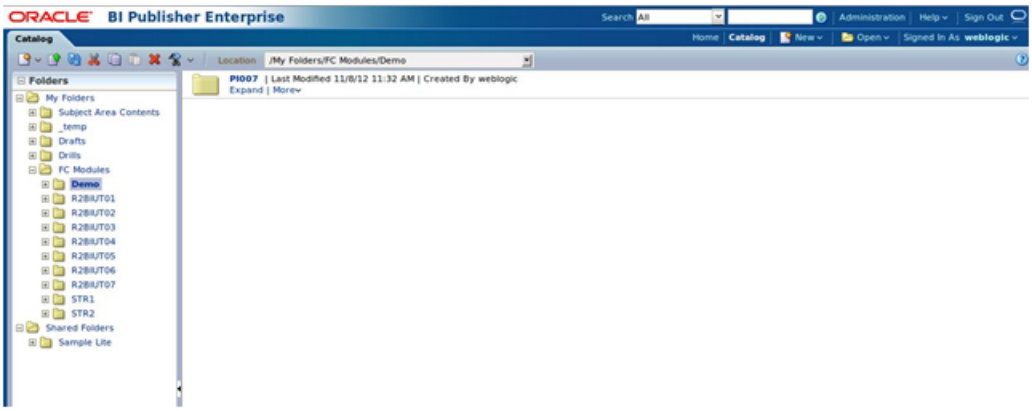
You can log in to the BIP application deployed on http: //<IP ADDRESS><PORT>/xmlpserver/ with the credentials *weblogic/weblogic1*.

11.2 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 11–7 Catalog Folder

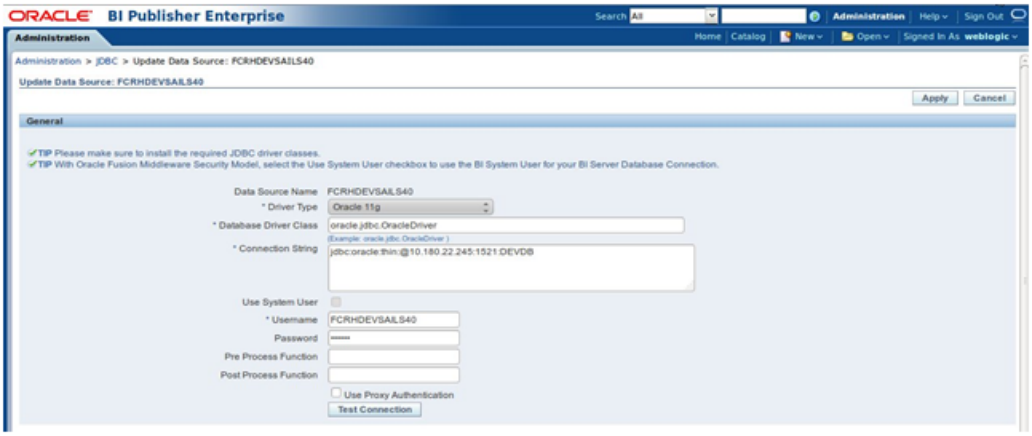


11.3 Data Source

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 11–8 Data Source



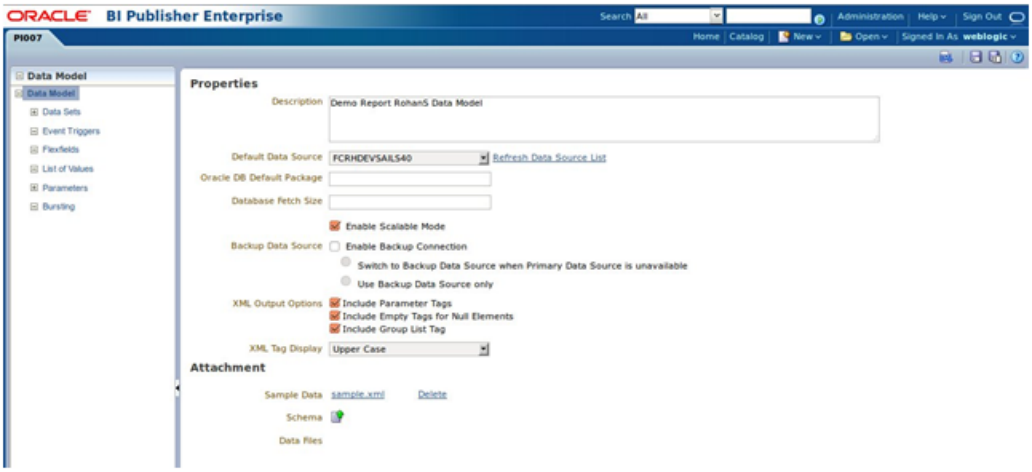
11.4 Data Model

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 11–9 Data Model



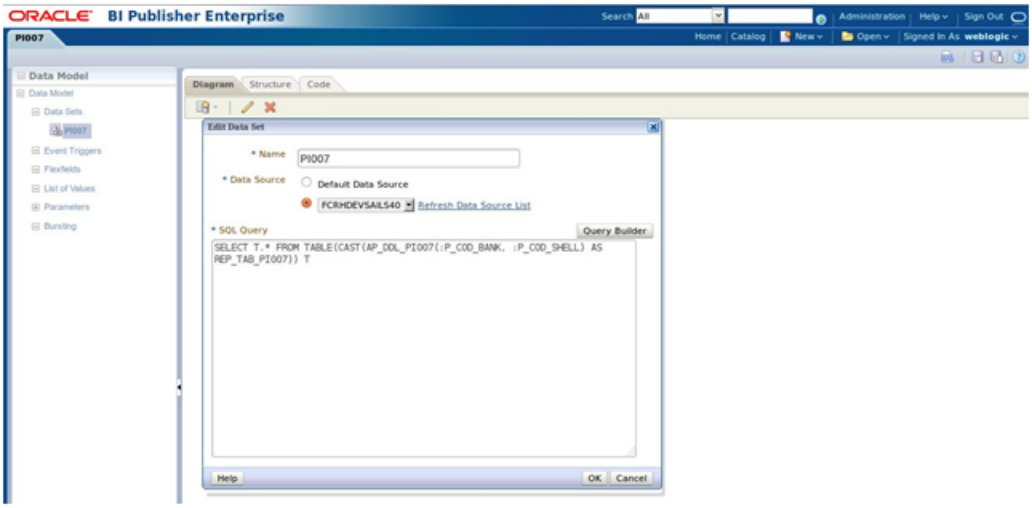
Data Set

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 11–10 Data Set



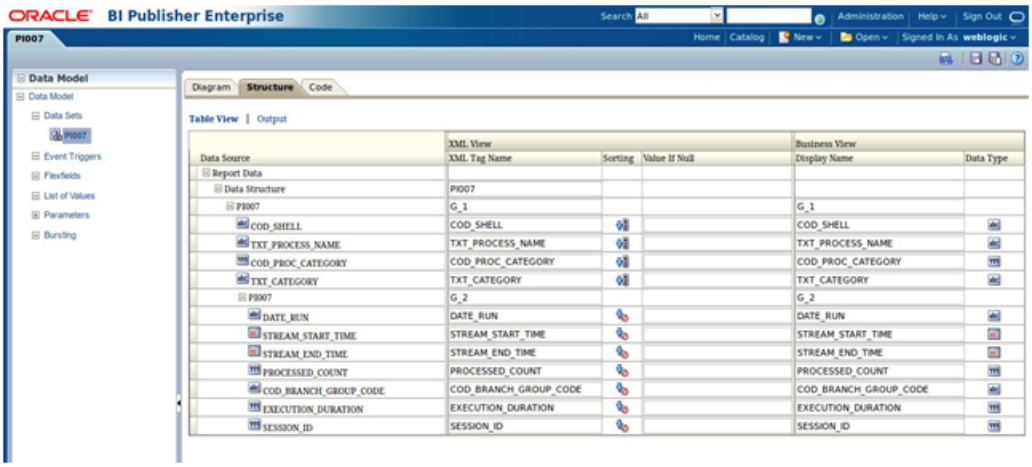
On click of OK, a data set will be created with all the fields as defined in the previously created *Report Record Type*.

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.

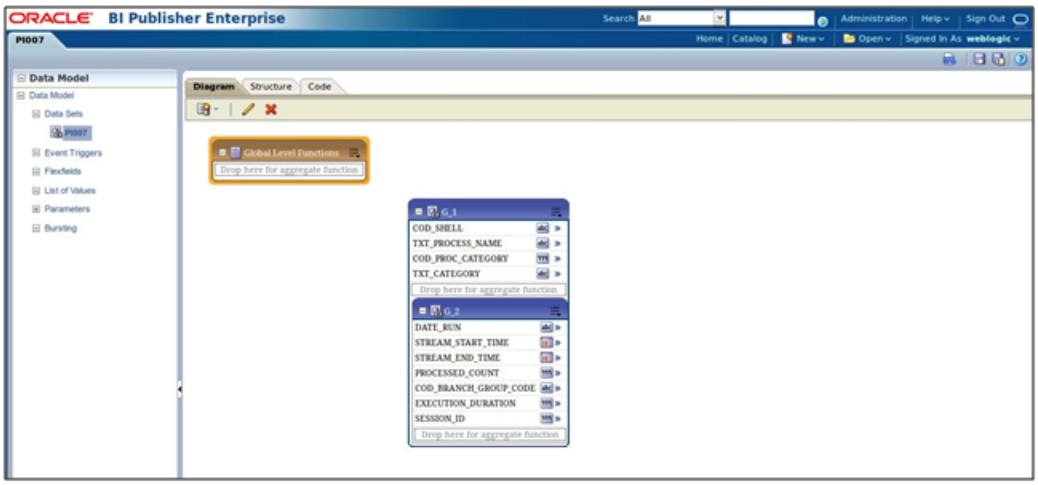
Figure 11–11 Group Fields



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:

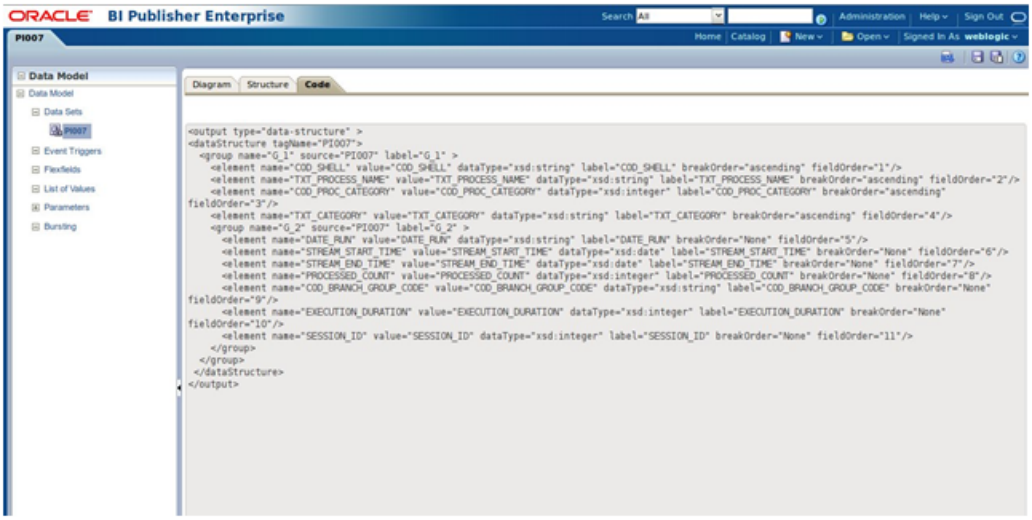
Figure 11–12 XML Structure and Labels



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 11–13 XML Code



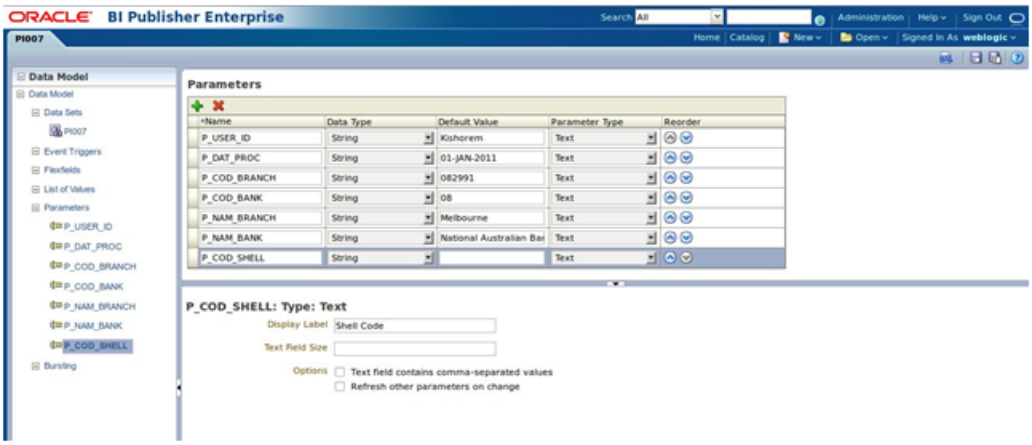
Input Parameters

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 11–14 Add Input Parameters



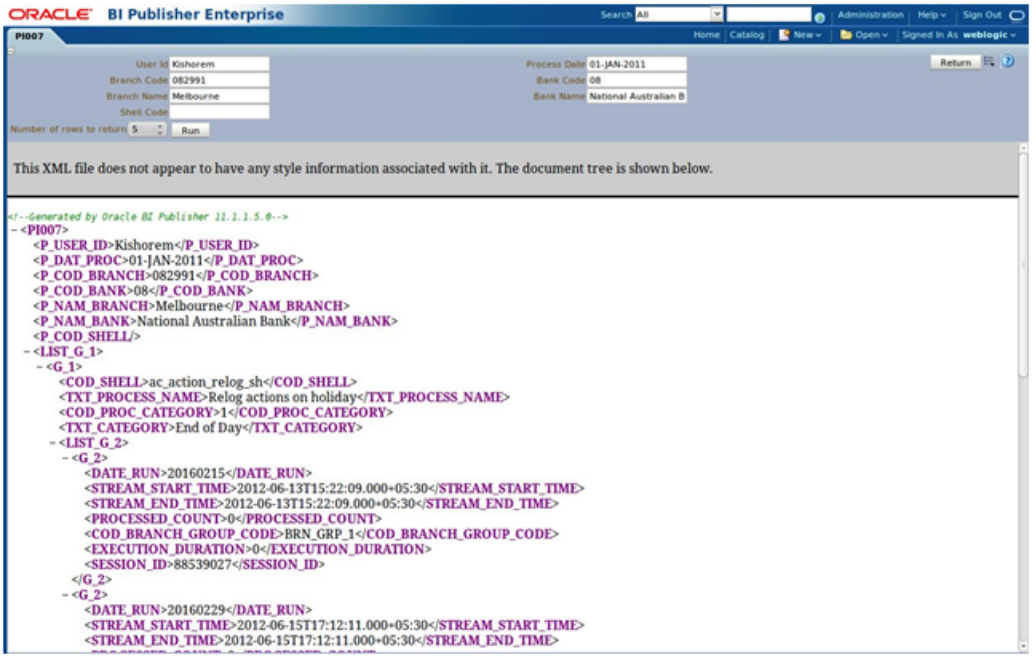
11.5 XML View of Report

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 11–15 XML View of Report



11.6 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.

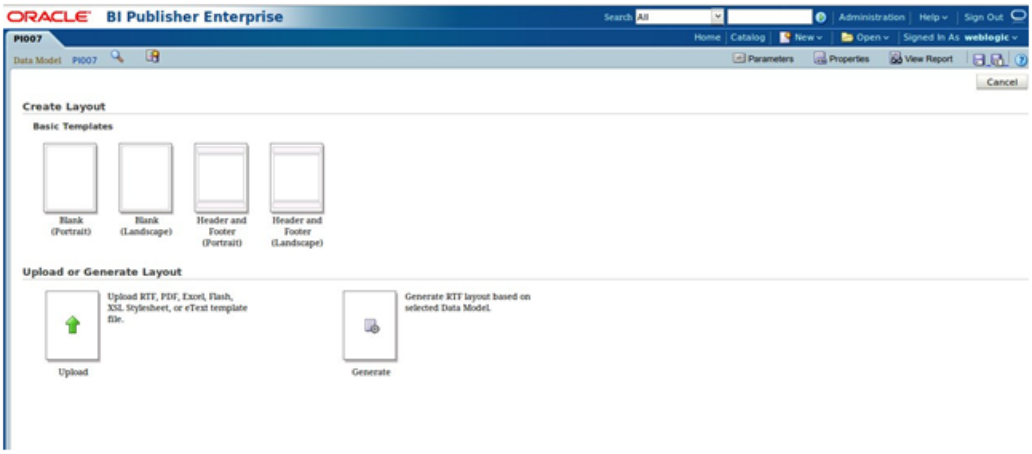
Anybody familiar with using Microsoft Word or Adobe Acrobat can use the corresponding plug-ins for these tools to create a layout for a report. You can create a rich layout using these standalone applications with BIP plug-ins and then upload them to the BIP application for use in your report.

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 11–16 Layout of the Report - Create Layout

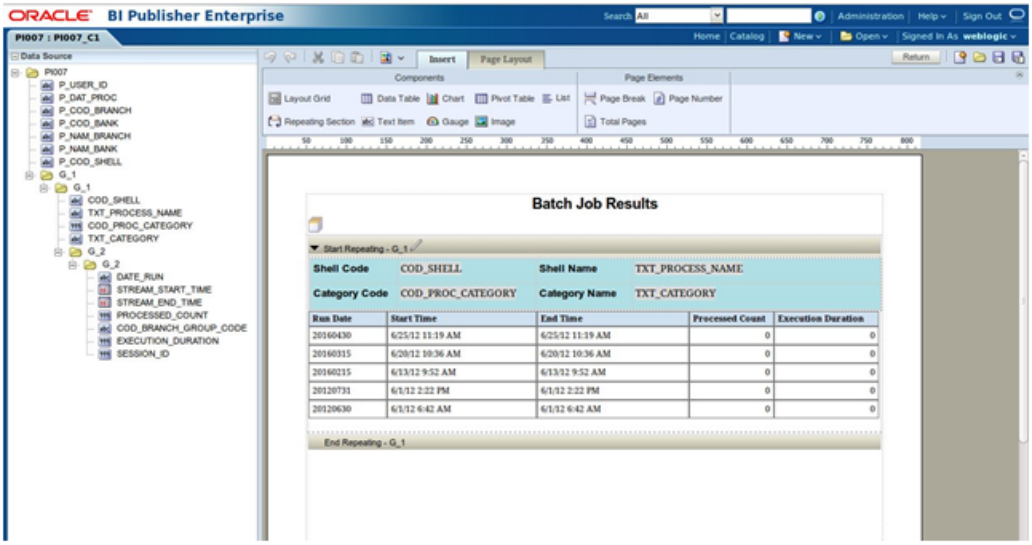


Choose from the *Basic Templates* to create a layout from a template. The layout editor screen will open. The previously created data set fields are present on the left pane of the screen. The toolbar present on top of the layout has tools to insert *Layout Grid, Data Table, Repeating Section, Text Item, 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.

Figure 11–17 Layout of the Report - Batch Job Results



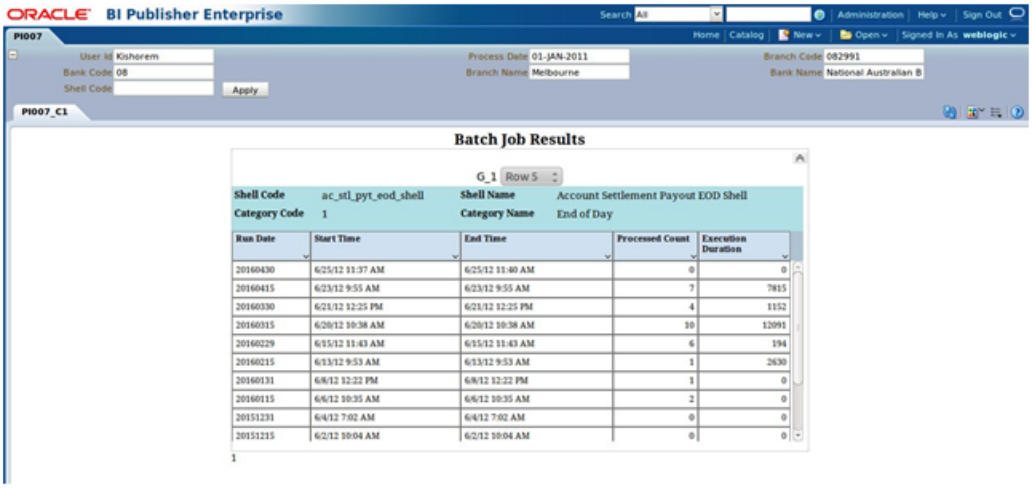
11.7 View Report in BIP

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.

Figure 11–18 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.

11.8 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.

11.9 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.

11.10 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.

11.11 OBP Batch Report Configuration

This section describes the OBP batch report configuration.

11.11.1 Batch Report Generation for a Branch Group Code

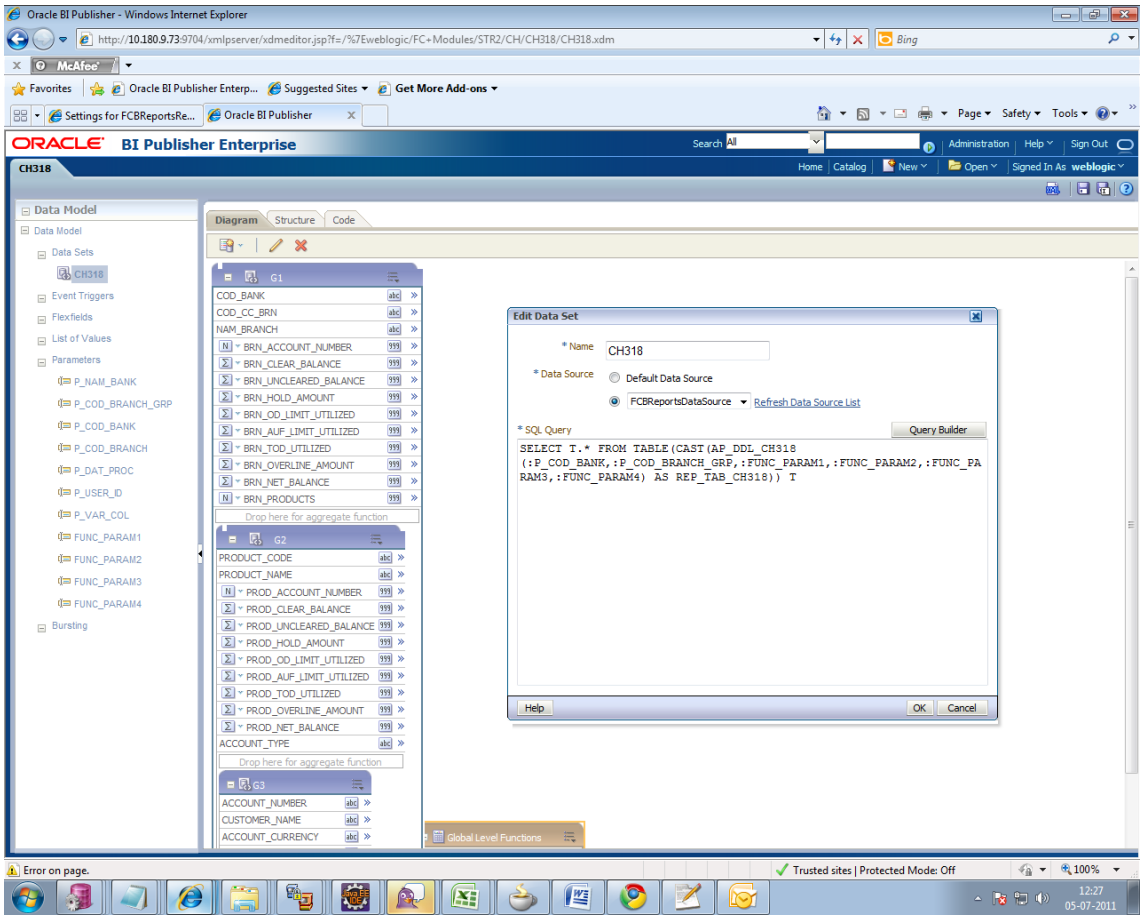
During Batch Process, a report should be generated for all branches linked to the respective Branch Group Code.

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 11–19 Batch Report Generation for a Branch Group Code



11.11.2 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.

11.11.3 Batch Report Generation Path

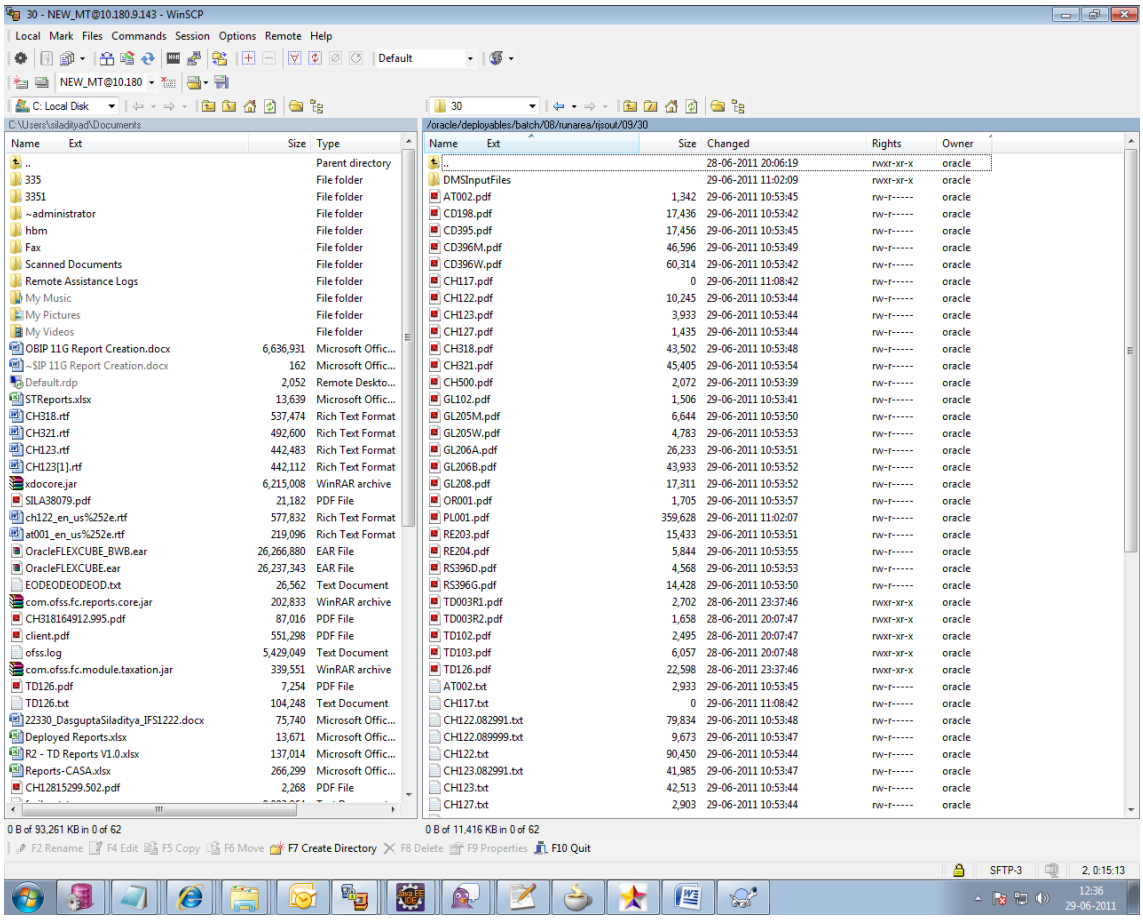
The reports (for example, 30th September 2008) will be generated as shown in the host side screen-shot.

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>

Figure 11–20 Batch Report Generation Path



11.12 OBP Adhoc Report Configuration

This section describes the OBP adhoc report configuration.

11.12.1 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');
```

11.12.2 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.

11.12.3 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.

11.12.4 Adding Screen Tab for Report Module

For adding a Screen Tab do the following:

```
com.ofss.fc.ui.view.brop.jar@
public_
html/com/ofss/fc/ui/view/brop/reportRequest/form/ReportRequest.jsff
<af:commandNavigationItem partialSubmit="true" text="#{rb7775.LBL_
Reconciliation}"
binding="#{ReportRequest.cni11}" id="cni11" immediate="true"
actionListener="#{ReportRequest.processMode}" selected="false">
```

```
<f:attribute name="mode" value="Reconciliation"/>
</af:commandNavigationItem>


com.ofss.fc.ui.view.brop.jar@
/com/ofss/fc/ui/view/brop/reportRequest/backing/ReportRequest.java
private RichCommandNavigationItem cni11;
Add following accessors:-
public void setCni11(RichCommandNavigationItem cni11) {
this.cni11 = cni11;
}
public RichCommandNavigationItem getCni11() {
return cni11;
}
```

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

11.13 Adhoc Report Generation – Screen 7775

The adhoc report can be generated using the following screen:

Figure 11–21 Adhoc Report Generation - Report Request

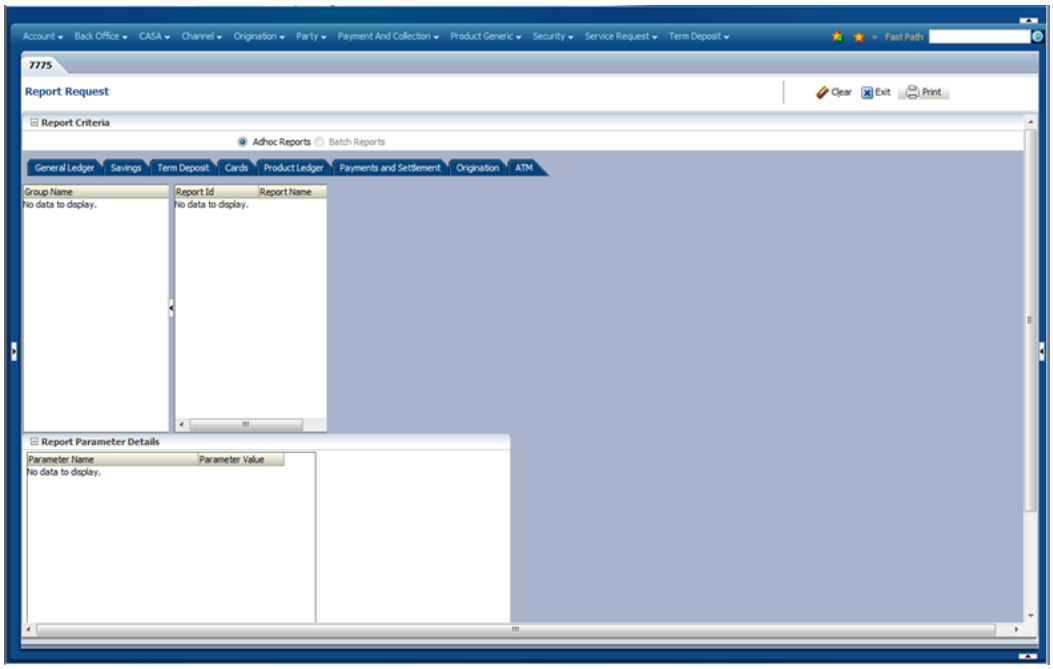
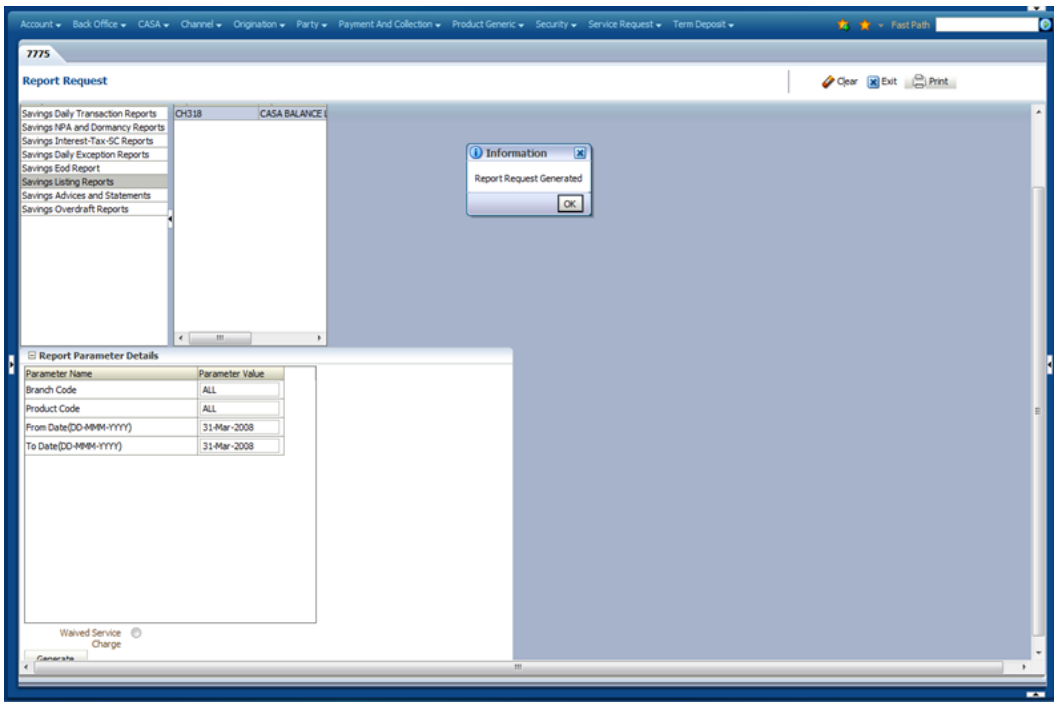


Figure 11–22 Adhoc Report Generation - Report Generated



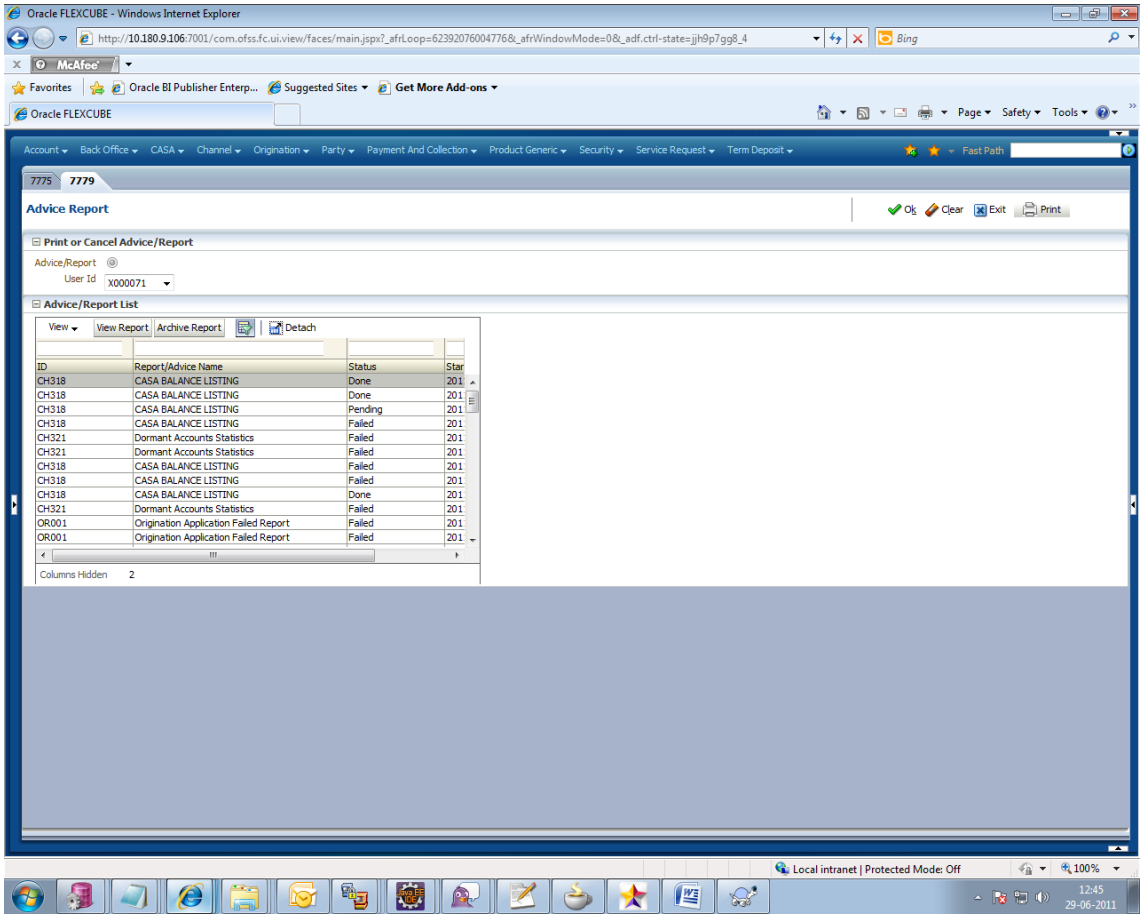
On filling the parameters and clicking on 'Generate' the report request gets successfully posted.

At the end of Adhoc report generation, RJS_REQUESTS gets logged with the generated report status as D -> Done, F-> Failed.

11.14 Adhoc Report Viewing – Screen 7779

The adhoc report can be viewed using the following screen:

Figure 11–23 Advice Report

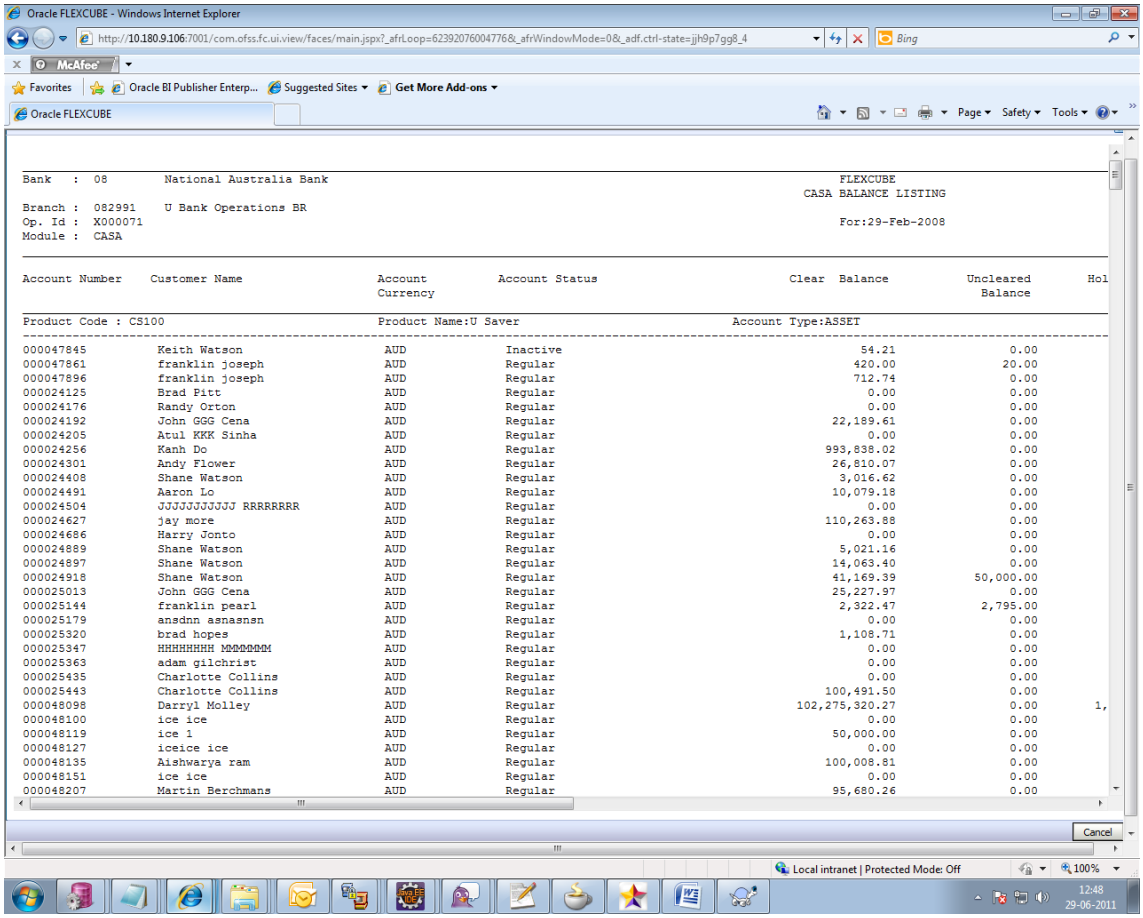


On selecting the correct user id that generated the report we get the reports generated by that user.

Now sort the Transaction Number (right most column) in the descending order.

Select the top record and click 'View Report'.

Figure 11–24 View Generated Adhoc Report



Oracle FLEXCUBE - Windows Internet Explorer

Bank : 08 National Australia Bank — FLEXCUBE

CASA BALANCE LISTING

Branch : 082991 U Bank Operations BR

Op. Id : X000071 — For:29-Feb-2008

Module : CASA

Account Number	Customer Name	Account Currency	Account Status	Clear Balance	Uncleared Balance	Hol
Product Code : CS100		Product Name:U Saver		Account Type:ASSET		
000047845	Keith Watson	AUD	Inactive	54.21	0.00	
000047861	franklin joseph	AUD	Regular	420.00	20.00	
000047896	franklin joseph	AUD	Regular	712.74	0.00	
000024125	Brad Pitt	AUD	Regular	0.00	0.00	
000024176	Randy Orton	AUD	Regular	0.00	0.00	
000024192	John GGG Cena	AUD	Regular	22,189.61	0.00	
000024205	Atul KKK Sinha	AUD	Regular	0.00	0.00	
000024256	Kanh Do	AUD	Regular	993,838.02	0.00	
000024301	Andy Flower	AUD	Regular	26,810.07	0.00	
000024408	Shane Watson	AUD	Regular	3,016.62	0.00	
000024491	Aaron Lo	AUD	Regular	10,079.18	0.00	
000024504	JJJJJJJJJJJ RRRRRRRR	AUD	Regular	0.00	0.00	
000024627	jay more	AUD	Regular	110,263.88	0.00	
000024686	Harry Jonto	AUD	Regular	0.00	0.00	
000024889	Shane Watson	AUD	Regular	5,021.16	0.00	
000024897	Shane Watson	AUD	Regular	14,063.40	0.00	
000024918	Shane Watson	AUD	Regular	41,169.39	50,000.00	
000025013	John GGG Cena	AUD	Regular	25,227.97	0.00	
000025144	franklin pearl	AUD	Regular	2,322.47	2,795.00	
000025179	ansdnn asnasnsn	AUD	Regular	0.00	0.00	
000025320	brad hopes	AUD	Regular	1,108.71	0.00	
000025347	HHHHHHHH MMMMMM	AUD	Regular	0.00	0.00	
000025363	adam gilchrist	AUD	Regular	0.00	0.00	
000025435	Charlotte Collins	AUD	Regular	0.00	0.00	
000025443	Charlotte Collins	AUD	Regular	100,491.50	0.00	
000048098	Darryl Molley	AUD	Regular	102,275,320.27	0.00	1,
000048100	ice ice	AUD	Regular	0.00	0.00	
000048119	ice 1	AUD	Regular	50,000.00	0.00	
000048127	iceice ice	AUD	Regular	0.00	0.00	
000048135	Aishwarya ram	AUD	Regular	100,008.81	0.00	
000048151	ice ice	AUD	Regular	0.00	0.00	
000048207	Martin Berchmans	AUD	Regular	95,680.26	0.00	

Cancel

Local intranet | Protected Mode: Off — 100% — 12:48 29-06-2011

The report is rendered in the front end.

12 Security Customizations

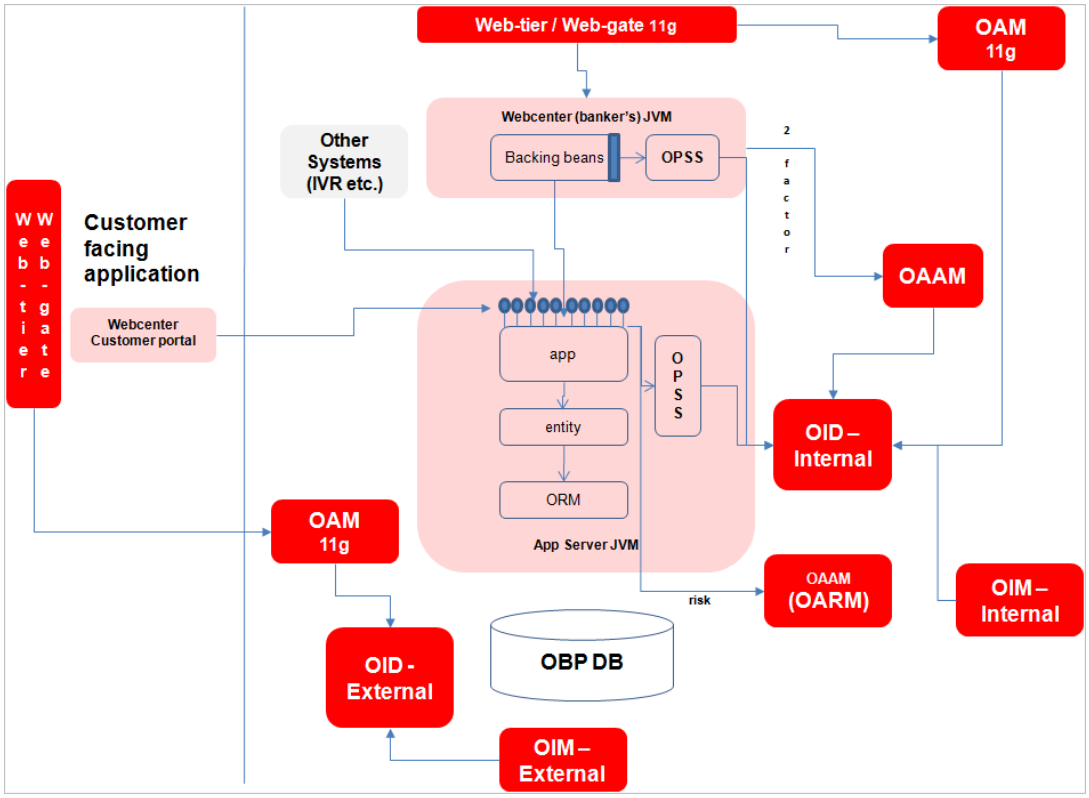
OBP comprising of several modules has to interface with various systems in an enterprise to transfer or share data which is generated during business activity that takes place during teller operations or processing. While managing the transactions that are within OBP's domain, it is needed to consider 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

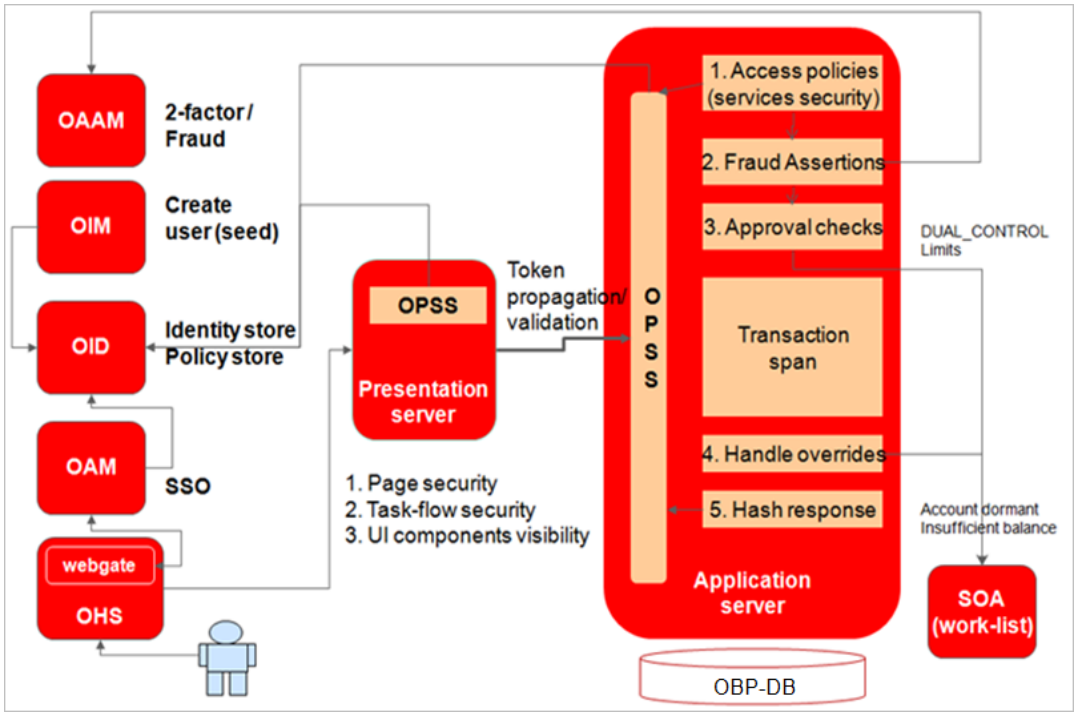
Figure 12–1 Security Customizations Interface



- 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.
- SM502- Policy Management screen 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 12–2 Security Use Case with Access Checks and Assertions



12.1 OPSS Access Policies / Matrix Auth – 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.DemandDepositCashT
ransactionService.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.

12.1.1 Steps

Following steps are needed to add an extra attribute to an access policy rule.

You can use SM500 screen or use ‘PolicyStoreSetup’ -> script ‘seedFreshPolicyStore.sh’.

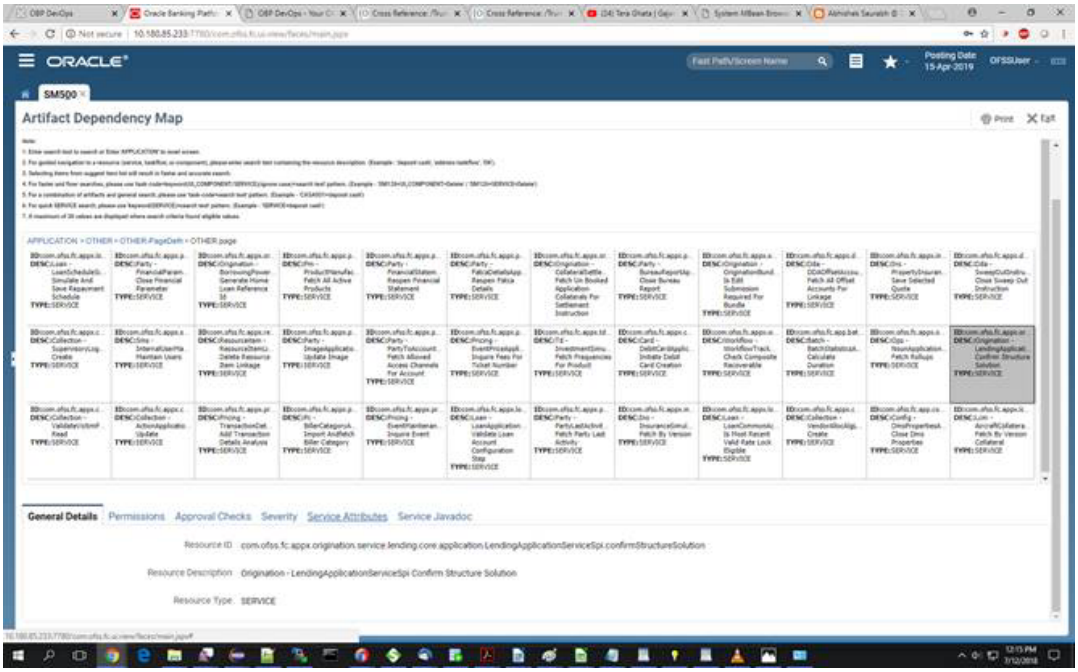
Below are the steps to use consulting atributes in 'com.ofss.fc.approval.lendingspi_confirmstructuresolution' routing rule:

1. Add the attribute name for service_id in SM500:

   com.ofss.fc.appx.origination.service.lending.core.application.LendingApplicationServiceSpi.confirmStructureSolution,AllowedPolicyAttributes,LendingStructureSolution_IsTaskIncludesSecuredProduct,N,2/1

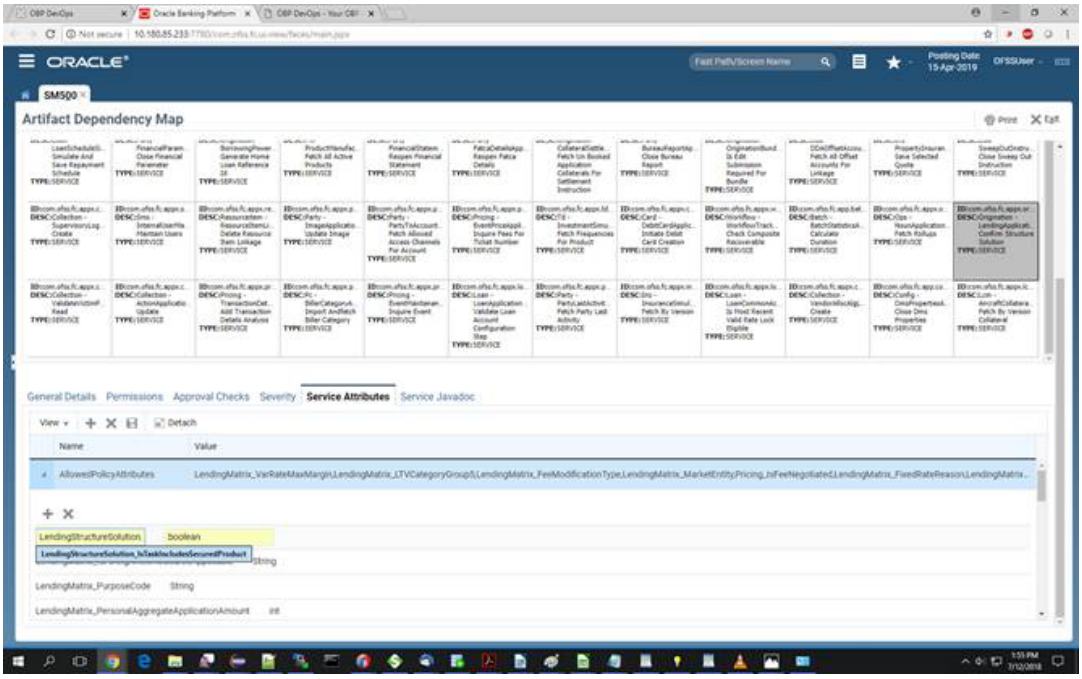
   a. Search for serviceId 'com.ofss.fc.appx.origination.service.lending.core.application.LendingApplicationServiceSpi.confirmStructureSolution' in SM500

   Figure 12–3 Add Attributes Name for Service ID

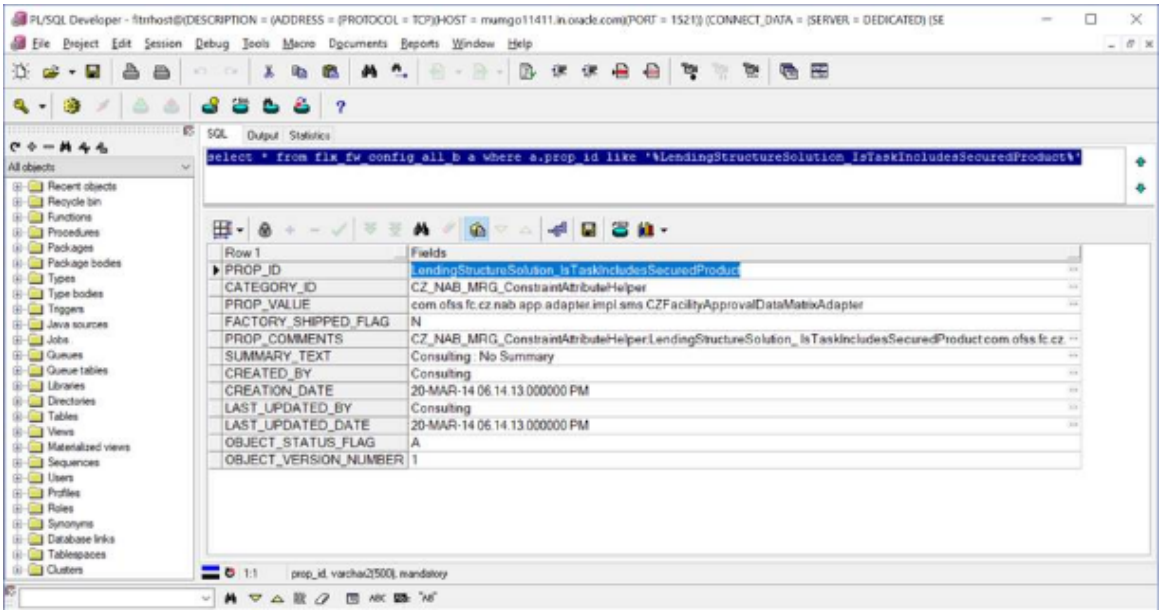
   b. Add attribute name 'LendingStructureSolution_IsTaskIncludesSecuredProduct' with appropriate datatype in 'allowed policy attributes' under 'Service Attributes' tab in SM500 as shown below:

Figure 12–4 Add Service Attributes in SM500



2. Define the attribute in Constraint Attribute Config with associated adapter to fetch attribute value.

Figure 12–5 Constraint Attribute Config with Associated Adapter



3. Add derivation logic to set the attribute value in Adapter that is mapped in config.

   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 - LendingStructureSolution_
IsTaskIncludesSecuredProduct
Adapter -
public
com.ofss.fc.app.adapter.impl.sms.FacilityApprovalDataMatrixAd
apter {
public Boolean getIsTaskIncludesSecuredProduct() {
if (facilityApprovalDataDTO != null &&
facilityApprovalDataDTO.getIsTaskIncludesSecuredProduct() !=
null)
return facilityApprovalDataDTO.getIsTaskIncludesSecuredProduct
();
else
return false;
} }
```

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.

```
Public final String LendingStructureSolution_
IsTaskIncludesSecuredProduct =
"com.ofss.fc.app.adapter.impl.sms.FacilityApprovalDataMatrixA
dapter";
```

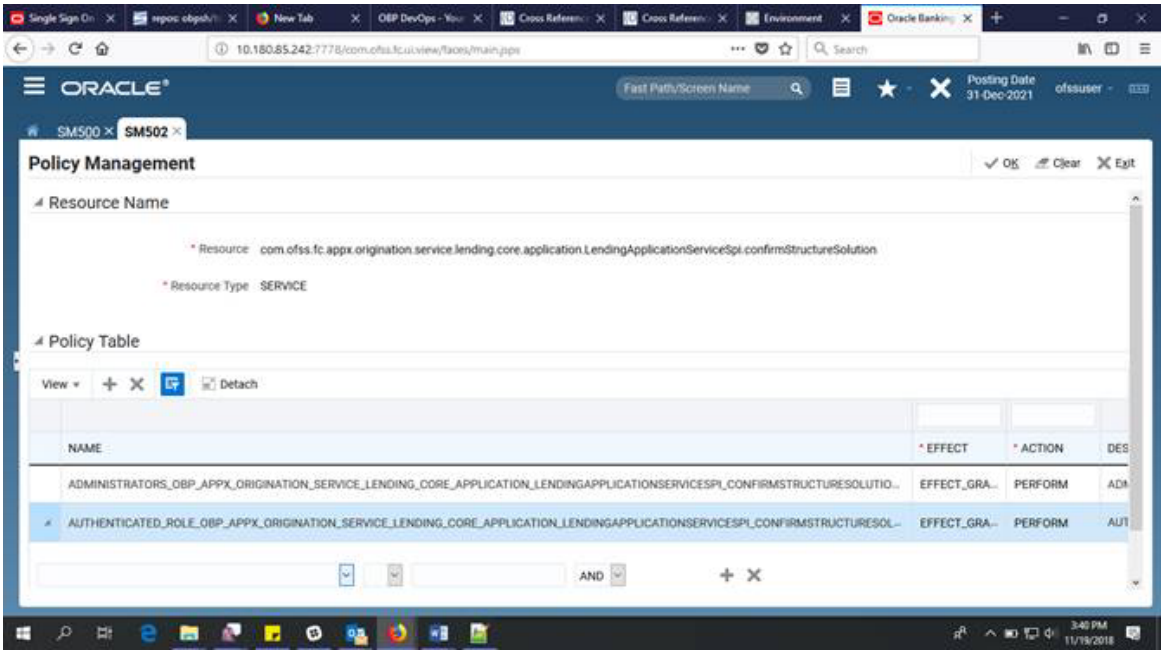
5. Add/Modify access policy/rule using SM502 screen or using ‘refreshMatrixAuthPolicies.sh’ script available in ‘PolicyStoreSetup’ utility.

12.1.1.1 Example of Matrix_auth conditional rule

After severity configuration in SM500, open SM502 screen to create Matrix_auth conditional rule for ServiceID.

1. In SM502, select ResourceType as ‘SERVICE’ and in Resource add ‘ServiceID’.and then press Enter.

Figure 12–6 Select Resource Type and Add Service ID

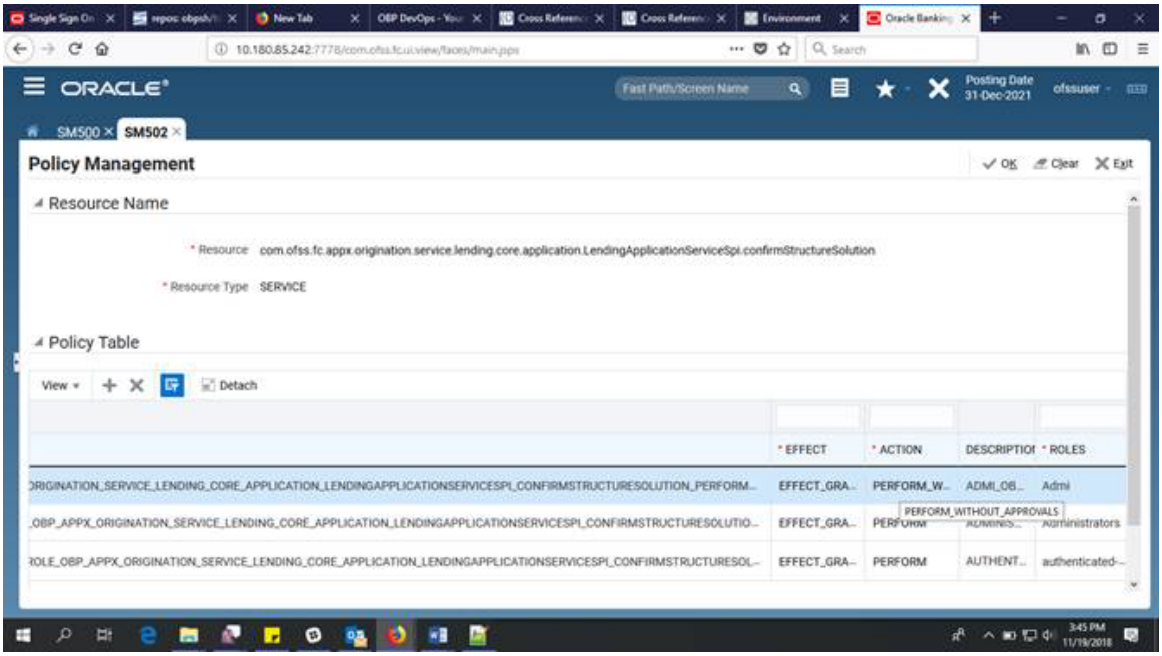


2. Now create policy for ‘PERFORM_WITHOUT_APPROVALS’ for the resource.

   Under PolicyTable, click on Add, and select following details:

   - EFFECT: GRANT
   - ACTION: PERFORM_WITHOUT_APPROVALS
   - ROLES: Administrators

Figure 12–7 Add PolicyTable Details



3. Then, add condition using attribute ‘LendingStructureSolution_Margin’.

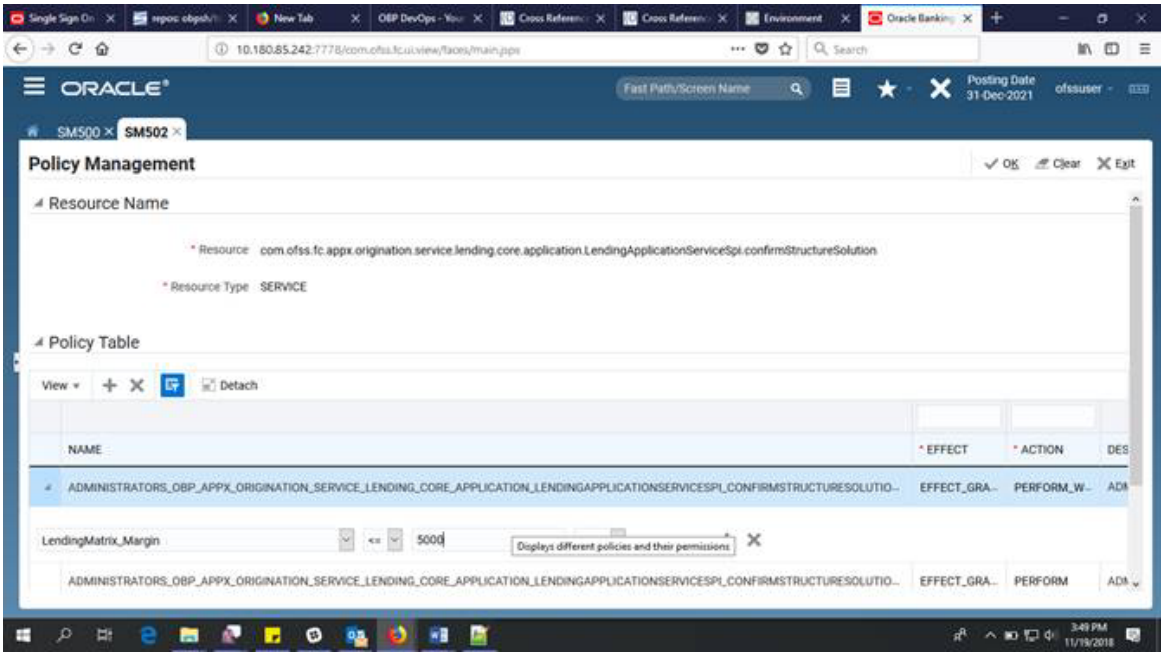
   Condition:

   If LendingStructureSolution_Margin<=5000 , then PerformWithOutApprovals

   else perform,

   So, if LendingStructureSolution_Margin>5000 then Transaction will go for Approval.

Figure 12–8 Add Condition



Note

The only difference between the policy semantics in matrix_auth and access policy is the 'Action'. ['perform' versus 'performWithoutApprovals']

12.2 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.

12.2.1 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 12–9 Add or Modify Fraud Rules in OAAM - Data Tab

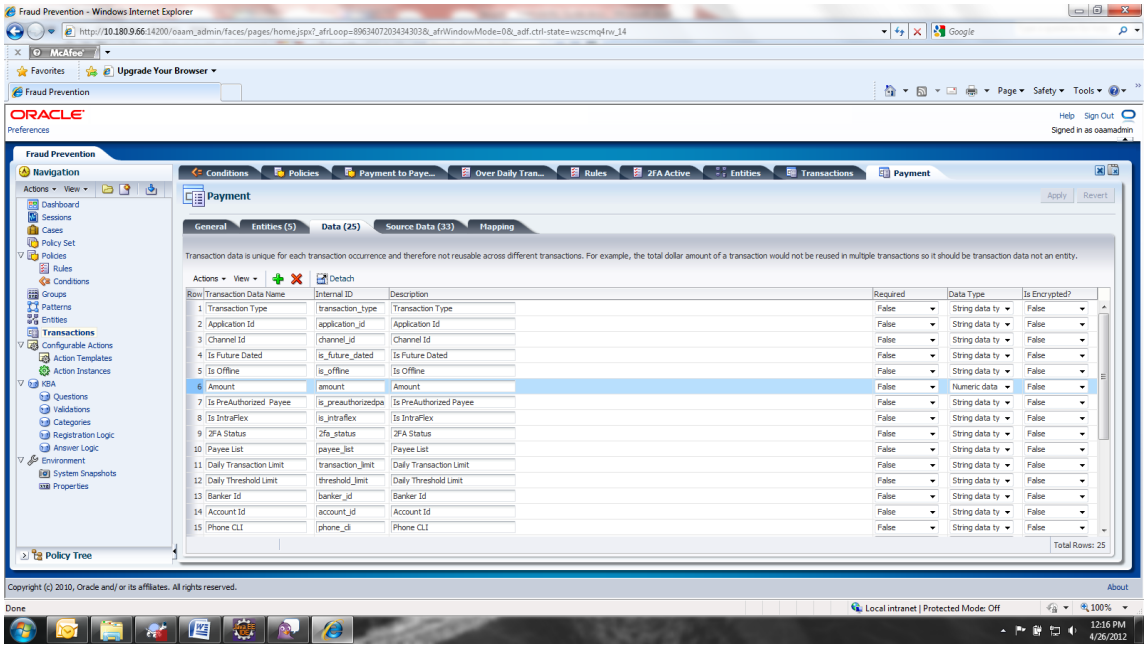
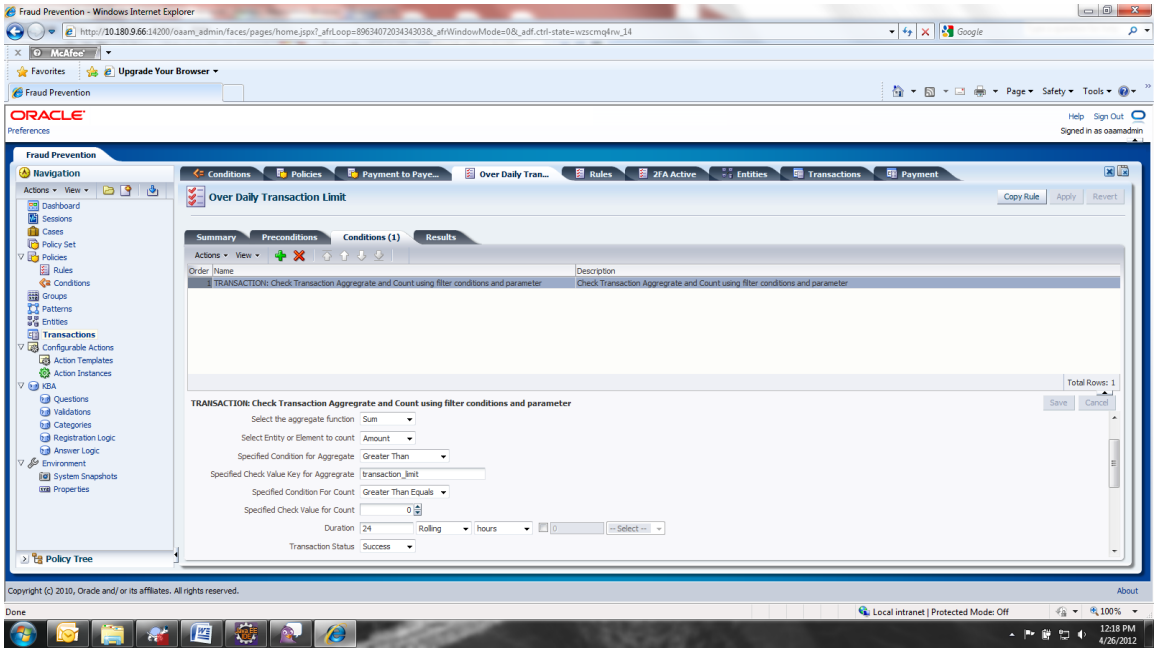


Figure 12–10 Add or Modify Fraud Rules in OAAM - Conditions Tab



12.3 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'
```

12.3.1 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.DDAInquiryApplicationServ
ice.fetchBasicDetails.CustomerValidators=RestrictedAccountApplicat
ionValidator,EmployeeAccountApplicationValidator
```

Step 2

Develop custom validator along the lines of existing adapters.

12.3.2 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.DDAInquiryApplicationServ
ice.fetchBasicDetails.AccountValidators=RestrictedAccountApplicati
onValidator,EmployeeAccountApplicationValidator
```

Step 2

Develop custom validator along the lines of existing adapters.

12.3.3 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.DDAInquiryApplicationServ
ice.fetchBasicDetails.BusinessUnitValidators=BusinessUnitApplicati
onValidator
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

12.4 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.

12.4.1 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.CustomUserSearch
Adapter.retrieveUserUsingExtendableAttributes
CUSTOM_SEARCH_PARAM=nagactualaccessid
```

Step 2

Develop custom user search adapter.

12.5 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.

12.5.1 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.

12.6 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.

12.6.1 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.roleEvaluationCriteri
a.SimpleRoleEvaluator
```

Step 2

Develop custom role evaluator.

Currently, the default role evaluator returns the role that has the maximum limits for the service.

12.7 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.

12.7.1 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.DemandDepositFundsTransfer
Service.
transferFundsToBeneficiaries=TransferFundsExclusionValidator
```

Step 2

Develop custom limits exclusions adapter.

12.8 Customizing Business Rules

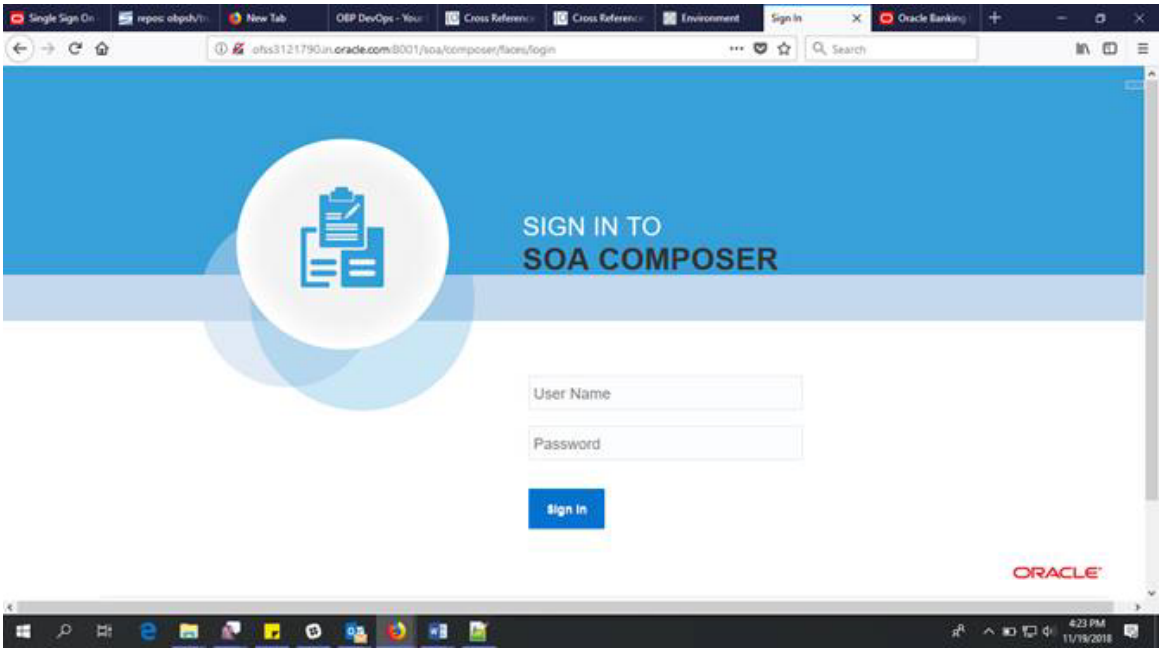
BPEL approval process business rules can be configured and it is based on input authorizations raised during transaction processing at OBP host. The steps for configuring the business rules of the approvals are given in the below section.

12.8.1 Steps to Update the Business Rules by Browser

Following is the Rules setup to be done in SOA Composer:

1. Log in to SOA Composer application of the OBP.

Figure 12–11 Log in to SOA Composer Application screen



2. Search rules file with .rules extension in search box.

   Ex: Search ‘HT_SubmissionSpi_ConfirmSubmissionRules.rules’ in search box.

Figure 12–12 Search Rules file

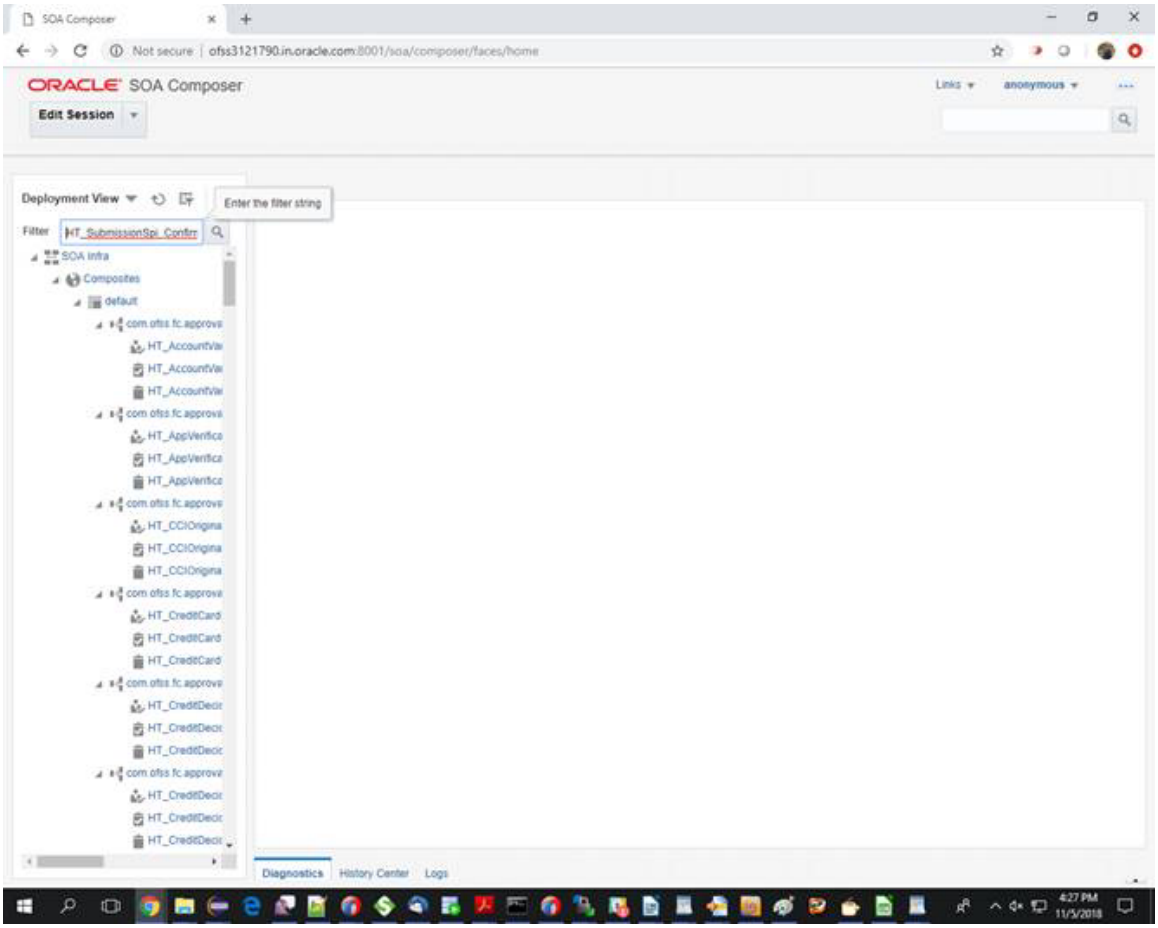
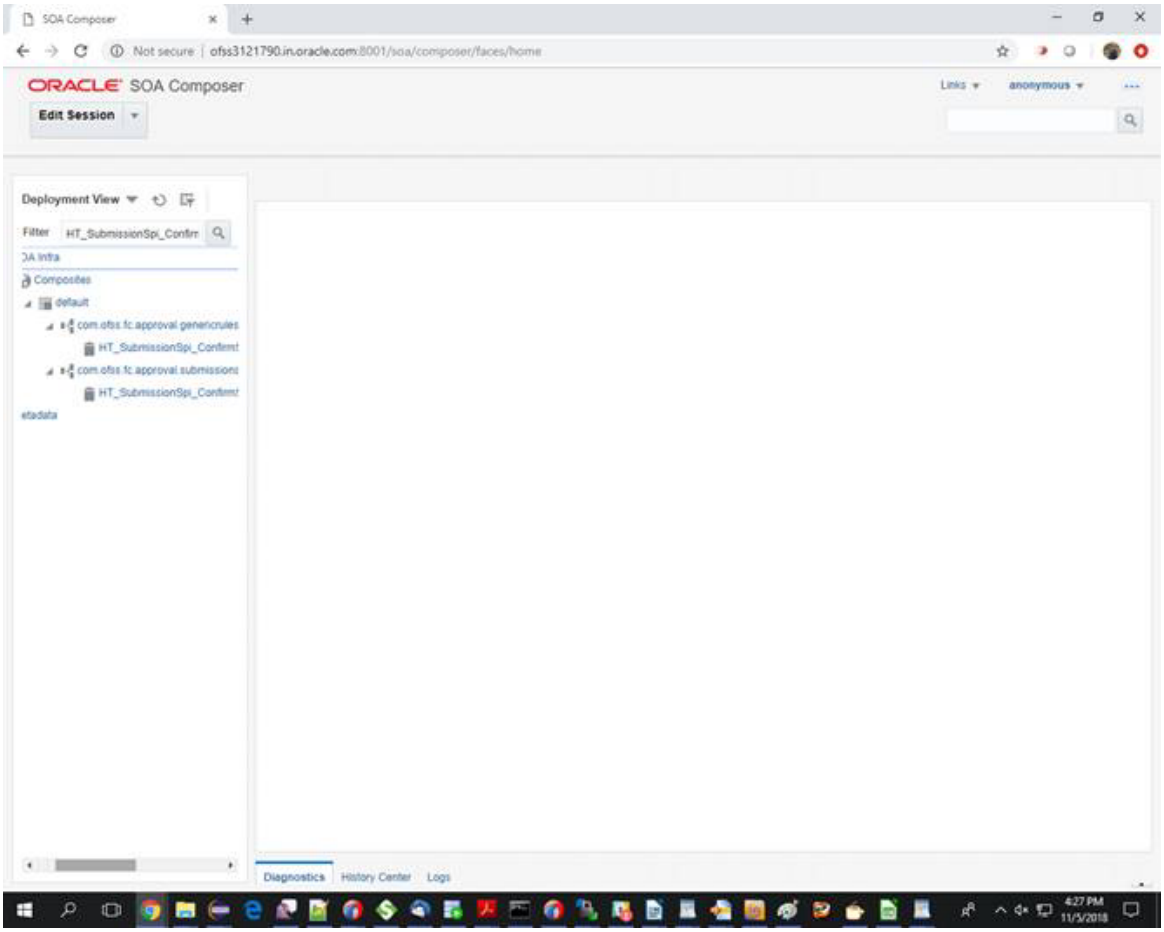
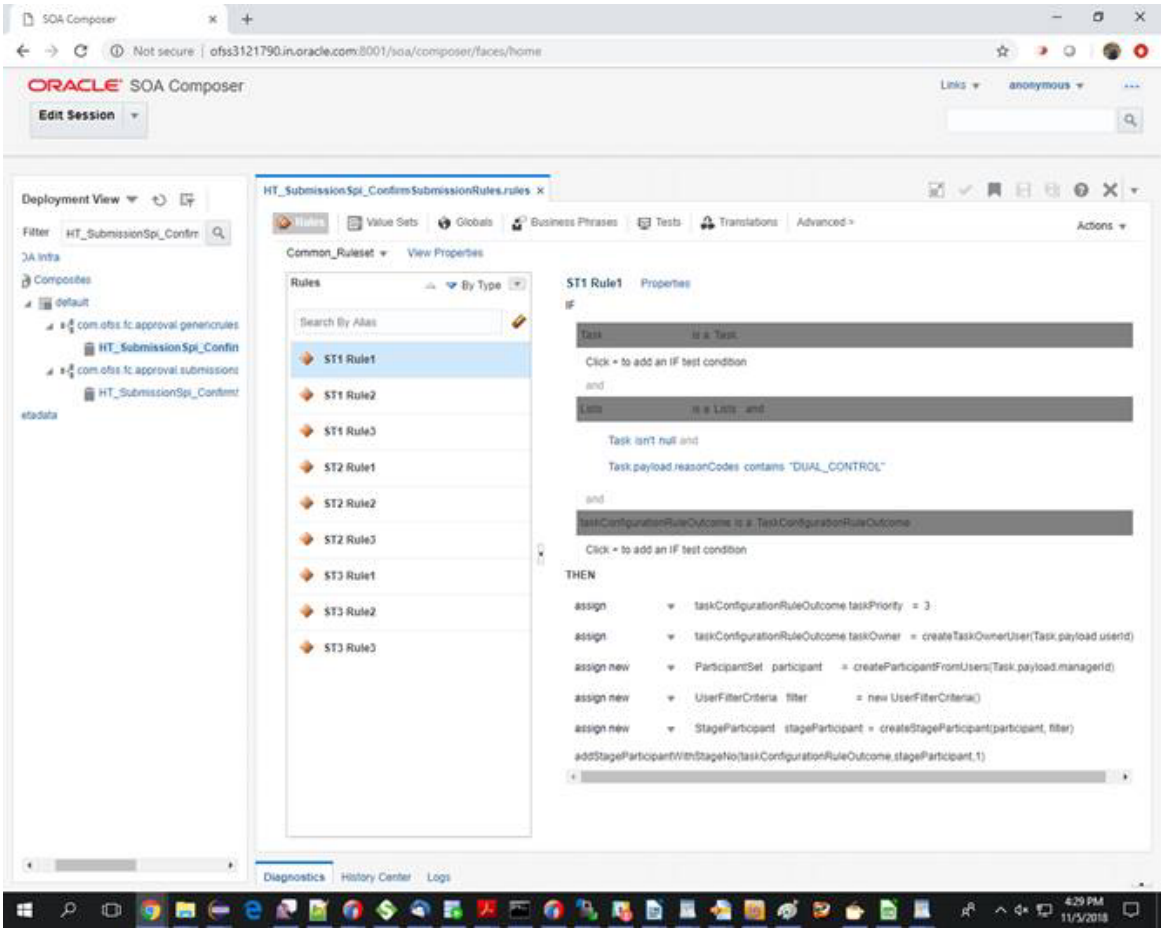


Figure 12–13 Search Rules file -View



3. Select ‘com.ofss.fc.approval.genericrulesapprovalspi.executeapprovalrulesorigcore\SOA\oracle\rules\HT_SubmissionSpi_ConfirmSubmissionRules.rules’ composite to configure routing rules.

Figure 12–14 Composite to Configure Routing Rules

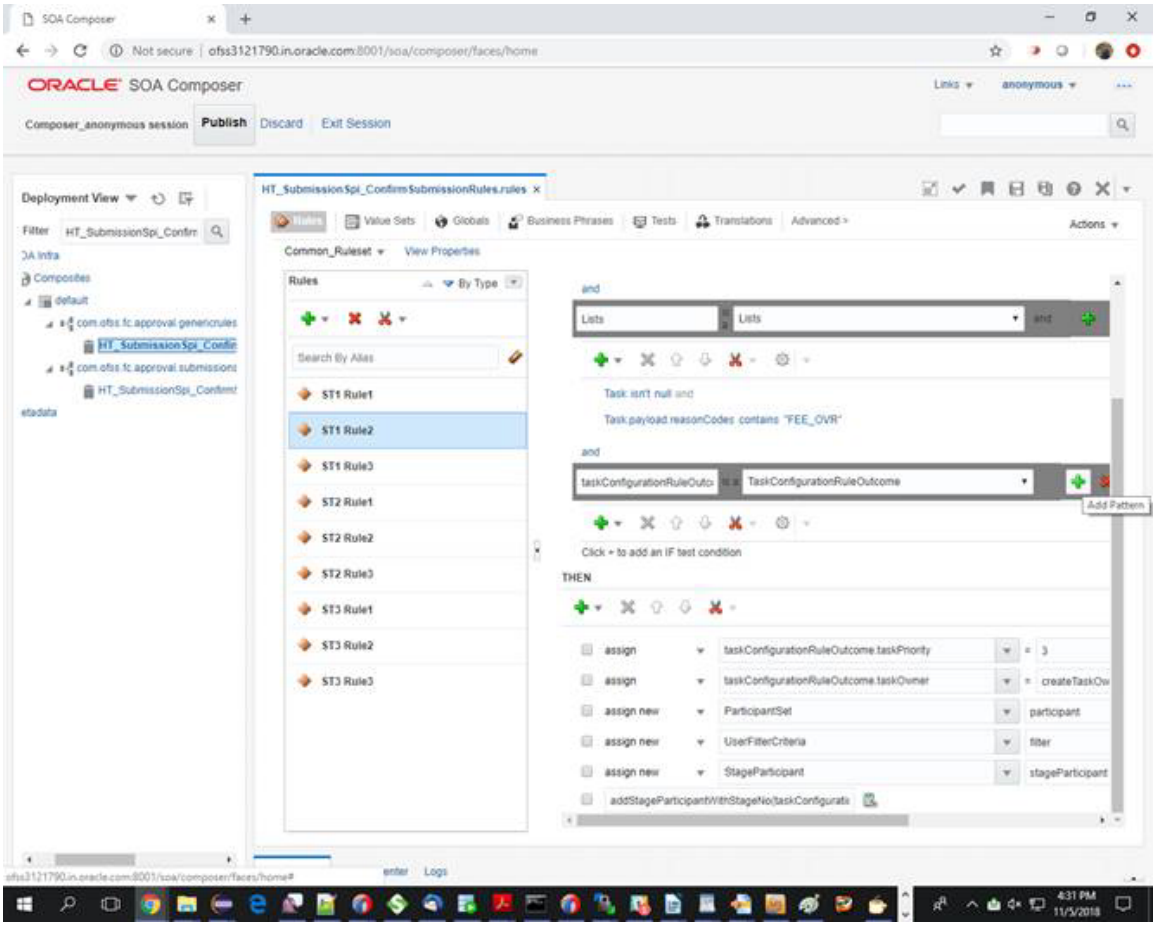


4. Select the stages of approval where the condition in rule is required to be updated.

   To edit rules file, click on the ‘EDIT Session’ button and then select RuleID under ‘common_ruleset’.

   In the below case, select ‘ST1Rule2’ and then, add new condition using ‘Add Pattern’ option.

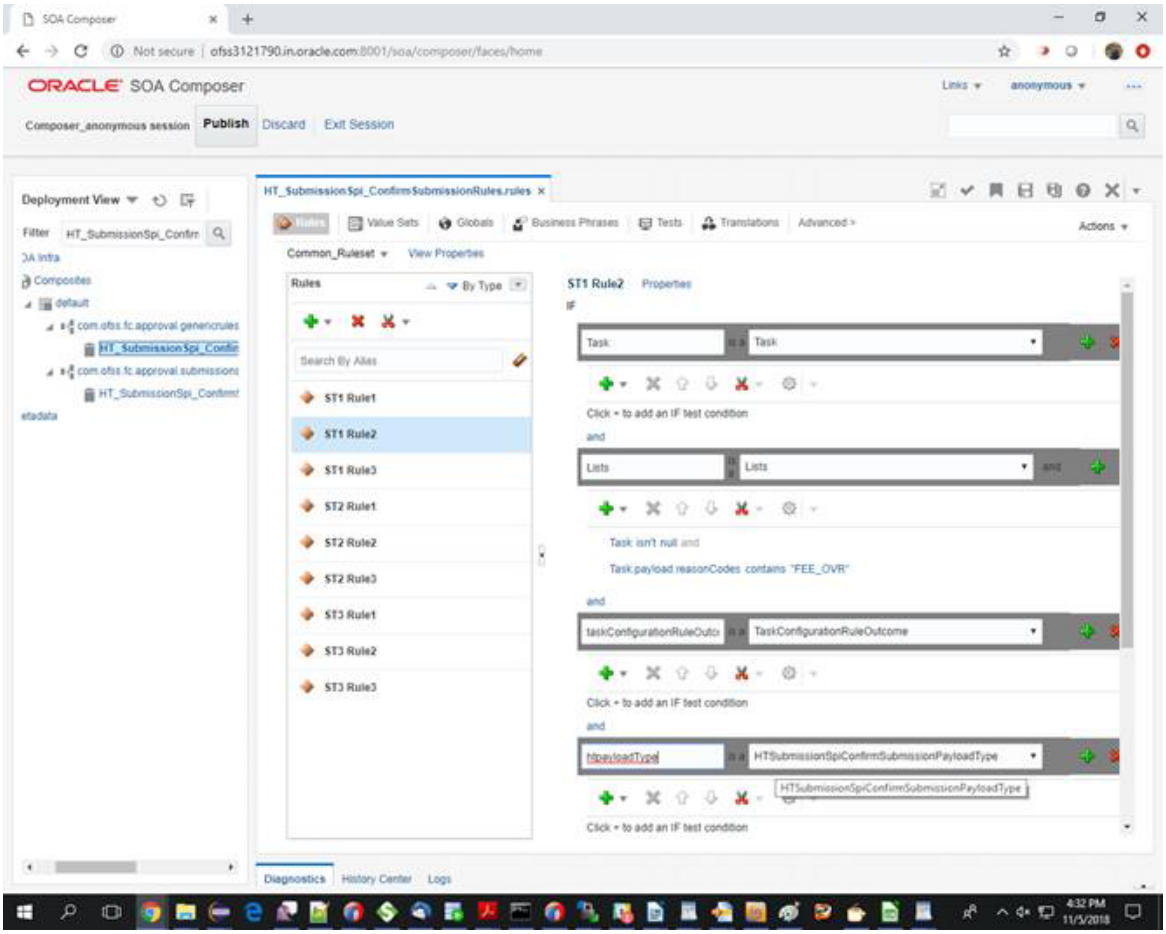
Figure 12–15 Stages of Approval



5. On clicking the Add Pattern, select fact type from drop down and then select payloadType.

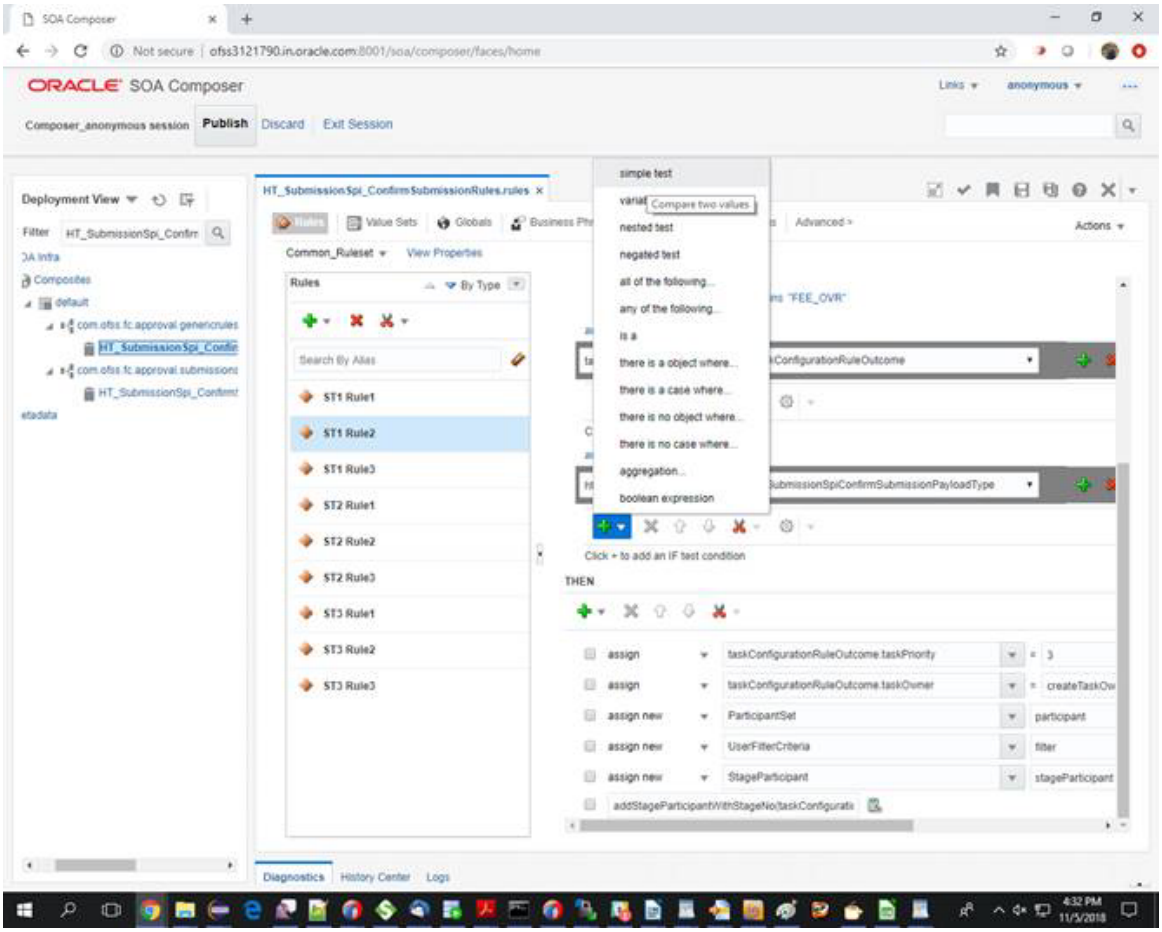
   Ex: select ‘HTSubmissionSpiConfirmSubmissionPayloadType’ from drop down as shown below:

Figure 12–16 Add pattern-Submission Payload Type



6. 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. You can add new test using below option.

Figure 12–17 Add New Test

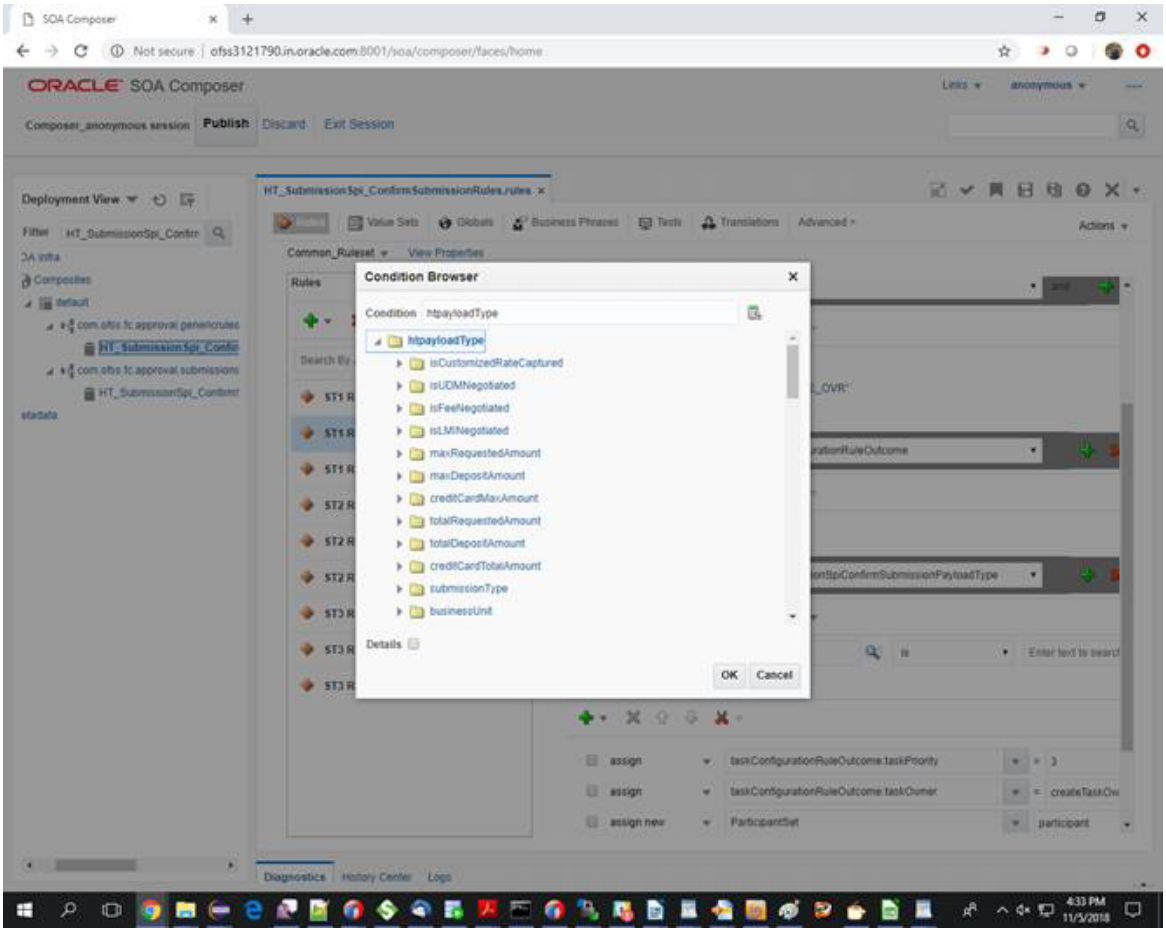


7. After selection of the test condition, the new expression row appears where the variable, the operator and the expression value can be selected.

   On selection of the search button next to the variable select box, the condition browser opens where the specific task can be selected.

   Select 'fact' in left side expression with appropriate value on right side.

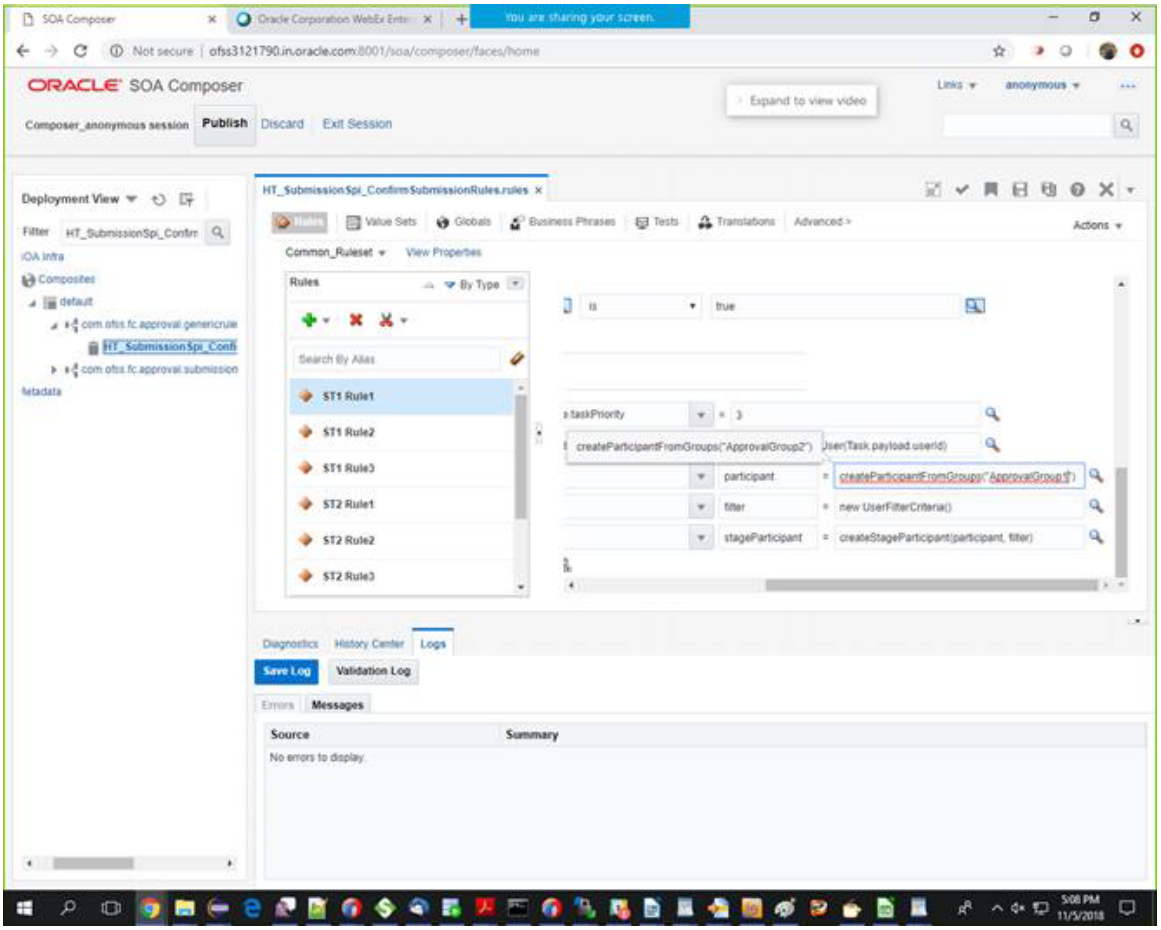
Figure 12–18 Select Fact with Appropriate Value



8. If you want to assign to a particular group then use below function:

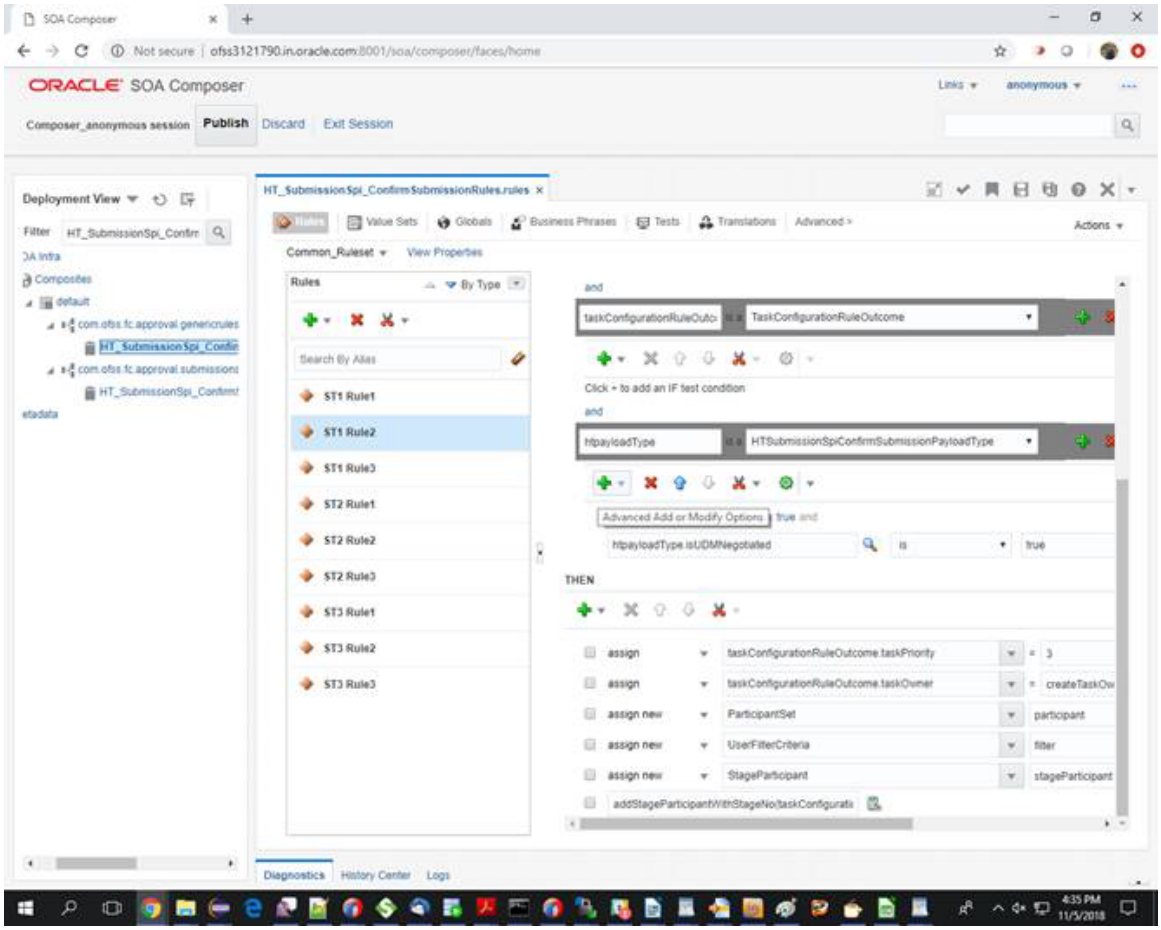
   createParticipantFromGroups("ApprovalGroup2")

Figure 12–19 Create Participant From Group



9. Validate rules file and then save all changes and publish.

Figure 12–20 Validate Rules File



12.8.2 Steps to Update the Business Rules in JDeveloper

Following are the steps to update the business rules in JDeveloper.

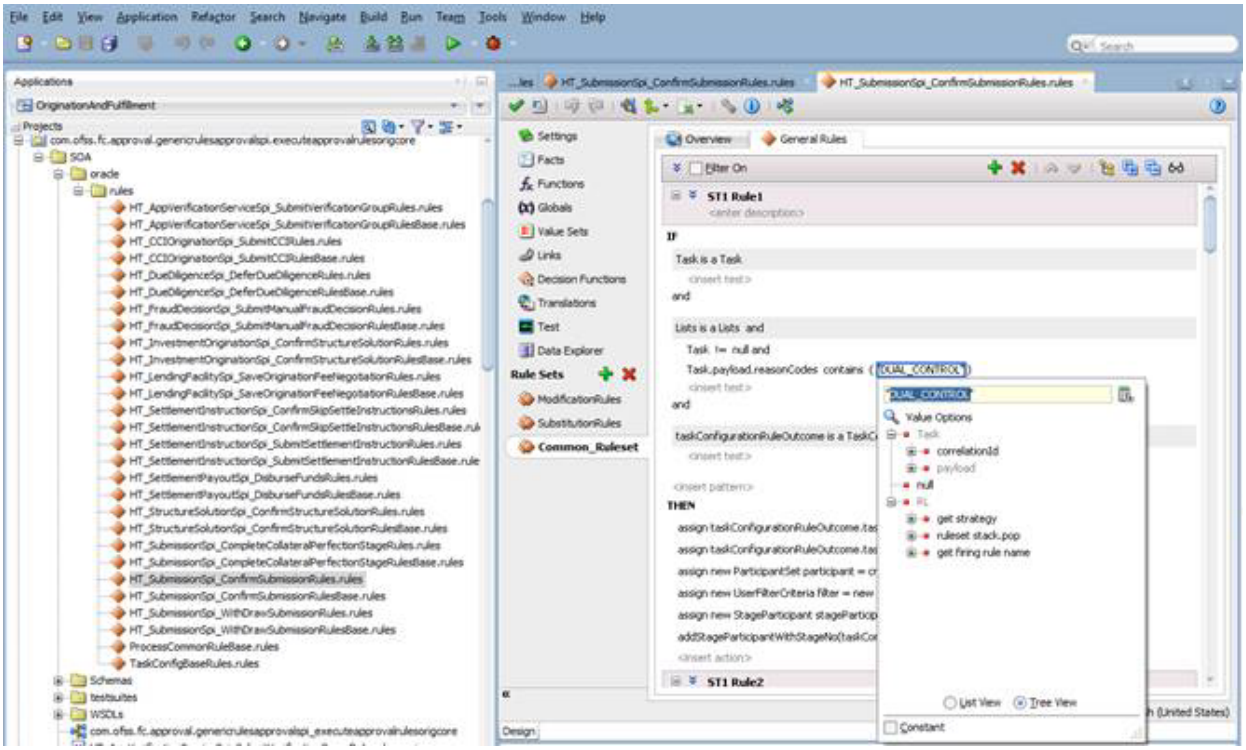
Step 1

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 the Oracle Banking Enterprise Product Manufacturing SOA Extensibility Guide in the SOA Customization section.

Step 2

Expand the Business Rules which are inside oracle folder of your SOA 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.

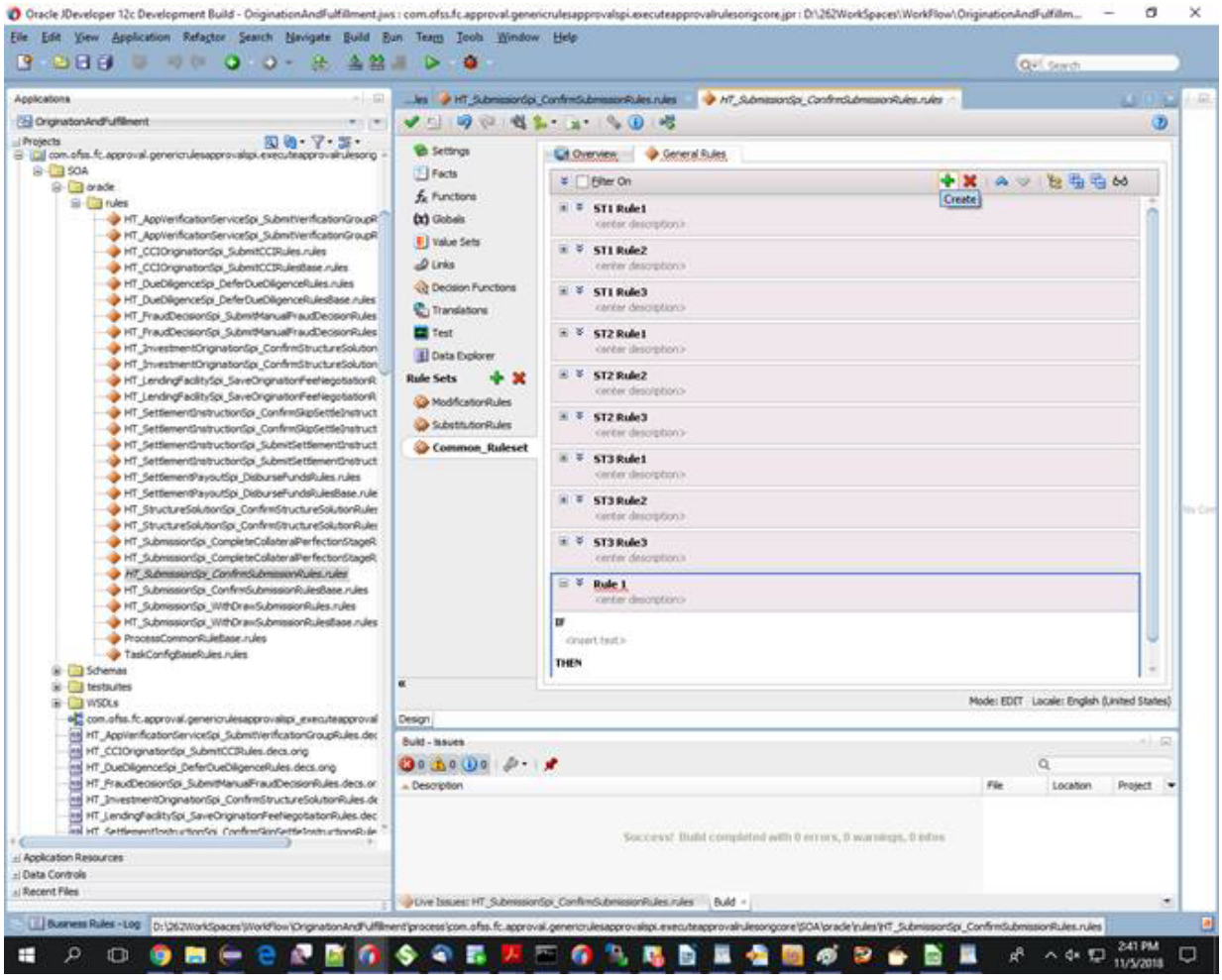
Figure 12–21 Expand Business Rules



Step 3

Create a new rule in the format 'ST<Stage Number>_Rule<Rule Number>' by clicking the Create button in the Rulesets. The new rules/decision table can be added to a stage. Populate the rule with the conditions in 'if' and 'then' block.

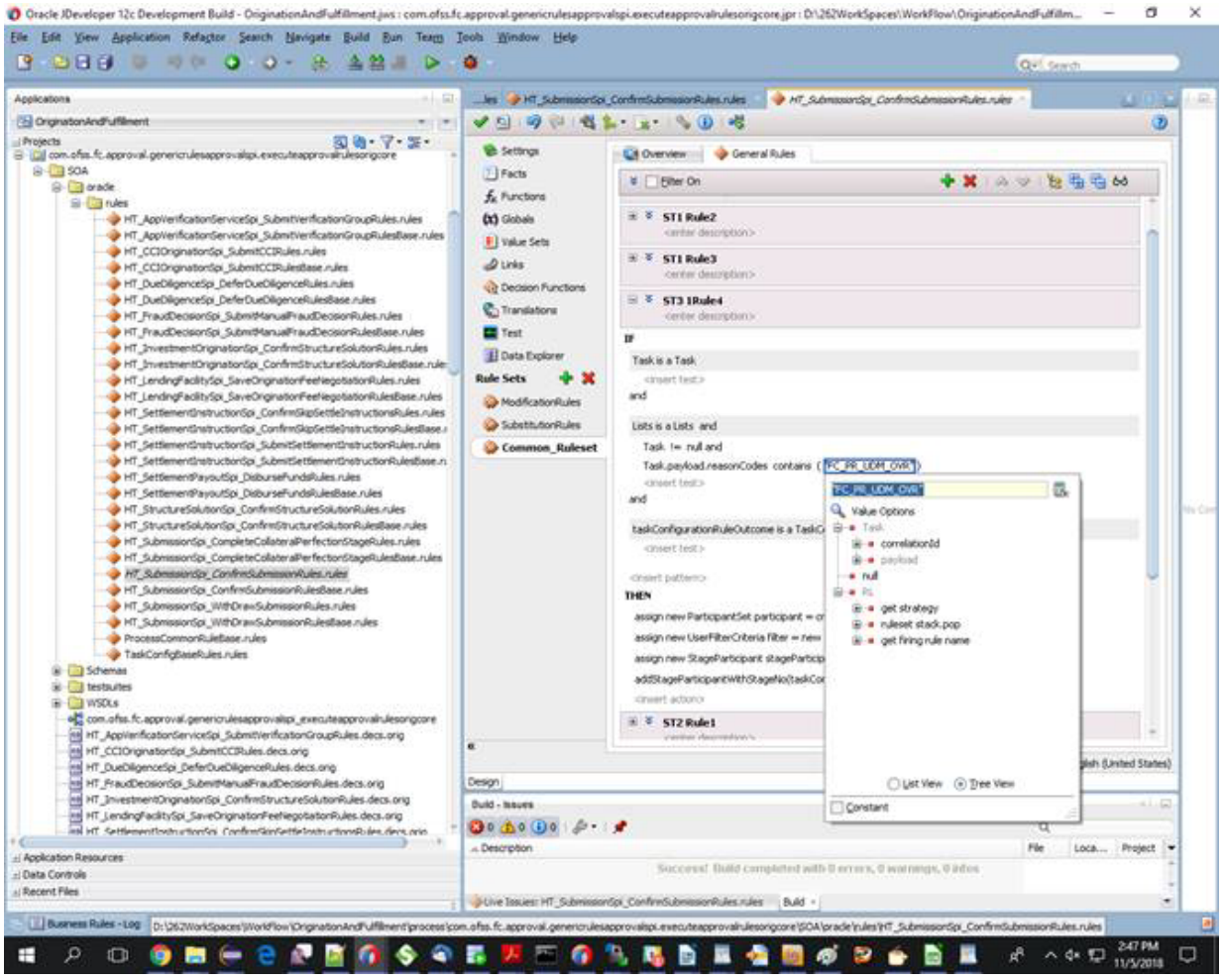
Figure 12–22 Create New Rule



Step 4

In the exiting stage, existing rule can also be added or modified.

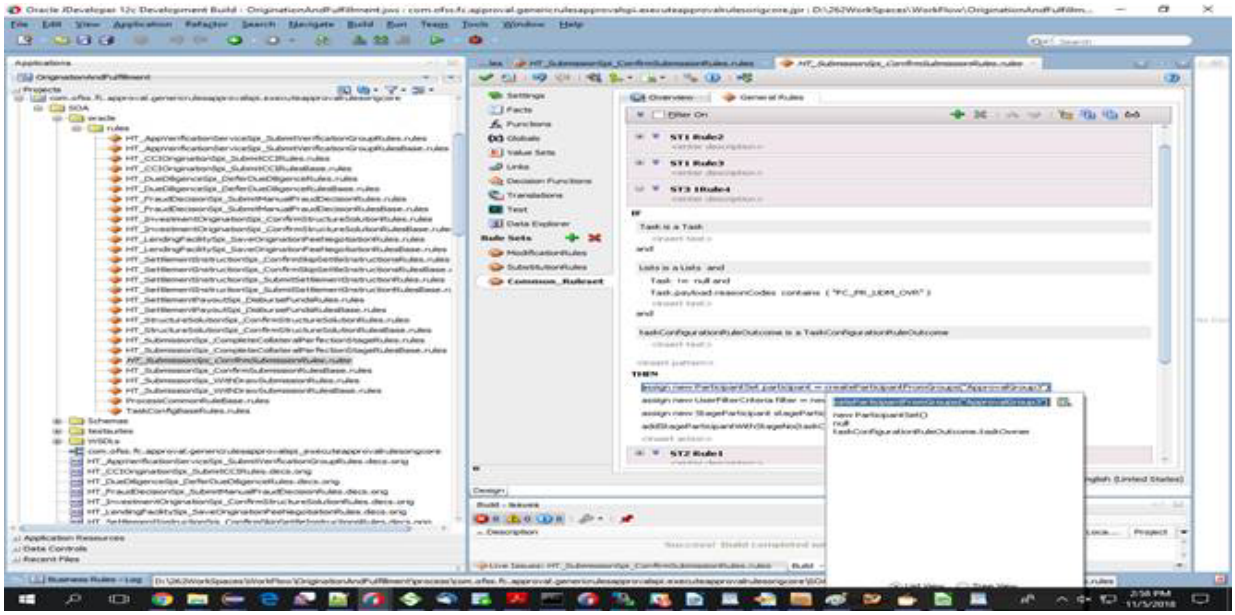
Figure 12–23 Existing Rule



Step 5

Change the participant or approval group by editing below function.

Figure 12–24 Change the approval group



Step 6

Deploy the project jar as explained in the Oracle Banking Enterprise Product Manufacturing SOA Extensibility Guide in the SOA Customization section.

> **Note**
>
> Make sure to activate rules which have to be executed by editing properties of each rules.

13 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.

13.1 Adding a New Algorithm

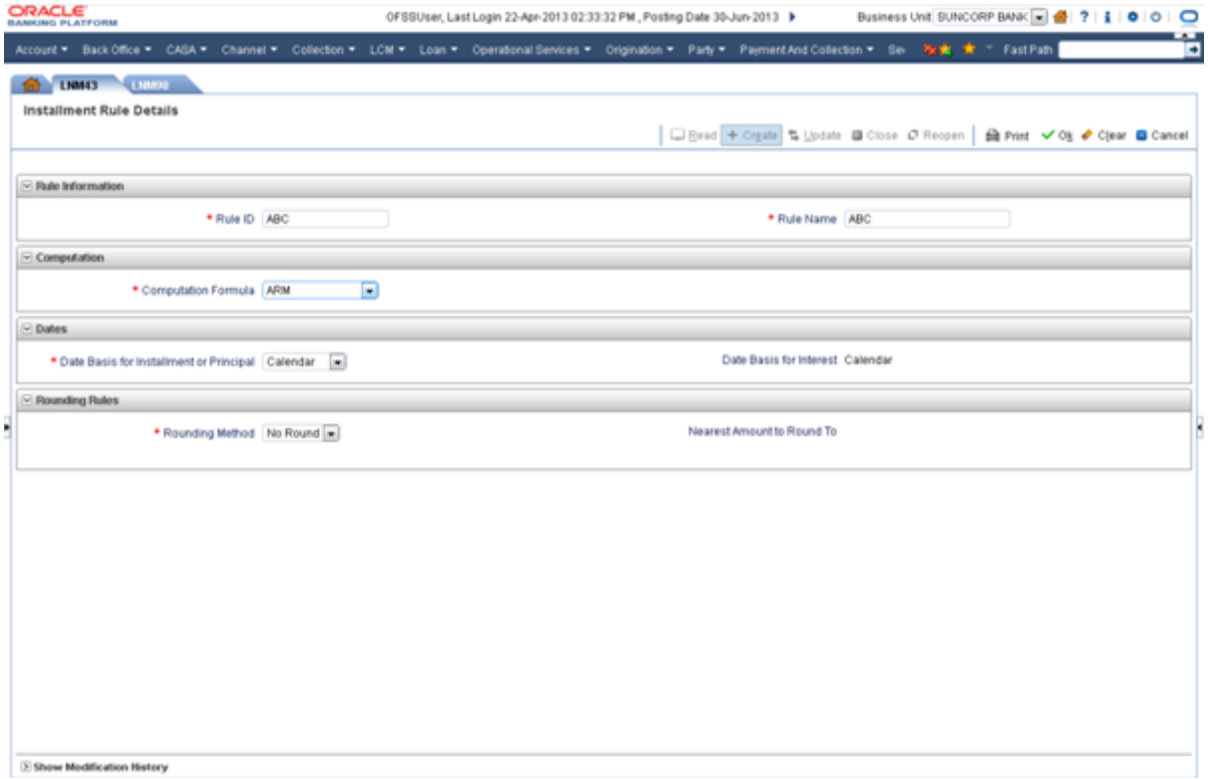
For adding a new algorithm following additions need to be done.

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.

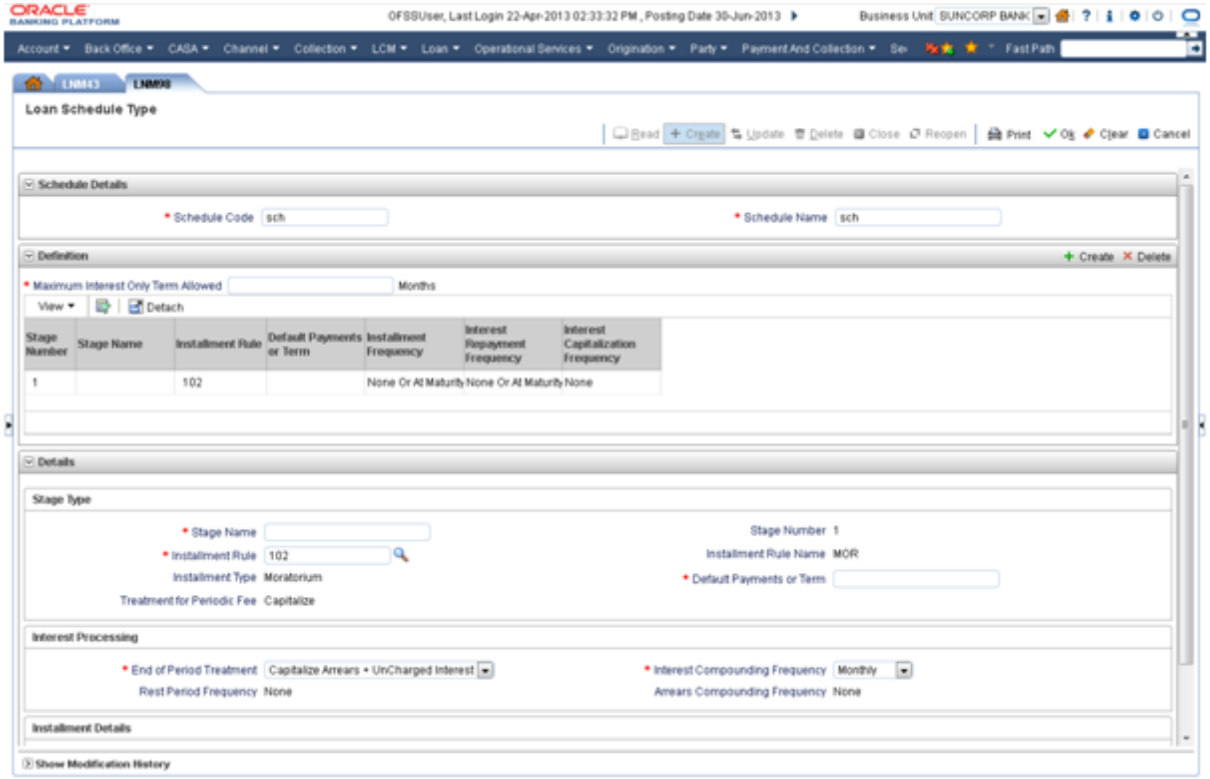
An entry has to be made in this file to appear as a choice in the drop-down list.

Figure 13–1 Add New Algorithm



This screen is used to create a new Installment Rule. For example: ABC. We can choose the new algorithm for the new rule.

Figure 13–2 Create New Installment



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.NewPrincipalRepaymentSc
heduleGenerator;
```

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
().createScheduleGeneratorInstance
(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.EPIARMRepaymentStag
eGenerator
```

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())
```

13.2 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;
```

14 Facts and Rules Configuration

This chapter explains the facts and rules configuration details.

14.1 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.

14.1.1 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
- JPQL DataSource - JPQL 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.
- **Fact Enum Definition** - Definition to Derive Permissible values for a fact. It is used mostly in Rule Creation.

14.1.2 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.

14.1.3 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.

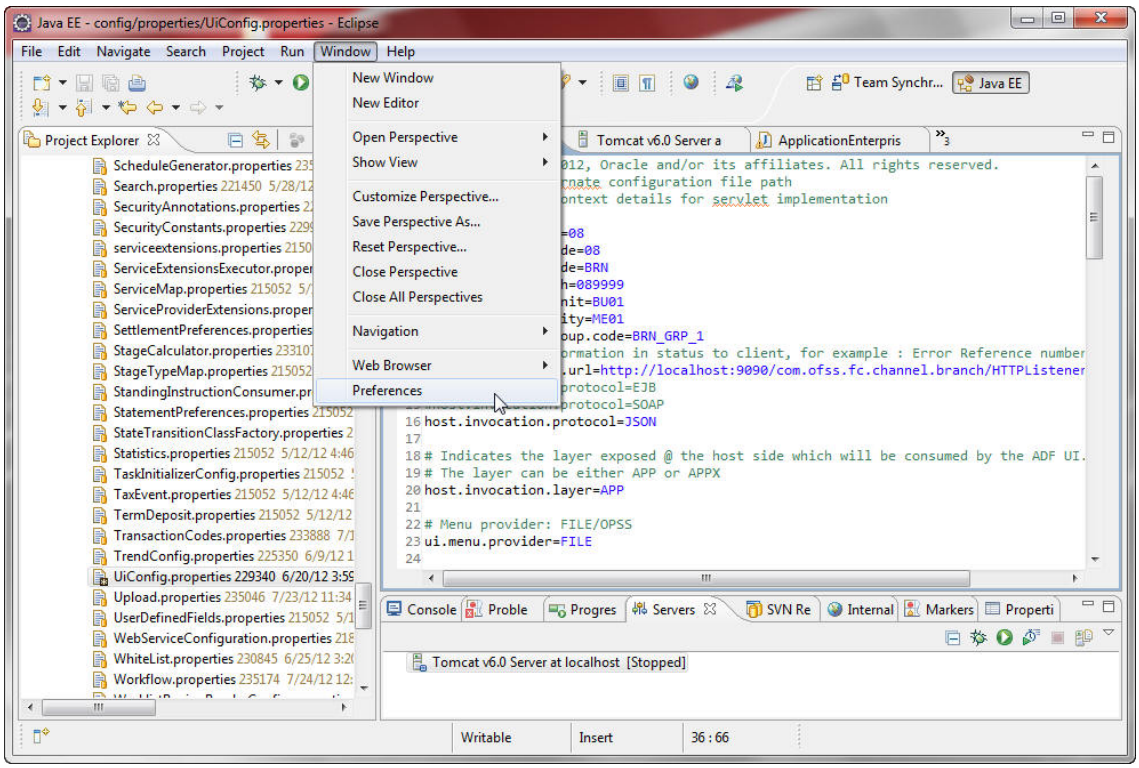
Get the Fact Plugin from the development team.

Put the latest fact generation plugin (**com.ofss.fc.util.plugin.fact_x.x.x.jar**) in the plug-in folder of eclipse.

Restart Eclipse

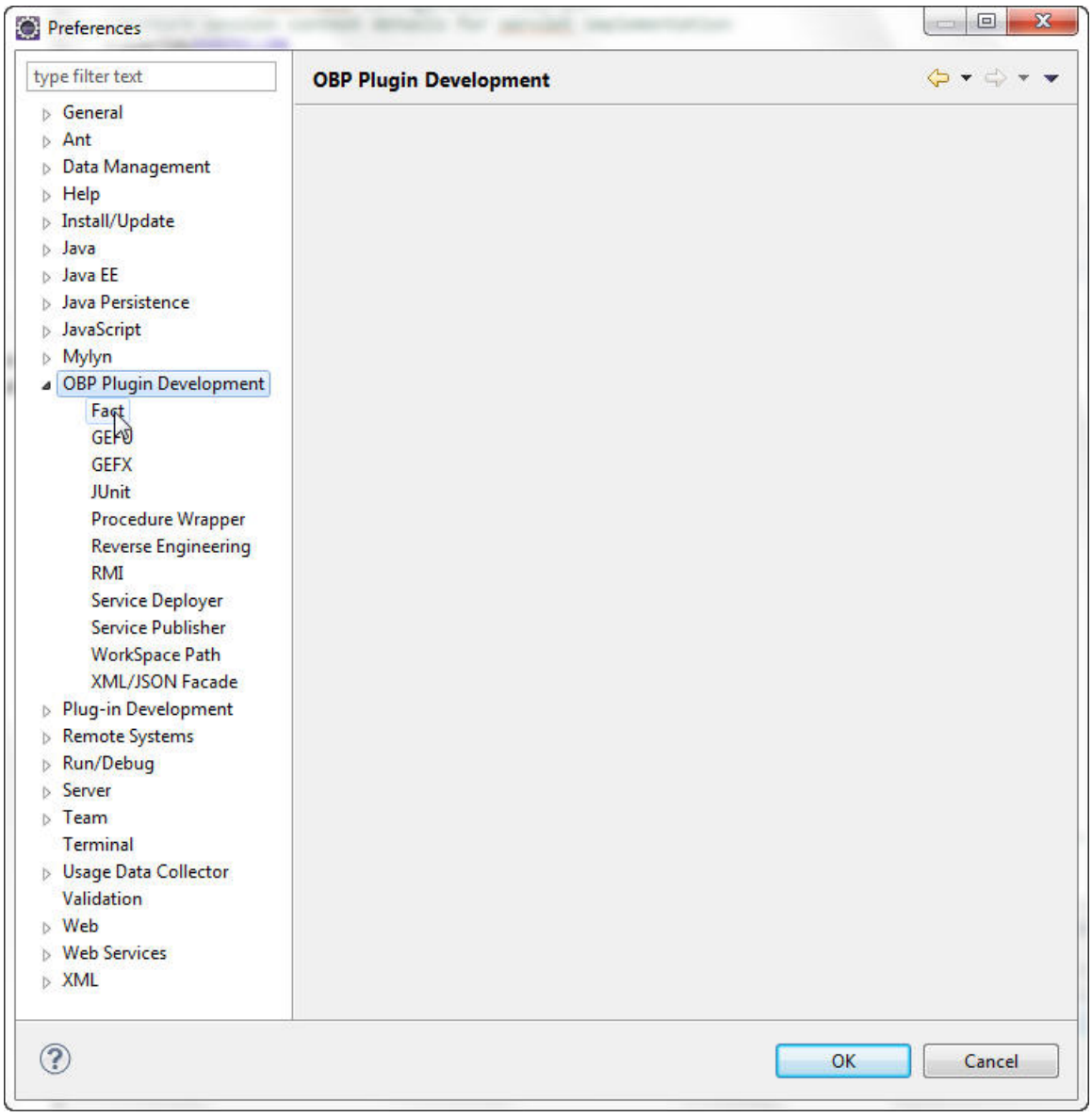
1. In eclipse, go to Window -> Preferences.

Figure 14–1 Select Window Preferences



2. Now in Preferences Window, go to **OBP Plugin Development -> Fact**.

Figure 14–2 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

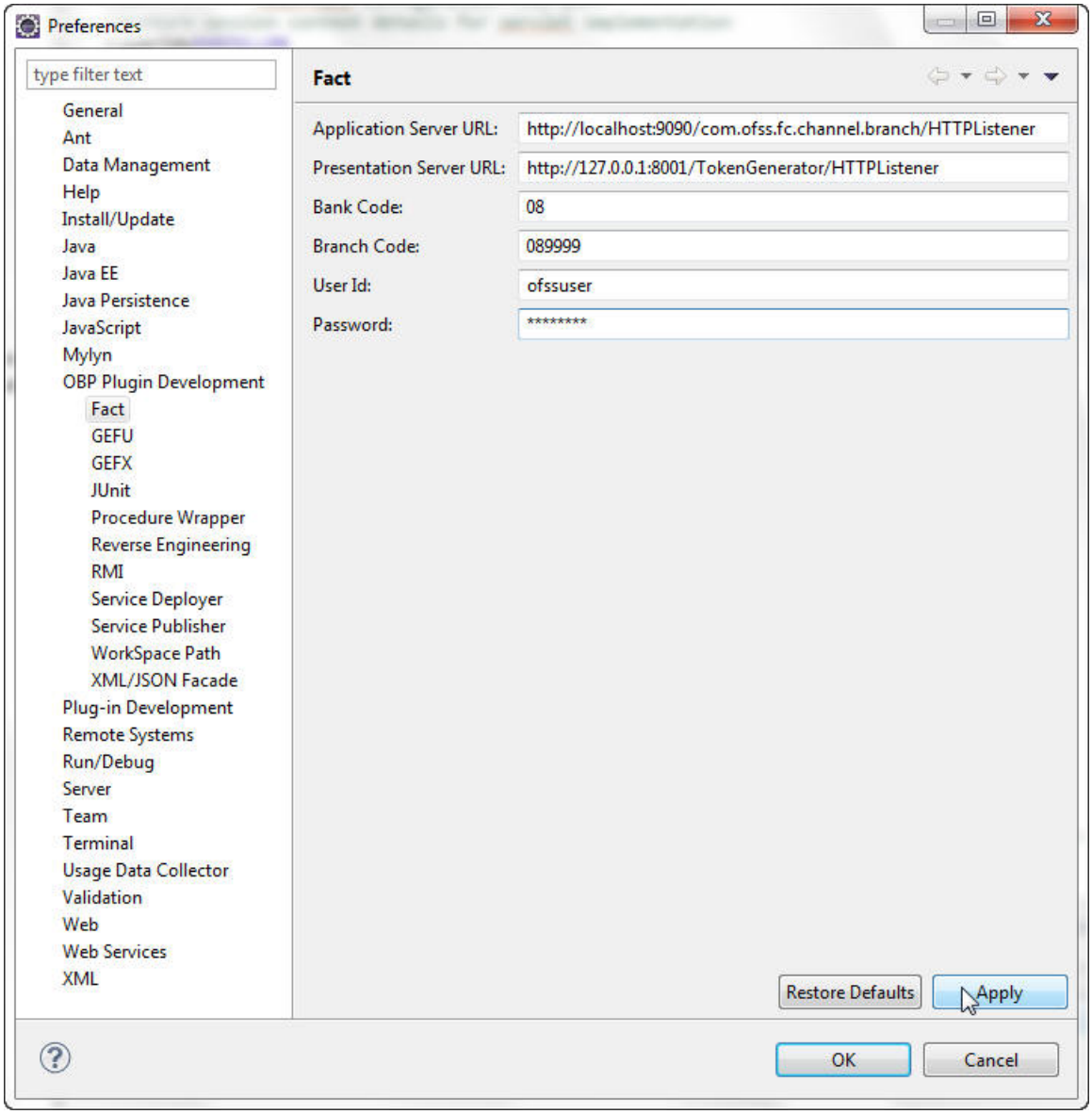
     Example: http: //127.0.0.1:8001/TokenGenerator/HTTPListener

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 14–3 Enter the Preferences Fact values



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 14–4 Fact Properties - aggregateCodeFilePath

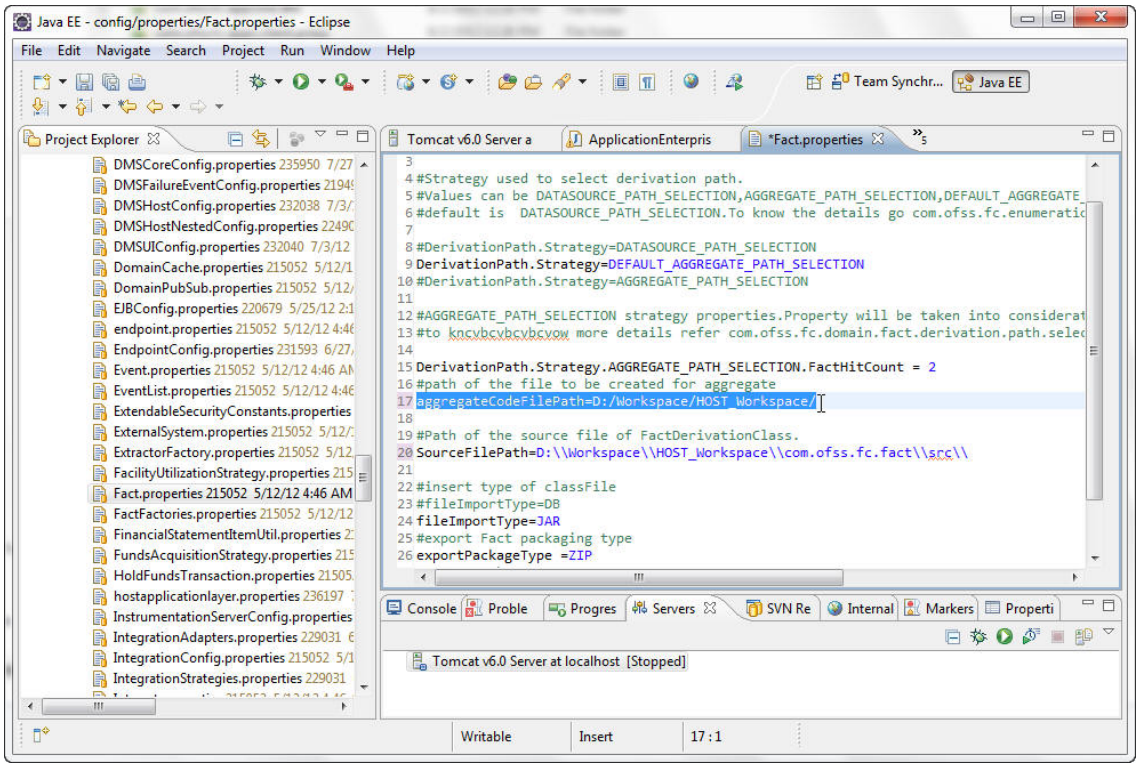
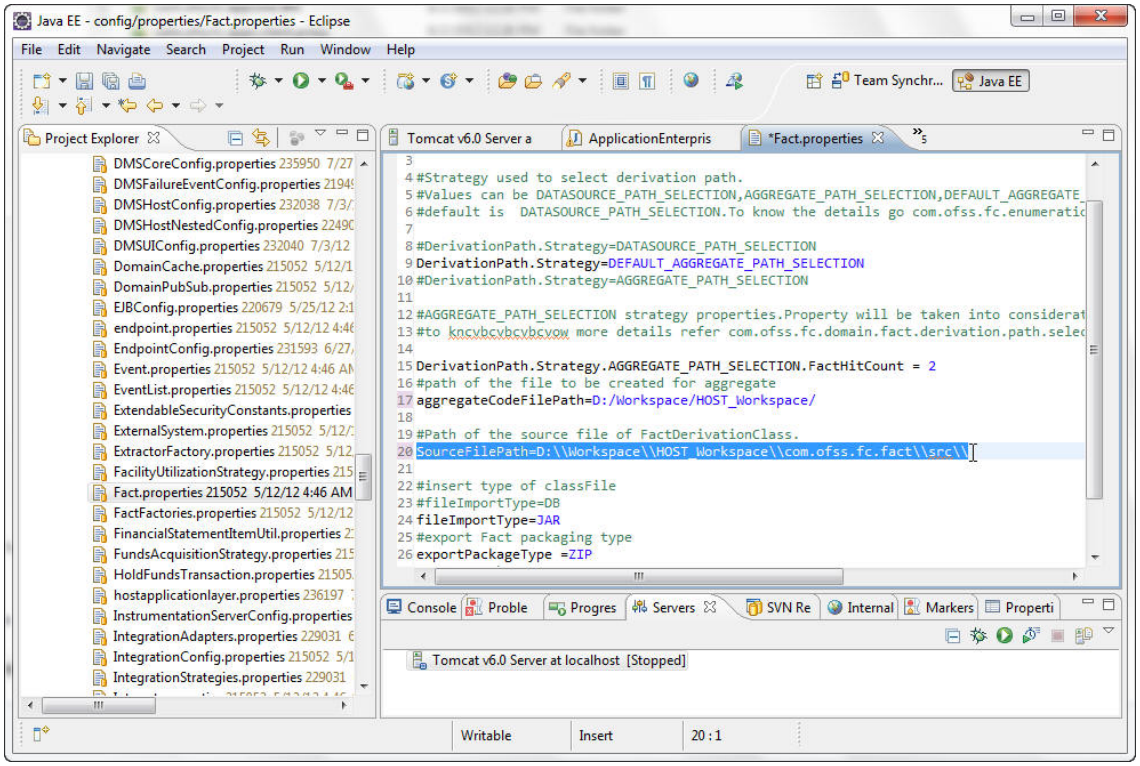
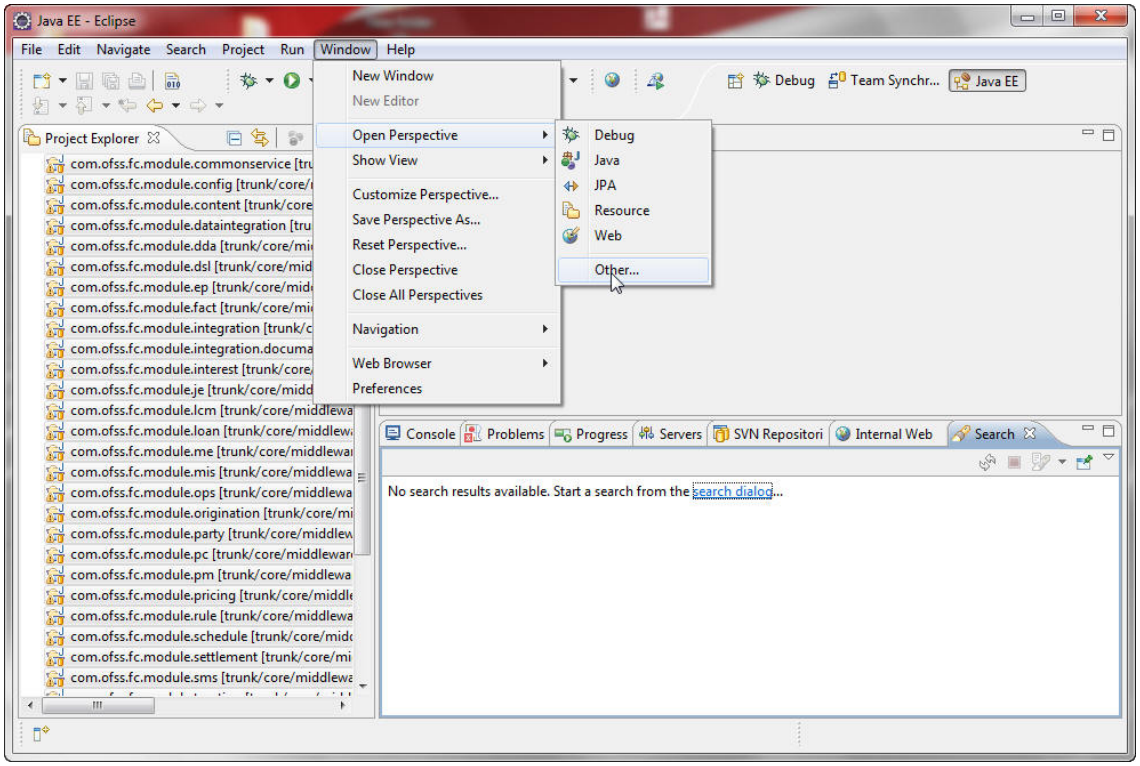


Figure 14–5 Fact Properties - sourceFilePath



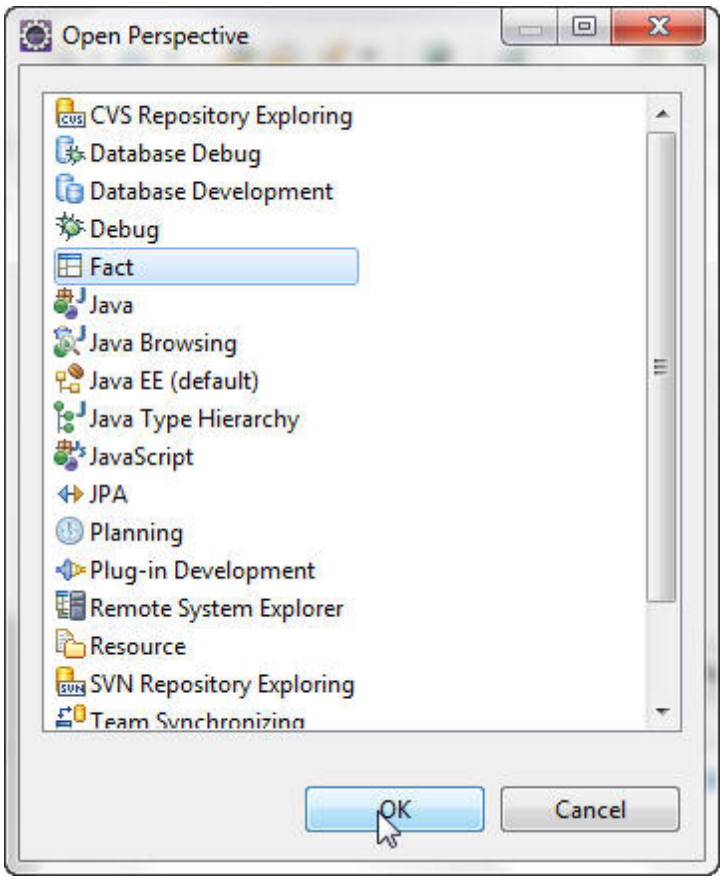
6. Now start the Host server.
7. In eclipse, go to Window -> Open Perspective -> Other.

Figure 14–6 Start Host Server



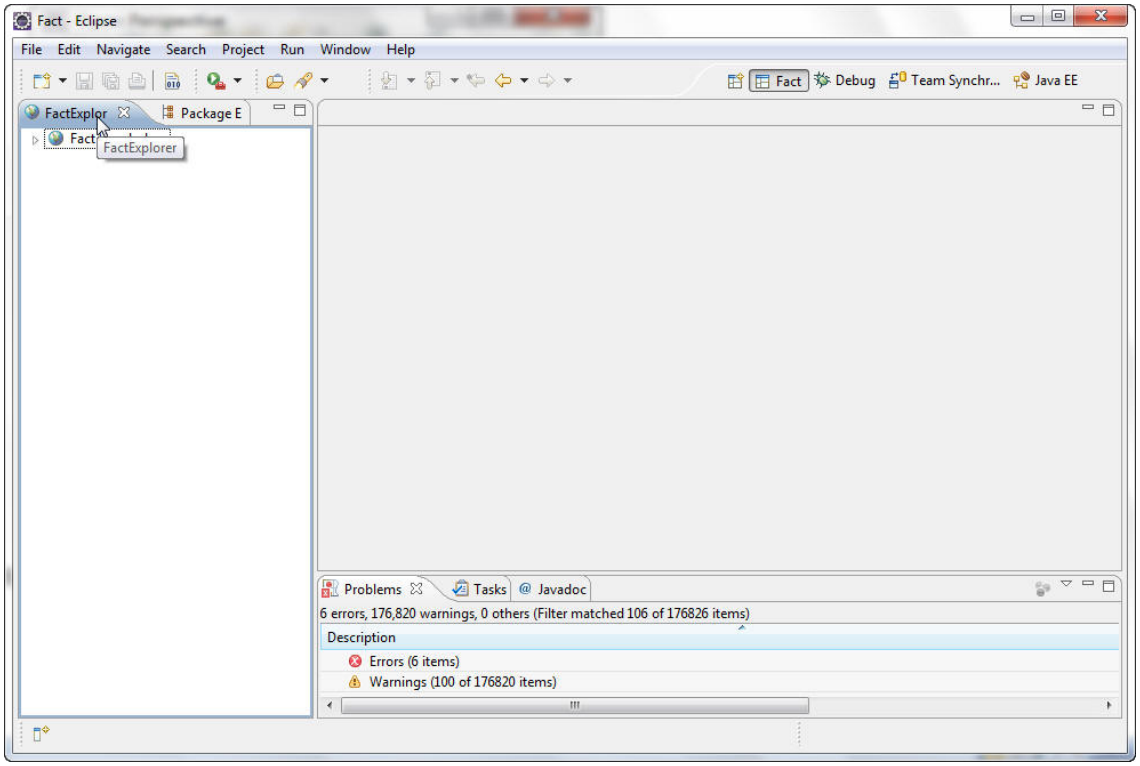
8. Now in Open Perspective window select **Fact**.
9. Click **Ok**.

Figure 14–7 Select Open Perspective value



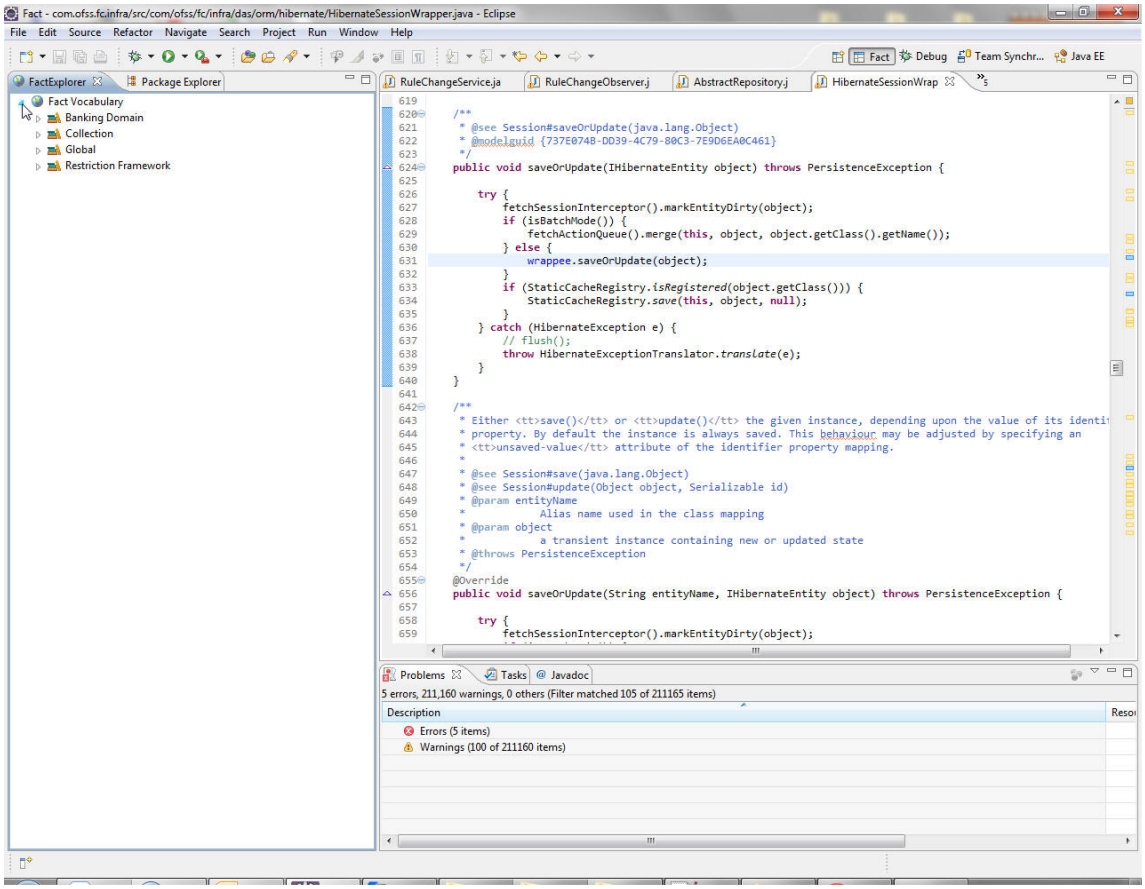
It will open **Fact Explorer** perspective, where **Fact Vocabulary** is available.

Figure 14–8 Fact Explorer



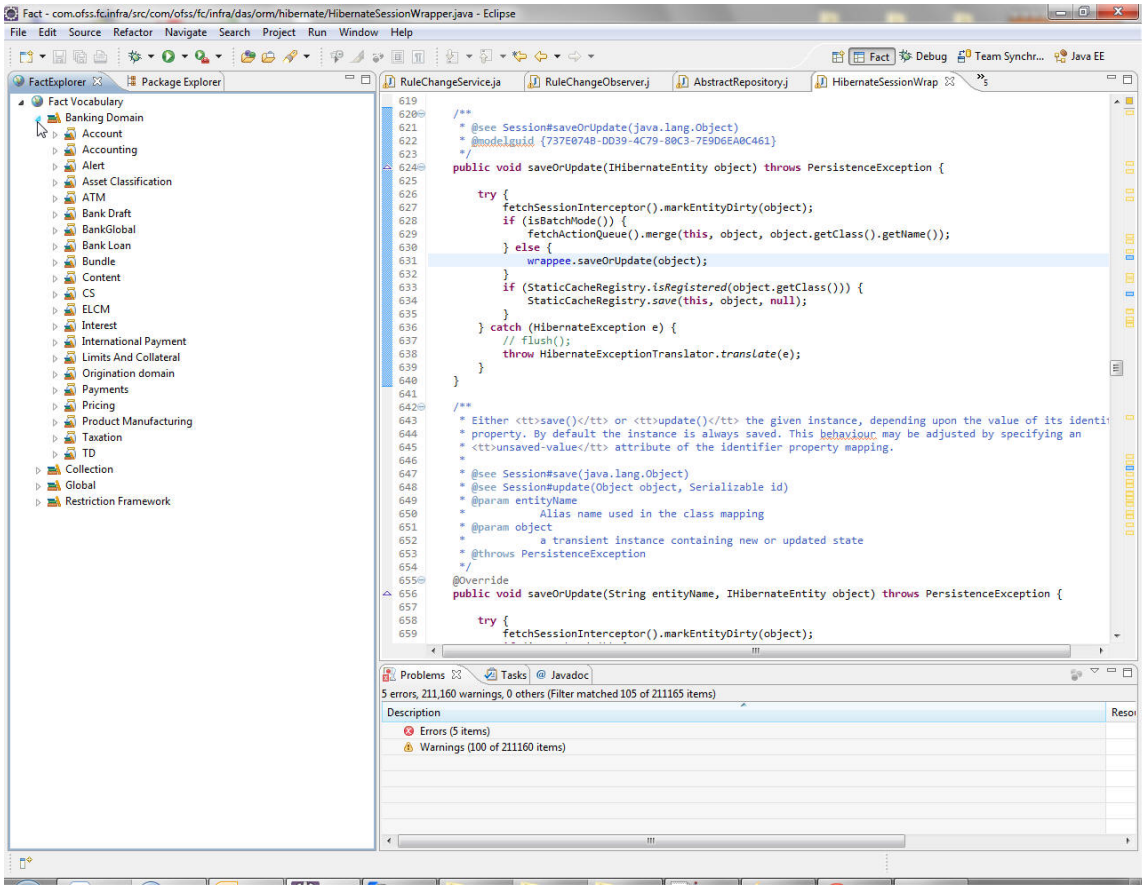
10. Now refresh and expand **Fact Vocabulary**. Expanding Fact Vocabulary will show the **Domain** names.

Figure 14–9 Fact Vocabulary



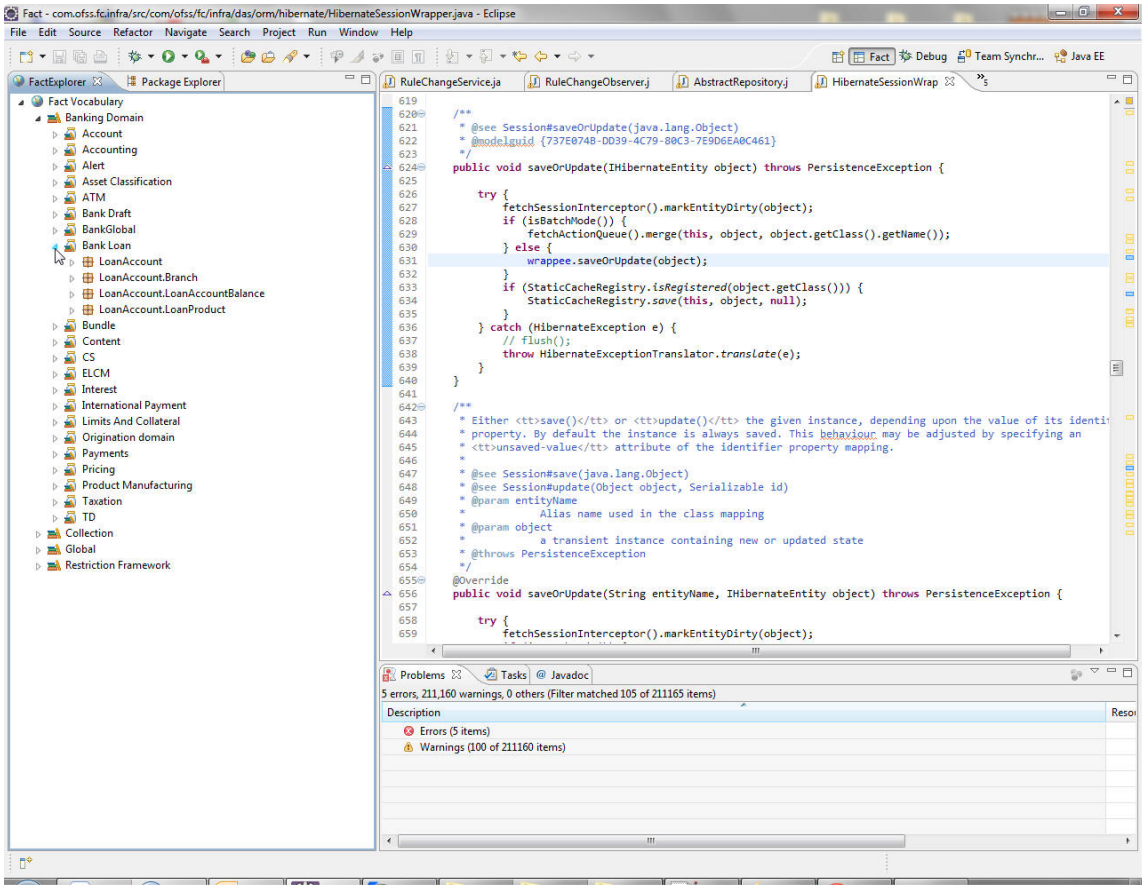
Each Domain contains its **Domain Category** names.

Figure 14–10 Domain Category



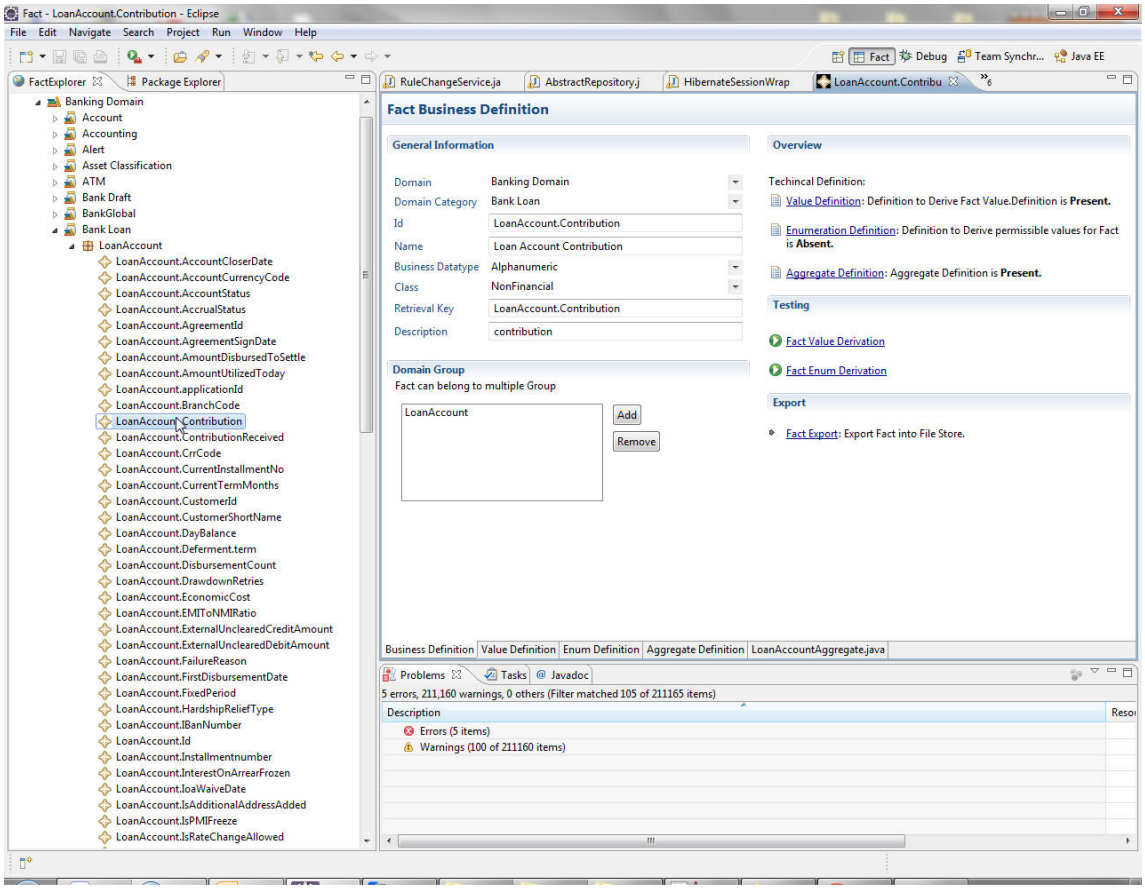
Each Domain category contain its **Fact Groups**

Figure 14–11 Fact Groups



Each Fact Groups contains its **Facts**.

Figure 14–12 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 14–13 Business Definition Tab

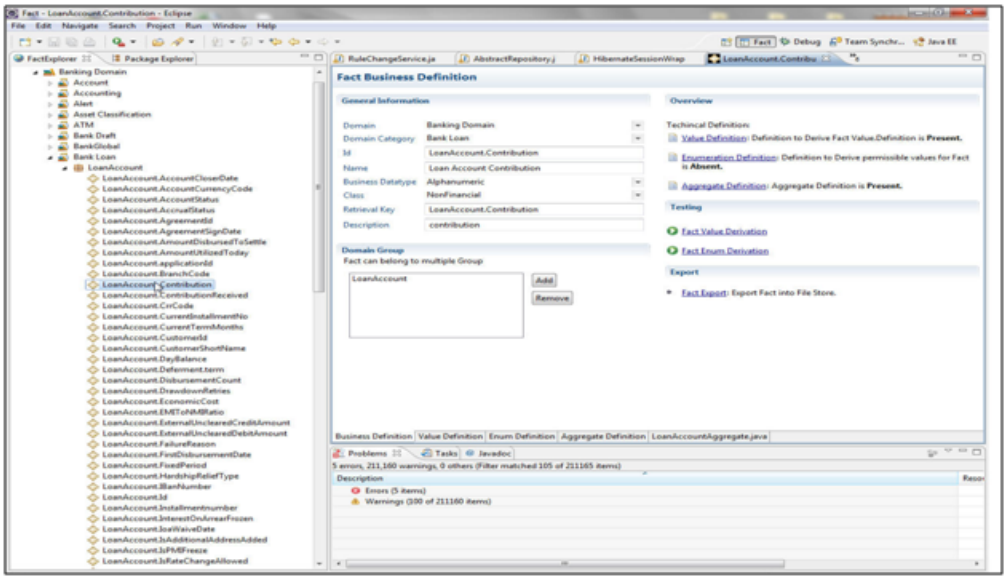


Figure 14–14 Value Definition Tab

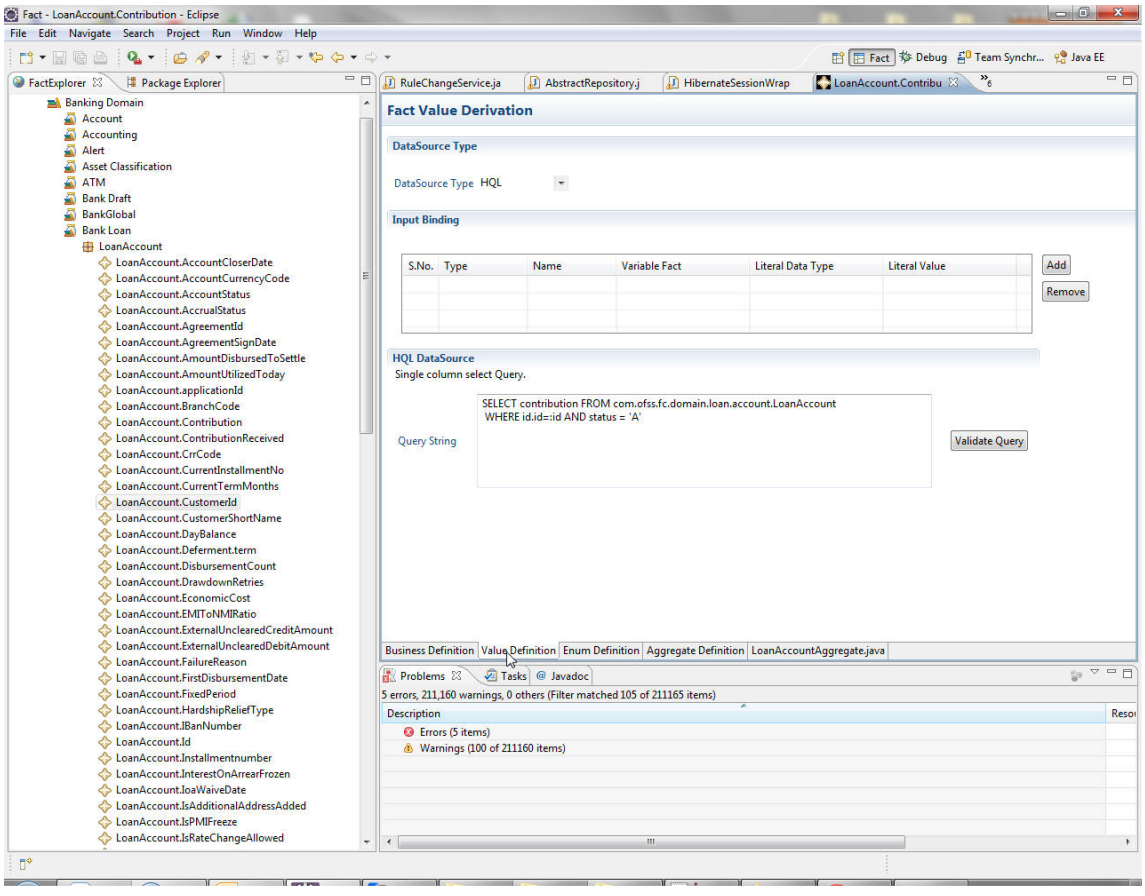


Figure 14–15 Enum Definition Tab

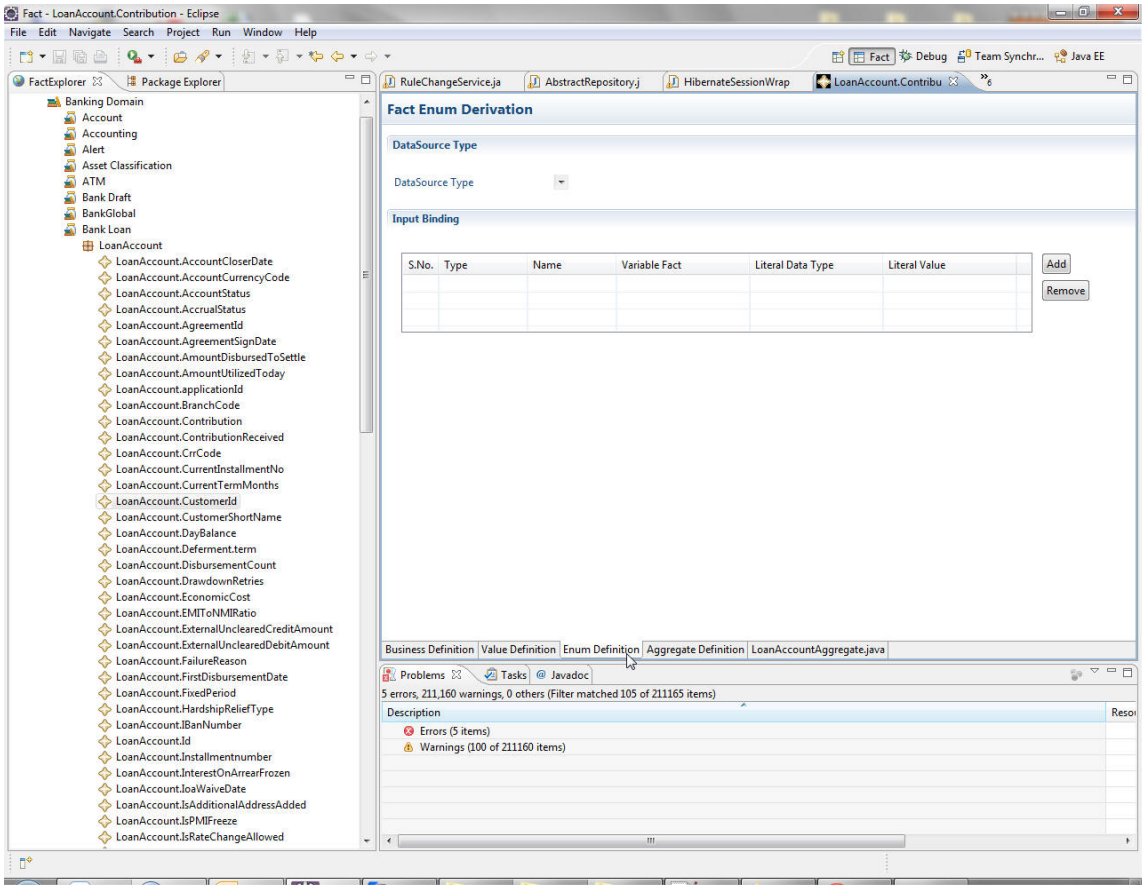


Figure 14–16 Aggregate Definition Tab

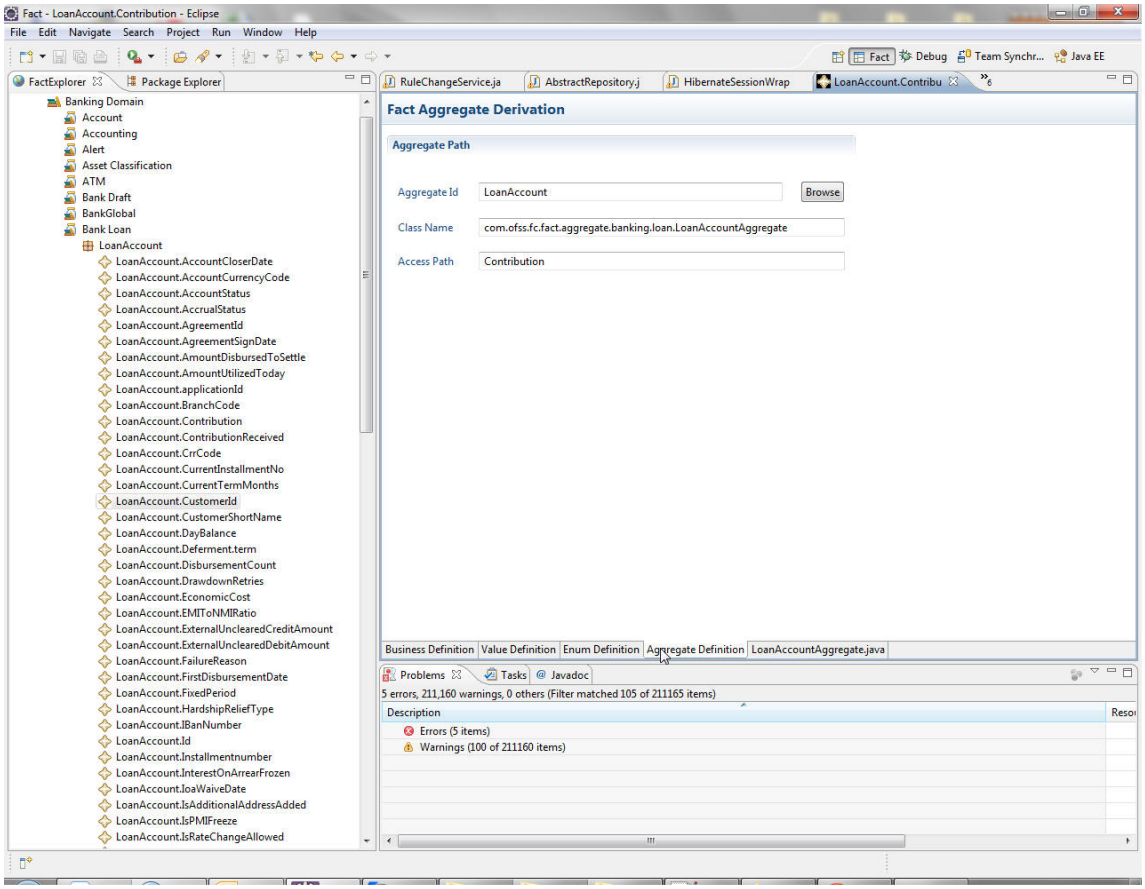
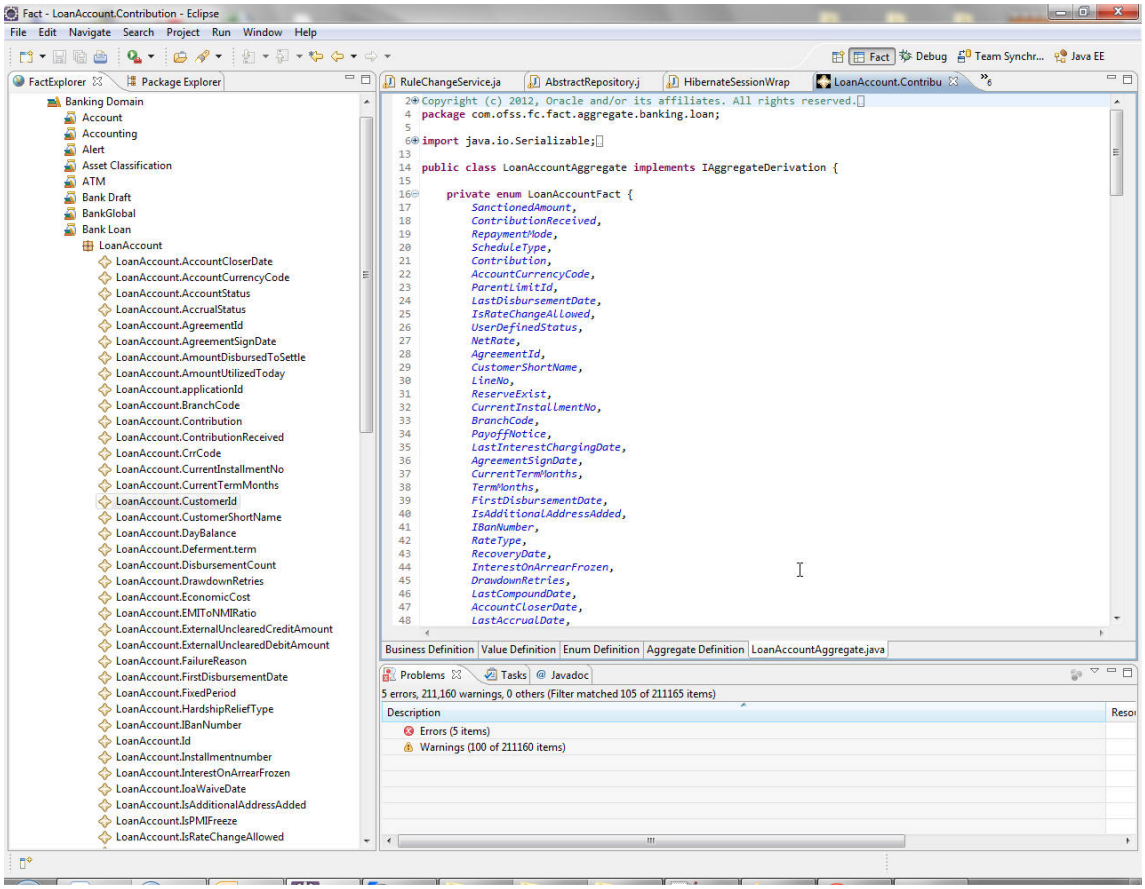
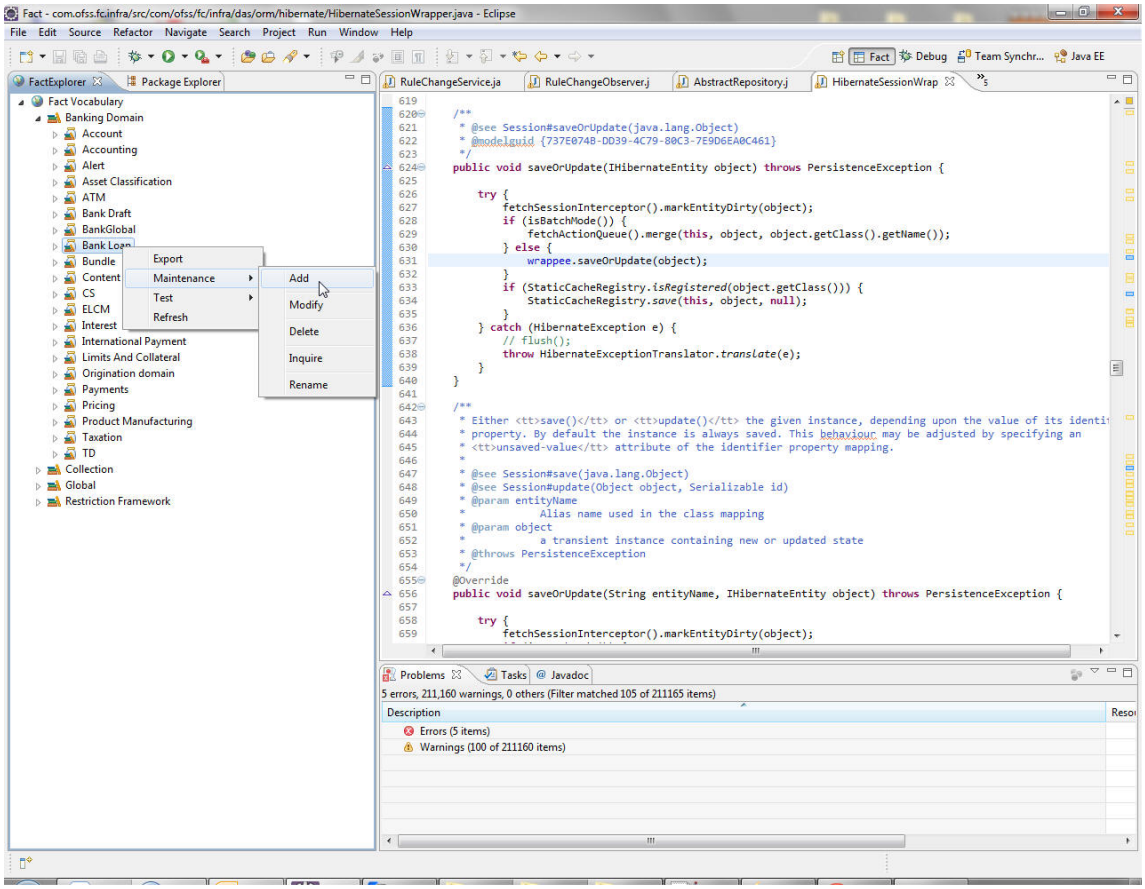


Figure 14–17 Aggregate File Tab



12. Creating **New Fact**: Right-click any domain Category in which Fact is to be created. Go to Maintenance -> Add.

Figure 14–18 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 14–19 Creating New Fact - Fact Business Definition

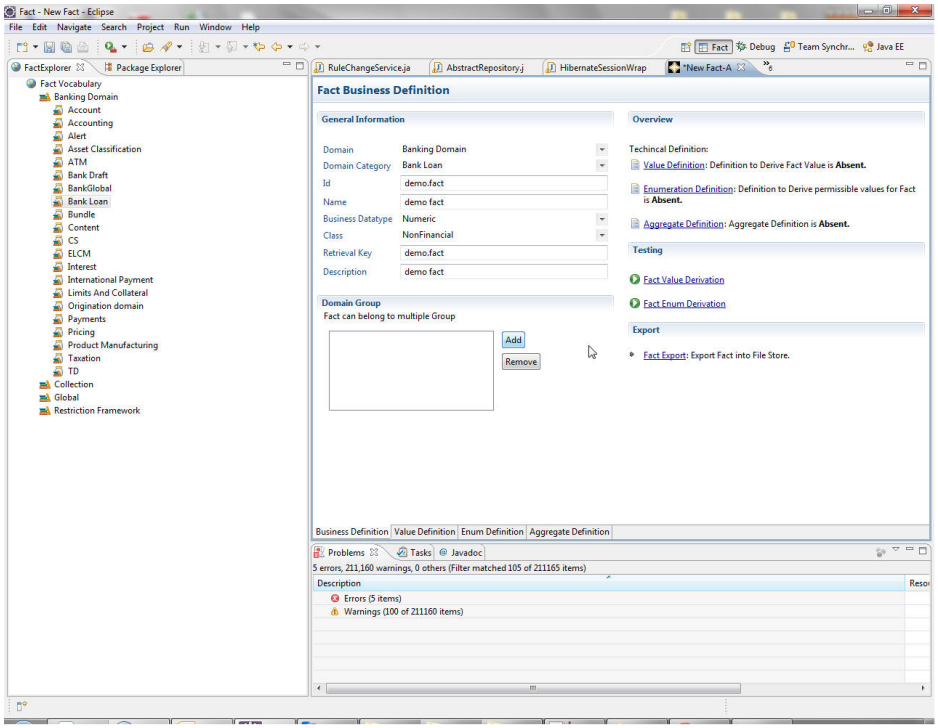
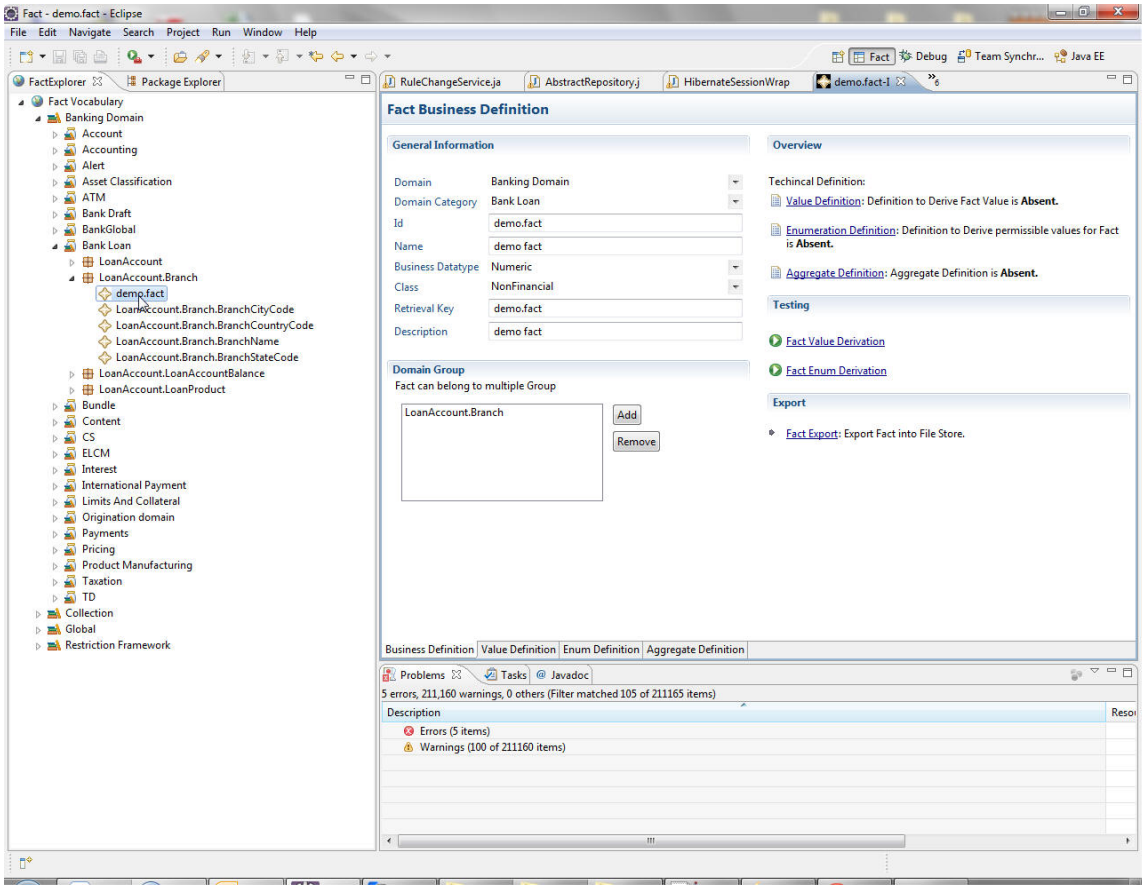


Figure 14–20 Creating New Fact - Domain Group



14. Enter the values in the fields and press CTRL+S, click **Yes** to save and fact will be created.

Figure 14–21 Saving New Fact

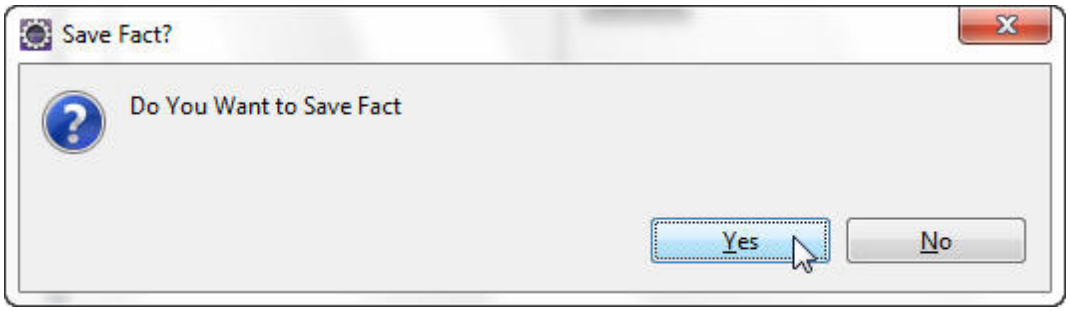
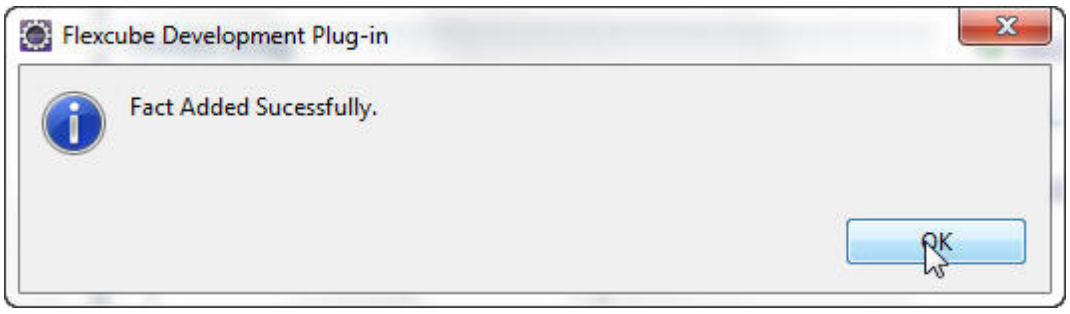


Figure 14–22 Saving New Fact - Fact Added



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.

14.1.4 Object Facts

Apart from the normal facts that have to be maintained explicitly, there is a way to define an object as a fact. The idea behind having object fact is to ease the fact definition phase when a particular class holds maximum attributes that are likely to be used in a given rule along the execution path. The advantages are as follows:

- No need of having individual fact definitions for each of the attribute in the class.
- The entire class can be made an object fact and the fact derivation takes the responsibility of scanning through this class object for fact value.
- The caller module will have the object already loaded in most of the scenarios.
- Ease of passing the facts through fact context, no need to remember the fact IDs of all the facts to a granular level. Once the parent fact is passed in the fact context with the class name as the fact id, the attributes are automatically scanned for the respective values as required.

Designate a class as Object Fact

To make a class an object fact, an entry for it needs to be made in the table: "flx_fa_object_facts_b".

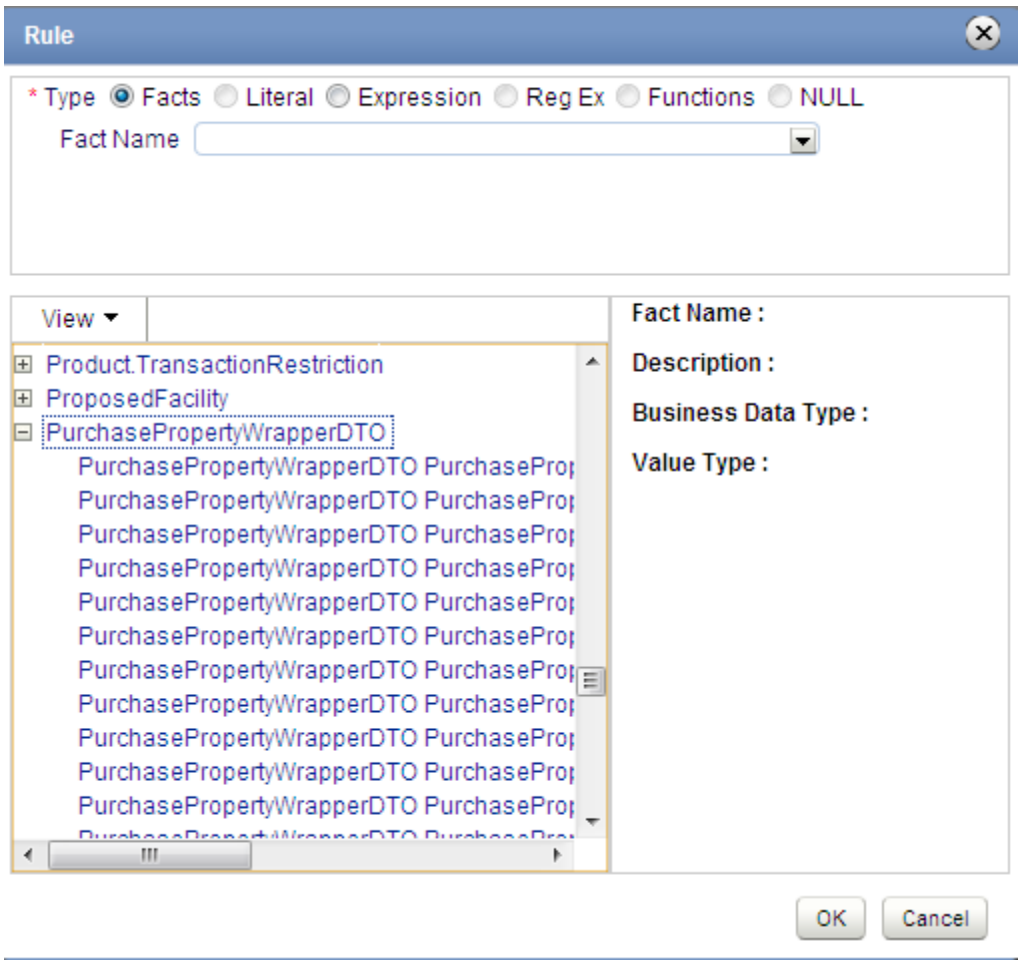
Figure 14–23 Designate Class as Object Fact

Row 1	Fields
FACT_GRP_NAME	PurchasePropertyWrapperDTO
FULL_QUAL_NAME	com.ofss.fc.app.origination.dto.lending.core.pg.PurchasePropert
DOMAIN_CODE	Banking
DOMAIN_CATEGORY_CODE	OR
CREATED_BY	
CREATION_DATE	
LAST_UPDATED_BY	
LAST_UPDATE_DATE	
OBJECT_VERSION_NUMBER	1
LAST_UPDATE_LOGIN	
DOMAIN_OBJECT_EXTN	CZ
FACT_NAME	PurchasePropertyWrapperDTO

Object Fact in UI

The usage of the object fact will be same as any other fact in the UI.

Figure 14–24 Object Fact in UI



Fact definitions for Object Fact

Building the fact definitions for an object fact is done as follows:

1. Once a class is designated as an object fact, it will be looked up at the time of loading the fact vocabulary.
2. The individual attribute access methods (getters or Boolean access methods that is, ones that start with "is") will be scanned to get the name of the attributes.
3. Once the attribute names and their data types are obtained, the FactBusinessDefinition object is created for it.
4. A variable fact object is also created and registered in the fact registry on the host.
5. The step 3 and 4 will be recursive, done for all the nested objects with the object fact till the leaf fact is found (that is, the one that can be used in the rule for instance data type could be any Java data types like String or Integer, or the OBP data types like Money or Duration)

14.2 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.

14.2.1 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.

14.2.2 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]
If
[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

14.2.3 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 14–1 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.

14.2.4 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.

Table 14–2 Actions

ActionID	Outvariable	Expression	Datatype
ACTION1	Discount Fee	0.2*App Fee	Double
ACTION2	Discount Fee	0	Double
ACTION3	Discount Fee	0	Double

Table 14–3 Conditions

ConditionID	LeftExpression	Relational Operator	RightExpression	LinkedConditionID	LinkedConditionalOperator	ActionId	RuleID	Version
CON1	CustomerType	==	HNI	CON2	&&	ACTION1	RULE1	1

ConditionID	LeftExpression	Relational Operator	RightExpression	LinkedConditionID	LinkedConditionalOperator	ActionId	RuleID	Version
CON2	Location Type	==	METRO	CON3	&&		RULE1	1
CON3	Location	==	MUMBAI				RULE1	1
CON4	CustomerType	==	HNI	CON5	&&	ACTION2	RULE1	1
CON5	Location Type	==	METRO	CON6	&&		RULE1	1
CON6	Location	==	DELHI				RULE1	1
CON7	CustomerType	==	HNI	CON8	&&	ACTION3	RULE1	1
CON8	Location	==	JAIPUR				RULE1	1

14.2.5 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.

14.2.6 Rules Versioning

Each time rule is modified new version is created for the rule and stored.

Table 14–4 Rules Versioning

RuleID	Version	Name	Effective Date
RULE1	1	DiscountRule	01/01/2009
RULE1	2	DiscountRule	31/03/2009

14.3 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("111111111111");
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");
```

14.3.1 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.

Figure 14–25 Generic Rule Configuration

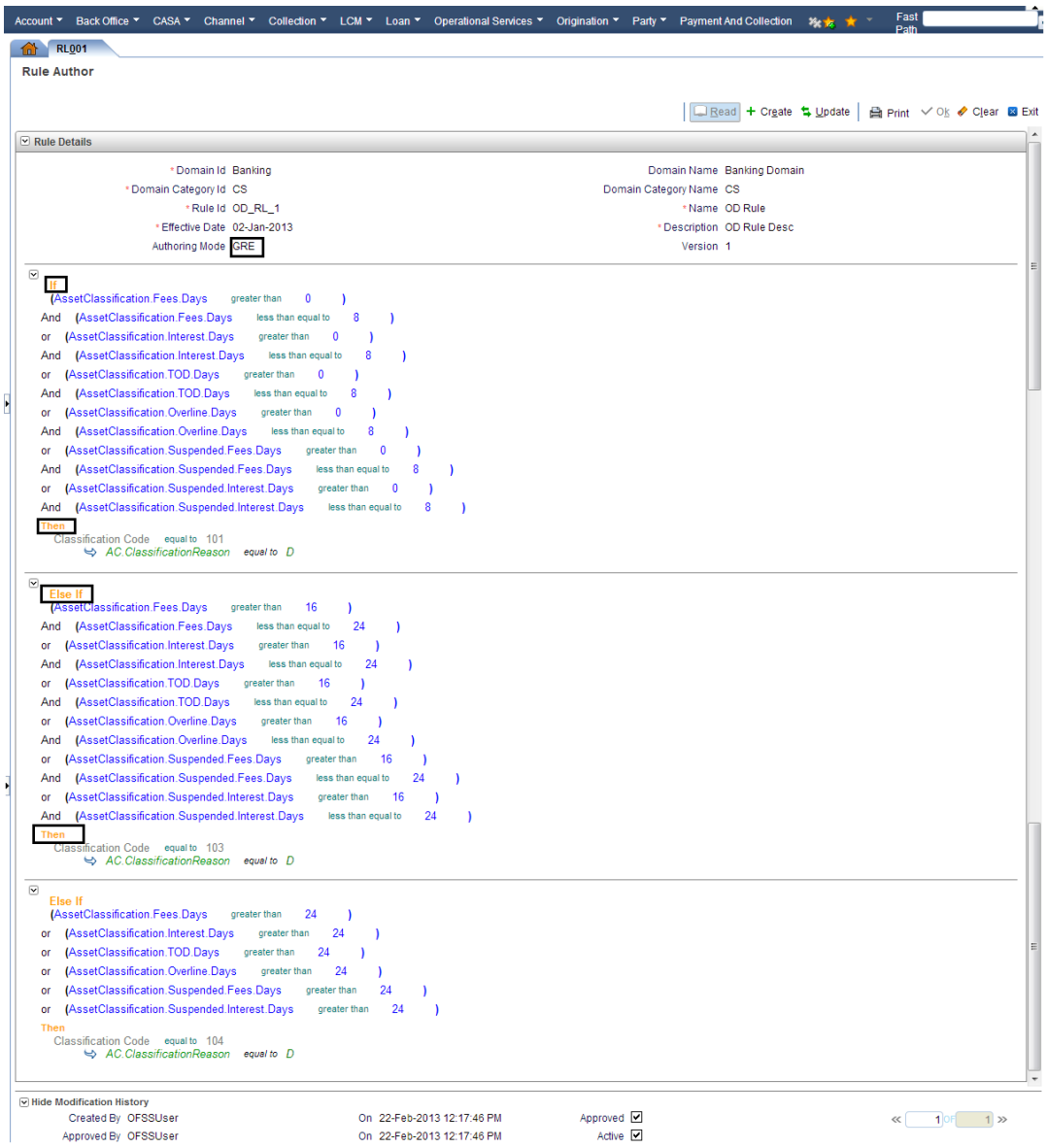
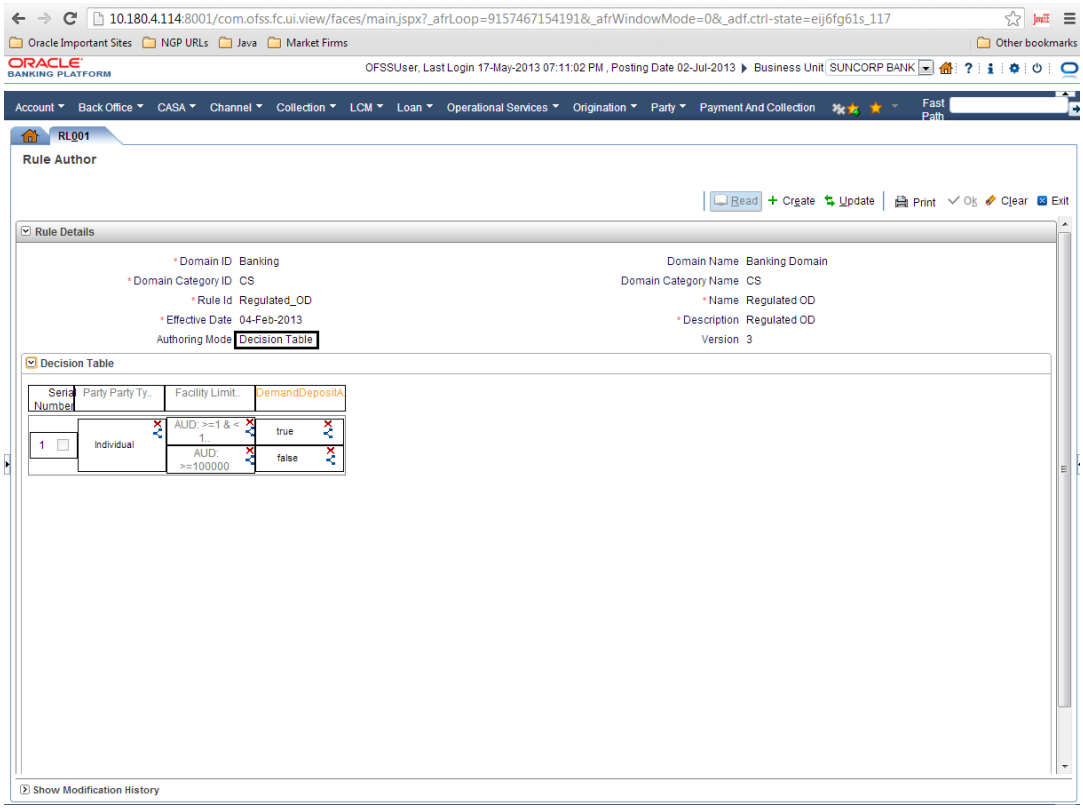
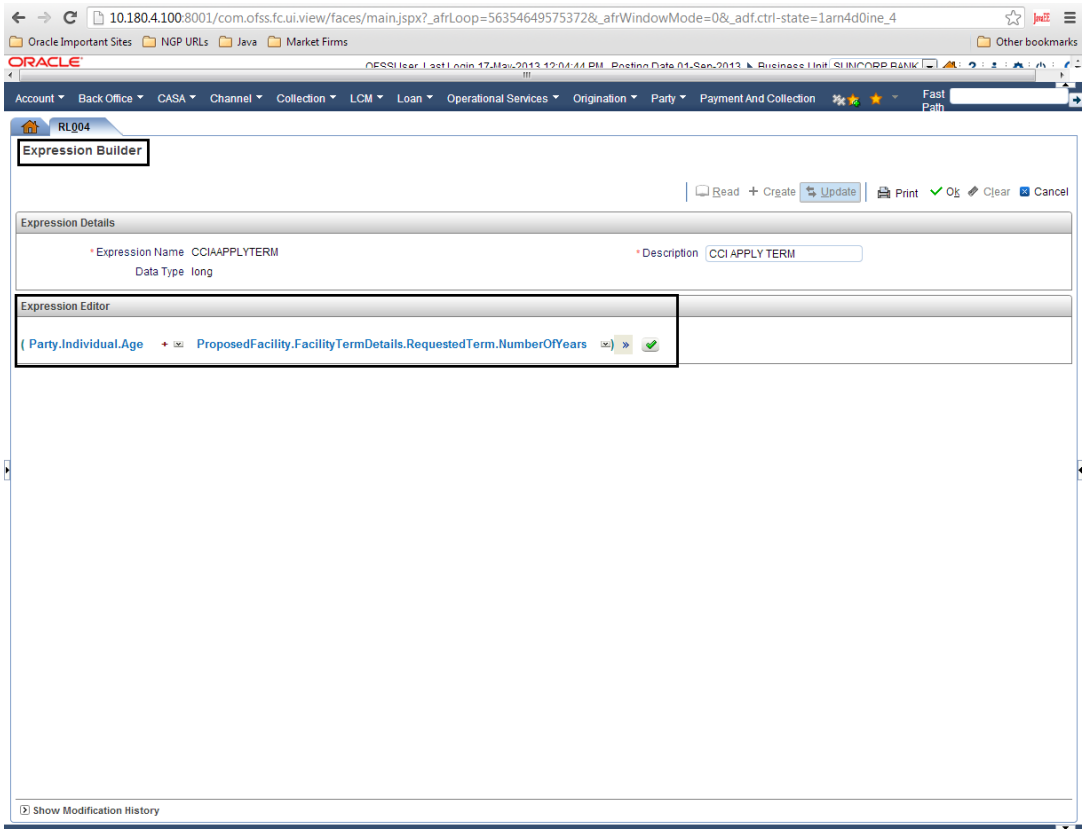


Figure 14–26 Rule Author - Decision Table



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.

Figure 14–27 Rule Author - Expression Builder



14.4 Rules Migration

This section describes the rules migration.

14.4.1 Rules Configured for Modules

Rule taskflows can be added to different modules. User can set up different rules based on the screen requirements.

Table 14–5 Details of Configured Rules in Modules

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

Module	Screen	Rule Type	Rule Description
			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/Margin 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 Computation Analysis	Generic Rule Author	This screen provides analysis as how the fee for particular transaction (happened in past) was computed. 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 Rate Derivation Analysis	Generic Rule Author	This screen provides analysis as how the interest rate for particular account was computed. 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.
Product Manufacturing	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 derive 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 Classification	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).

Module	Screen	Rule Type	Rule Description
			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/Decision 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.

15 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.

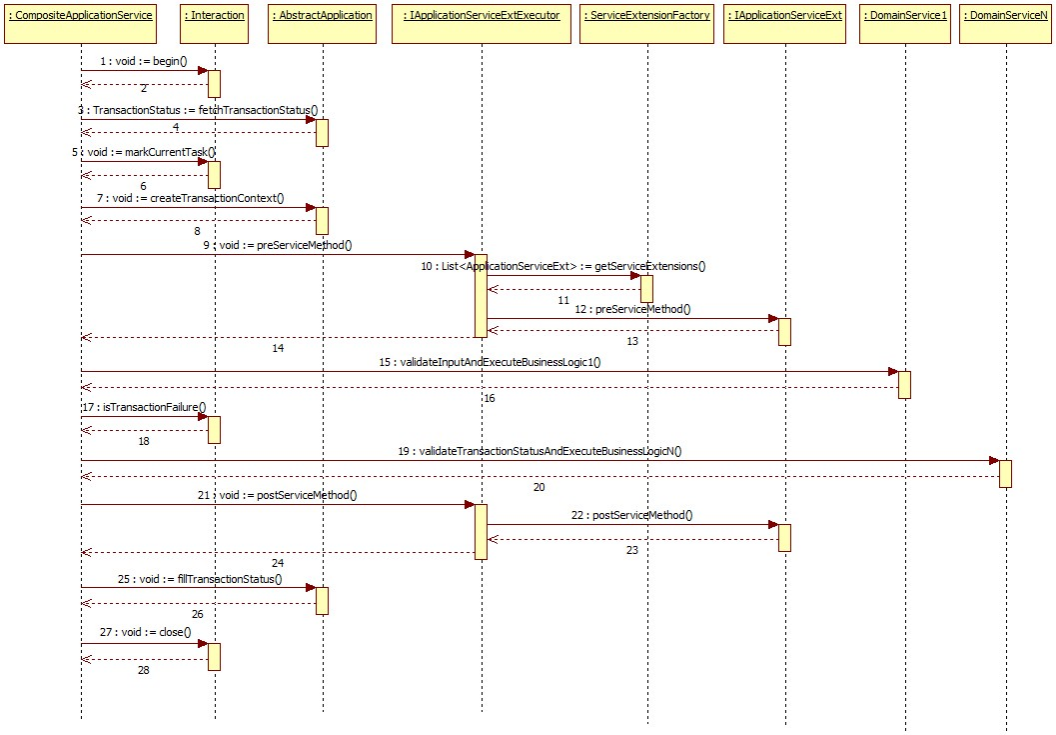
For the transaction control of the transaction manager, there are two defined patterns:

- **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

15.1 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.

Figure 15–1 Composite Application Service Architecture



15.2 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 15–1 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.
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.

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.TDAccountPayinRespo
nse;
import
com.ofss.fc.app.extensibility.td.service.composite.ext.IExtendedTe
rmDepositApplicationServiceExtExecutor;
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.ITermDepositAccountApplicationS
ervice;
import
com.ofss.fc.app.td.service.account.TermDepositAccountApplicationSe
rvice;
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();
/**
* 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.TDAccountPayinApplicationSe
rvice.openAccountWithPayin", sessionContext,
termDepositAccountOpenDTO, payinTransactionDTO,
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)) {
logger.log(Level.FINE, formatter.formatMessage("Exiting from
openAccountWithPayin()."));
}
return tdAccountPayinResponse;
}
}
```

16 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:

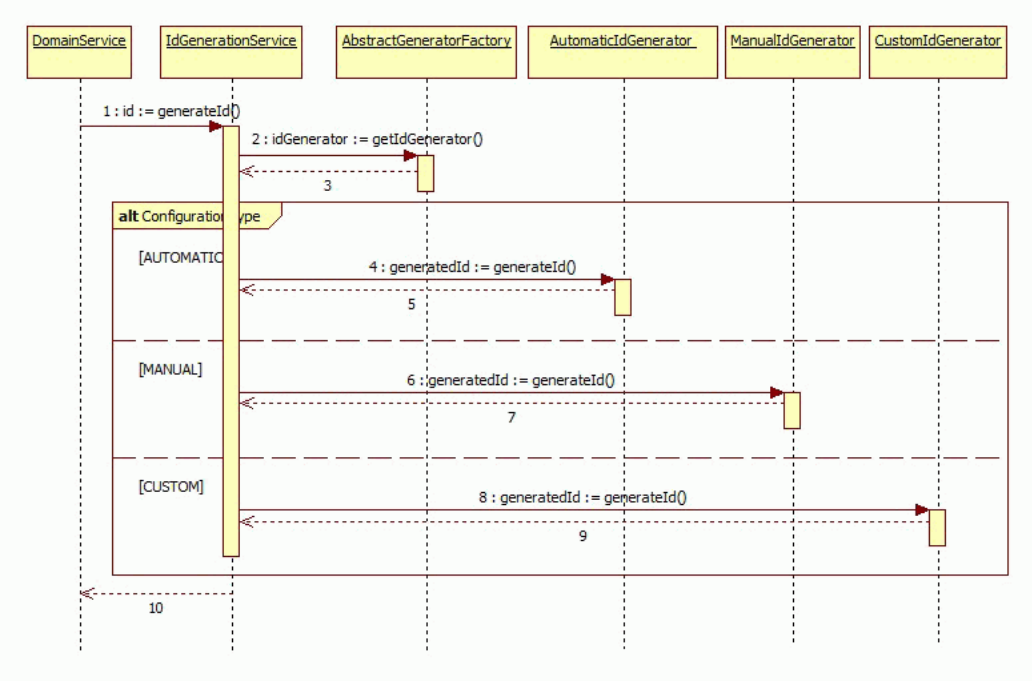
If the configuration type for the ID generation is set to automatic, the ID is generated as per the defined generation logic for the automated ID generation. You can set the pattern, sequence, weights and check digit modulo and modify the automatic generation logic.

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 16–1 Configuration of ID Generation Process



From the implementation perspective, the following sections describe the change in configurations required for customizing the ID generation.

16.1 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.

Table 16–1 FLX_CS_ID_CONFIG_B

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, NNNBBBBYYYYSSSSSSS
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', '3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1'
CHK_DIGIT_MODULO	Check digit modulo
CREATED_BY	Indicates the User who created the row
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

- FLX_CS_ID_RANGE: This table is used to determine the range of the values which the ID can take.

Table 16–2 FLX_CS_ID_RANGE

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

- FLX_CS_ID_USF: This table is used to determine the user selected fields for the ID generation logic.

Table 16–3 FLX_CS_ID_USF

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

16.1.1 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.

16.2 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

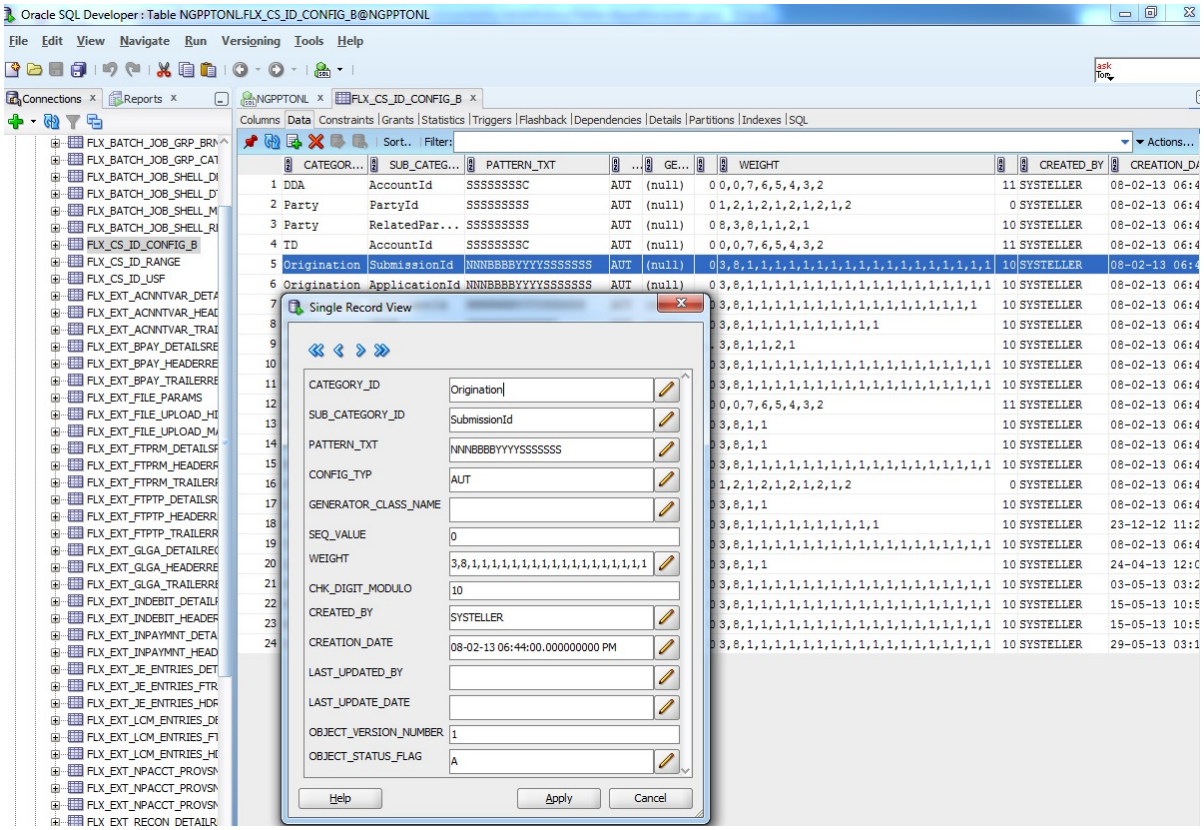
current position value of range until it's size becomes pattern's length. For example, the pattern is 'SSSSSS' and the generated range gives the value as '2345' then the actual value will become '002345'.

- 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 NNNYYYYBBBSSSSS in the database.

Figure 16–2 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 16–3 Automated ID Generation - Generate Submission ID

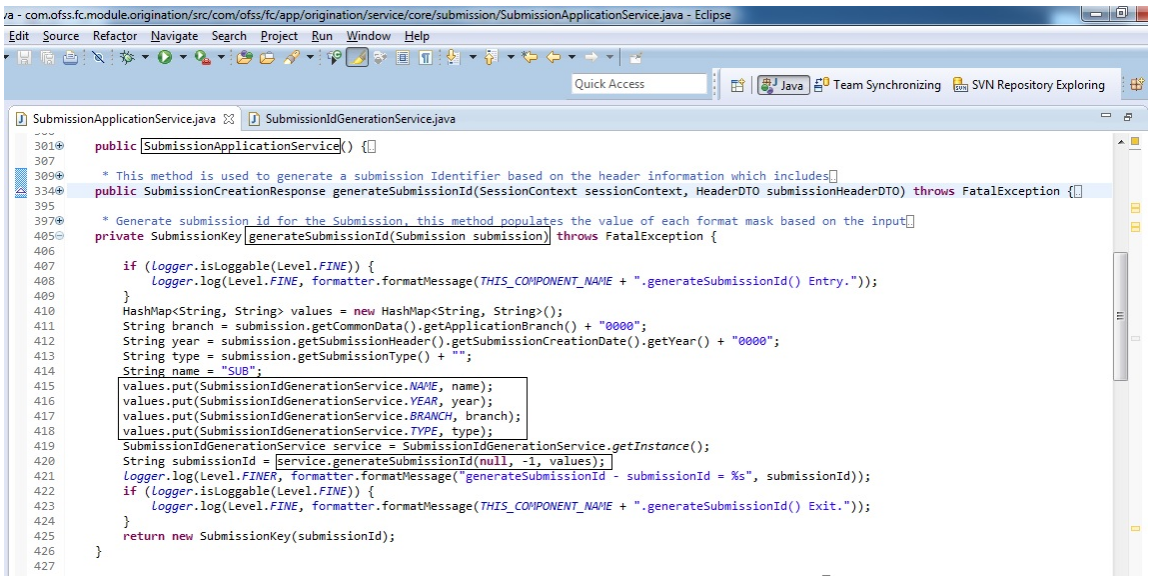
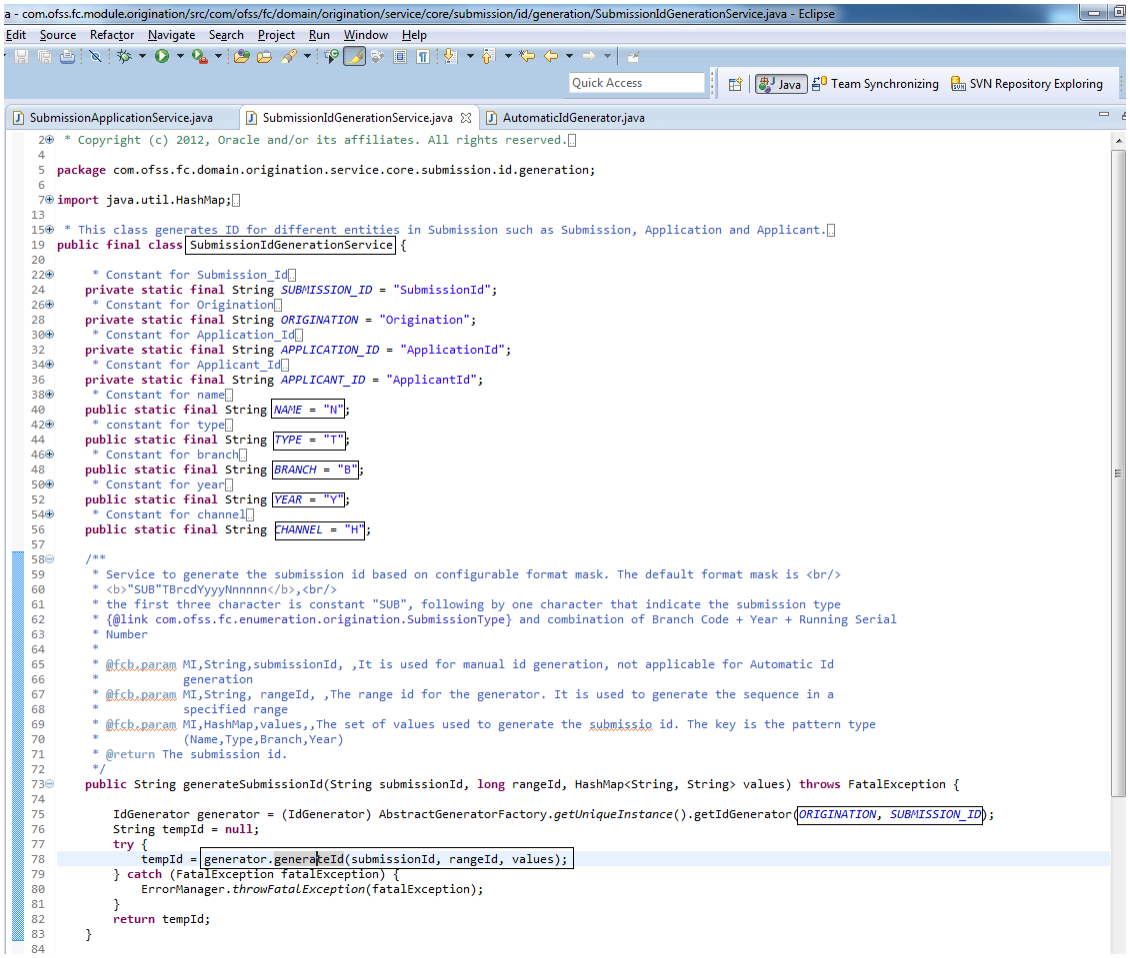


Figure 16–4 Automated ID Generation - Submission ID Generation Service



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.

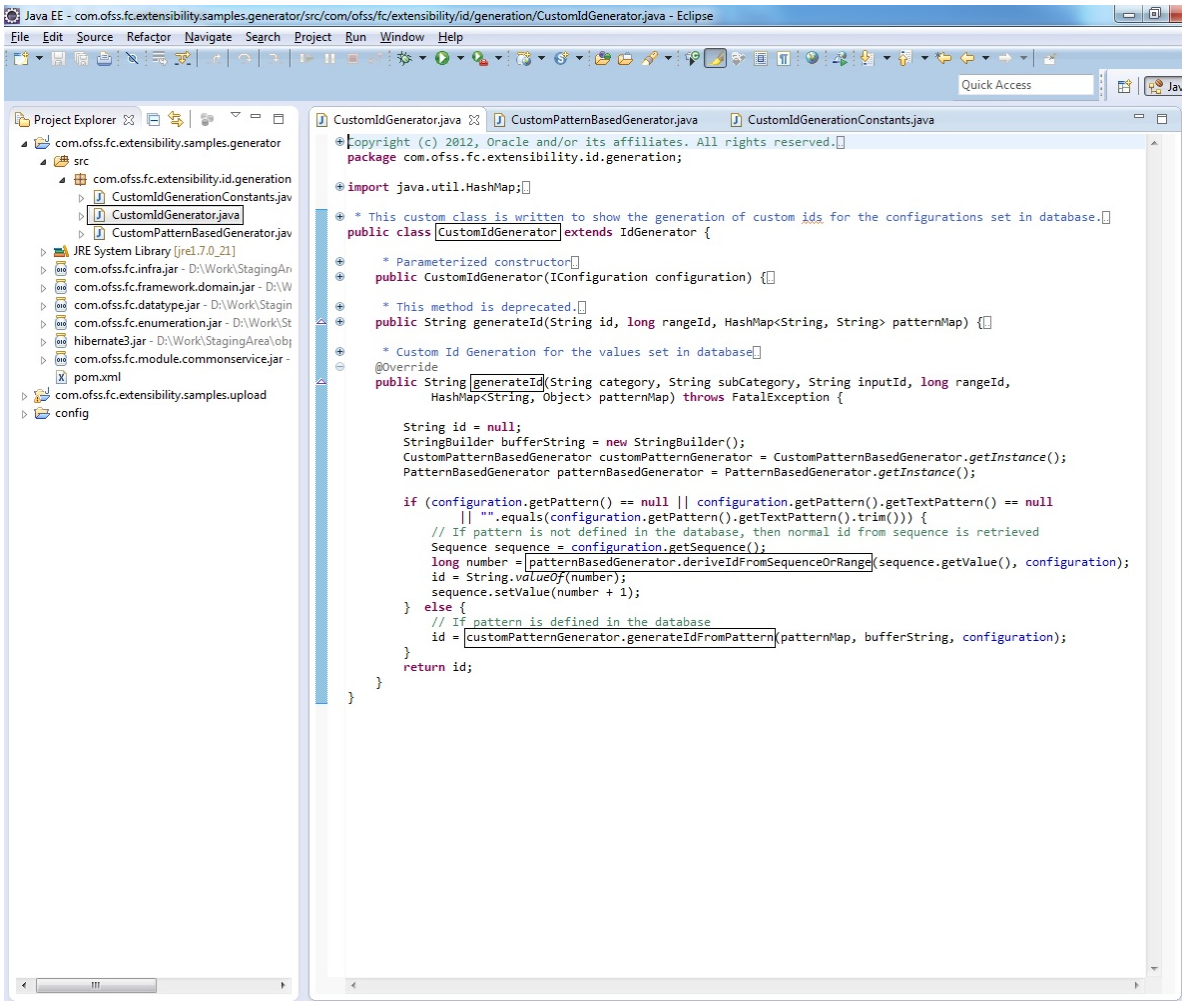
In case of without mask configuration's pattern. If range ID is -1, it means that there is no range defined for the mask configuration, it then picks up the range details with range ID based on the category and sub-category. The generated ID will become the current position of range. If range is not 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.

16.3 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 16–5 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 16–6 Custom ID Generation - Custom ID Generation Constants

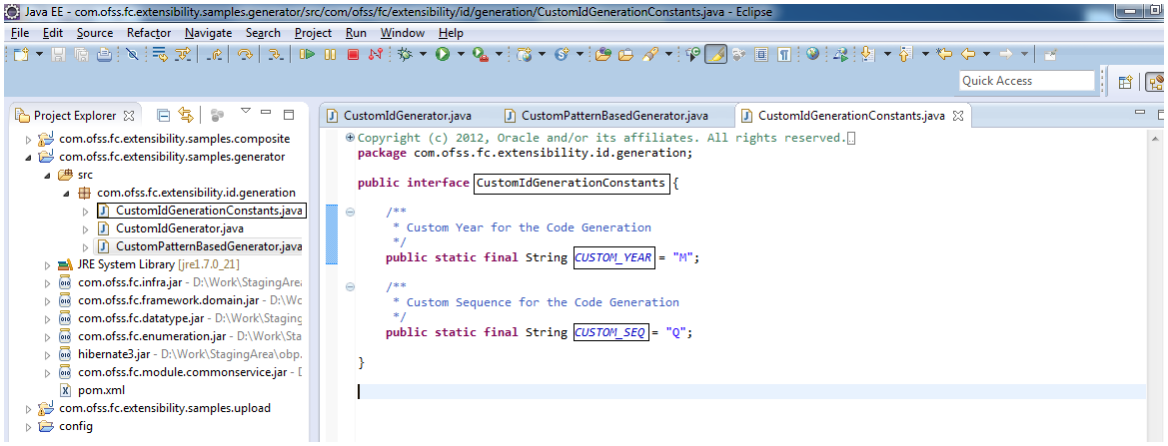
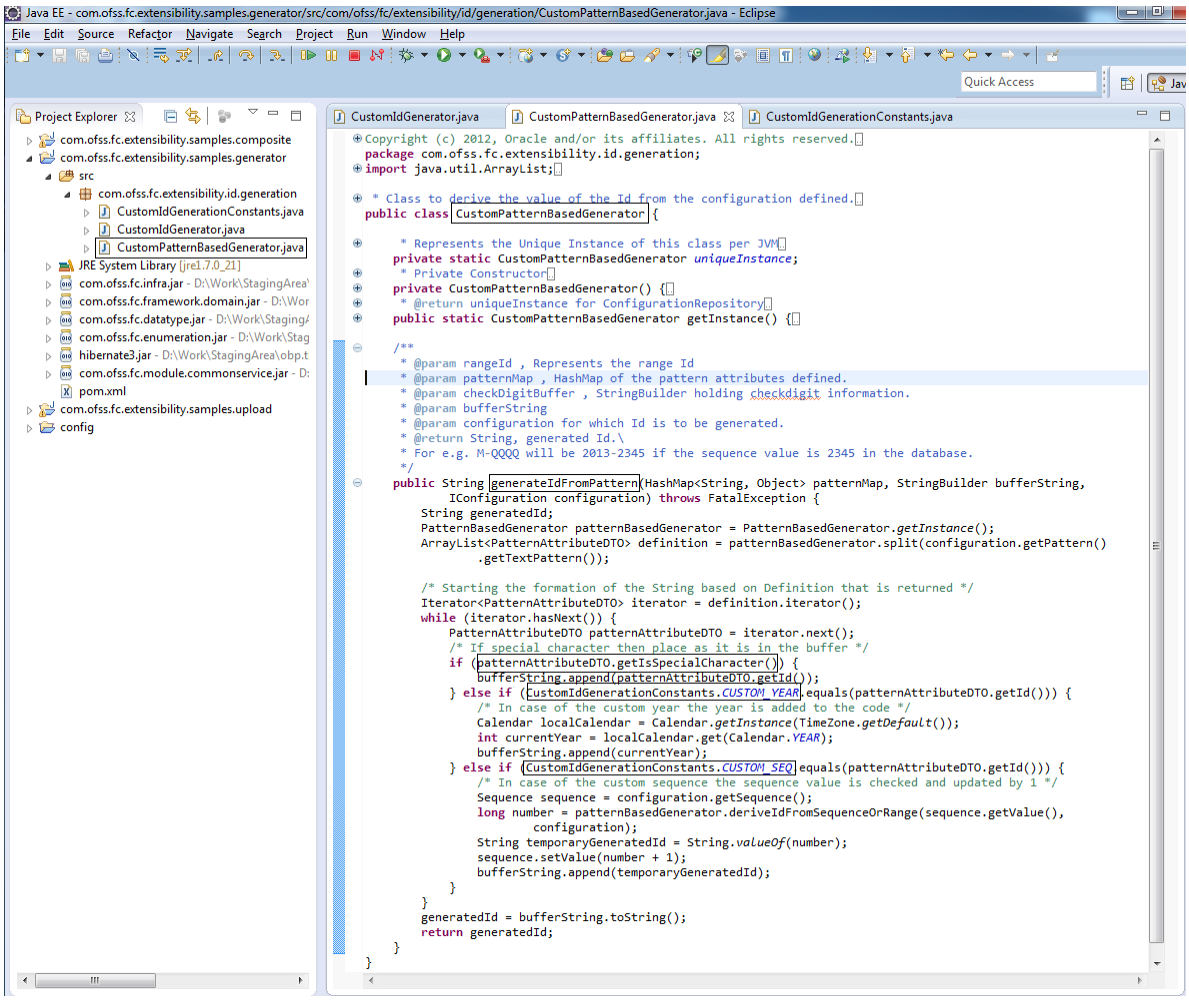


Figure 16–7 Custom ID Generation - Custom Pattern Based Generator



17 Extensibility of Domain Objects using Flex Fields

This chapter describes about the Flex Field provisioning by the product at the Service Layer. Flex Fields are additional attributes provisioned upfront via configuration and with basic validations. By using e Flex Field, banks IT, consultants or partners can configure additional data elements to be part of the domain entities, without the need to add custom code. It provides ability to use these additional data elements in the business logic of the application. These additional elements are available in the service interface in the form of key-value pairs in Dictionary Object. Some important features of Flex Fields are:

- A Flex Field is a flexible data field that can be customized as per business needs without programming.
- Only Seed Metadata Configuration is required to enable Flex Fields.
- Flex field attributes can be provisioned at each entity level [Max 30 attributes per entity].
- Flex Field Attribute are strongly typed at an definition & busiless logic validation level. However, the Data Type is maintained as String in the entity and VARCHAR (250) in database.
- Service input or data transfer is supported via Dictionary Object (Separate indicator is provided in Dictionary Object to distinguish flex field dictionary object)

Utilizing Flex Fields in OBP requires following the two steps mentioned below:

- **Flex Field Provisioning:** This step denotes ensuring that Table of Entity which is candidate for extension has required number of columns in database. Product out of the box ships sufficient number of Flex Field columns for all Entities eligible for extension. Still if Consultant wants more Flex Field columns to utilize, then this is Optional step is needed.
- **Flex Field Utilization:** This step denotes utilizing required number of Flex Fields by providing its Metadata. Utilized Flex Fields should be always less or equal to Provisioned flex Fields.

17.1 Flex Field - Provisioning

Maximum 30 attributes per entity can be provisioned at each entity level. Each Attribute data type is declared as String in Java Entity (inside superclass AbstractDomainObject) and VARCHAR (255) in table for flexibility.

17.1.1 How to know Maximum Flex Fields Provisioned for Entity?

- FLX_FW_FLX_FLD_ENTITIES_ALL_B seed table contains list of all Product Entity tables for which Flex Fields are provisioned.
- MAX_FLX_FLD_COLUMN_COUNT column in this table denotes Maximum number of Flex Fields currently provisioned for that Entity. This value cannot exceed 30.
- Consultant can utilize only those number of Flex Fields which are provisioned as per MAX_FLX_FLD_COLUMN_COUNT column value.
- If Consultant wants to utilize more Flex Fields than that are already provisioned as per MAX_FLX_FLD_COLUMN_COUNT column value, then it can be done using steps mentioned in section below.

17.1.2 Increase Maximum Flex Fields Provisioned for Entity (Optional Step)

- In order to increase number of Provisioned Flex Fields for Entity, Consultant needs to update value of column MAX_FLX_FLD_COLUMN_COUNT in table FLX_FW_FLX_FLD_ENTITIES_ALL_B. This number should not exceed 30.
- After that Consultant needs to execute stored procedure AP_FW_PROVISION_FLEX_FIELDS.
- This stored procedure will read seed table FLX_FW_FLX_FLD_ENTITIES_ALL_B and alter those entity tables to add missing Flex Fields columns in the table.
- This procedure assumes that few Flex Field Columns might be already present in table and will not impact those columns or tables.

17.2 Flex Field - Utilization

Utilization of Flex Field can be done using two steps:

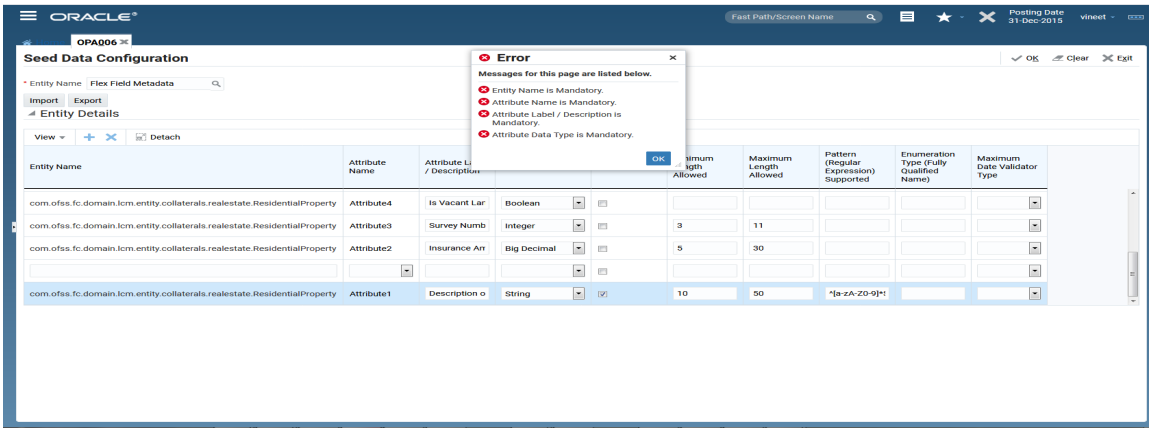
- Maintain Flex Field Metadata by:
  - Using Seed Data Configuration (Fast Path: OPA006) page.
  - Directly making seed entries in table FLX_FW_FLX_FLD_METADATA_B.
- Restart host Server.

17.2.1 Maintain Flex Field Metadata using Seed Data Configuration (Fast Path: OPA006) Page

- Metadata gives information about Domain Object Name and following information about each Flex Field Attribute:
  - Name - Mandatory
  - Label - Mandatory
  - Data Type - Mandatory
  - Is Mandatory - Optional
  - Min Length - Optional
  - Max Length - Optional
  - RegEx Pattern - Optional
  - Enum Type - Mandatory if Data type is Enum
  - Max Date Validator - Optional and Applicable only for Date Data Type
- Go to OPA006 page. Select Entity Name as Flex Field Metadata. Details of existing Flex Field entity Metadata will be displayed in read mode.
- First two columns (Entity Name, Attribute Name) will be in read only mode for existing values (As these are Key values). However, when user clicks Add, these two columns will be editable only for Newly added rows.

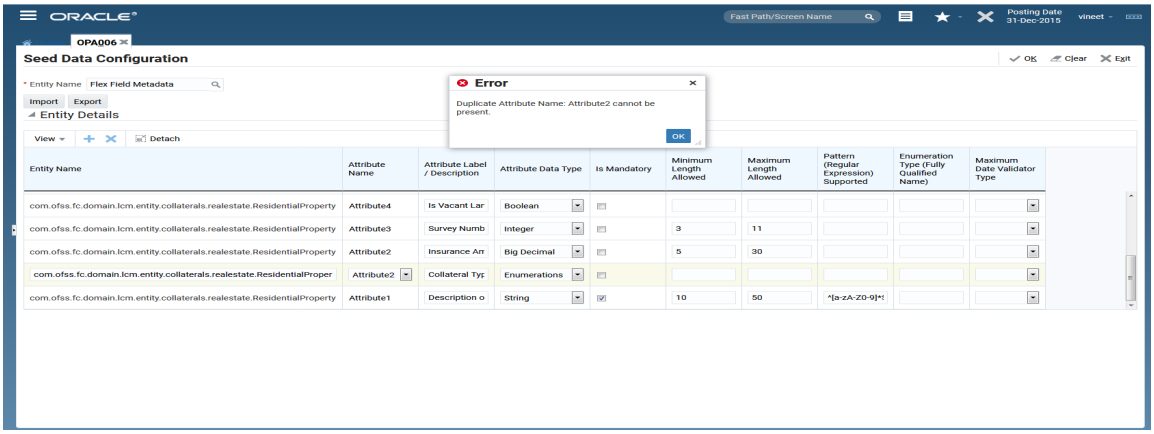
- If user clicks Add to add new row and if a user clicks final OK without entering Entity Name, Attribute Name, Attribute Label or Description, and Attribute Data Type, then the following error is displayed.

Figure 17–1 OPA006 - Error



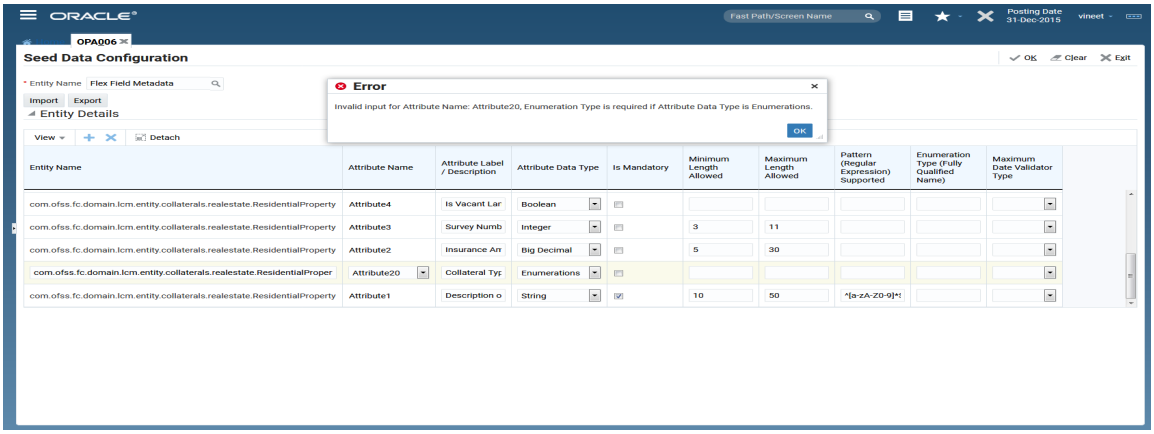
- Duplicate Attribute names cannot be present. If present, then the following error is displayed.

Figure 17–2 OPA006 - Duplicate Attribute



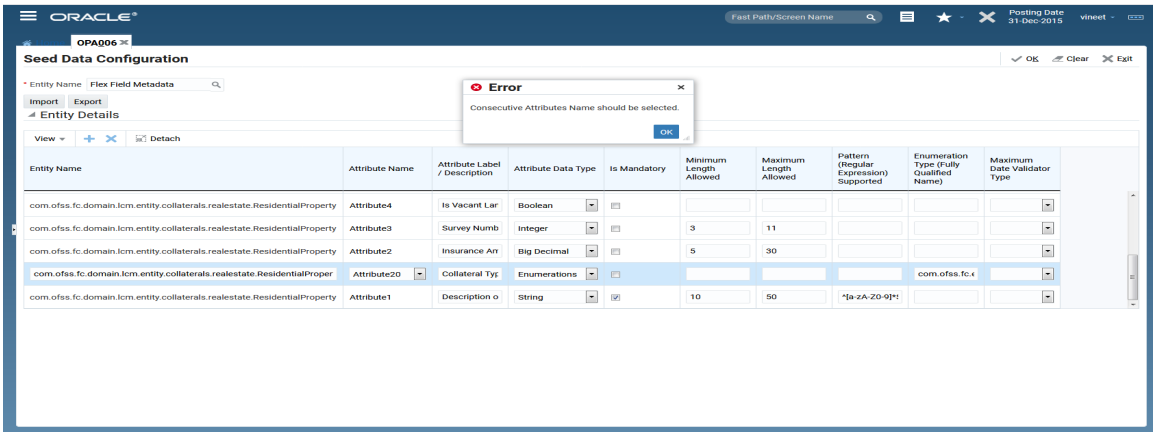
- When Attribute Data type is chosen as Enumerations then Enumeration Type should be present. If not, then the following error is displayed.

Figure 17–3 OPA006 - Enumeration Type



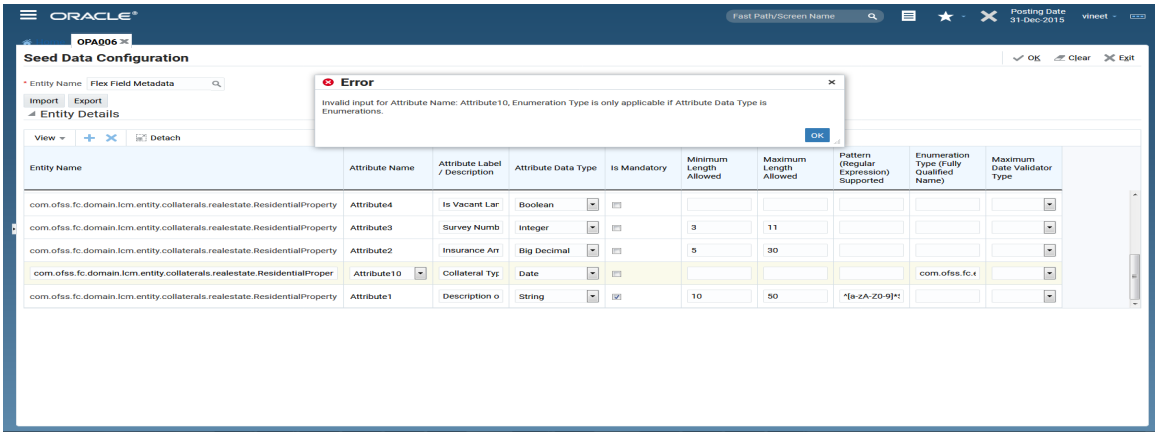
- Consecutives attributes names should be present. User cannot skip any attribute name otherwise the following error is displayed.

Figure 17–4 OPA006 - Consecutive Attributes Names



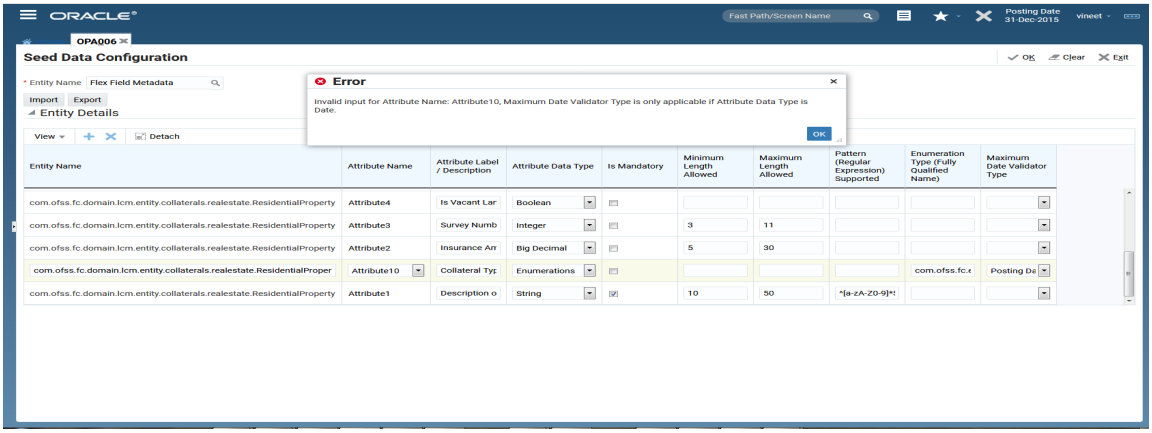
- Enumeration type is only applicable if Attribute Data Type is Enumerations. If other than Enumerations is chosen as an Attribute Data Type, then the following error is displayed.

Figure 17–5 OPA006- Invalid Input for Attribute Data Type.



- Maximum Date Validator type is only applicable if Attribute Data Type is Date. If other than Date is chosen as an Attribute Data Type, then the following error is displayed.

Figure 17–6 Attribute Data Type as Date



- If all information is valid Flex Field Metadata is saved in table FLX_FW_FLX_FLD_METADATA_B.
- Alternatively instead of using OPA006 page, Consultant can directly add seed data in FLX_FW_FLX_FLD_METADATA_B table for required Flex Fields and check-in the same.
- Flex field metadata is mandatory and useful to achieve following:
  - Clearly indicates out of Provisioned Flex Fields which are actually utilized and for which purpose. Flex Field Metadata is mandatory (at least label and data type). This is extremely important when, as a part of previous customization if few Flex Fields are utilized then Consulting should clearly know used fields and available fields.
  - Attribute Description or Label mapping.
  - Attribute Data Type mapping and supporting basic validations.
  - Throwing validation message with respective label for better interpretation.
- After populating Flex Field Metadata, Host Server restart is needed.

- On Host Restart, during ORM loading for each Entity in FLX_FW_FLX_FLD_METADATA_B table Flex Field related ORM mapping is added dynamically as per Metadata using EclipseLinkSessionCustomizer class.
- Out of Provisioned columns only optimal number of table columns are actually used during this dynamic ORM mapping based on fields specified in FLX_FW_FLX_FLD_METADATA_B table.
- Supported sizes are: 5/10/15/20/25/30 fields.
- AbstractDomainObject holds following objects to support these flexible sizes. One of these fields is chosen for ORM mapping per Entity as per Metadata configuration.
    - FiveAttributeFlexField
    - TenAttributeFlexField which extends FiveAttributeFlexField
    - FifteenAttributeFlexField which extends TenAttributeFlexField
    - TwentyAttributeFlexField which extends FifteenAttributeFlexField
    - TwentyFiveAttributeFlexField which extends TwentyAttributeFlexField
    - ThirtyAttributeFlexField which extends TwentyFiveAttributeFlexField

17.3 Runtime Storage and Retrieval of Flex Field Attribute values

- Service input or data transfer is supported via Dictionary Object. Separate indicator named flexField is provided to distinguish flex field dictionary object.
- The attributes which are passed as part of the Dictionary object with the Flex Field indicator, will be persisted as Flex Field in the respective Entity Table. The attribute name follow the name convention as “Attribute<<attribute number>>” [‘A’ caps, like Attribute1, Attribute2, Attribute3, …. and Attribute30].
- For example, com.ofss.fc.domain.lcm.entity.collaterals.realestate.Industrialproperty this Entity is configured to make use of Flex Field attribute by defining metadata.
- Data is saved using Dictionary input as follows, if API is being tested using SOAP UI Toolkit:

Figure 17–7 Runtime Storage Saved Data

```
<dto:dictionaryArray>
   <dto:fullyQualifiedClassName>com.ofss.fc.domain.lcm.entity.collaterals.rea
   <dto:flexField>true</dto:flexField>
   <dto:nameValuePairDTOArray>
      <dto:dat/>
      <dto:genericName/>
      <!-- Description of Land -->
      <dto:name>Attribute1</dto:name>
      <dto:value>Industrial Property Land Description</dto:value>
   </dto:nameValuePairDTOArray>
   <dto:nameValuePairDTOArray>
      <dto:dat/>
      <dto:genericName/>
      <!-- Vacant Land -->
      <dto:name>Attribute2</dto:name>
      <dto:value>N</dto:value>
   </dto:nameValuePairDTOArray>
</dto:dictionaryArray>
```

- Data is retrieved using Dictionary as follows, if API is being tested using SOAP UI Toolkit:

Figure 17–8 Runtime Storage Retrieved Data

```
</responseservice:status>
<collateralsdtolcmapp:collateralArrayDTO xsi:type="realestatecollateralsdtolcmapp:industrialPropertyDTO" xmlns:xsi="http://www.w3.org/2001/XMLSchema-ir
   <dtocommondomainframework:dictionaryArray>
      <dtocommondomainframework:flexField>true</dtocommondomainframework:flexField>
      <dtocommondomainframework:fullyQualifiedClassName>com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty</dtocommondomainframewo
      <dtocommondomainframework:nameValuePairDTOArray>
         <dtocommondomainframework:name>Attribute1</dtocommondomainframework:name>
         <dtocommondomainframework:value>Industrial Property Land Description</dtocommondomainframework:value>
      </dtocommondomainframework:nameValuePairDTOArray>
      <dtocommondomainframework:nameValuePairDTOArray>
         <dtocommondomainframework:name>Attribute2</dtocommondomainframework:name>
         <dtocommondomainframework:value>N</dtocommondomainframework:value>
      </dtocommondomainframework:nameValuePairDTOArray>
      <dtocommondomainframework:nameValuePairDTOArray>
         <dtocommondomainframework:name>Attribute3</dtocommondomainframework:name>
      </dtocommondomainframework:nameValuePairDTOArray>
      <dtocommondomainframework:nameValuePairDTOArray>
         <dtocommondomainframework:name>Attribute4</dtocommondomainframework:name>
      </dtocommondomainframework:nameValuePairDTOArray>
      <dtocommondomainframework:nameValuePairDTOArray>
         <dtocommondomainframework:name>Attribute5</dtocommondomainframework:name>
      </dtocommondomainframework:nameValuePairDTOArray>
      <dtocommondomainframework:nameValuePairDTOArray>
         <dtocommondomainframework:name>Attribute6</dtocommondomainframework:name>
      </dtocommondomainframework:nameValuePairDTOArray>
      <dtocommondomainframework:nameValuePairDTOArray>
```

17.4 Flex Field - Fact support

Flex Fields provisioned can be consumed as facts as below,

- Derived facts can be created, with following Derivation Type.
  - Java Based Derivation
  - JPQL Query Based Derivation
  - SQL Query Based Derivation
- Following is example of JPQL based Fact which uses Flex Field attribute.
- com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty this Entity is configured to make use of FiveAttributeFlexField by defining Flex Field attribute metadata.
- Data will be saved using Dictionary input as follows, if API is being tested using SOAP UI Toolkit:

Figure 17–9 Flex Field - Fact support Saved Data

```
<dto:dictionaryArray>
    <dto:fullyQualifiedClassName>com.ofss.fc.domain.lcm.entity.collaterals.rea
    <dto:flexField>true</dto:flexField>
    <dto:nameValuePairDTOArray>
        <dto:dat/>
        <dto:genericName/>
        <!-- Description of Land -->
        <dto:name>Attribute1</dto:name>
        <dto:value>Industrial Property Land Description</dto:value>
    </dto:nameValuePairDTOArray>
    <dto:nameValuePairDTOArray>
        <dto:dat/>
        <dto:genericName/>
        <!-- Vacant Land -->
        <dto:name>Attribute2</dto:name>
        <dto:value>N</dto:value>
    </dto:nameValuePairDTOArray>
</dto:dictionaryArray>
```

- After this save API call, Data will be stored in table as follows:

Figure 17–10 Flex Field - Fact support Data in Table

Query Result

SQL | All Rows Fetched: 1 in 0.002 seconds

	COLLATERAL_ID	FF_ATTRIBUTE_1	FF_ATTRIBUTE_2	FF_ATTRIBUTE_3	FF_ATTRIBUTE_4	FF_ATTRIBUTE_5
1	COL201600100000193	Industrial Property Land Description	N	(null)	(null)	(null)

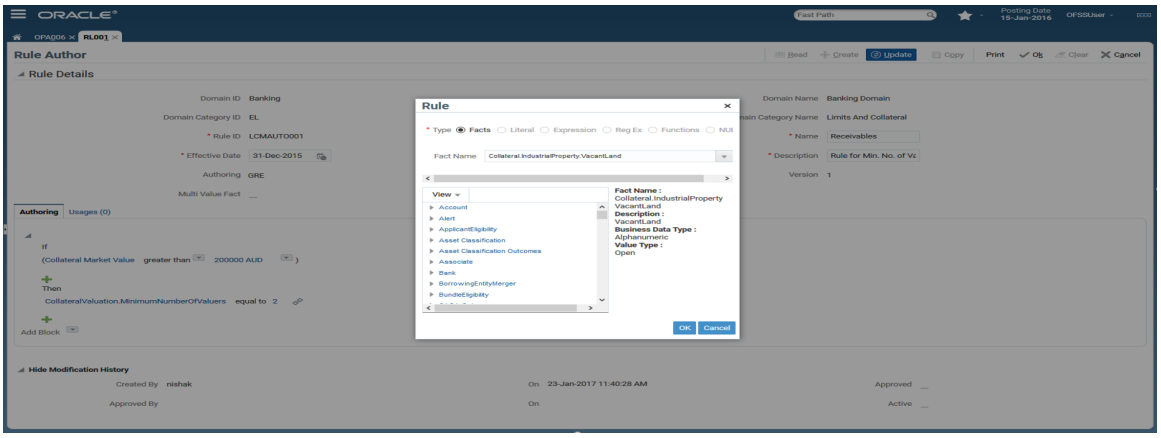
- Fact seed can be added to create fact named Collateral.IndustrialProperty.VacantLand, which will return value of Flex Field Attribute1, by defining following Fact Seed. Note that in JPQL query a.fiveAttributeFlexField.attribute1 is used to select value of Flex Field, as for this entity Metadata is given for less than or equal to 5 Fields, hence AbstractDomainObject.fiveAttributeFlexField is chosen as field to be mapped for dynamic Flex Field ORM mapping. Depending upon number of Flex Fields actually utilized, appropriate field out of following AbstractDomainObject class fields can be used in query: fiveAttributeFlexField, tenAttributeFlexField, fifteenAttributeFlexField, twentyAttributeFlexField, twentyFiveAttributeFlexField, thirtyAttributeFlexField.

Figure 17–11 Flex Field - Fact support Seed

```
insert into FLX_FA_FACTS_B (FACT_CODE, FACT_NAME, FACT_DESC, DOMAIN_CODE, DOMAIN_CATEGORY_CODE, VALUE_TYPE, BUSINESS_TYPE_CODE, RETRIEVAL_KEY, CREATED_BY,
CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATE_DATE, OBJECT_VERSION_NUMBER, LAST_UPDATE_LOGIN, FACT_CLASS, SHORT_NAME)
values ('Collateral.IndustrialProperty.VacantLand', 'Collateral.IndustrialProperty VacantLand', 'VacantLand', 'Banking', 'EL', 'Open', 'Alphanumeric',
'Collateral.IndustrialProperty.VacantLand', 'SYSTELLER', to_timestamp('01-02-2012 18:39:25.461000', 'dd-mm-yyyy hh24:mi:ss.ff'), 'SYSTELLER', to_timestamp(
'01-02-2012 18:39:25.479000', 'dd-mm-yyyy hh24:mi:ss.ff'), 2, null, 'NonFinancial', 'VacantLand');

insert into FLX_FA_GROUP_XREF (FACT_CODE, GROUP_CODE, CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATE_DATE, OBJECT_VERSION_NUMBER, LAST_UPDATE_LOGIN)
values ('Collateral.IndustrialProperty.VacantLand', 'Collateral.IndustrialProperty', null, null, null, null, null, null);

insert into FLX_FA_VALUE_BINDINGS (FACT_CODE, PARAM_NAME, DATA_TYPE_CODE, PARAM_DESC, VARIABLE_BASED_FACT, LITERAL_FACT_VALUE, BINDING_TYPE)
values ('Collateral.IndustrialProperty.VacantLand', 'id', null, null, 'Collateral.Id', null, 'Variable');

insert into flx_fa_value_datasources (FACT_CODE, JDBC_DERIVATION_QUERY,JAVA_DERIVATION_CLASS,DATA_SOURCE_CODE,HQL_DERIVATION_QUERY,DSN_CODE,DB_FUNCTION_NAME)
values ('Collateral.IndustrialProperty.VacantLand',null,null,'HQL','SELECT a.fiveAttributeFlexField.attribute1 FROM
com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty a WHERE a.collateralKey.id = :id',null,null);
```

- After adding this seed, Collateral.IndustrialProperty.VacantLand will be available as Fact while defining Pricing or Rules in OBP as follows:

Figure 17–12 Flex Field - Fact support After Seed



17.5 Flex Field – Validation Support

Basic validations are supported for Flex Fields using Metadata configuration (Refer Section 17.2.1 Maintain Flex Field Metadata using Seed Data Configuration (Fast Path: OPA006) Page). Using Flex field Metadata for each attribute in Entity, supported validation can be configured. Find below the details on the metadata configuration for supported validations. This needs be seeded or can be maintained using OPA006 page. This requires host restart to reflect.

Table 17–1 metadata configuration details for validations

Column	Description	Example
ENTITY_NAME	Name of the entity where flex field is applicable. Full qualified name.	com.ofss.fc.-domain.lcm.entity.collaterals.realestate.ResidentialProperty
ATTRIBUTE_NAME	Name of the attribute of the flex field. Attribute1, Attribute 2, or so on...	Attribute1
LABEL	Label or description of the attribute. When validation error message is thrown, this is used to throw exception.	Description of land

Column	Description	Example
	If not maintained, then the attribute name will be used for the validation message.	
ATTRIBUTE_DATA_TYPE	Attribute data type. Supported Data Types are: STRING / BIG_DECIMAL/ INTEGAR/ BOOLEAN/ DATE/ ENUM. Used when validating the field based on the datatype.	STRING
IS_MANDATORY	Validator field: Indicates	Y

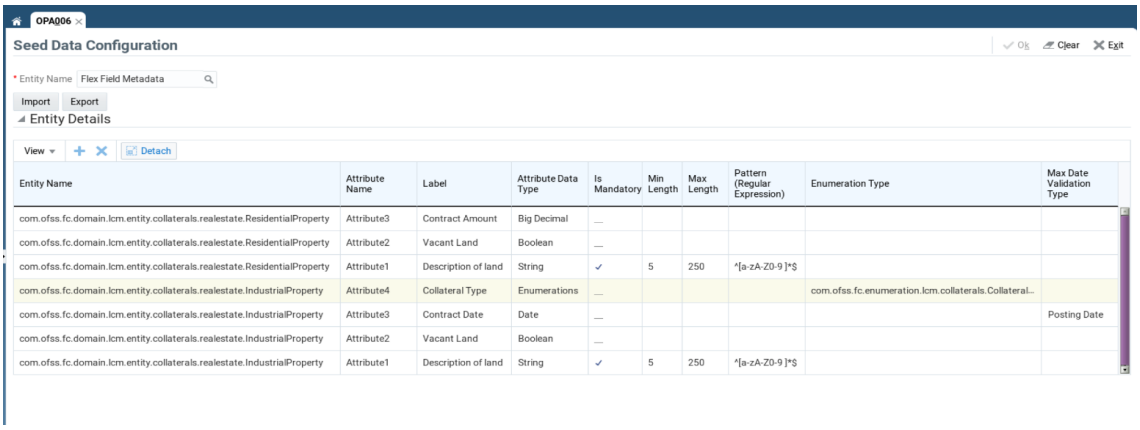
Column	Description	Example
	whether attribute value is mandatory. Check for not null / empty values.	
MIN_ LENGTH	Validator field: Indicates the minimum length required for the attribute. Validates, if maintained some value.	5
MAX_ LENGTH	Validator field: Indicates the maximum length required for the attribute. Validates, if maintained some value.	250

Column	Description	Example
PATTERN_REGEX	Validator field: Indicates the regular expression supported by the attribute. Validates, if maintained some value.	^[a-zA-Z0-9]*$
ENUM_TYPE_NAME	Validator field: Indicates the enumeration type supported by the attribute. Fully qualified name. Checks with the enumeration value sent. Validates, if maintained some value and data type	<<Applicable only for Enum type, e.g., com.ofss.fc.enumeration.lcm.collaterals.CollateralType>>

Column	Description	Example
	is Enum.	
MAX_DATE_VALIDATOR_TYPE	Validator field: Indicates the maximum date validator type. (Posting date / System date / Value date). Validates whether the date is not more than the mentioned date validator type. Date validator type supported are Posting date, System date and Value date. Validates, if maintained some value and data type is date.	<<Applicable only for Date type, e.g., POSTING_DATE>>

- For Example, suppose com.ofss.fc.domain.lcm.entity.collaterals.releaserate. IndustrialProperty and com.ofss.fc.domain.lcm.entity.collaterals.releaserate. Residentialproperty these Entities are configured to make use of Flex Fields by defining attribute metadata as follows:

Figure 17–13 Flex Field - Validation Support - Metadata



Entity Name	Attribute Name	Label	Attribute Data Type	Is Mandatory	Min Length	Max Length	Pattern (Regular Expression)	Enumeration Type	Max Date Validation Type
com.ofss.fc.domain.lcm.entity.collaterals.realestate.ResidentialProperty	Attribute3	Contract Amount	Big Decimal						
com.ofss.fc.domain.lcm.entity.collaterals.realestate.ResidentialProperty	Attribute2	Vacant Land	Boolean						
com.ofss.fc.domain.lcm.entity.collaterals.realestate.ResidentialProperty	Attribute1	Description of land	String	✓	5	250	^[a-zA-Z0-9]*$		
com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty	Attribute4	Collateral Type	Enumerations					com.ofss.fc.enumeration.lcm.collaterals.Collateral...	
com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty	Attribute3	Contract Date	Date						Posting Date
com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty	Attribute2	Vacant Land	Boolean						
com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty	Attribute1	Description of land	String	✓	5	250	^[a-zA-Z0-9]*$		

- Following is examples depict output of Flex Field validation framework, at runtime, when tested using SOAP UI toolkit.
- Mandatory Validation
  - Input:

Figure 17–14 Flex Field - Validation Support - Mandatory Validation - Input

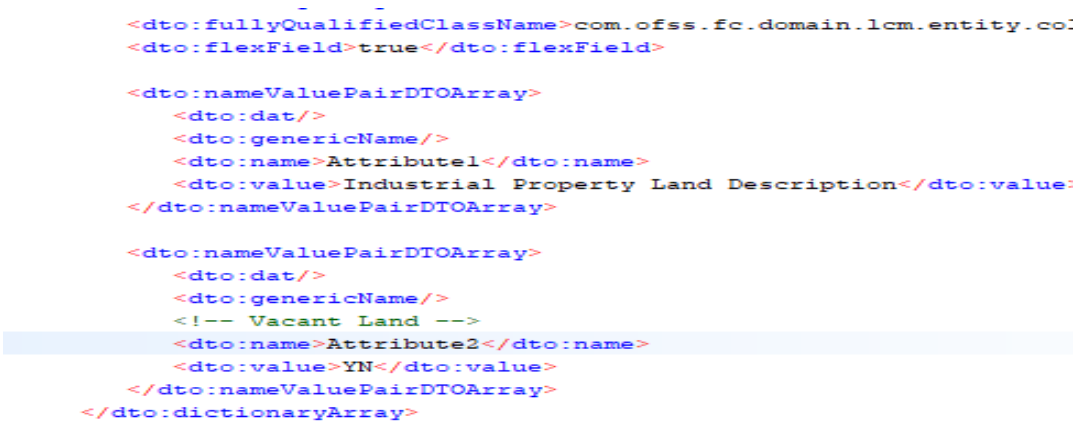
```
<dto:dictionaryArray>
   <dto:fullyQualifiedClassName>com.ofss.fc.domain.lcm.entity.collaterals.re
   <dto:flexField>true</dto:flexField>

   <!--<dto:nameValuePairDTOArray>
      <dto:dat/>
      <dto:genericName/>
      <dto:name>Attribute1</dto:name>
      <dto:value>Industrial Property Land Description</dto:value>
   </dto:nameValuePairDTOArray>-->

   <dto:nameValuePairDTOArray>
      <dto:dat/>
      <dto:genericName/>
      <!-- Vacant Land -->
      <dto:name>Attribute2</dto:name>
      <dto:value>N</dto:value>
   </dto:nameValuePairDTOArray>
</dto:dictionaryArray>
```

- Output:

Figure 17–15 Flex Field - Validation Support - Mandatory Validation -Output

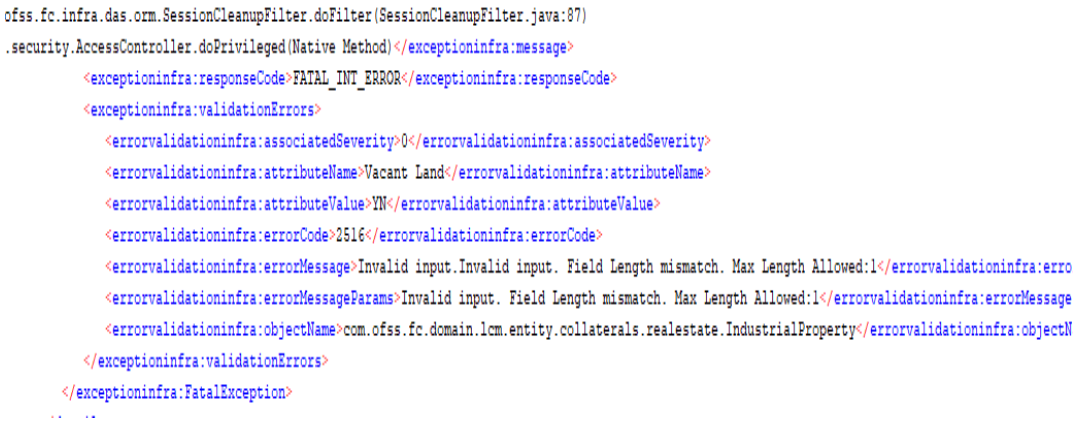
- Min/Max Length Validation
  - Input:

Figure 17–16 Flex Field - Validation Support - Min/Max Lenght Validation Input



- Output:

Figure 17–17 Flex Field - Validation Support - Min/Max Lenght Validation Output



```
ofss.fc.infra.das.orm.SessionCleanupFilter.doFilter(SessionCleanupFilter.java:87)
.security.AccessController.doPrivileged(Native Method)</exceptioninfra:message>
          <exceptioninfra:responseCode>FATAL_INT_ERROR</exceptioninfra:responseCode>
          <exceptioninfra:validationErrors>
            <errorvalidationinfra:associatedSeverity>0</errorvalidationinfra:associatedSeverity>
            <errorvalidationinfra:attributeName>Vacant Land</errorvalidationinfra:attributeName>
            <errorvalidationinfra:attributeValue>YN</errorvalidationinfra:attributeValue>
            <errorvalidationinfra:errorCode>2516</errorvalidationinfra:errorCode>
            <errorvalidationinfra:errorMessage>Invalid input.Invalid input. Field Length mismatch. Max Length Allowed:1</errorvalidationinfra:erro
            <errorvalidationinfra:errorMessageParams>Invalid input. Field Length mismatch. Max Length Allowed:1</errorvalidationinfra:errorMessage
            <errorvalidationinfra:objectName>com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty</errorvalidationinfra:objectN
          </exceptioninfra:validationErrors>
        </exceptioninfra:FatalException>
```

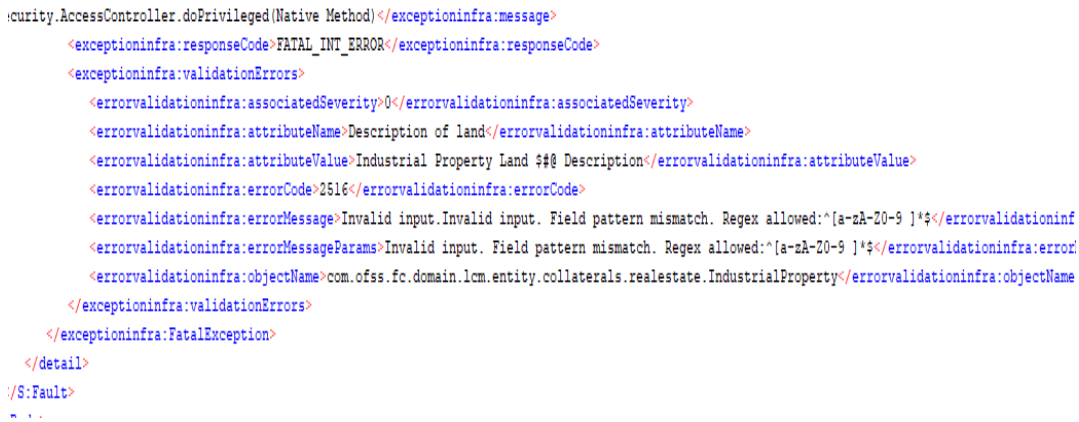
- Pattern (Regex) Validation
  - Input:

Figure 17–18 Flex Field - Validation Support - Pattern (Regex) Validation Input

```
<dto:dictionaryArray>
   <dto:fullyQualifiedClassName>com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty</dto:fullyQual
   <dto:flexField>true</dto:flexField>

   <dto:nameValuePairDTOArray>
      <dto:dat/>
      <dto:genericName/>
      <dto:name>Attribute1</dto:name>
      <dto:value>Industrial Property Land $#@ Description </dto:value>
   </dto:nameValuePairDTOArray>

   <dto:nameValuePairDTOArray>
      <dto:dat/>
      <dto:genericName/>
      <!-- Vacant Land -->
      <dto:name>Attribute2</dto:name>
      <dto:value>N</dto:value>
   </dto:nameValuePairDTOArray>
</dto:dictionaryArray>
```

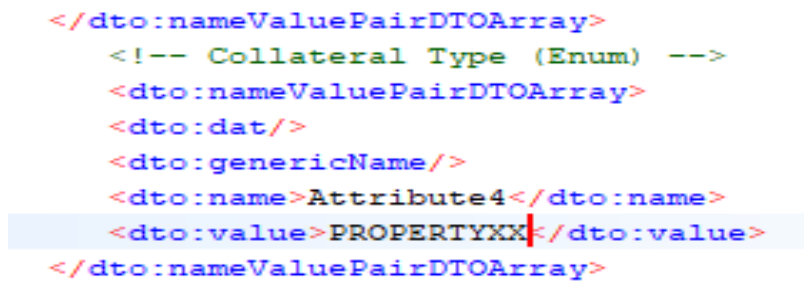
- Output:

Figure 17–19 Flex Field - Validation Support - Pattern (Regex) Validation Output



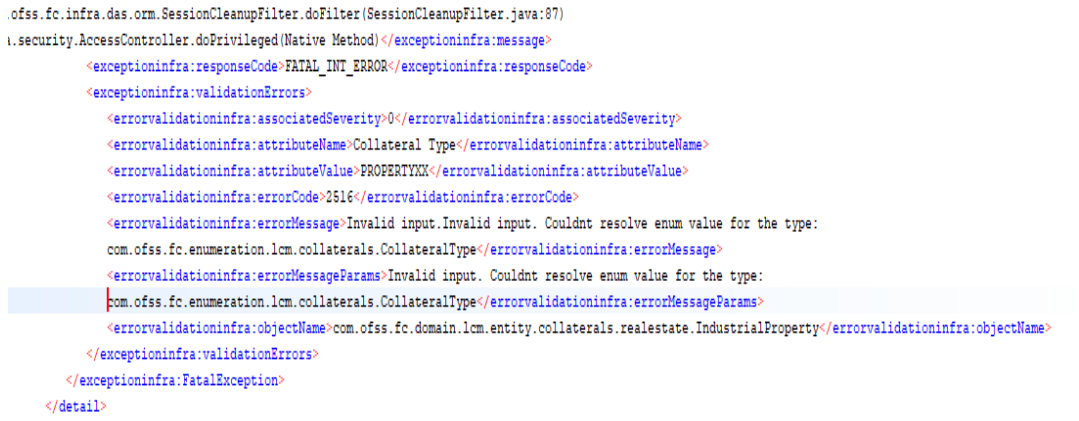
- Enum Validation
  - Input:

Figure 17–20 Flex Field - Validation Support - Enum Validation Input



  - Output

Figure 17–21 Flex Field - Validation Support - Enum Validation Output



- Date Validation
  - Input:

Figure 17–22 Flex Field - Validation Support - Date Validation Input

```
<dto:fullyQualifiedClassName>com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProp
<dto:flexField>true</dto:flexField>
<!-- Description of land (String) -->
<dto:nameValuePairDTOArray>
<dto:dat/>
<dto:genericName/>
<dto:name>Attribute1</dto:name>
<dto:value>Industrial Property Land Desc</dto:value>
</dto:nameValuePairDTOArray>
<!-- Is Vacant Land (Boolean) -->
<dto:nameValuePairDTOArray>
<dto:dat/>
<dto:genericName/>
<dto:name>Attribute2</dto:name>
<dto:value>N</dto:value>
</dto:nameValuePairDTOArray>
<!-- Contract Date (Date) -->
<dto:nameValuePairDTOArray>
<dto:dat/>
<dto:genericName/>
<dto:name>Attribute3</dto:name>
<dto:value>20170115</dto:value>
</dto:nameValuePairDTOArray>
<!-- Collateral Type (Enum) -->
<dto:nameValuePairDTOArray>
<dto:dat/>
<dto:genericName/>
<dto:name>Attribute4</dto:name>
<dto:value>PROPERTY</dto:value>
</dto:nameValuePairDTOArray>
</dto:dictionaryArray>
```

  - Output:

Figure 17–23 Flex Field - Validation Support - Date Validation Output

```
om.ofss.fc.domain.lcm.service.collaterals.CollateralService.updateCollateral(CollateralService.java:366)
om.ofss.fc.app.lcm.service.collaterals.CollateralApplicationService.updateCollateral(CollateralApplicationService.java:562)
om.ofss.fc.appx.lcm.service.collaterals.CollateralApplicationServiceSpi.updateCollateral(CollateralApplicationServiceSpi.java:132)
un.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
om.ofss.fc.webservice.servlet.DefaultValuesFilter.doFilter(DefaultValuesFilter.java:94)
ava.security.AccessController.doPrivileged(Native Method)
om.ofss.fc.webservice.servlet.XSIFilter.doFilter(XSIFilter.java:122)
om.ofss.fc.infra.das.orm.SessionCleanupFilter.doFilter(SessionCleanupFilter.java:87)
ava.security.AccessController.doPrivileged(Native Method)</exceptioninfra:message>
<exceptioninfra:responseCode>FATAL_INT_ERROR</exceptioninfra:responseCode>
<exceptioninfra:validationErrors>
<errorvalidationinfra:associatedSeverity>0</errorvalidationinfra:associatedSeverity>
<errorvalidationinfra:attributeName>Contract Date</errorvalidationinfra:attributeName>
<errorvalidationinfra:attributeValue>20170115</errorvalidationinfra:attributeValue>
<errorvalidationinfra:errorCode>2516</errorvalidationinfra:errorCode>
<errorvalidationinfra:errorMessage>Invalid input.Invalid input. Date is greater than POSTING_DATE:20160115000000</errorvalidationinfra:error
<errorvalidationinfra:errorMessageParams>Invalid input. Date is greater than POSTING_DATE:20160115000000</errorvalidationinfra:errorMessageP
<errorvalidationinfra:objectName>com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty</errorvalidationinfra:objectName>
</exceptioninfra:validationErrors>
</exceptioninfra:FatalException>
</detail>
</S:Fault>
</S:Body>
```

17.6 Flex Field – Usage Instructions

Perform the following steps for usage:

- Identify the Entity for which Flex Field support is required.
- Refer seed table FLX_FW_FLX_FLD_ENTITIES_ALL_B to ascertain that Flex Field is already provisioned that Entity Table with sufficient number of MAX_FLX_FLD_COLUMN_COUNT.
- If Flex Field is not provisioned in FLX_FW_FLX_FLD_ENTITIES_ALL_B then, verify with the product team.
- In case Flex Field is provisioned but MAX_FLX_FLD_COLUMN_COUNT is not sufficient then follow steps mentioned in section to get extra columns provisioned Section 17.1 Flex Field - Provisioning
- Pass or Retrieve the attributes via Dictionary object, as per Section 17.3 Runtime Storage and Retrieval of Flex Field Attribute values.

- If validation required for any of the attributes in the Flex Field, configure or seed the metadata as per Section 17.5 Flex Field – Validation Support . By dynamic provisioning & configuration page, this will be enabled via configuration page: OPA006.
- If fact required, follow Section 17.2 Flex Field - Utilization for details.

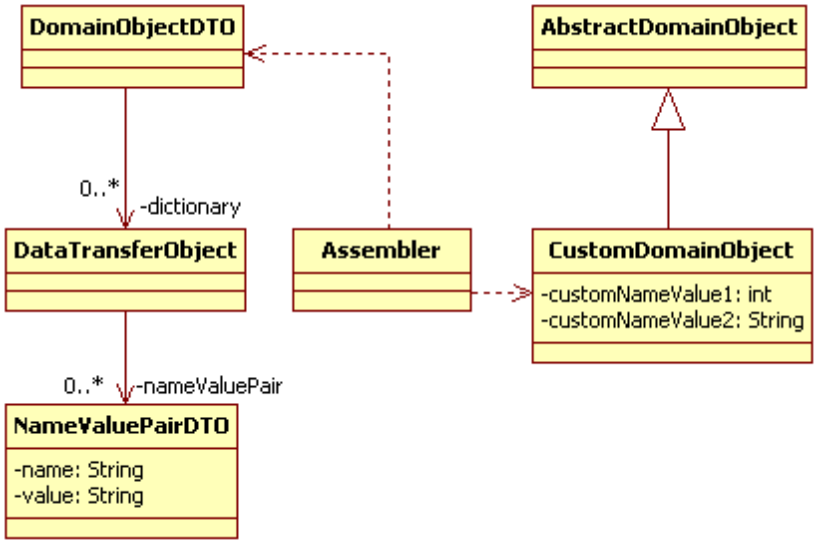
18 Extensibility of Domain Objects - Dictionary Pattern

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. In this approach, the data model for the custom fields is extended from that of the domain objects itself and hence can 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.

Figure 18–1 Extensibility of Domain Objects - Framework



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.

18.1 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 18–2 Code Extract

```
com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePairDTO1= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO();
nameValuePairDTO1.setGenericName("com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate.CustomValue1");
nameValuePairDTO1.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];
dictionaryArray[0]=dictionary;
```

18.2 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.

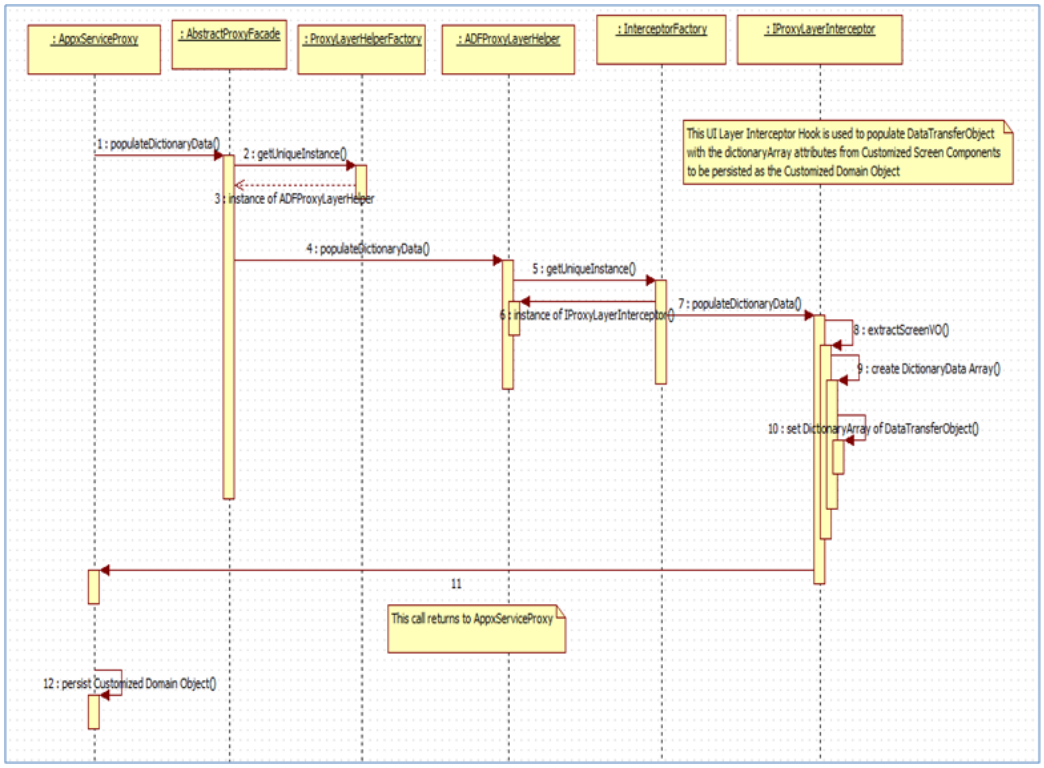
18.2.1 Interceptor Hook to Persist Customized Domain Object Attributes

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 18–3 Interceptor Hook to Persist Customized Domain Object



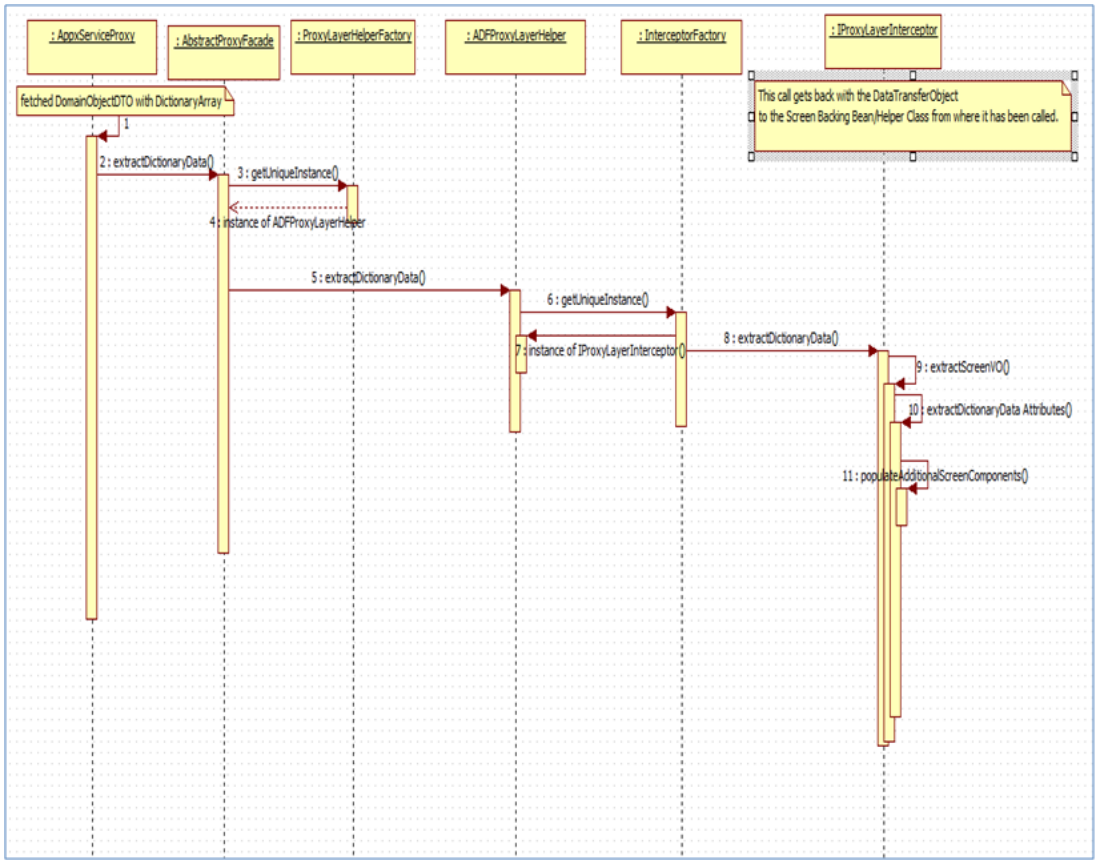
18.2.2 Interceptor Hook to Fetch Customized Domain Object Attributes

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 18–4 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.MessageTemplateApplicationServiceProxyFacade=com.ofss.fc.ui.taskflows.ep.messageTemplateUI.view.interceptor.MessageTemplateUIInterceptor

com.ofss.fc.appx.party.service.contact.service.client.proxy.ContactPointApplicationServiceProxyFacade=com.ofss.fc.ui.view.party.contactPoint.interceptor.ContactPointUIInterceptor

18.3 Dictionary Data Transfer from UI to Host

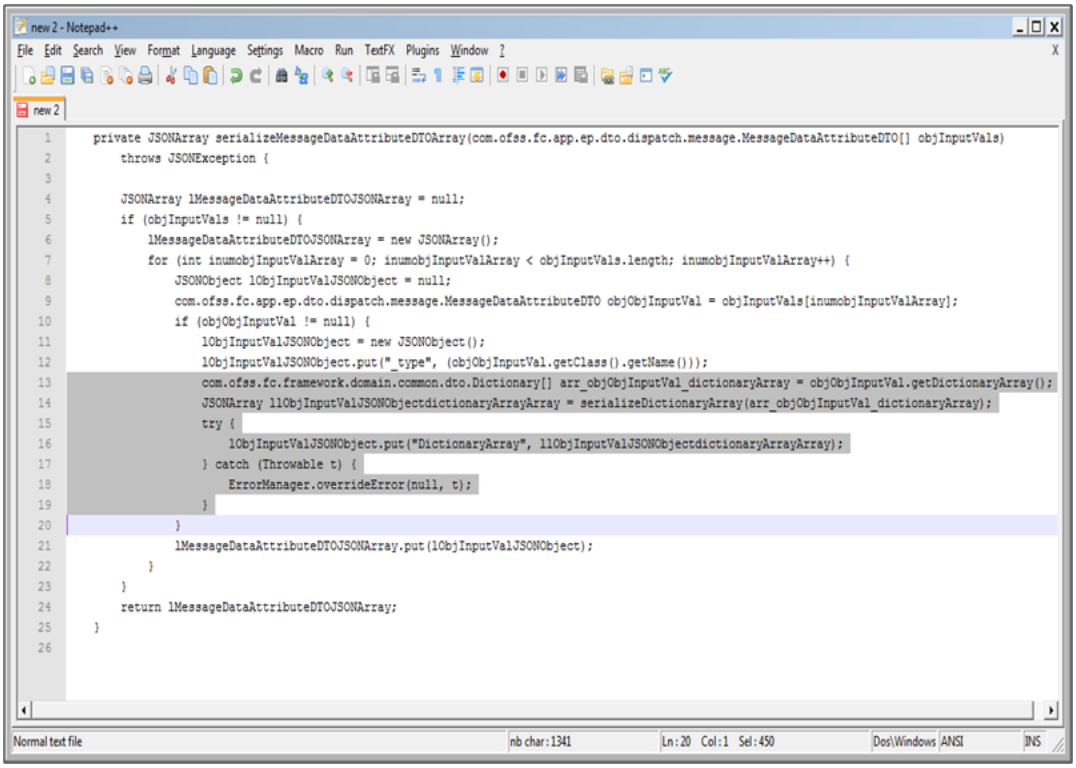
The section describes the dictionary data transfer from UI to Host.

18.3.1 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.

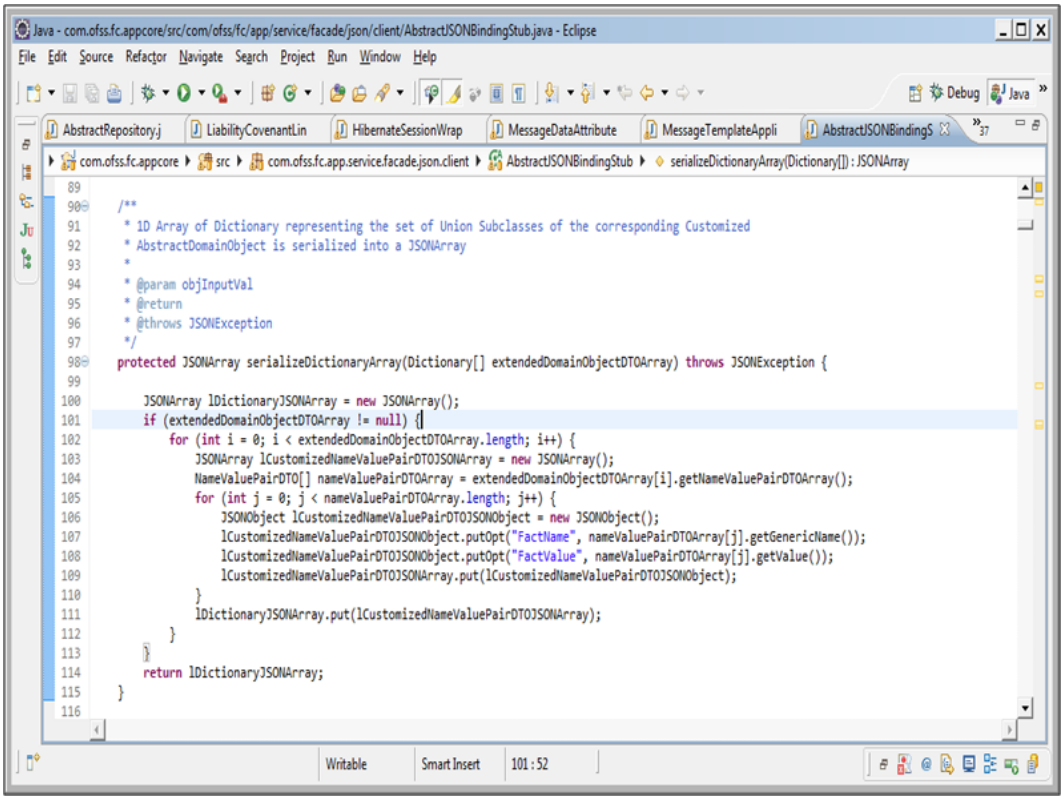
Figure 18–5 JSONClient constructs the JSON Object



<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 18–6 SerializeDictionaryArray to include GenericName and Value attributes

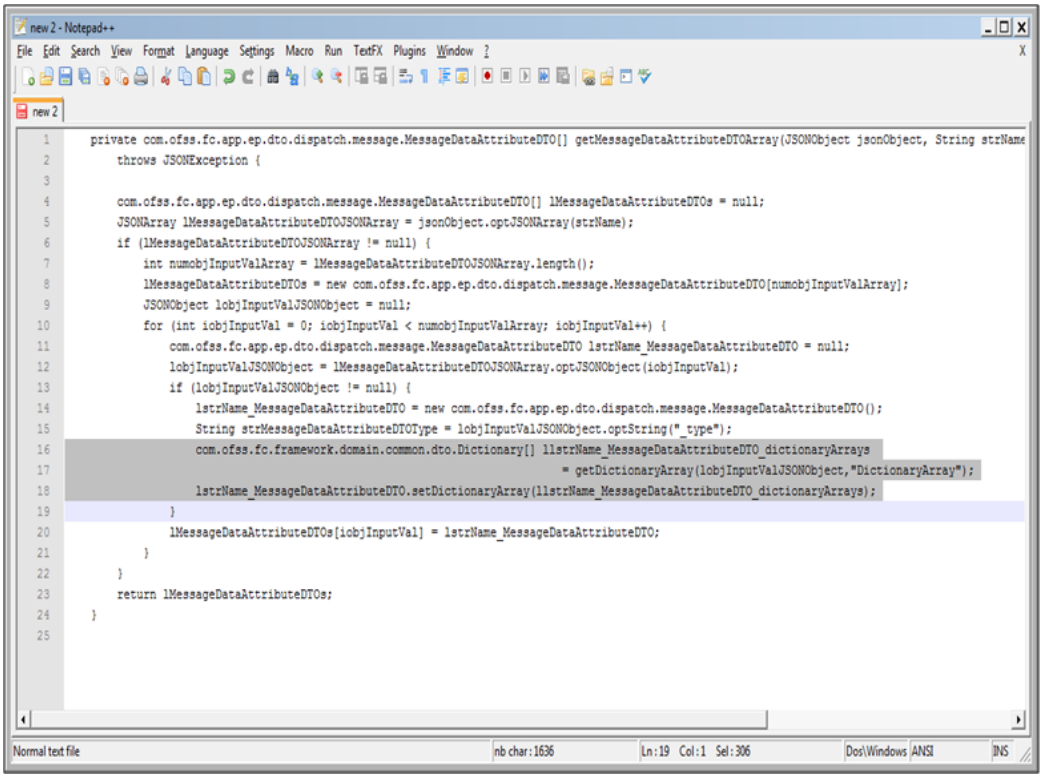


AbstractJSONBindingStub class's serializeDictionaryArray to include "genericName" and "value" attributes of NameValuePairDTOArray

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.

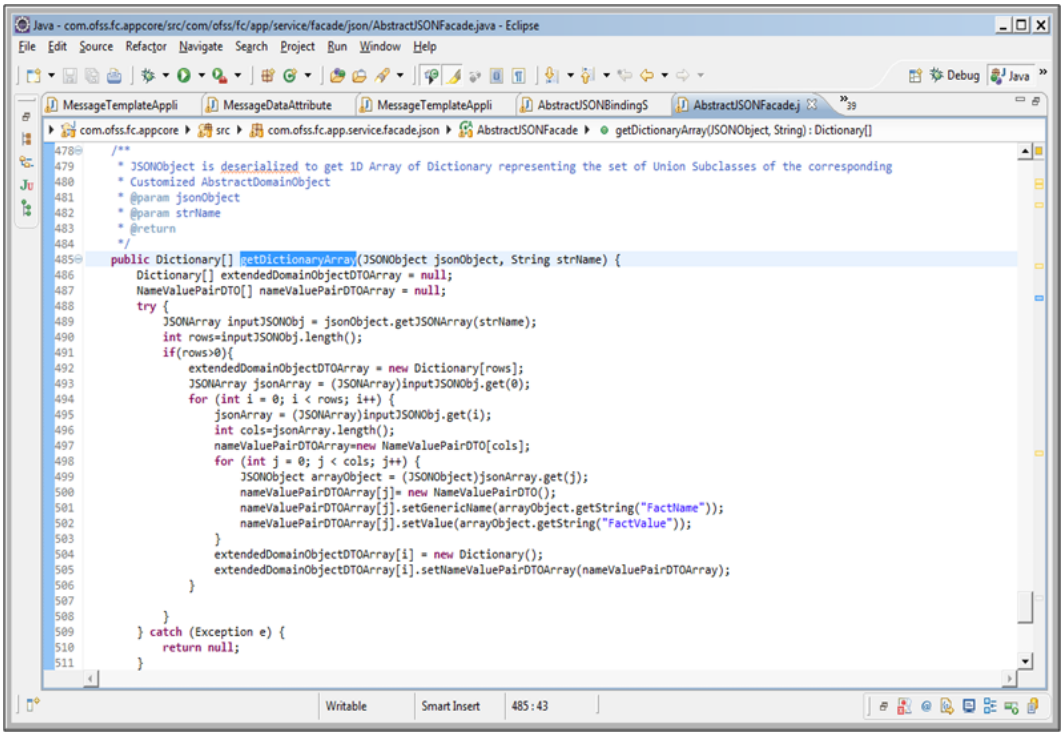
Figure 18–7 Host Server JSONFacade extracts the attribute of JSON Object



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 18–8 AbstractJSONFacade's getDictionaryArray method



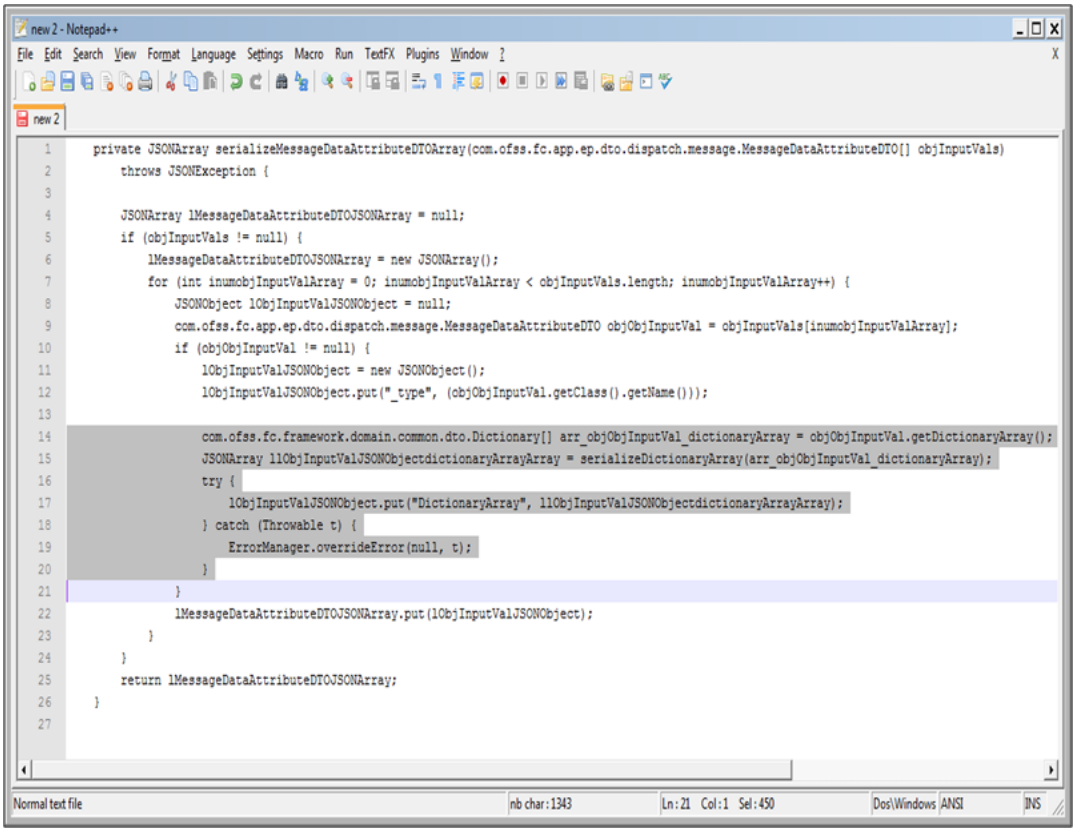
AbstractJSONFacade's getDictionaryArray method that unmarshalls the "genericName" and "value" from JSON Object to get the dictionaryArray attribute

18.3.2 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:

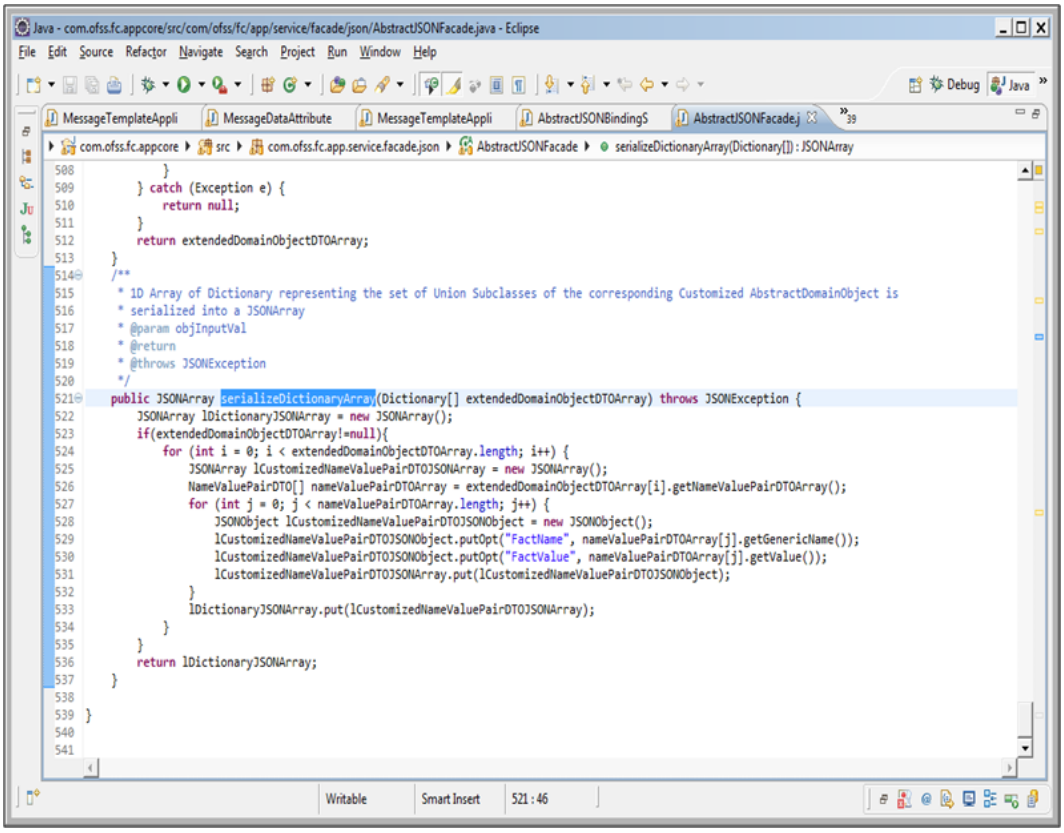
Figure 18–9 Host Server JSONFacade constructs the JSON Object



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 18–10 AbstractJSONFacade's serializeDictionaryArray to include Generic Name and Value attributes

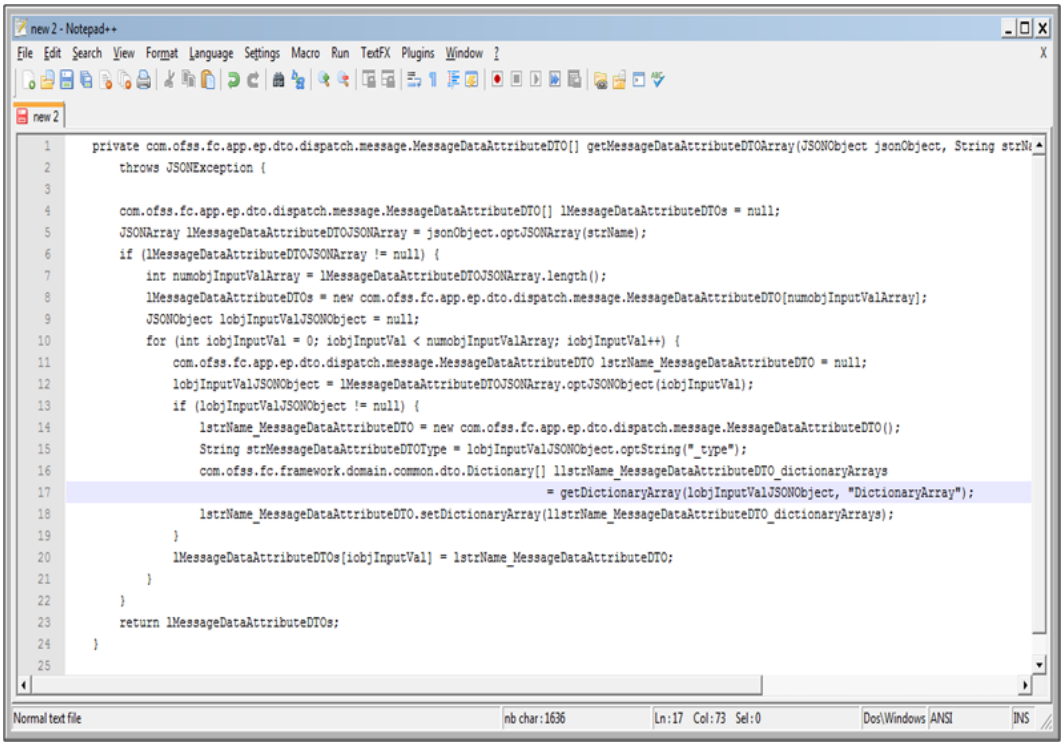


AbstractJSONFacade's serializeDictionaryArray to include "genericName" and "value" attributes of NameValuePairDTOArray

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.

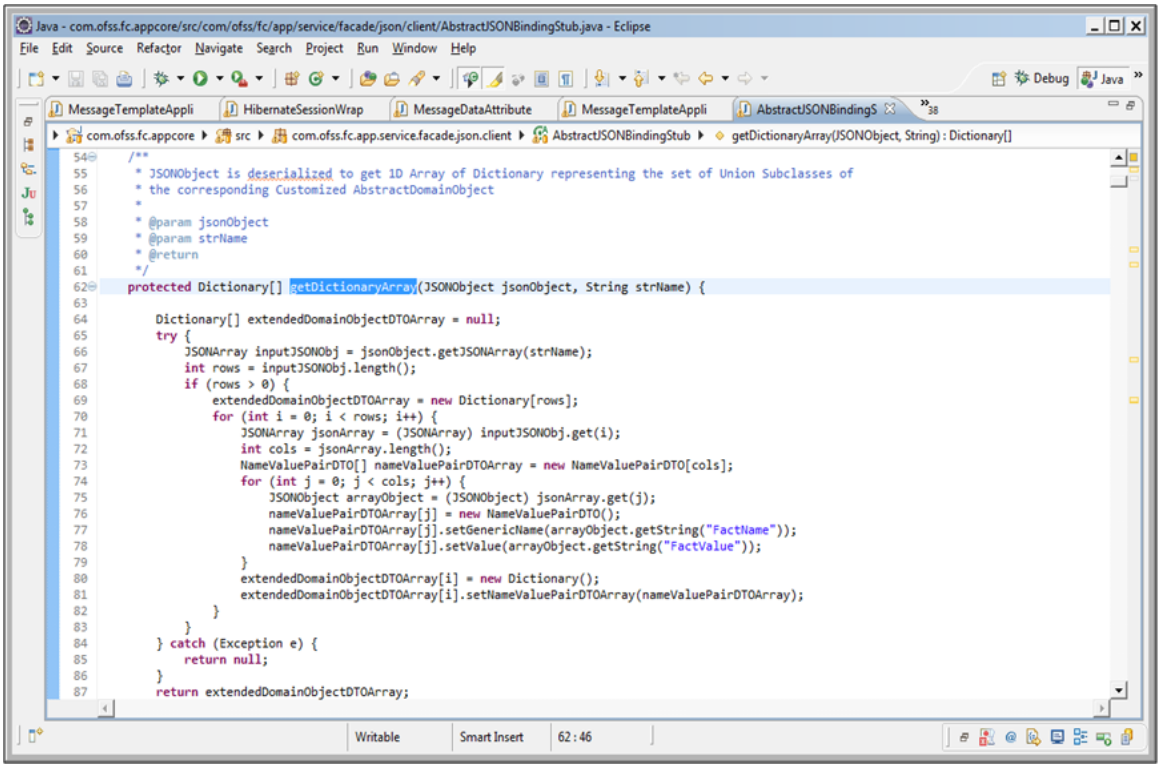
Figure 18–11 UI Server JSONClient extracts the DictionaryArray attribute



In the UI Server, <ApplicationService>JSONClient extracts the "DictionaryArray" attribute of JSON Object and sets it as <DomainObjectDTO>DTO's dictionaryArray attribute

The above process uses AbstractJSONBindingStub's getDictionaryArray method that unmarshalls the 'genericName' and 'value' from JSON Object to get the dictionaryArray attribute.

Figure 18–12 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.

18.4 Translating Dictionary Data into Custom Domain Object

This section describes the details of translating dictionary data into custom domain object.

18.4.1 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;
try {
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>;
}
```

18.4.2 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.

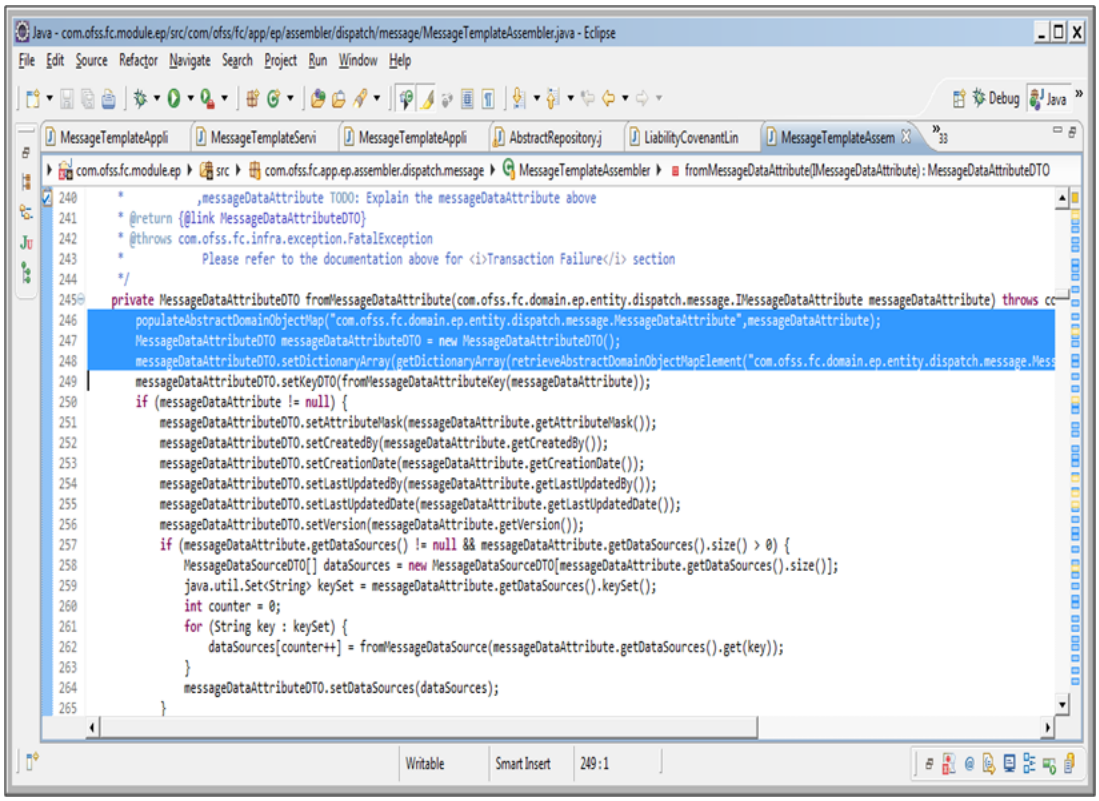
In AbstractAssembler.java, we have defined the method à

```
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 18–13 Instantiation of DataTransferObjects



18.4.3 Defining of Customized Domain Objects

When we are viewing the customized attributes on the screen, we need to fetch the Customized Abstract Domain Object data into the Domain Object DTO. This is why the customized attributes in the Customized Domain Object have to be populated in the dictionary array of the Domain Object DTO.

This is done in the AbstractAssembler which returns the dictionary array of the Domain Object DTO based on the Abstract Domain Object passed to it, through a method called "getDictionaryArray". To achieve this, the AbstractAssembler firstly needs to understand which is a customized domain object.

In preferences.xml we have defined the following:

```
<Preference name="CustomizedAbstractDomainObjectConfig"
PreferencesProvider="com.ofss.fc.infra.config.impl.DBBasedProperty
Provider"
parent="jdbcpreference"
propertyFileName="select prop_id, prop_value from flx_fw_config_
all_b where category_id = 'CustomizedAbstractDomainObjectConfig'"
syncTimeInterval="600000" />
```

We have to insert a record in table flx_fw_config_all_b to identify a Customized Domain Object in the following manner.

```
INSERT INTO FLX_FW_CONFIG_ALL_B (PROP_ID, CATEGORY_ID, PROP_VALUE,
FACTORY_SHIPPED_FLAG, PROP_COMMENTS, SUMMARY_TEXT, CREATED_BY,
CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATED_DATE, OBJECT_STATUS_
FLAG, OBJECT_VERSION_NUMBER)
VALUES ('com.ofss.fc.domain.ep.entity.action.ActivityEventAction',
'CustomizedAbstractDomainObjectConfig',
'com.ofss.fc.domain.ep.entity.action.CustomizedActivityEventActio
n', 'y', '',
'Customized object of
com.ofss.fc.domain.ep.entity.action.ActivityEventAction',
'ofssuser',
'09-SEP-14 05.53.56.000000 PM', 'ofssuser', '09-SEP-14
05.53.56.000000 PM', 'A', 1);
```

The AbstractAssembler identifies a customized domain object by deciphering the above information.

So every Customized Domain Object has to be defined in flx_fw_config_all_b with category_id = 'CustomizedAbstractDomainObjectConfig'.

Only if such a definition exists, the abstract domain object passed is identified to be a customized domain object and the corresponding Domain Object DTO is provided with its dictionary array.

However, if the abstract domain object passed is not identified to be a customized domain object, the corresponding Domain Object DTO is provided with a dictionary array which has null value.

18.5 Customized Domain Object ORM Configuration

This section describes the details of customized domain object ORM configuration.

18.5.1 Case 1 - Non-Inheritance based mapping

Non-inheritance based mapping refers to those domain objects that are not mapped as a 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.orm.xml.

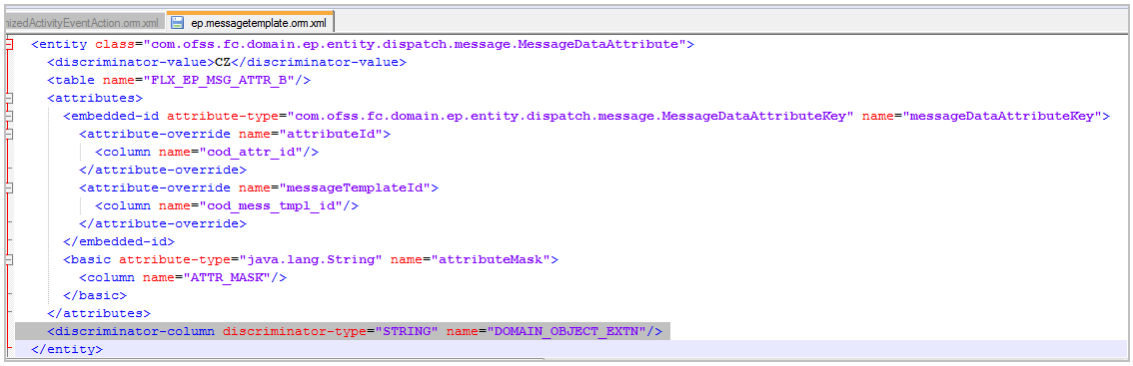
Adding Discriminator column mapping in existing ORM file

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 ORM 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.

Figure 18–14 Adding Discriminator Column Mapping in Existing ORM file



A new ORM file mapping to Customized Domain Object is added

The following figure explains adding a new ORM file mapping to Customized Domain Object.

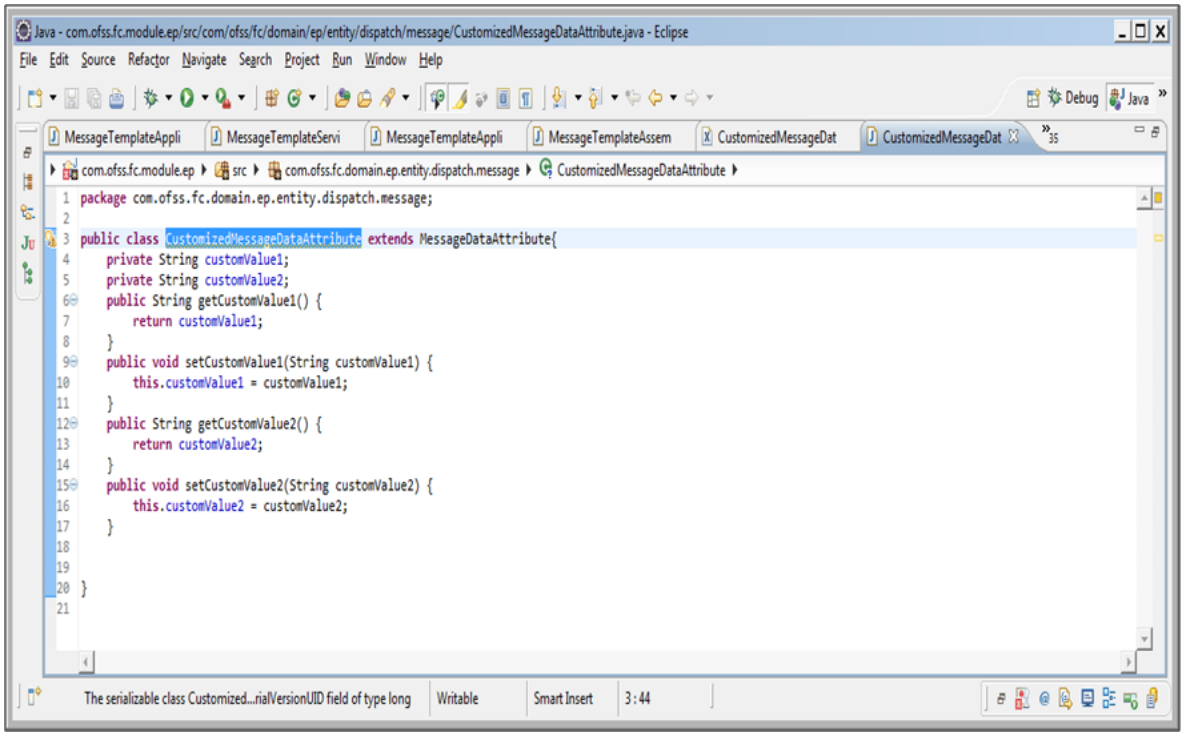
Figure 18–15 ORM File Mapping to Customized Domain Object

For example a new file CustomizedMessageDataAttribute.orm.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 18–16 Adding New Java File to the Customized Domain Object

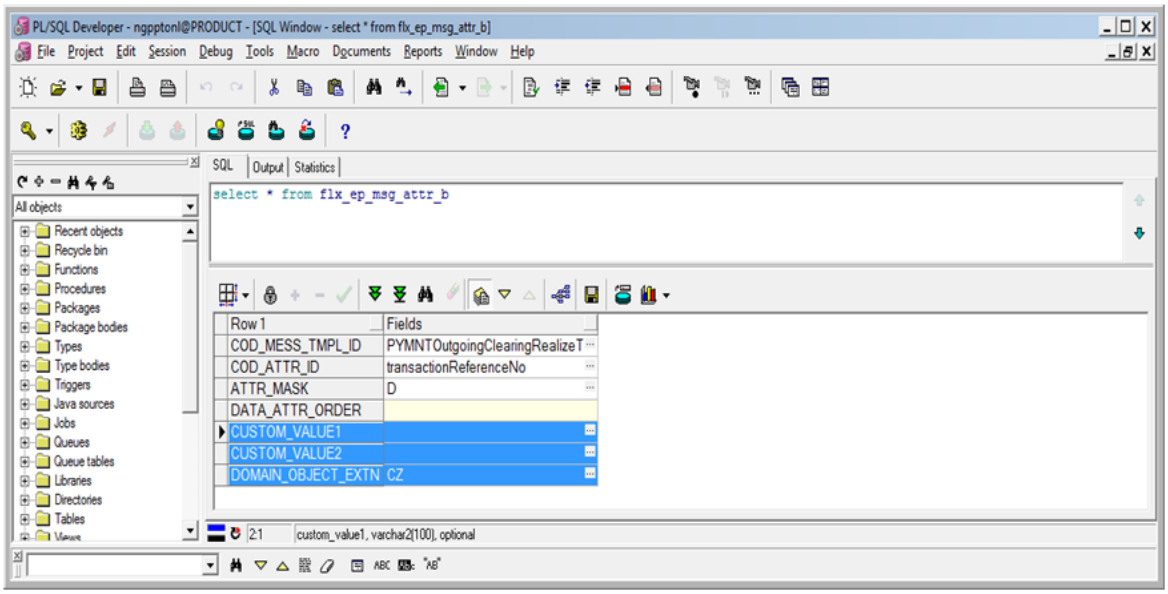


A Java File is added corresponding to the existing Abstract Domain Object. This will be extending the Abstract Domain Object that we are extending.

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 18–17 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.

18.5.2 Case 2 - Mapped as ORM Subclass

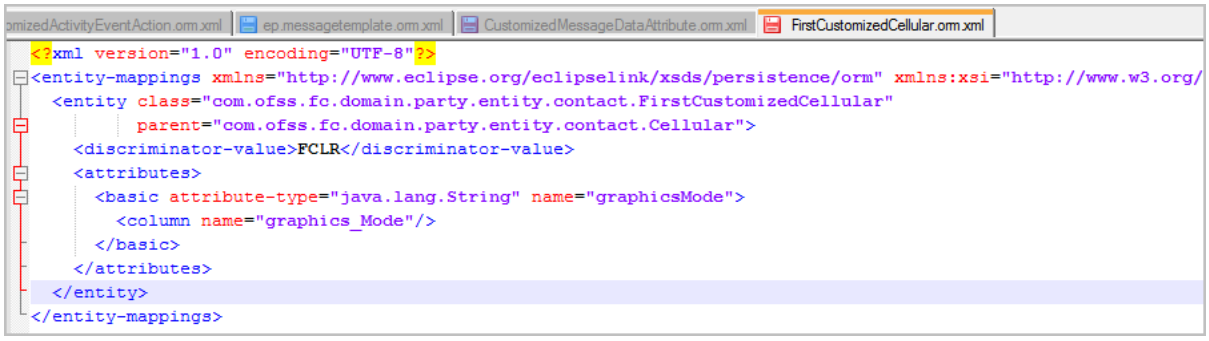
The maintenance domain objects which are mapped as a Subclass already have an existing discriminator. For the purpose of identifying the extended domain object in the same table, we shall be using the existing discriminator.

Let us take the example of 'com.ofss.fc.domain.party.entity.contact.Cellular'. This is mapped as a subclass in ContactPoint.orm.xml.

A new ORM file mapping to Customized Domain Object is added

The following figure explains adding a new ORM file mapping to Customized Domain Object.

Figure 18–18 Adding a New ORM File Mapping to Customized Domain Object

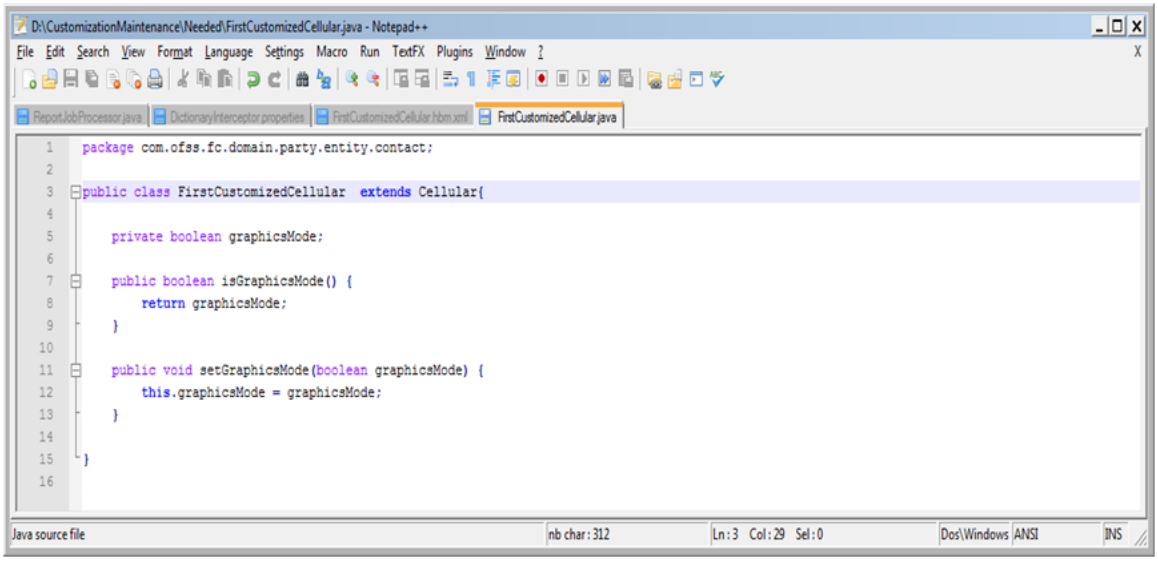


A new file FirstCustomizedCellular.orm.xml is introduced to include the extra attributes added by consulting or any other third party along with the discriminator value 'FCLR'. This file will map to the new customized domain object 'com.ofss.fc.domain.party.entity.contact.FirstCustomizedCellular' and 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 18–19 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.

18.5.3 Case 3 - Mapped as ORM 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.orm.xml as a union subclass.

Use the customized entity 'com.ofss.fc.cz.nab.domain.lcm.entity.limits.facility.proposedFacility.CustomizedProposedFacility' for the purpose of extensibility of this domain object.

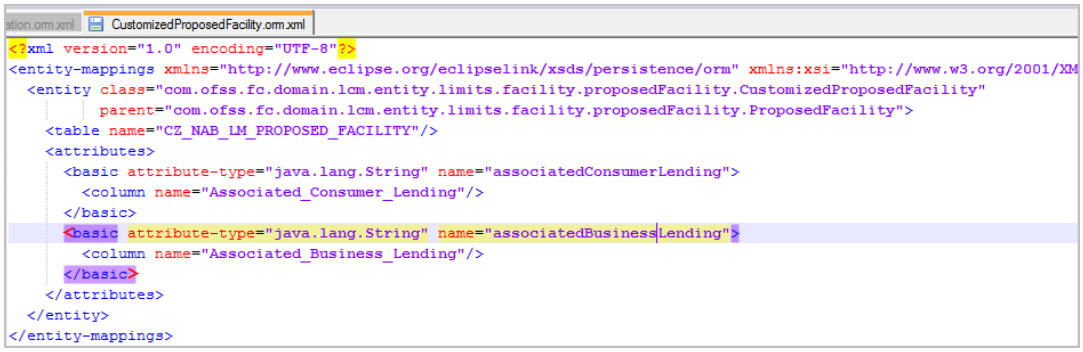
Adding Discriminator in ORM file where base class has been mapped is not required

The existing Facility.orm.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 ORM file mapping to Customized Domain Object is added

The following figure explains adding a new ORM file mapped to new Customized Domain Object.

Figure 18–20 New ORM File Mapping



For example, a new file CustomizedProposedFacility.orm.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 18–21 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;
    }

    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 18–22 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)
11 /
```

Adding Customized JPQL Queries whenever the Domain Object is Referred

The following file has the attribute 'CustomizedORMQueriesConfig' to fire the Customized JPQL if required: Preferences.xml.

The attribute is as follows:

```
<Preference name="CustomizedORMQueriesConfig"

PreferencesProvider="com.ofss.fc.infra.config.impl.JavaConstantsCo
nfigProvider"
overriddenBy="CustomizedORMQueriesConfigOverride"
parent="jdbcpreference"
propertyFileName="com.ofss.fc.common.CustomizedORMQueriesConfig"
syncTimeInterval="600000" />
```

The following files have also been changed to fire the Customized JPQL if required.

com.ofss.fc.framework.domain@/com/ofss/fc/framework/repository/AbstractRepository.java

com.ofss.fc.common.jar@/src/com/ofss/fc/common/CustomizedORMQueriesConfig.java

The following file has the attribute 'CustomizedORMQueriesConfigOverride' to fire the Customized JPQL if required.

<lzn>/au/config/Preferences.xml

```
<Preference name="CustomizedORMQueriesConfigOverride"

PreferencesProvider="com.ofss.fc.infra.config.impl.JavaConstantsCo
nfigProvider"
parent=""
propertyFileName="com.ofss.fc.lz.au.common.CustomizedORMQueriesCon
fig"
syncTimeInterval="600000"/>
```

Therefore, com.ofss.fc.lz.au.common.CustomizedORMQueriesConfig.java file needs to have the old JPQL query name mapped to the customized query name for this domain object.

Similarly, extensibility of domain objects mapped as joined-subclass can also be done.

18.5.4 Case 4 - Mapped as ORM 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 ORM file.

The corresponding additional columns have to be added in the domain object table in question.

18.6 Extensibility using Dictionary in Origination Application

In this section, the Application Form page (Fast path: OR097) of the Oracle Banking Platform is taken as an example.

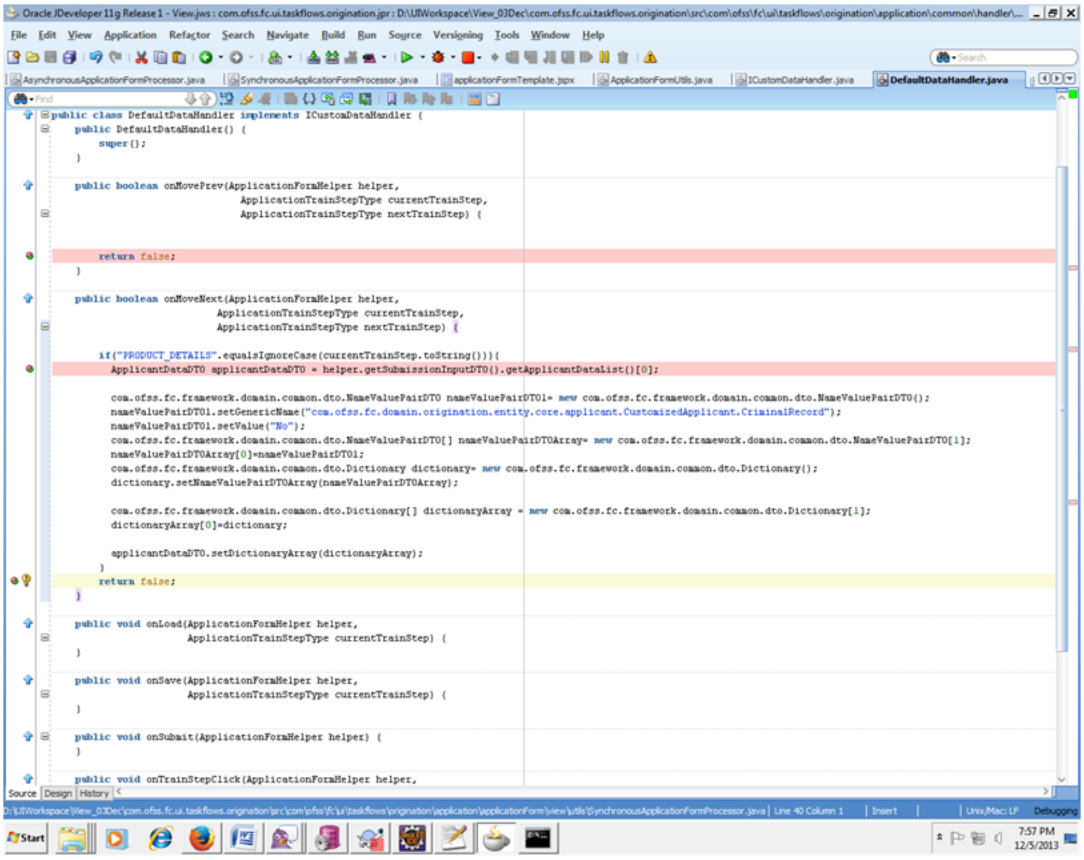
18.6.1 ICustomDataHandler's as DictionaryArray Interceptor

The backing bean method of OR097 - Application Form 'com.ofss.fc.ui.taskflows.origination.application.applicationForm.view.backing.ApplicationForm.moveNext ()' calls implementation of com.ofss.fc.ui.taskflows.origination.application.common.handler.ICustomDataHandler.

Implementation of com.ofss.fc.ui.taskflows.origination.application.common.handler.ICustomDataHandler can be configured in OriginationConfiguration.properties. Property name is **customDataHandler**

ApplicationFormHelper.getSubmissionInputDTO() will give the master DTO for the application form.

Figure 18–23 CustomDataHandler's as DictionaryArray Interceptor

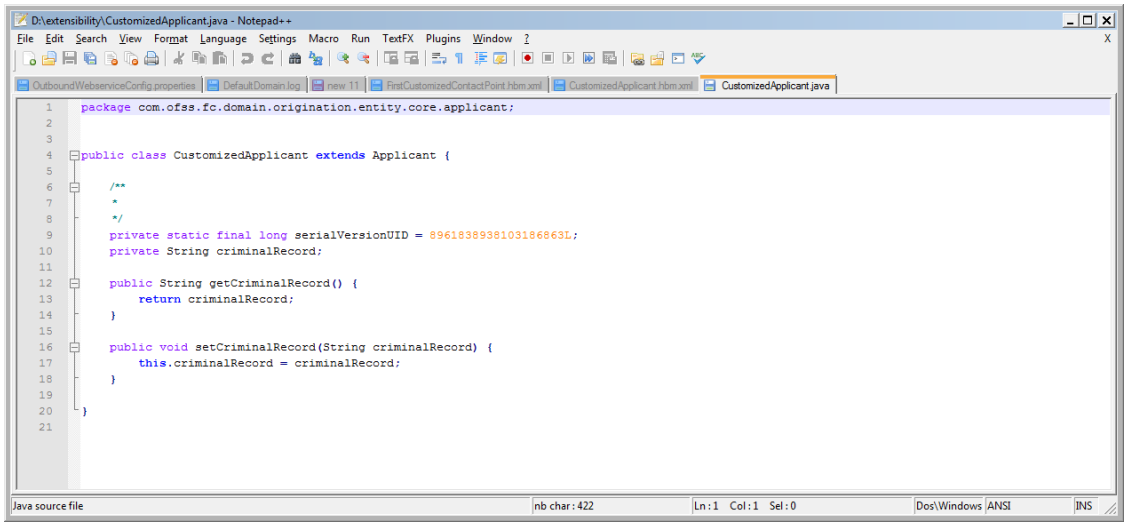


This hook should be used to populate the dictionary array of the concerned DTO at the correct stage of application form entry.

18.6.2 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.

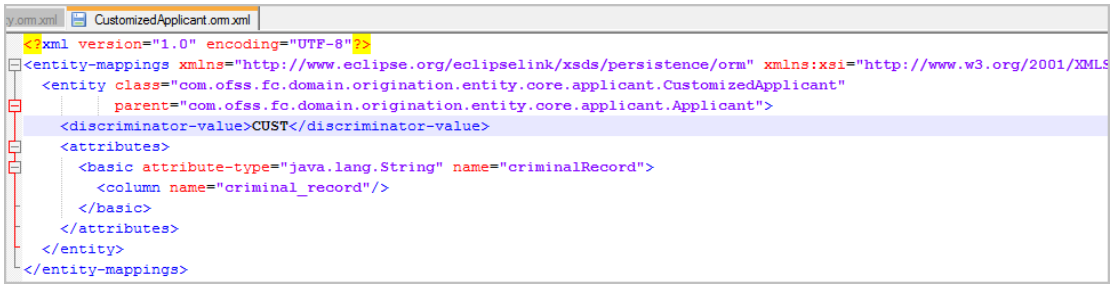
Figure 18–24 Create Customized Abstract Domain Object Class



18.6.3 Create Customized Abstract Domain Object ORM Mapping File

A new file .orm.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.

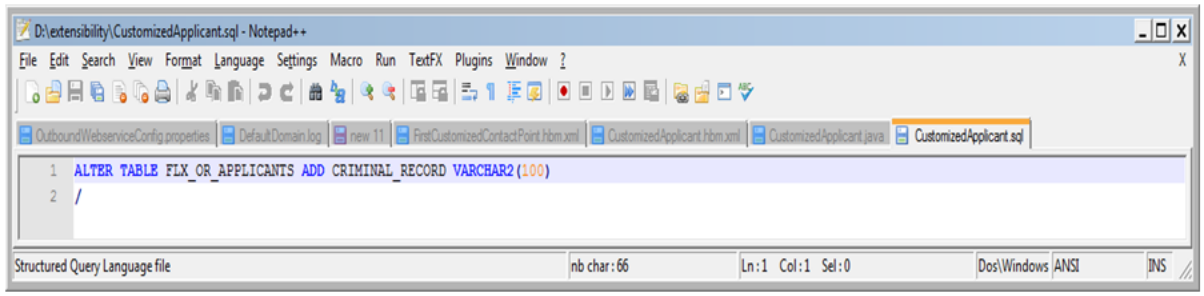
Figure 18–25 Create Customized Abstract Domain Object ORM Mapping File



18.6.4 Create Customized Abstract Domain Object Attribute Columns

The extra columns have to be added to the domain object table of this domain object.

Figure 18–26 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.

18.7 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 18–27 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 = 3762836902405427 91L;

    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() {
```

The following extra columns have been added in the domain object table "flx_ep_msg_tmpl_b".

Figure 18–28 Domain Object Table

Name	Type	Nullable	Default	Storage	Comments
COD_TMPL_ID	VARCHAR2(100)	☐			Indicates unique message template id
DESTINATION_TYPE	VARCHAR2(20)	☑			Indicates destination type like SMS,Email..
MSG_TMPL_NAME	VARCHAR2(100)	☑			Indicates message template name
MSG_TMPL_DESC	VARCHAR2(100)	☑			Indicates message template description
TXT_MSG_TMPL	CLOB	☑			Indicates message template buffer
CREATED_BY	VARCHAR2(64)	☑			Indicates the creator
CREATION_DATE	DATE	☑			Indicates the creation Date
LAST_UPDATED_BY	VARCHAR2(64)	☑			Indicates the approver of the transaction
LAST_UPDATE_DATE	DATE	☑			Indicates the last updated date
OBJECT_VERSION_NUMBER	NUMBER(9)	☑			Indicates the version number. Defaults to 1 for new instances.
OBJECT_STATUS	VARCHAR2(5)	☑			Indicates current status of the entity.
TXT_SUBJECT_TMPL	CLOB	☑			Indicates message for subject Buffer
DOMAIN_OBJECT_EXTN	VARCHAR2(100)	☑			
CUST_INTEGER	NUMBER(9)	☑			
CUST_BOOLEAN	VARCHAR2(5)	☑			
CUST_DATE	DATE	☑			
CUST_STRING	VARCHAR2(100)	☑			
CUST_ENUM	VARCHAR2(100)	☑			

The following ORM file maps the customized class attributes with the table columns.

Figure 18–29 ORM File

```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings xmlns="http://www.eclipse.org/eclipselink/xsds/persistence/orm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <entity class="com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate"
          parent="com.ofss.fc.domain.ep.entity.dispatch.message.MessageTemplate">
    <discriminator-value>CUST</discriminator-value>
    <attributes>
      <basic attribute-type="int" name="attributeInteger">
        <column name="cust_integer"/>
      </basic>
      <basic attribute-type="java.lang.Boolean" name="attributeBoolean">
        <column name="cust_boolean"/>
        <convert>yesno</convert>
      </basic>
      <basic attribute-type="java.lang.String" name="attributeString">
        <column name="cust_string"/>
      </basic>
      <basic attribute-type="com.ofss.fc.datatype.Date" name="attributeDate">
        <column name="cust_date"/>
        <convert>Date</convert>
      </basic>
      <basic attribute-type="com.ofss.fc.enumeration.ep.DestinationType" name="attributeEnum">
        <column name="cust_enum"/>
        <enumerated>VALUE</enumerated>
      </basic>
    </attributes>
  </entity>
</entity-mappings>
```

The following JUnit test case has been used to test a "create" operation.

Figure 18–30 JUnit Test Case

```
@Test
public void testAddMessageTemplate() {
    String testCase = "testAdd.messageTemplateDTO.";
    MessageTemplateApplicationService applicationService = new MessageTemplateApplicationService();
    SessionContext sessionContext = getSessionContext();
    MessageTemplateDTO messageTemplateDTO = populateMessageTemplateDTO(testCase);
    try {
        deleteMessageTemplate(testCase);

        com.ofss.fc.framework.domain.common.dto.Dictionary[] dictionaryArray= new com.ofss.fc.framework.domain.common.dto.Dictionary[1];
        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.CustomizedMessageTemplate.AttributeInteger");
        nameValuePairDTO0.setValue("100");
        nameValuePairDTOArray[0]=nameValuePairDTO0;

        com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePairDTO1= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO();
        nameValuePairDTO1.setGenericName("com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate.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();
        nameValuePairDTO2.setGenericName("com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate.AttributeString");
        nameValuePairDTO2.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;

        com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePairDTO4= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO();
        nameValuePairDTO4.setGenericName("com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate.AttributeEnum");
        nameValuePairDTO4.setValue(com.ofss.fc.enumeration.ep.DestinationType.EMAIL.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("MessageTemplateApplicationService", "testAddMessageTemplate", e);
        fail("Unexpected failture from " + THIS_COMPONENT_NAME + ".testAddMessageTemplate");
    }
}
```

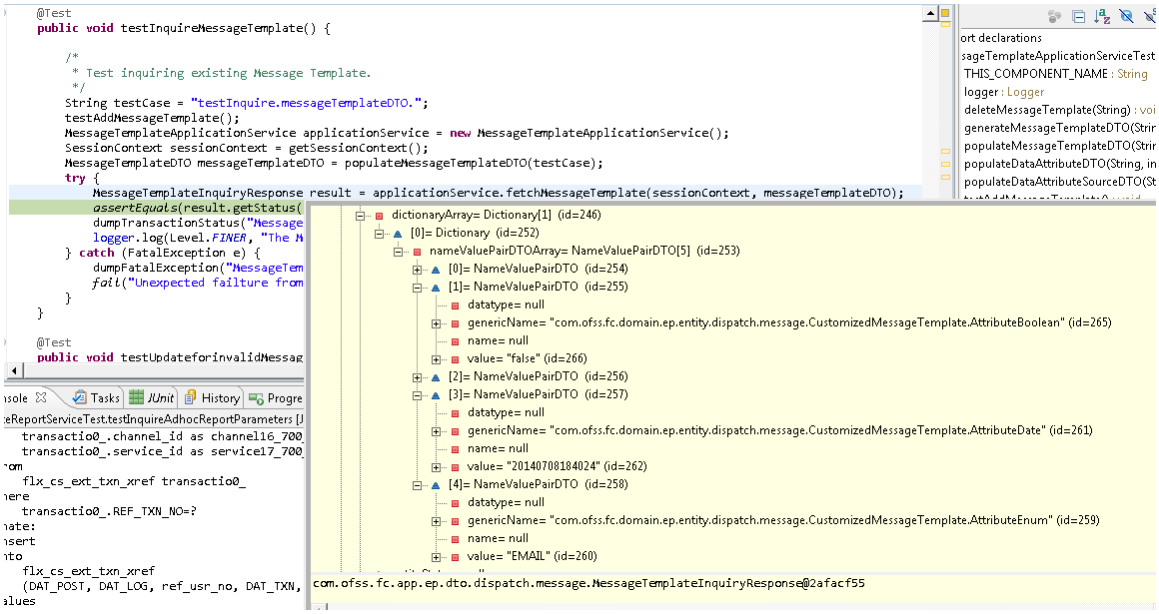
The above JUnit runs to add the following record in the table.

Figure 18–31 JUnit Adds Table Record

Row 1	Fields
COD_TMPL_ID	Junit_Message
DESTINATION_TYPE	
MSG_TMPL_NAME	Junit message template
MSG_TMPL_DESC	Message template description via junit test cas
TXT_MSG_TMPL	<CLOB>
CREATED_BY	ofssuser
CREATION_DATE	7/8/2014 6:40:34 PM
LAST_UPDATED_BY	ofssuser
LAST_UPDATE_DATE	7/8/2014 6:40:34 PM
OBJECT_VERSION_NUMBER	1
OBJECT_STATUS	A
TXT_SUBJECT_TMPL	<CLOB>
DOMAIN_OBJECT_EXTN	CUST
CUST_INTEGER	100
CUST_BOOLEAN	0
CUST_DATE	7/8/2014 6:40:24 PM
CUST_STRING	ABCDEFR
CUST_ENUM	EMAIL

Similarly, a JUnit is run to do "fetch" operation. This fetches the customized record whose dictionary array values have been shown below.

Figure 18–32 Dictionary Array Values



18.8 Customized Domain Object having Collection of Objects as Attributes

Figure 18–33 Customized Domain Object having collection of Objects as Attributes

```
package com.ofss.fc.dictionary;
import java.util.List;

public class CustomizedMessageTemplate extends MessageTemplate{

    private static final long serialVersionUID = 3762836902405427 91L;

    private int attributeInt;

    private boolean attributeBool;

    private char attributeChar;

    private Money attributeMoney;

    private Integer attributeInteger;

    private Boolean attributeBoolean;

    private String attributeString;

    private Date attributeDate;

    private DestinationType attributeEnum;

    private List<MessageAttribute> messageAttributeList;

    private List<MessageRecipient> messageRecipientList;

    public Integer getAttributeInteger() {
        return attributeInteger;
    }

    public List<MessageRecipient> getMessageRecipientList() {
        return messageRecipientList;
    }

    public void setMessageRecipientList(List<MessageRecipient> messageRecipientList) {
```

Figure 18–34 Member Attributes of Customized Domain Object

```java
package com.ofss.fc.dictionary;

public class MessageAttribute {

    private String messageTag;

    private String masking;

    public String getMessageTag() {
        return messageTag;
    }

    public void setMessageTag(String messageTag) {
        this.messageTag = messageTag;
    }

    public String getMasking() {
        return masking;
    }

    public void setMasking(String masking) {
        this.masking = masking;
    }


}
```

```java
package com.ofss.fc.dictionary;

public class MessageRecipient {

    private String name;

    private int age;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }


}
```

Figure 18–35 Dictionary Array Elements

```java
private void populateDictionaryData(MessageTemplateDTO messageTemplateDTO){
    Dictionary[] dictionaryArray= new Dictionary[4];
    Dictionary dictionaryObject = new Dictionary();
    dictionaryObject.setFullyQualifiedClassName("com.ofss.fc.dictionary.CustomizedMessageTemplate");
    NameValuePairDTO[] nameValuePairDTOArray= new NameValuePairDTO[6];
    nameValuePairDTOArray[0]=getNameValuePairDTO("com.ofss.fc.dictionary.CustomizedMessageTemplate.AttributeInteger","100");
    nameValuePairDTOArray[1]=getNameValuePairDTO("com.ofss.fc.dictionary.CustomizedMessageTemplate.AttributeBoolean","true");
    nameValuePairDTOArray[2]=getNameValuePairDTO("com.ofss.fc.dictionary.CustomizedMessageTemplate.AttributeString","ABCDEFR");
    nameValuePairDTOArray[3]=getNameValuePairDTO("com.ofss.fc.dictionary.CustomizedMessageTemplate.AttributeInt","101");
    nameValuePairDTOArray[4]=getNameValuePairDTO("com.ofss.fc.dictionary.CustomizedMessageTemplate.AttributeBool","true");
    nameValuePairDTOArray[5]=getNameValuePairDTO("com.ofss.fc.dictionary.CustomizedMessageTemplate.AttributeChar","C");
    dictionaryObject.setNameValuePairDTOArray(nameValuePairDTOArray);
    dictionaryArray[0]=dictionaryObject;
    Dictionary dictionaryObject1 = new Dictionary();
    dictionaryObject1.setFullyQualifiedClassName("com.ofss.fc.dictionary.CustomizedMessageTemplate.MessageAttributeList");
    NameValuePairDTO[] nameValuePairDTOArray1= new NameValuePairDTO[2];
    nameValuePairDTOArray1[0]=getNameValuePairDTO("com.ofss.fc.dictionary.MessageAttribute.MessageTag","100");
    nameValuePairDTOArray1[1]=getNameValuePairDTO("com.ofss.fc.dictionary.MessageAttribute.Masking","D");
    dictionaryObject1.setNameValuePairDTOArray(nameValuePairDTOArray1);
    dictionaryArray[1]=dictionaryObject1;
    Dictionary dictionaryObject2 = new Dictionary();
    dictionaryObject2.setFullyQualifiedClassName("com.ofss.fc.dictionary.CustomizedMessageTemplate.MessageAttributeList");
    NameValuePairDTO[] nameValuePairDTOArray2= new NameValuePairDTO[2];
    nameValuePairDTOArray2[0]=getNameValuePairDTO("com.ofss.fc.dictionary.MessageAttribute.MessageTag","111");
    nameValuePairDTOArray2[1]=getNameValuePairDTO("com.ofss.fc.dictionary.MessageAttribute.Masking","DD");
    dictionaryObject2.setNameValuePairDTOArray(nameValuePairDTOArray2);
    dictionaryArray[2]=dictionaryObject2;
    Dictionary dictionaryObject3 = new Dictionary();
    dictionaryObject3.setFullyQualifiedClassName("com.ofss.fc.dictionary.CustomizedMessageTemplate.MessageRecipientList");
    NameValuePairDTO[] nameValuePairDTOArray3= new NameValuePairDTO[2];
    nameValuePairDTOArray3[0]=getNameValuePairDTO("com.ofss.fc.dictionary.MessageRecipient.Name","Raju");
    nameValuePairDTOArray3[1]=getNameValuePairDTO("com.ofss.fc.dictionary.MessageRecipient.Age","35");
    dictionaryObject3.setNameValuePairDTOArray(nameValuePairDTOArray3);
    dictionaryArray[3]=dictionaryObject3;
    messageTemplateDTO.setDictionaryArray(dictionaryArray);
}
```

To construct a CustomizedMessageTemplate having 2 elements in messageAttributeList and 1 element in messageRecipientList, set the dictionaryArray of MessageTemplateDTO as follows:

The dictionaryArray has four elements as highlighted in the above figure.

- The 0th dictionaryArray element will have NameValuePairDTO array of non-collection attributes. This element's fullyQualifiedClassName will be the fully qualified class name of the customized domain object that is being constructed.

- The 1st dictionaryArray element will have NameValuePairDTO array of 1st element of 1st collection attribute. This element's fullyQualifiedClassName will be the fully qualified class name of the customized domain object that is being constructed, appended with "." and 1st collection attribute name.
- The 2nd dictionaryArray element will have NameValuePairDTO array of 2nd element of 1st collection attribute. This element's fullyQualifiedClassName will be the fully qualified class name of the customized domain object that is being constructed, appended with "." and 1st collection attribute name.
- The 3rd dictionaryArray element will have NameValuePairDTO array of 1st element of 2nd collection attribute. This element's fullyQualifiedClassName will be the fully qualified class name of the customized domain object that is being constructed, appended with "." and 2nd collection attribute name.

Figure 18–36 Customized Domain Object constructed by AbstractAssembler

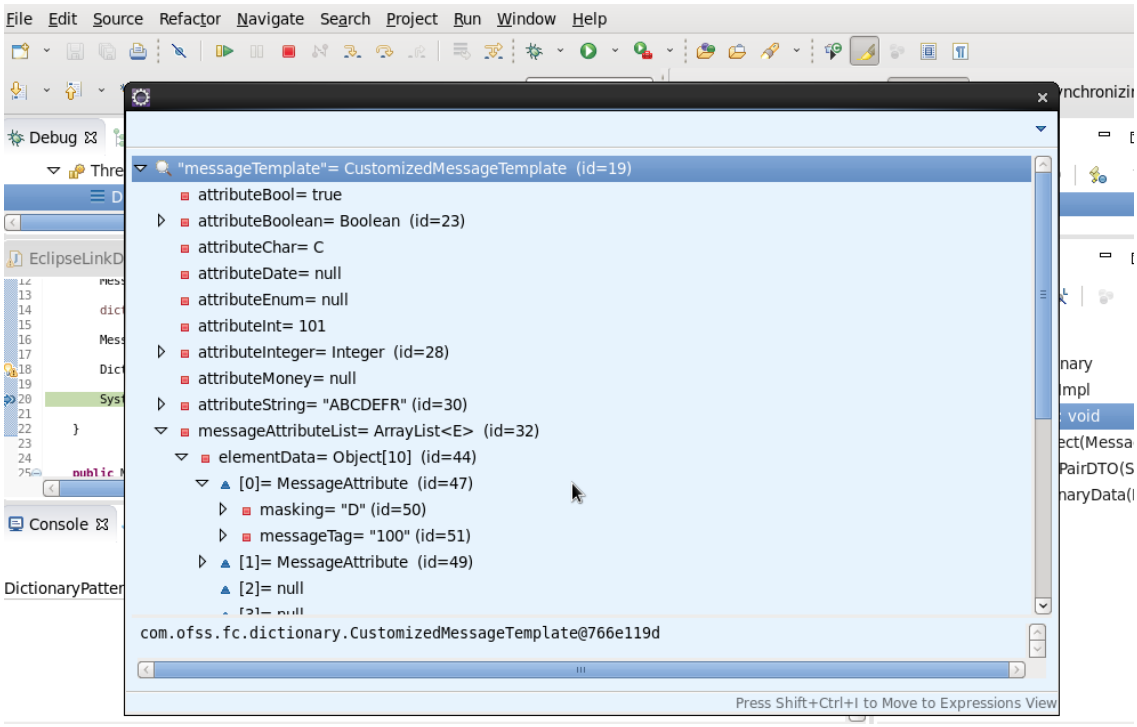
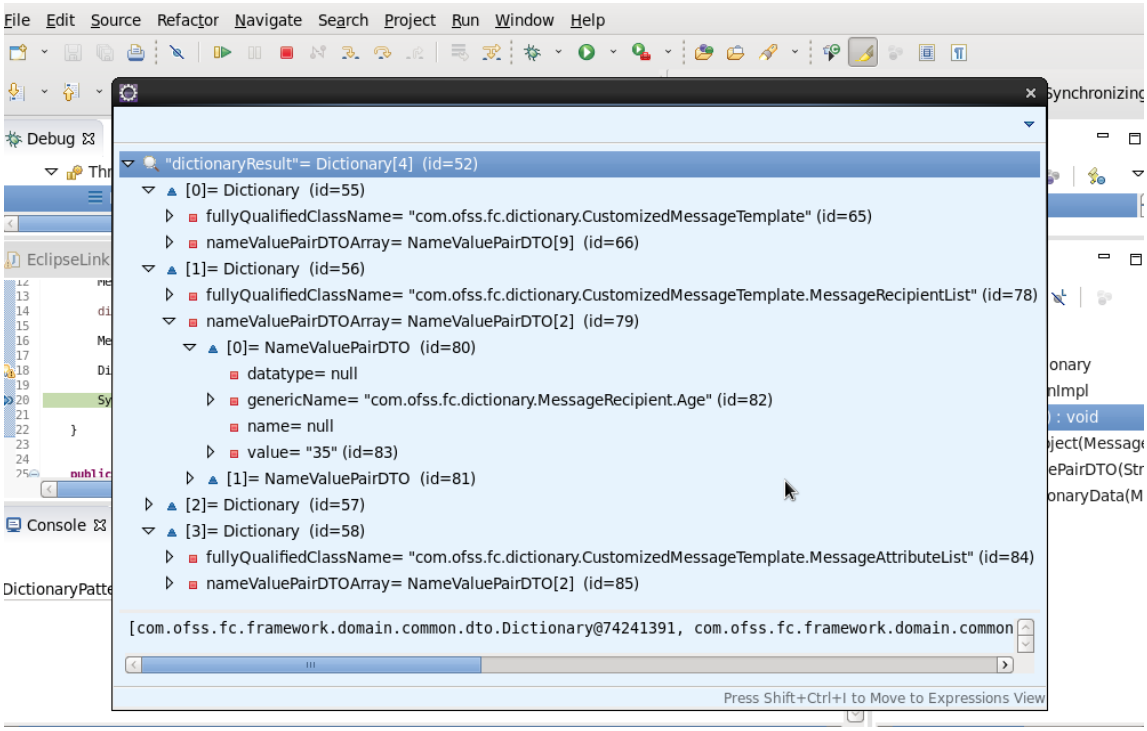


Figure 18–37 Dictionary Array returned by AbstractAssembler



18.9 Limitation to Extensibility using Dictionary Pattern

Extensibility of domain objects using Dictionary pattern is not applicable to those Maintenance Domain Objects that implement com.ofss.fc.framework.domain.search.ISearchableEntity.

The following is the list of the ISearchableEntity:

- com.ofss.fc.domain.config.entity.OBPConfigurationProperty
- com.ofss.fc.domain.origination.entity.core.submission.Submission
- com.ofss.fc.domain.party.entity.textsearch.PartyAggregateSummary
- com.ofss.fc.domain.account.entity.statement.impl.TDFinancialStatementItem
- com.ofss.fc.domain.account.entity.statement.impl.LoanFinancialStatementItem
- com.ofss.fc.domain.account.entity.statement.impl.DDFinancialStatementItem
- com.ofss.fc.domain.account.entity.statement.impl.DDNonFinancialStatementItem
- com.ofss.fc.domain.account.entity.statement.impl.LoanNonFinancialStatementItem
- com.ofss.fc.domain.account.entity.transactingparty.TransactingParty
- com.ofss.fc.domain.lcm.entity.collaterals.businessassets.AllPAPExcept
- com.ofss.fc.domain.lcm.entity.collaterals.businessassets.BusinessAssets
- com.ofss.fc.domain.lcm.entity.collaterals.businessassets.AllPAP
- com.ofss.fc.domain.lcm.entity.collaterals.fixedasset.computerhardware.ComputerHardware
- com.ofss.fc.domain.lcm.entity.collaterals.fixedasset.Machinery

- com.ofss.fc.domain.lcm.entity.collaterals.fixedasset.computersoftware.ComputerSoftware
- com.ofss.fc.domain.lcm.entity.collaterals.fixedasset.FixedAsset
- com.ofss.fc.domain.lcm.entity.collaterals.fixedasset.Furniture
- com.ofss.fc.domain.lcm.entity.collaterals.industrybusinessvalue.IndustryBusinessValue
- com.ofss.fc.domain.lcm.entity.collaterals.agriculture.Agriculture
- com.ofss.fc.domain.lcm.entity.collaterals.agriculture.crop.Crops
- com.ofss.fc.domain.lcm.entity.collaterals.agriculture.livestock.LiveStocks
- com.ofss.fc.domain.lcm.entity.collaterals.agreementandundertaking.NonFinancialAgreementAndUndertaking
- com.ofss.fc.domain.lcm.entity.collaterals.agreementandundertaking.AgreementAndUndertaking
- com.ofss.fc.domain.lcm.entity.collaterals.currentassets.inventorystock.InventoryStocks
- com.ofss.fc.domain.lcm.entity.collaterals.currentassets.CurrentAssets
- com.ofss.fc.domain.lcm.entity.collaterals.currentassets.bookdebt.BookDebts
- com.ofss.fc.domain.lcm.entity.collaterals.currentassets.receivable.Receivable
- com.ofss.fc.domain.lcm.entity.collaterals.automobile.PassengerVehicle
- com.ofss.fc.domain.lcm.entity.collaterals.automobile.Automobile
- com.ofss.fc.domain.lcm.entity.collaterals.automobile.GoodsVehicle
- com.ofss.fc.domain.lcm.entity.collaterals.investmentsecurities.InvestmentSecurities
- com.ofss.fc.domain.lcm.entity.collaterals.investmentsecurities.SharesStock
- com.ofss.fc.domain.lcm.entity.collaterals.investmentsecurities.InvestmentSecurity
- com.ofss.fc.domain.lcm.entity.collaterals.intangibleasset.IntangibleAsset
- com.ofss.fc.domain.lcm.entity.collaterals.other.OtherCollateral
- com.ofss.fc.domain.lcm.entity.collaterals.insurance.lifeinsurance.LifeInsurance
- com.ofss.fc.domain.lcm.entity.collaterals.insurance.Insurance
- com.ofss.fc.domain.lcm.entity.collaterals.bullion.Bullion
- com.ofss.fc.domain.lcm.entity.collaterals.cash.TermDeposit
- com.ofss.fc.domain.lcm.entity.collaterals.cash.CashDeposit
- com.ofss.fc.domain.lcm.entity.collaterals.Collateral
- com.ofss.fc.domain.lcm.entity.collaterals.proposedcollateral.ProposedCollateralRequest
- com.ofss.fc.domain.lcm.entity.collaterals.proposedcollateral.IPARequest
- com.ofss.fc.domain.lcm.entity.collaterals.proposedcollateral.SubDivisionRequest
- com.ofss.fc.domain.lcm.entity.collaterals.proposedcollateral.ConsolidationRequest
- com.ofss.fc.domain.lcm.entity.collaterals.guarantee.PersonalGuarantee
- com.ofss.fc.domain.lcm.entity.collaterals.guarantee.Guarantee
- com.ofss.fc.domain.lcm.entity.collaterals.guarantee.FamilyGuarantee

- com.ofss.fc.domain.lcm.entity.collaterals.guarantee.BankGuarantee
- com.ofss.fc.domain.lcm.entity.collaterals.guarantee.GuaranteeAndIndemnity
- com.ofss.fc.domain.lcm.entity.collaterals.guarantee.GovernmentGuarantee
- com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty
- com.ofss.fc.domain.lcm.entity.collaterals.realestate.WaterProperty
- com.ofss.fc.domain.lcm.entity.collaterals.realestate.CommercialProperty
- com.ofss.fc.domain.lcm.entity.collaterals.realestate.RealEstate
- com.ofss.fc.domain.lcm.entity.collaterals.realestate.ResidentialProperty
- com.ofss.fc.domain.lcm.entity.collaterals.realestate.RuralProperty
- com.ofss.fc.domain.lcm.entity.collaterals.artwork.ArtWork
- com.ofss.fc.domain.lcm.entity.collaterals.aircraft.smallaircraft.SmallAirCraft
- com.ofss.fc.domain.lcm.entity.collaterals.aircraft.cargoaircraft.CargoAirCraft
- com.ofss.fc.domain.lcm.entity.collaterals.aircraft.airframe.AirFrame
- com.ofss.fc.domain.lcm.entity.collaterals.aircraft.passengeraircraft.PassengerAirCraft
- com.ofss.fc.domain.lcm.entity.collaterals.aircraft.helicopter.HeliCopter
- com.ofss.fc.domain.lcm.entity.collaterals.aircraft.aircraftengine.AirCraftEngine
- com.ofss.fc.domain.lcm.entity.collaterals.aircraft.otheraircraft.OtherAirCraft
- com.ofss.fc.domain.lcm.entity.collaterals.aircraft.AirCraft
- com.ofss.fc.domain.lcm.entity.collaterals.ship.Ship
- com.ofss.fc.domain.lcm.entity.collaterals.license.WaterLicense
- com.ofss.fc.domain.lcm.entity.collaterals.license.License
- com.ofss.fc.domain.lcm.entity.collaterals.license.liquorlicense.LiquorLicense
- com.ofss.fc.domain.lcm.entity.collaterals.license.fishinglicense.FishingLicense
- com.ofss.fc.domain.lcm.entity.collaterals.license.managementrights.ManagementRights
- com.ofss.fc.domain.lcm.entity.collaterals.license.taxilicense.TaxiLicense
- com.ofss.fc.domain.pc.entity.institution.FinancialInstitution
- com.ofss.fc.framework.audit.AuditItem

19 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.

19.1 Integration Adapter Interface

OBP framework contains an interface, IIntegrationAdapter which provides two basic methods for OCH integration.

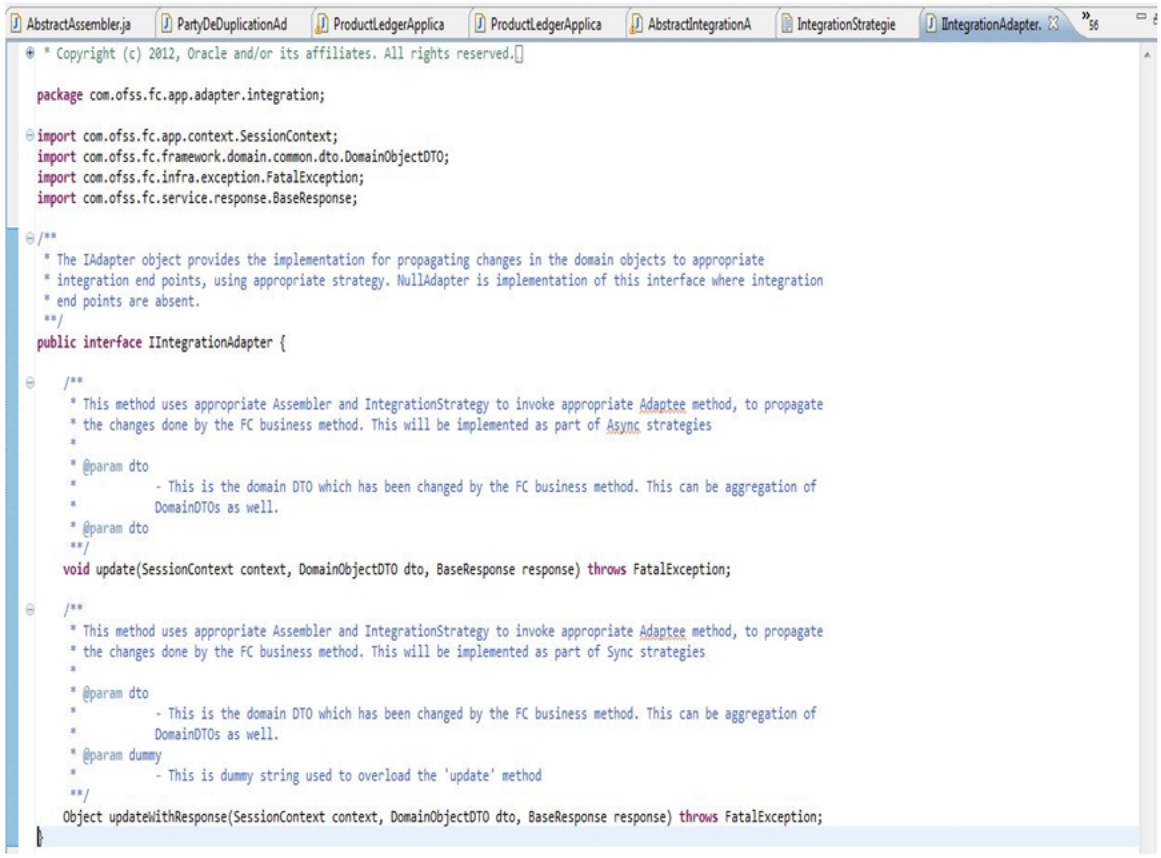
These two methods must be implemented by any adapter implementing the interface and use them for publishing data to OCH. Signature of these two methods are:

```
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 19–1 Integration Adapter Interface



19.2 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.

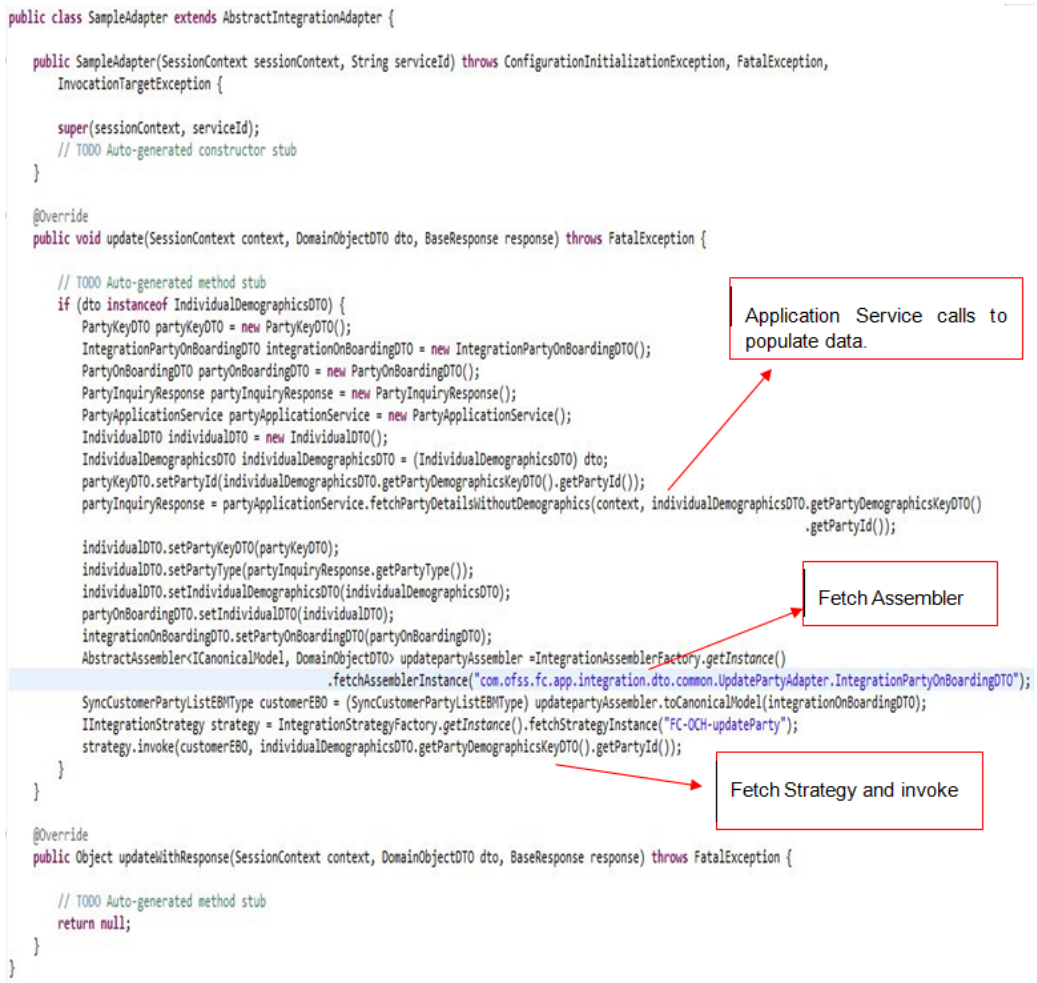
Figure 19–2 Abstract Integration Adapter Class

```
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;
    }
```

19.3 Sample Integration Adapter

The following figure is a sample adapter for customer information:

Figure 19–3 Sample Integration Adapter




```
public class SampleAdapter extends AbstractIntegrationAdapter {

    public SampleAdapter(SessionContext sessionContext, String serviceId) throws ConfigurationInitializationException, FatalException,
        InvocationTargetException {

        super(sessionContext, serviceId);
        // TODO Auto-generated constructor stub
    }

    @Override
    public void update(SessionContext context, DomainObjectDTO dto, BaseResponse response) throws FatalException {

        // TODO Auto-generated method stub
        if (dto instanceof IndividualDemographicsDTO) {
            PartyKeyDTO partyKeyDTO = new PartyKeyDTO();
            IntegrationPartyOnBoardingDTO integrationOnBoardingDTO = new IntegrationPartyOnBoardingDTO();
            PartyOnBoardingDTO partyOnBoardingDTO = new PartyOnBoardingDTO();
            PartyInquiryResponse partyInquiryResponse = new PartyInquiryResponse();
            PartyApplicationService partyApplicationService = new PartyApplicationService();
            IndividualDTO individualDTO = new IndividualDTO();
            IndividualDemographicsDTO individualDemographicsDTO = (IndividualDemographicsDTO) dto;
            partyKeyDTO.setPartyId(individualDemographicsDTO.getPartyDemographicsKeyDTO().getPartyId());
            partyInquiryResponse = partyApplicationService.fetchPartyDetailsWithoutDemographics(context, individualDemographicsDTO.getPartyDemographicsKeyDTO()
                                                                                                 .getPartyId());

            individualDTO.setPartyKeyDTO(partyKeyDTO);
            individualDTO.setPartyType(partyInquiryResponse.getPartyType());
            individualDTO.setIndividualDemographicsDTO(individualDemographicsDTO);
            partyOnBoardingDTO.setIndividualDTO(individualDTO);
            integrationOnBoardingDTO.setPartyOnBoardingDTO(partyOnBoardingDTO);
            AbstractAssembler<ICanonicalModel, DomainObjectDTO> updatepartyAssembler =IntegrationAssemblerFactory.getInstance()
                                                .fetchAssemblerInstance("com.ofss.fc.app.integration.dto.common.UpdatePartyAdapter.IntegrationPartyOnBoardingDTO");
            SyncCustomerPartyListEBMType customerEBO = (SyncCustomerPartyListEBMType) updatepartyAssembler.toCanonicalModel(integrationOnBoardingDTO);
            IIntegrationStrategy strategy = IntegrationStrategyFactory.getInstance().fetchStrategyInstance("FC-OCH-updateParty");
            strategy.invoke(customerEBO, individualDemographicsDTO.getPartyDemographicsKeyDTO().getPartyId());
        }
    }

    @Override
    public Object updateWithResponse(SessionContext context, DomainObjectDTO dto, BaseResponse response) throws FatalException {

        // TODO Auto-generated method stub
        return null;
    }
}
```

19.4 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 19–4 Integration Abstract Assembler

```
public abstract class AbstractAssembler<T extends ICanonicalModel, D extends DomainObjectDTO> {

    /**
     * This method needs to be implemented to convert from a DTO array to a canonical object.
     *
     * @param dto
     *            The input DTO which implements Serializable.
     * @return The canonical object instance.
     */
    public abstract T toCanonicalModel(D dto) throws FatalException;

    /**
     * This method needs to be implemented to convert from a canonical object to a DTO array.
     *
     * @param domainObject
     *            Instance of canonical model.
     * @return Instance of DTO
     */
    public abstract D fromCanonicalModel(T domainObject) throws FatalException;
}
```

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.

19.5 Sample Assembler

A sample assembler which extends AbstractAssembler should be like:

Figure 19–5 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.

20 Algorithm Extensions

This chapter explains the Algorithm Extensions for Oracle Banking Platform (OBP) Collections

20.1 Overview

Where the system requires a customization, OBP Collections provide for customizable algorithms. Algorithms provide a powerful and flexible way of extending applications. Base algorithms exist, but can be cloned and modified. Unlike Change Handlers, they are more related to the business functions and events. Also, unlike Change Handlers, they use configurable ("soft") parameters. At upgrades, custom algorithms will not be overwritten.

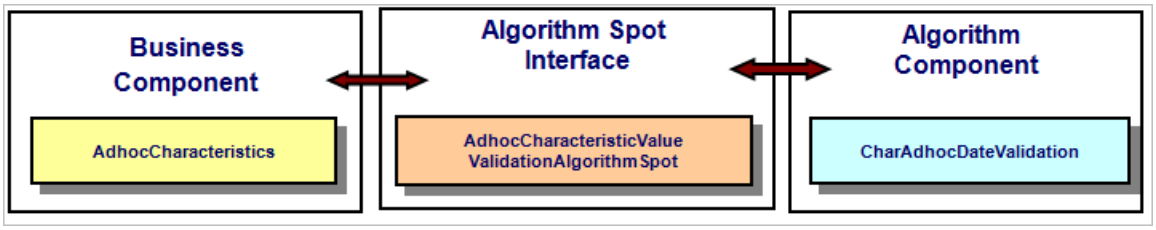
Algorithms are defined in 2 places:

- Database tables: The online Admin menu is used to define the following database components:
  - Algorithm Types
  - Algorithms
  - The event or activity to which the algorithm applies (For example, Characteristics, Date validations, and so on.)
- Framework: The framework requires the implementation class, that is the program that contains the logic, and various generated artifacts.

20.2 Algorithm Spots

The call out places in the system (For example, Date validation for ad hoc characteristics) are known as algorithm spots. Each algorithm spot has an interface class. Communication with an algorithm takes place through the interface. An interface provides abstraction between the base and the customization.

Figure 20–1 Algorithm Spot Interface

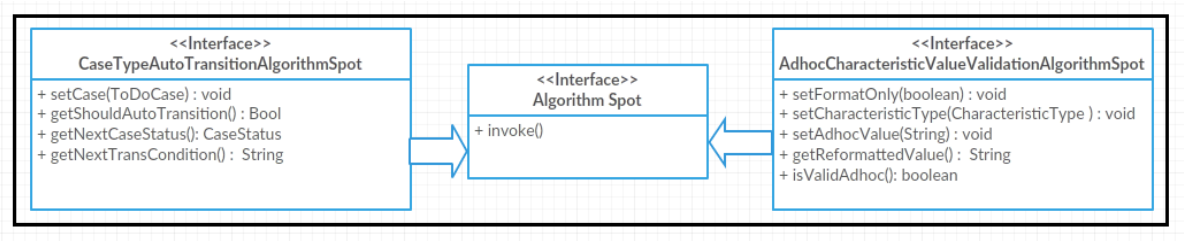


Attributes of an algorithm spot interface class:

- The API to the algorithm component (from the base application).
- It is specific to the algorithm entity type (or system event).
- It defines the hard input parameters for an algorithm. These are the parameters associated with a specific event.
- It defines the output parameters that can be retrieved after the algorithm has been invoked.

- It also specifies the schema defined for a plug-in script.
- It is invoked from the base code at appropriate times (events).
- The methods on the interface are related to the algorithm type. For example, setAdhocValue (String value) is only relevant to AdhocCharacteristicValueValidationAlgorithmSpot.

Figure 20–2 Example: Algorithm Spot Interface

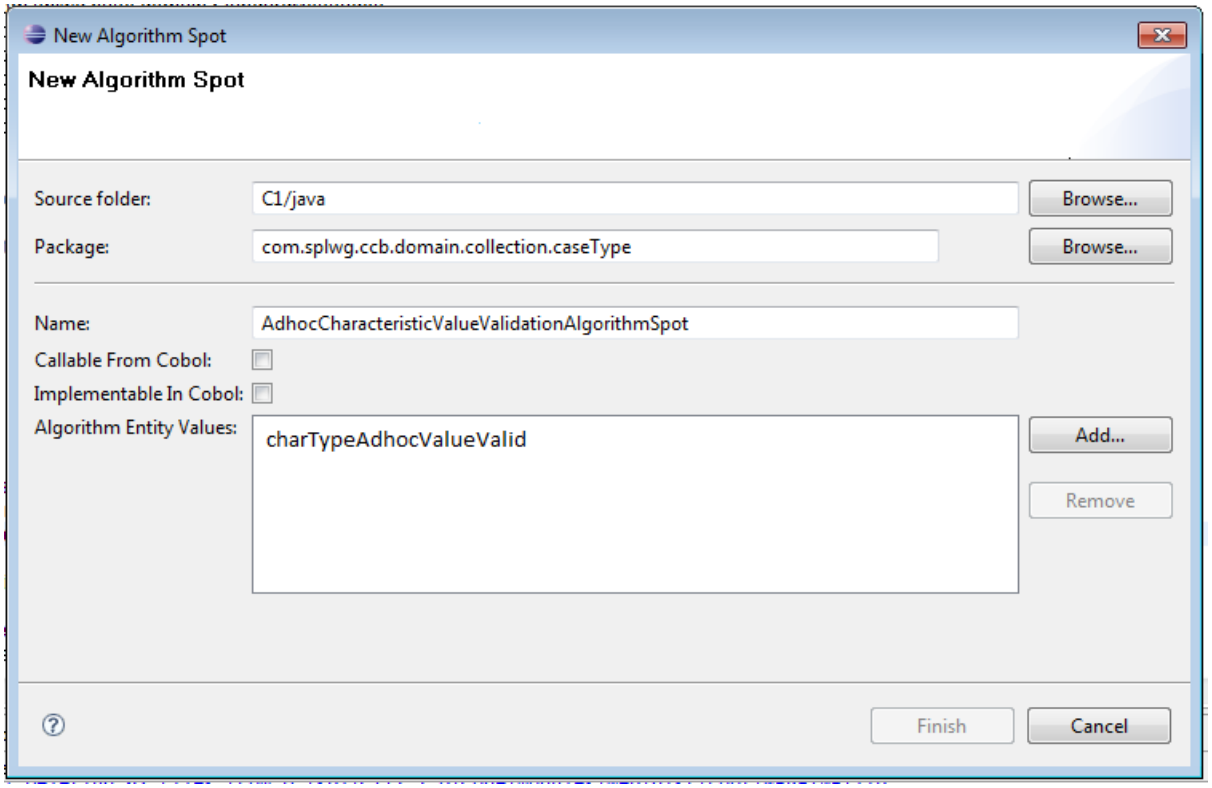


Adding Algorithm Spots:

- Algorithm Spots reference an AlgorithmEntityLookup (ALG_ENTITY_FLG) value, so a new lookup value must be added to correspond to the new spot.
- Add an interface that defines the spot using the @AlgorithmSpot annotation.
- Properties include:
  - algorithmEntity: One or more AlgorithmEntity values corresponding to the lookup value described above.
  - calledFromCobol: A boolean attribute that lets the framework know if inbound call support is to be supported from COBOL.
  - implementableInCobol: A boolean attribute that lets the framework know if it must be able to call an algorithm implemented in COBOL.
- Extend the AlgorithmSpot interface.
- Wire up the call to the spot by accessing the AlgorithmComponent via Algorithm.getAlgorithmComponent(…)

Example: See AdhocCharacteristicValueValidationAlgorithmSpot

Figure 20–3 Example: Adding New Algorithm Spot



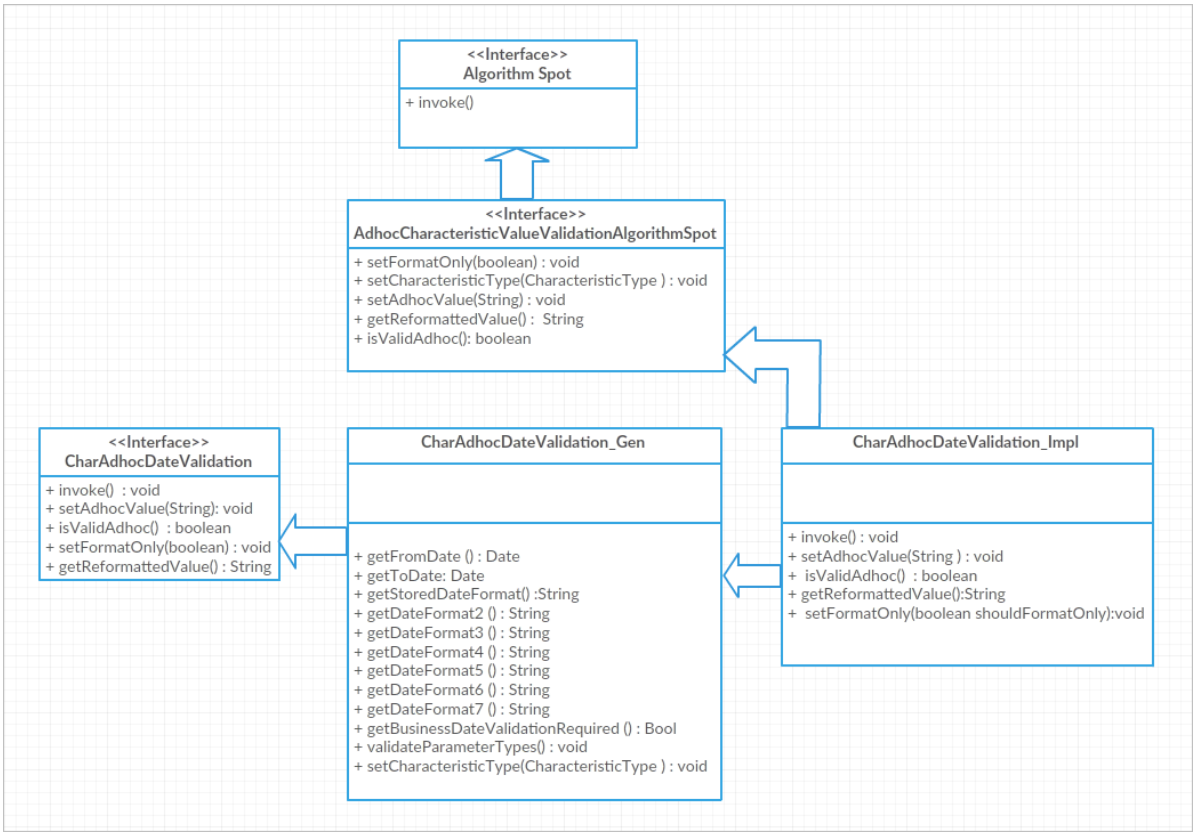
20.3 Algorithm Components

An algorithm requires a programmatic implementation. The Algorithm Type definition carries the program name, for examplecom.splwg.ccb.domain.collection.caseType. CharAdhocDateValidation. This name in fact specifies another interface which is generated from the implementation class. The implementation class name = the interface name + Impl, for example com.splwg.ccb.domain.collection.caseType.CharAdhocDateValidation_Impl.

The following diagram describes the Date Validation algorithm component. Remember:

- An interface is empty. It requires an implementation to perform appropriate tasks.
- The implementation for an algorithm spot is an Algorithm Component, that is Business Component.

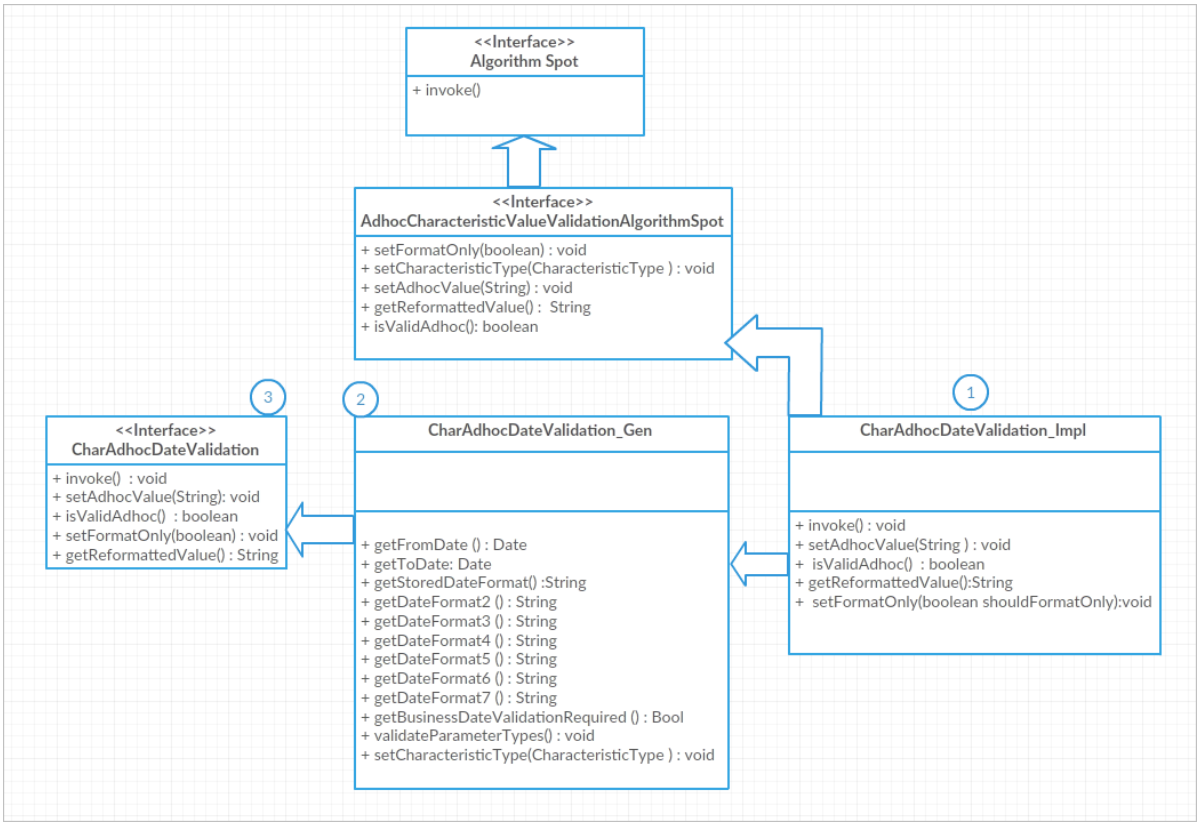
Figure 20–4 Example: Data Validation Algorithm Component



Algorithm Components Example - CharAdhocDateValidation

The following diagram presents an example of the CharAdhocDateValidation algorithm component.

Figure 20–5 Example: CharAdhocDateValidation



The annotations marked in the above diagram are explained as follows:

- The implementation class (CharAdhocDateValidation_Impl) is hand-coded - it can be customized.
- The component interface is generated by the artifact generator - a customized version will be generated for a custom impl class. The _Gen class (CharAdhocDateValidation_Gen) has the methods for the soft parameters (as specified on the Algorithm Type definition). Note: These are generated from the annotations in the "_Impl" class.
- An algorithm is invoked via its component interface (CharAdhocDateValidation).

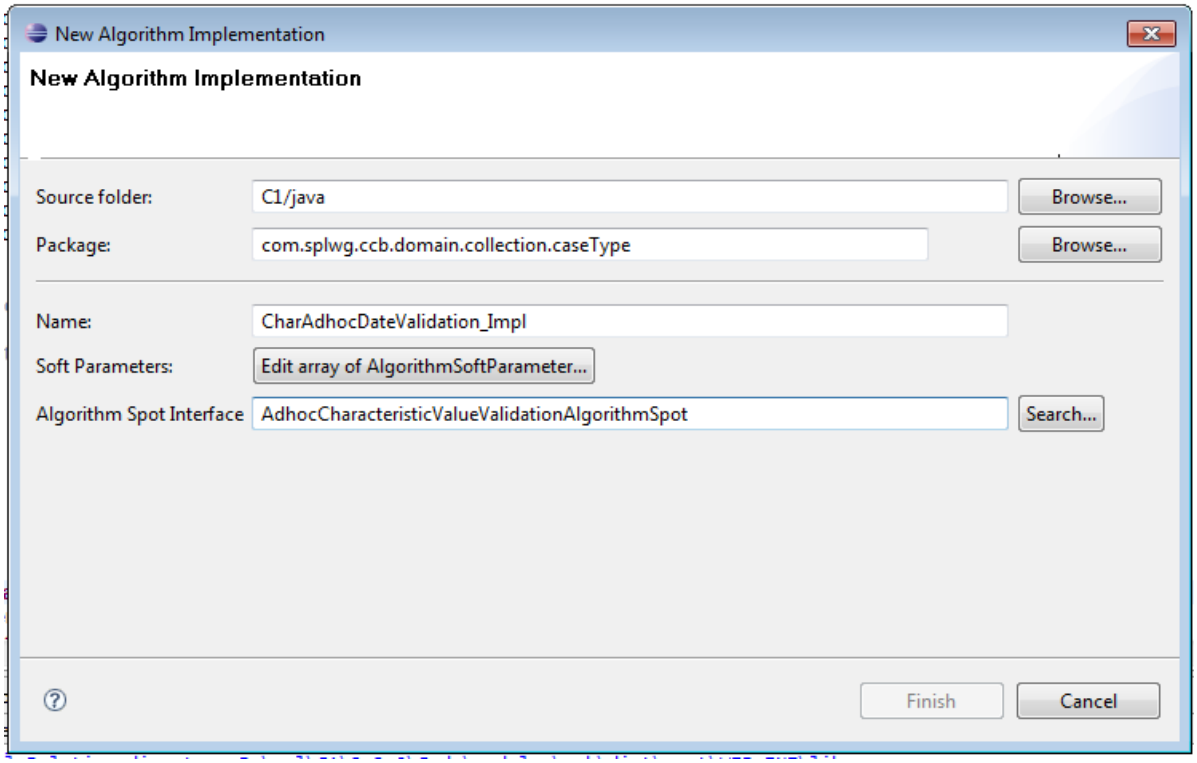
Algorithm Implementation Class:

The base versions of all algorithms are provided. To create a new one, it is easiest to duplicate the appropriate base one if it exists and modify it.

The basic Java elements of a new algorithm are:

- An "_Impl" class, the hand-coded implementation class that contains the logic
- A "_Gen" class, the implementation class for the "soft" parameters, generated by the artifact generator
- A component interface class, generated by the AG
- A message method, if required

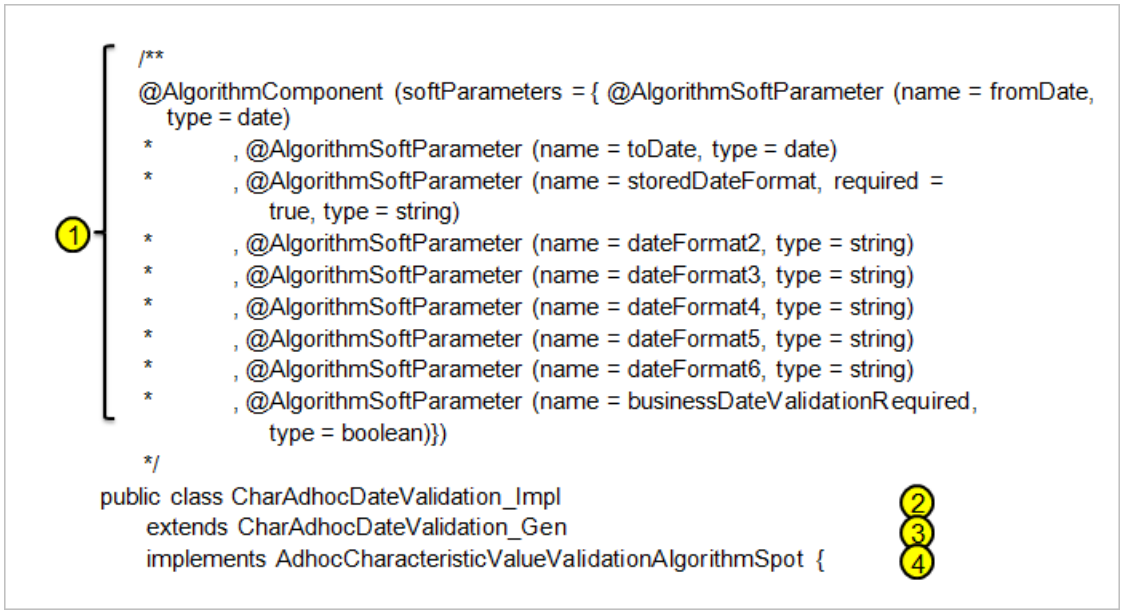
Figure 20–6 New Algorithm Implementation



Adhoc Characteristic Date Validation Example:

The following diagram presents an example of Adhoc Characteristic Date Validation.

Figure 20–7 Adhoc Characteristic Date Validation



The annotations marked in the above diagram are explained as follows:

1. The soft parameters expected by the algorithm - these correspond with the Algorithm Type parameter definitions
2. Has an Algorithm Component name (as specified on the Algorithm Type) + "_Impl"
3. Extends the "_Gen" class - the "_Gen" class is generated by the Artifact Generator
4. Implements the base Algorithm Spot class for the algorithm type

Algorithm Spot interface methods that are implemented in _Impl class:

The following diagrams present the Algorithm Spot interface methods that are implemented in _Impl class.

Figure 20–8 Algorithm Spot Interface Methods

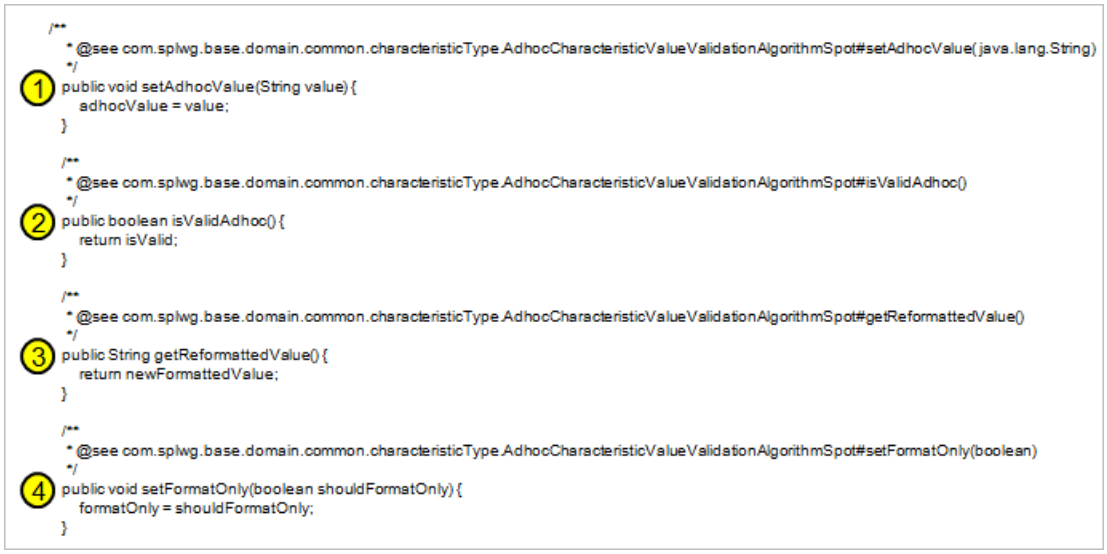


Figure 20–9 Algorithm Spot Interface Methods (continued)

```
/**
 * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#invoke()
 */
public void invoke() {
    logger.debug("Executing adhoc date validation");
    if (adhocValue == null || adhocValue.trim().length() == 0) {
        isValid = true;
        newFormattedValue = "";
        return;
    }
    initializeFormats();
    DateTime date = parseDateFromAnyFormat();
    if (date == null) {
        isValid = false;

        for (Iterator iter = validFormats.iterator(); iter.hasNext();) {
            DateFormat format = (DateFormat) iter.next();
            if (format != null) {
                notNullValidFormats.add(format);
            }
        }
        ServerMessage message = CustomMessageRepository.invalidFormatForDate(adhocValue, notNullValidFormats,
                notNullValidFormats.size());
        addError(message);

        return;
    }
    isValid = true;

    if (!formatOnly && adhocValue.length() <= Date.TO_STRING_SIZE) {
        validateDateInRange(date.getDate());
    }

    if (getBusinessDateValidationRequired().isTrue()) {
        if (!DateUtility.isWorkingDay(date.getDate()))
            addError(CustomMessageRepository.businessDateValidationFailed(date.getDate().toString()));
    }

    newFormattedValue = storedFormat.format(date);
}
```

The annotations marked in the above diagrams are explained as follows:

1. This method is invoked by the business component to set the hard parameters. This sets the char value to validate. It is stored here for use later.
2. This returns a true/false to indicate the validity of the date characteristics.
3. This returns the reformatted value.
4. This method set the required format.
5. The invoke () method is called to validate and format the date.

Generated artifacts that are based on the _Impl class annotation:

- The _Gen class has the methods for the soft parameters
- The _Impl class calls these methods to get the soft parameter values, as set on the Algorithm definition

Figure 20–10 Generated Artifacts

```
/* Generated by com.splwg.tools.artifactgen.ArtifactGenerator */
package com.splwg.ccb.domain.collection.caseType;

import com.splwg.base.api.BusinessComponent;

/**
 * Interface for the charAdhocDateValidation component
 *
 * @author Generated by com.splwg.tools.artifactgen.ArtifactGenerator
 */
public interface CharAdhocDateValidation extends BusinessComponent
        , AdhocCharacteristicValueValidationAlgorithmSpot
        {


    /**
     * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#invoke()
     */
    void invoke();

    /**
     * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#setAdhocValue(java.lang.String)
     */
    void setAdhocValue(java.lang.String value);

    /**
     * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#isValidAdhoc()
     */
    boolean isValidAdhoc();

    /**
     * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#getReformattedValue()
     */
    java.lang.String getReformattedValue();

    /**
     *
     * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#setFormatOnly(boolean)
     */
    void setFormatOnly(boolean shouldFormatOnly);

}
```

- The component interface defines the required methods for the _Impl class as viewed from the application (the business component).

Figure 20–11 Generated Artifacts

```
/* Generated by com.splwg.tools.artifactgen.ArtifactGenerator */
package com.splwg.ccb.domain.collection.caseType;

import com.splwg.base.api.BusinessComponent;

/**
 * Interface for the charAdhocDateValidation component
 *
 * @author Generated by com.splwg.tools.artifactgen.ArtifactGenerator
 */
public interface CharAdhocDateValidation extends BusinessComponent
        , AdhocCharacteristicValueValidationAlgorithmSpot
        {


    * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#invoke()

    void invoke() ;

    * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#setAdhocValue(java.lang.String)

    void setAdhocValue(java.lang.String value) ;

    * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#isValidAdhoc()

    boolean isValidAdhoc() ;

    * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#getReformattedValue()

    java.lang.String getReformattedValue() ;

   /**
    *
    * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#setFormatOnly(boolean)

    */

    void setFormatOnly(boolean shouldFormatOnly) ;

}
```

The steps to create a new algorithm Impl class are:

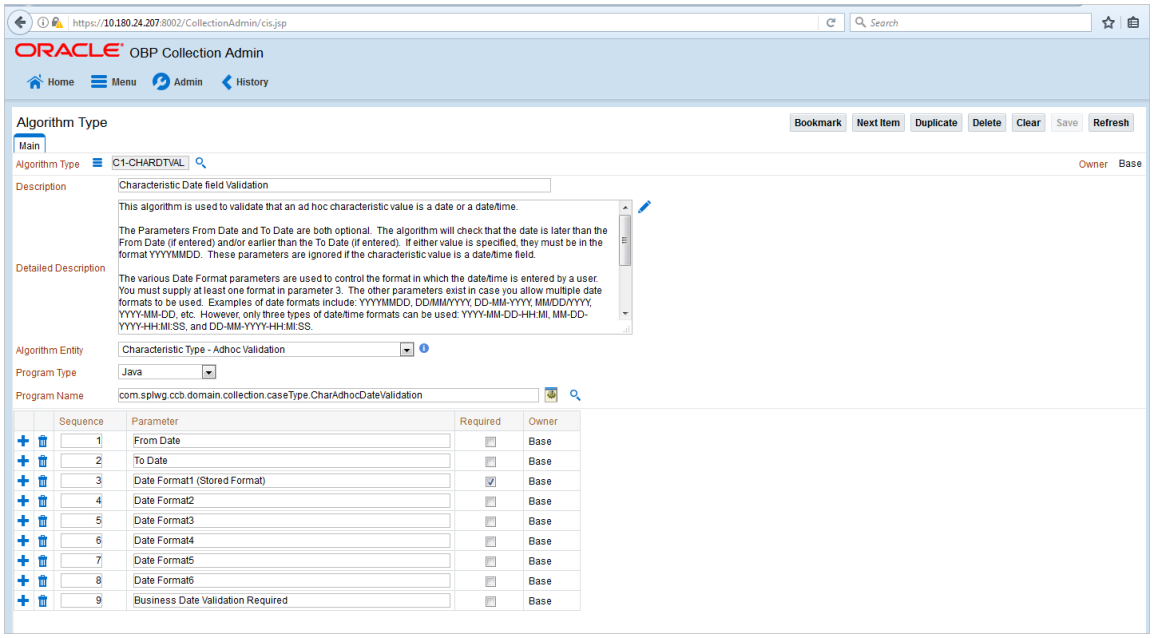
1. Determine the Algorithm Spot interface name. The Javadocs can be used for this.
2. Create the "_Impl" class, implementing the appropriate Algorithm Spot interface.
3. Add default implementations for all the Algorithm Spot methods (For example, using the Eclipse *Source, Override/implement Methods*… menu item).
4. Code the annotation.
5. Run the Artifact Generator to create the "_Gen" and component interface classes.

In Eclipse, you must refresh the project after this.

The steps to create a new algorithm (Admin UI) are:

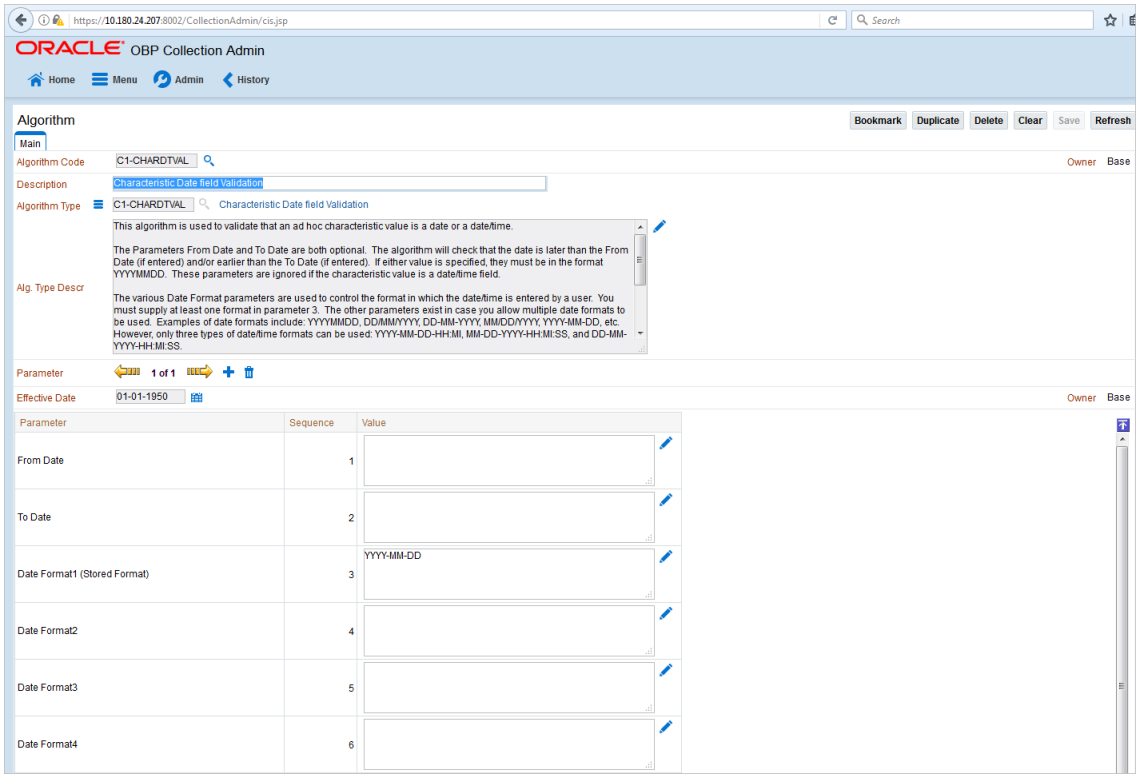
1. Create the Algorithm Type and attached with algorithm component.

Figure 20–12 Create Algorithm Type



2. Attach the algorithm to the Algorithm Type and test.

Figure 20–13 Attach Algorithm



20.4 List of Algorithm Spots

The detailed list of algorithm spots which can be used for extending and customizing the product are listed in the following table.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
AdhocCharacteristicValueValidationAlgorithmSpot	This	void setFormatOnly (boolean formatOnly); void setChar-	com.splwg.cb.domain.collection.caseType.CharAdhocDateValidation	com.splwg.cb.domain.collection.caseType.CharAdhocDateValidation_Impl	Characteristic Date field	This algorithm is used to validate that an ad hoc characteristic value is a date or

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	algorithmspot	acteristicType (CharacteristicType type); void setAd-			Validation:C-1-CHARDT-VAL	a date/time. The Parameters From Date and To Date are both optional. The algorithm will check that the date is later than

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	oked onchara-	tring value); String getReformattedValue (); boolean				the From Date (if entered) and/or earlier than the To Date (if entered). If either value is specified, they must be in the format YYYYMMDD.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	order to: 1) validate	isValidAdhoc();				The various Date Format parameters are used to control the format in which the date/-time is entered by a user. You must supply at

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	that the value is cor-					least one format in parameter 3. The other parameters exist in case you allow multiple date formats to be used. Examples of date formats include: YYYYMMDD,

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	2) possibly perfor-					MM-DD-YYYY-HH:MI:SS, and DD-MM-YYYY-HH:MI:SS. Regardless of the format entered by the user, the date is stored in the format defined by parameter 3.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	m a re- fo- r- m- at of th- e v- al- u-					We strongly recommend this parameter be set to YYYY-MM-DD for dates and YYYY-MM-DD-HH:MI:SS for date/time fields as this is how all dates are stored in our system. Parameter 9 :

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	nthetable					valid values are true/false. When Business date validation required is true, algorithm will validate the given date to check if its a valid business date.
CureEntityAlgorithmSpot	This	void setAccountId	com.splwg.ccb.-	com.splwg.ccb.-	Cure Ac-	This algorithm

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	salgorithmspotisus-	(Account_Id acctId);	domain.collection.batch.algorithm.CureEntityAlgorithm	domain.collection.batch.algorithm.CureEntityAlgorithm_Impl	count: C1-FINCOLL	performs following activities: - Invoke OBP service to set the incollection flag in host as "N". - Mark incollection flag as "N" in collections. - Set end date in

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to cure the account.					CI_PARTY_ COLLECT as posting date. - Update number of times account is self cured (used for statistics). - Remove strategy review date. Parameter:

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						contactMethods : This soft parameter accept the comma separated value of customer contact method which is used to calculate the no. of times account is self cured
SaTypeSaStopAl-	T-	void	com.s-	com.s-	Stop	This algorithm

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
gorithmSpot	his algorithms pot is u-	setServiceAgreement (ServiceAgreement serviceAgreement);	plwg.ccb.domain.collection.batch.algorithm.FinalizeCollectionContractStopAlgoComp	plwg.ccb.domain.collection.batch.algorithm.FinalizeCollectionContractStopAlgoComp_Impl	Contract: C1-CURENTITY	will stop the contract for the account which is to be cured.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ed to stop the contarct.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
AllocationGroupQueueAlgorithmSpot	This algorithm spot	void setAllocationGroup (String allocationGroup); void setBusinessDat-	Com.splwg.cb.domain.collection.batch.algorithm.AllocationGroupQueueAlgoComp	com.splwg.cb.domain.collection.batch.algorithm.AllocationGroupQueueAlgoComp_Impl	Queue Allocation: C1-ALLOCQUEU	This Algorithm type is used to allocate the entities such as cases to queues. ci_allocation_monitor_vw view is shipped from product to filter cases.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is used to allocate the	e(Date businessDate); void seToQueueBean (AllocationGroupCasesToQueu-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		eBean case-AllocTo-Queue); Alloc-ationGro-upCases-ToQueu-eBean getCase-ToQueu-eBean(); Alloc-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ationGroupCasesToQueueBean getCaseAllocationMap();				
CaseTypeEnterStatusAlgorithmSpot	T-	void setCase (ToDoCase	com.splwg.ccb.domain.col-	com.splwg.ccb.domain.col-	Update Cu-	Update customer level case status on case enter processing.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	he purpose of the al-	toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShoul-	lection.batch.algorithm.CustomerLevelSwitchUpdateAlgorithm	lection.batch.algorithm.CustomerLevelSwitchUpdateAlgorithm_Impl	stomer Switch: C1-CUST-SW	Customer Level Switch Name: Please enter the customer level case status switch which needs to update. eg. BANKRUPT_SW, HARDSHIP_SW,

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	rithms-pot is to exec-	dAutoTransition() String getNextCaseStatus() String getNextTransCondition()				IMPRISONED_SW, DECEASED_SW, ABSCONDING_SW etc Switch Value: Please enter the switch value as Y or N

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	hen Case is moved in-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.batch.algorithm.RepoAndLegalCaseUpdateAlgorithm	com.splwg.ccb.domain.collection.batch.algorithm.RepoAndLegalCaseUpdateAlgorithm_Impl	Update Legal/Repo Switch: C1-LE-	Algorithm Type to update Legal and Repo case status on enter process Legal Repo Switch Name: Please enter the Legal or Repo case switch

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ose of th-e alg-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()			RE-PO-CT	column name of account extension eg. LEGAL_ CASE_EXISTS_ SW or REPO_ CASE_EXISTS_ SW etc Switch Value: Please enter the switch value as Y or N

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
UserAllocationAlgorithmSpot	This spot being	void setUserToQueueBean (UserAllocationToQueueBean userAllocTo-	com.splwg.cb.domain.collection.batch.algorithm.UserAllocationRoundRobinAlgorithm	com.splwg.cb.domain.collection.batch.algorithm.UserAllocationRoundRobinAlgorithm_Impl	User Allocation Round Robin:	User Allocation Round Robin algorithm type allocates cases to users on the basis of capacity set during configuration on queue admin. OverFlow cases will get assigned to Exception

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	used for allocation of	Queue); UserAllocationToQueueBean getUserToQueueBean(); UserAllocationTo-			C1-USRA-LC-RR	User.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	usalgorithms.	QueueBean getUserAllocationMap();				
UserAllocationAlgorithmSpot	This	void setUserToQueueBean	com.splwg.ccb.domain.col-	com.splwg.ccb.domain.col-	User Allocation	User Allocation Percentage based algorithm type allocates

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spot being used for	(UserAllocationToQueueBean userAllocToQueue); UserAllocationToQueueB-	lection.batch.algorithm.UserAllocationPercentageBaseAlgorithm	lection.batch.algorithm.UserAllocationPercentageBaseAlgorithm_Impl	tion - % Based: C1-US-RA-LC-PR	cases to users on the basis of percentage allocations set during configuration on queue admin. OverFlow cases will get assigned to Exception User.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ocation of user using	ean getUserToQueueBean(); UserAllocationToQueueBean getUserAllocationMap();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
VendorAllocationAlgorithmSpot	This spot being	void setVendorToQueueBean (VendorAllocationToQueueBean	com.splwg.ccb.domain.collection.batch.algorithm.VendorAllocationRoundRobinAlgorithm	com.splwg.ccb.domain.collection.batch.algorithm.VendorAllocationRoundRobinAlgorithm_Impl	Vendor Allocation - Round Robin:	This algorithm will allocate cases to vendors in round robin fashion. This algorithm is invoked from the User Allocation batch (C1-USALC). OverFlow cases will get assigned

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	used for allocation	vendorAllocToQueue); VendorAllocationToQueueBean getVendorToQueueBean();			C1-VENALC-RR	to Exception User of the queue.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	various algorithms.	VendorAllocationToQueueBean getVendorAllocationMap();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
VendorAllocationAlgorithmSpot	This spot being us-	void setVendorToQueueBean (VendorAllocationToQueueBean vendorAl-	com.splwg.cb.domain.collection.batch.algorithm.VendorAllocationPercentageBaseAlgorithm	com.splwg.cb.domain.collection.batch.algorithm.VendorAllocationPercentageBaseAlgorithm_Impl	VendorAllocation - %Based: C1-VENA-	This algorithm will allocate cases to vendors in percentage base. This algorithm is invoked from the User Allocation batch (C1-USALC). OverFlow cases will get assigned to Exception

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ed for allocation of vehicle	locToQueue); VendorAllocationToQueueBean getVendorToQueueBean(); Ven-			LCPR	User of the queue.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	algorithms.	dorAllocationToQueueBean getVendorAllocationMap();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
BulkContactCreationAlgorithmSpot	This algorithm spot	void setAccountId (String accountId); void setContactClass (String con-	com.splwg.cb.domain.collection.batch.algorithm.BulkContactCreationAlgoComp	com.splwg.cb.domain.collection.batch.algorithm.BulkContactCreationAlgoComp_Impl	Bulk Contact Creation: C1-BLKCNTCR-	This algorithm type is called from Bulk Contact Creation Batch. It invokes business service 'C1-GenMultipleCorrespondence' which creates a customer contact for the accounts

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is used for creation of	tactClass); void setContactTypeCode (String contactTypeCode); void			E	filtered by the condition builder attached to the process codes in bulk contact admin.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	for accounts in bulk.	setMode(String mode); void setCharacteristicType(String characteristicType);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setCharacteristicValue (String characteristicValue); void setJoint-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		NominationForAcc(String jointNominationForAcc); void setContactDateTime (String				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		contactDateTime); void setGeneratedBy(String generatedBy);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CrossStrategyActionMatrixAlgorithmSpot	This algorithm	void setCase(ToDoCase toDoCase); void setCaseOriginalStatus(CaseStatus	com.splwg.cb.domain.collection.batch.algorithm.CrossStrategyActionMatrixAlgorithm	com.splwg.cb.domain.collection.batch.algorithm.CrossStrategyActionMatrixAlgorithm_Impl	Cross Strategy Action Matrix: C1-CSAM	Cross Strategy Action Matrix Algorithm Type is used by Strategy Monitor and case association process in order to take actions on existing strategies and recommended strategies based

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spot is used to e-	case-Status); String getNext-tCaseSt-atus();				on CSAM Matrix. Parameters : Check Status- It checks the status with which the matrix has to be dealt with. Possible values are "Y" or "N"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	nessl ogic when a					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	changes related to th-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CustomerClassFtFreezeAlgorithmSpot	The purpose	void setFinancialTransaction (FinancialTransaction fin-	com.splwg.ccb.domain.collection.batch.algorithm.LastPaymentDtAmtUpdateAlgorithm	com.splwg.ccb.domain.collection.batch.algorithm.LastPaymentDtAmtUpdateAlgorithm_Impl	Last Payment for Account: C1-PAYD-	This algorithm is used to stamp the last payment date and last payment amount for written off accounts.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithms-	ancialTransaction); void setFinancialTransactionType (FinancialTrans-			TAMTU	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	llalgorithmdef-	actionTypeLookup financialTransactionType); void setRegularFinancialTrans-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ezeSystemEventona	action (FinancialTransaction regularFinancialTransaction); Bool getFinancialTr-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ansactionProcessAdded();				
CaseTypeEnterStatusAlgorithmSpot	Th-	void setCase(ToDoCase toDoCase) void setCase-	com.splwg.cb.domain.collection.caseType.special-	com.splwg.cb.domain.collection.caseType.special-	Association Review Chec-	This is to decide if the system association of entities should be reviewed by the user or not.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e p- ur- p- o- se of th-	Ori- gin- alStatus (CaseSt- atus case- Status) Bool getShoul- dAutoTr- ansition () String	isedCol- lec- tions.legal.CheckAssociationReview	isedCol- lec- tion- s.legal.CheckAssociationReview_ Impl	k: C1- AS- OR- VC- HK	Soft Parameters: Next Status: Val- ues Possible for Next Status {ASSNEWLSP}. This is applic- able if Asso- ciation Review Required is set N. Association Review

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lgorithms spot is	getNextCaseStatus() String getNextTransCondition()				Required= Possible Values{Y,N} If Association Review is Required {Y}- Stay in current status for user review. Set display date to current business date. If association Review Not

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	sslogicwheneCa-					Required{N} - Transition to specified next status.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.DefaultNoticeExpiryCheck	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Validate Expired Default Notice: C1-DE-	System should check that for associated accounts default notice has expired, This check can be for primary account or for all associated delinquent account

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pose of the algo-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()		s.legal.DefaultNoticeExpiryCheck_Impl	FNOEXP	based on parameter. 1. Association Type={P,A}.P-P=Primary Type Association,A=Primary as well as Secondary type association 2. To Do Type=To Do will be created if validation failure option is

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				N. 3. To Do Role= To Do Role for the specified To Do Type. 4.Val-idationfailure Option= {Y,N}. If it is Y then case transition will be failed elsen a To Do will be cre-ated.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into sp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	a-m-pl-e al-g-or-it-h-m cr-e-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.CheckLegalCase	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.CheckLegalCase_Impl	Associate Related Entity: C1-ASSO-	The algorithm checks the associated accounts of the primary account.The association of the primary account is done on the basis of the persons

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()			EN-TY	attached to the account and their financially responsible status.if the account has the same set of fin-ancially repons-ible persons attached as in the case for the primary account,

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				1)To Do Role:Specifies the role for the To Do Type cre-ated in case of any exception arrising in asso-ciation of accounts. 2)To Do Type:Specifies

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					the To Do Type created in case of any exception arrising in association of accounts. 3)Host Id:Specifies the host Id.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lections.legal.CheckLegalCase	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lections.legal.CheckLegalCase_Impl	Val-idat-e Le-gal Ca-se Exi-sts: C1-CH-KL-	The Algorithm checks if there is already open legal case for the primary accoun-t/Associated accounts linked to the case.The algorithm takes the parameters as follows:

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()			GL	1)To Do Role:Specifies the Role for the To Do Type. 2)To Do Type:Specifies the todo type cre-ated when the legal case has been created from batch mode and there is

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				open legal case for the Primary Accoun-t/Associated Accounts. 3)Case Cat-egory:Specifies the case cat-egory for the case(LEGL is for Legal Case)

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into sp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.AssignNewLSP	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.AssignNewLSP_Impl	Assign New LSP: C1-ASGN-LS-	This algorithm will assign a new LSP to the current case. LSP is a external vendor which is mapped LEGAL

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()			P	service Type. If manual review is not required then case will auto-matically trans-ition to next status met-ntioned in soft-parameter.Below

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pot is to execut-	String getNextTransCondition ()				1. Next Status: value can be possible next status example {PREPLGLDOC etc.} 2. Prv Allocation

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ismovedintosp-					Check: Possible values {Y, N}. If this switch is Y system will check if a legal case was created for any of the accounts associated with the current legal

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					3 .Reset Doc Sub Date Sw= Possible values {Y, N}. Value N means document submission date from previous

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					assignment will be copied to new assignment. 4. Change Allocation Option= Possible values {AUTO_WITH_REVIEW,AUTO_WITHOUT_REVIEW,MANU-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					AL}. AUTO_WITH_ REVIEW= System allocation with review option. AUTO_ WITHOUT_ REVIEW-W=System allocation without

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						review option. MANUAL-L=Manual allocation. System will not allocate LSP. 5. New Allocation And Review Option= Possible values {AUTO_WITH_

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						REVIEW,AUTO_WITH_REVIEW_PRVALLOC,AUTO_WITHOUT_REVIEW,MANUAL} AUTO_WITH_REVIEW= System allocation with review

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						option. AUTO_WITH_ REVIEW_ PRVALLOC-C=System allocation and review will be required if previous allocation was retained. AUTO_ WITHOUT_ REVIEW-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						=System allocation without review option. MANUAL=Manual allocation. System will not allocate LSP.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpo-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus(CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.CreateApprovalRequest	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.CreateApprovalRequest_Impl	CheckApprovalRequirement: C1-APPR-	This algorithm creates approval request if required based on certain conditions.This process will check if LSP assignment needs to be approved if LSP assignment

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	se of th- e al- g- or- it-	case- Status) Bool getShoul- dAutoTr- ansition () String getNex- tCaseSt- atus() String getNex-			CH- K	status = "Pend- ing Approval" Approval would be required if either of below is true • System alloc- ation override by user i.e. user has changed the LSP assigned by

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to execute the	tTransCondition()				the system. Set Approval Reason as "Allocation override" • Exposure i.e. sum of balances for all accounts associated with the case is more than a specified threshold.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ospecific status.					However if no threshold has been specified this parameter should be ignored. Set Approval Reason as "High Exposure" • In case approval is required for both the reason, con-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specific sam-					catenate the approval reasons before sending for approval. If approval is required • Transition the case to a specified status defined as the parameter

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lealgorithmcreatesT-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ksthe Case to it as FK Ch-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.ResumeStatusLSP	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.ResumeStatusLSP_Impl	Resume Status from Previous LS-	Algorithm to resume previous status

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()			P: C1-RE-SS-TA-TU-S	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot	The purpose of	void setCase (ToDoCase toDoCase); void setNextCaseStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	CheckSubmissionDate:C1-CHKS-	Check Submission Date

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the algorithms spot	(CaseStatus caseStatus);	s.legal.CheckSubmissionDateExitProcessing	s.legal.CheckSubmissionDateExitProcessing_Impl	UBDT-2	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	d-di-ti-o-n-al lo-gi-c w-h-e-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the next status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpo-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.UpdateLSPAssignment	com.splwg.cb.domain.collection.caseType.specialisedCollections.legal.UpdateLSPAssignment_Impl	Update LSP (CLOS): C1-LSPSTA-	Set LSP assignment status to value provided in the parameter. This should be done only for Latest LSP assignment and if it was done by current legal case. If Status = Closed or Can-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	se of th- e al- g- or- it-	case- Status) Bool getShoul- dAutoTr- ansition () String getNex- tCaseSt- atus() String getNex-			TU- S	celled set Assign- ment End date = Business Date Status possible values {CLOS,REJ,CA- N,PNAP} CLOS=Closed REJ=Rejected PNAP=Pending for Approval.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to execute the	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	o s- p- e- cif- ic st- at- u- s.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specific sam-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lealgorithmcreatesT-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ksthe Case to it as FK Ch-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal		Validate Expired Default Notice: C1-DE-	System should check that for associated accounts default notice has expired, This check can be for primary account or for all associated delinquent account

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()			FN-OE-XP	based on para-meter. 1. Association Type={P,A}.P-P=Primary Type Association,A= Primary as well as Secondary type association 2. To Do Type= To Do will be cre-ated if validation failure option is

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				N. 3. To Do Role= To Do Role for the specified To Do Type. 4.Val-idationfailure Option= {Y,N}. If it is Y then case transition will be failed elsen a To Do will be cre-ated.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-	Col-lat-era-l Ver-ific-atio-n: C1-VR-FY-	This will perform following val-idations for the collateral with the case • if soft para-meter Collateral type to this algorithm type is "PROPERTY"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()	s.AssetRepo.CollateralVerification	s.AssetRepo.CollateralVerification_ Impl	CO-LS	then, Only one col-lateral is asso-ciated with the case also that Collateral is associated with Facility for the primary account associated with the case • If collateral type soft para-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				meter is blank, then above val-idation should be ignored and Collateral status is set to not Sold • it will also val-idate that if There is not act-ive Asset repos-session case running for the collateral. If any

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					of the above validations fail case creation process should be terminated

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Account Association for Asset	This algorithm will perform following actions: • It will get all facilities to which this collateral is associated also it will get all accounts for these facilities. • It will Associate

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ose of the alg-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.AssetRepo.AccountAssociationForAssetRepossessionCase	s.AssetRepo.AccountAssociationForAssetRepossessionCase_Impl	RepossessionCase: C1-ARSACCTS	these accounts with the case. Scope of this association is limited to accounts already in collections. This process will not check for any accounts not in collections.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pot is to execut-	String getNextTransCondition()				This algorithm doesn't have any soft parameter.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Customer Association for Asset	This algorithm will perform following actions: • It will get all customers who are the owners for the selected collateral • It will Associate these customers with the case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ose of the alg-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.AssetRepo.CustomerAssociationForAssetRepossessionCase	s.AssetRepo.CustomerAssociationForAssetRepossessionCase_Impl	RepossessionCase: C1-ARSCUSTS	Scope of this association is limited to customers already in collections. This process will not check for any customers not in collections. This algorithm doesn't have any

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				soft parameter.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into sp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Update Collateral Property: C1-	This algorithm will perform following operations: 1)if the value of updateCollateralProperty soft parameter is "SET" and type of possession is "Warrant" then

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.AssetRepo.UpdateCollateralProperty	s.AssetRepo.UpdateCollateralProperty_Impl	UP-CO-LP-RO-P	Fetch the collateral for which case is created and update the IS_LEGAL_SW= "Y" and populate the case_id on this collateral. 2)if the value of updateCollateralProperty soft parameter is "RESET" then

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				Fetch the col-lateral for which case is created and update the IS_LEGAL_SW= "N" and IS_ REPO_SW= "N". nullify the case_id on this collateral

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot		void setCase (ToDoCase toDoCase); void setNextCaseStatus (CaseSt-	com.splwg.cb.domain.collection.caseType.specialisedCollections.CloseTodo	com.splwg.cb.domain.collection.caseType.specialisedCollections.CloseTodo_Impl	Close To do's Algorithm: C1-CLST-	This process will close all To-Do's of specific To-do types associated with the case. Up to 5 To-DO types can be given to this algorithm to close.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		atus caseStatus);			OD-O	
CaseTypeEnterStatusAlgorithmSpot	The	void setCase (ToDoCase toDoCase) void setCaseOri-	com.splwg.ccb.domain.collection.caseType.specialisedCol-	com.splwg.ccb.domain.collection.caseType.specialisedCol-	Validations for Mandato-	Subjective Validations for Mandatory Characteristics: This process will validate specified char-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pur-pose of the	ginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition () String getNex-	lection-s.AssetRepo.MandatoryCharacteristics	lection-s.AssetRepo.MandatoryCharacteristics_Impl	ry Characteristics: C1-CHAR-VALS	acteristics to be present on the case with reference to value selected by the user for one of the characteristics. This algorithm will have reference characteristic type

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ithms-pot is to e-	tCaseStatus() String getNextTransCondition()				and up to 5 validation characteristic type as parameters,So based on the reference characteristic type and value of this characteristic, system should validate that mandatory characteristic types

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	whenCaseismm-					If the parameter specifying mandatory characteristic type is blank, it should be ignored

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.s-plwg.c-cb.-domain.-col-lection.caseType.specialisedCollection-	com.s-plwg.c-cb.-domain.-col-lection.caseType.specialisedCollection-	Validations for Mandatory Character-	Subjective Validations for Mandatory Characteristics: This process will validate specified characteristics to be present on the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p- o- se of th- e al- g-	(CaseSt- atus case- Status) Bool getShoul- dAutoTr- ansition () String getNex- tCaseSt- atus()	s.As- setRepo.MandatoryCharacteristics	s.As- setRepo.MandatoryCharacteristics_ Impl	ist- ics: CI_ CH- AR- VA- L	case with ref- erence to value selected by the user for one of the char- acteristics. This algorithm will have ref- erence char- acteristic type and up to 5 val- idation char- acteristic type as

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				parameters,So based on the ref-erence char-acteristic type and value of this characteristic, system should validate that man-datory char-acteristic types have some value captured. If the parameter

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					specifying mandatory characteristic type is blank, it should be ignored

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	a-m-pl-e al-g-or-it-h-m cr-e-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Update Collateral Status in the	Subjective Validations for Mandatory Characteristics: This process will validate specified characteristics to be present on the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()	s.As-setRepo.Up-dateCollateralStatusInTheHost	s.As-setRepo.Up-dateCollateralStatusInTheHost_Impl	Ho-st: C1-CH-AR-VA-LZ	case with ref-erence to value selected by the user for one of the char-acteristics. This algorithm will have ref-erence char-acteristic type and up to 5 val-idation char-acteristic type as

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				parameters,So based on the ref-erence char-acteristic type and value of this characteristic, system should validate that man-datory char-acteristic types have some value captured. If the parameter

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					specifying mandatory characteristic type is blank, it should be ignored

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Initiate Collateral Valuation: C1-CO-	this alogrithm will work as follows: System should check if "X" days have elapsed since the last assessment was done for the collateral. That

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()	s.As-setRepo.Up-dateCollateralStatusInTheHost	s.As-setRepo.Up-dateCollateralStatusInTheHost_Impl	LL-VA-LX	is check if (Assessment date + X) <= Cur-rent business date. Number of days, X, will be set as Assess-ment Expiry Days parameter for this process. If yes - Create a

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				To Do to alert the user that col-lateral valuation is required. This To Do should be associated with the case. To Do Type is passed as a parameter to the process. However, To Do

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ismovedintosp-					should not be created if • A To Do of same To Do Type is already open for the case • A To Do of same To Do Type was closed within past "Y" days

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					To Do should be assigned to the To Do Role set as parameter to this process. If the parameter is left blank, To Do should be assigned to the default role.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot	The purpose of	void setCase (ToDoCase toDoCase); void setNextCaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.CloseTodo	com.splwg.cb.domain.collection.caseType.specialisedCollections.CloseTodo_Impl	Close To do's Algorithm: C1-CLST-OD-	This process will close all To-Do's of specific To-do types associated with the case. Up to 5 To-DO types can be given to this algorithm to close.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the algorithms spot	(CaseStatus casestatus);			O	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	d-di-ti-o-n-al lo-gi-c w-h-e-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the next status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpo-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus(CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	Validations for Mandatory Characterist-	Subjective Validations for Mandatory Characteristics: This process will validate specified characteristics to be present on the case with reference to value

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	se of th- e al- g- or- it-	case- Status) Bool getShoul- dAutoTr- ansition () String getNex- tCaseSt- atus() String getNex-	setRepo.MandatoryCharacteristics	setRepo.MandatoryCharacteristics_ Impl	ics:- C1- CH- AR- VA- LS	selected by the user for one of the char- acteristics. This algorithm will have ref- erence char- acteristic type and up to 5 val- idation char- acteristic type as parameters,So based on the ref-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to execute the	tTransCondition()				erence characteristic type and value of this characteristic, system should validate that mandatory characteristic types have some value captured. If the parameter specifying mandatory char-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ospecific status.					acteristic type is blank, it should be ignored

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specific sam-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lealgorithmcreatesT-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ksthe Case to it as FKCh-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purp-	void setCase (ToDoCase toDoCase); void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Update Collateral Status in the	Subjective Validations for Mandatory Characteristics: This process will validate specified characteristics to be present on the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	o-se of th-e al-g-or-	(CaseStatus caseStatus); Bool getShouldAutoTransition(); String getNextCaseStatus();	s.AssetRepo.UpdateCollateralStatusInTheHost	s.AssetRepo.UpdateCollateralStatusInTheHost_Impl	Host: C1-CHAR-VALZ	case with reference to value selected by the user for one of the characteristics. This algorithm will have reference characteristic type and up to 5 validation characteristic type as

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	t is to e-x-e-c-ut-e	String getNex-tTransC-ondition ();				parameters,So based on the ref-erence char-acteristic type and value of this characteristic, system should validate that man-datory char-acteristic types have some value captured. If the parameter

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	moved into specif-					specifying mandatory characteristic type is blank, it should be ignored

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	oitasF-KCh-ar-a-ct-er-ist-ic					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase); void setNextCaseStatus (CaseSt-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	Validation Settlement: C1-VALSET	This algorithm will perform following actions: Before completing the asset repossession case below validations should be done for the case 1. Collateral should have a settlement date

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	atus caseStatus);	setRepo.ValidateCollateralSettlementStatus	setRepo.ValidateCollateralSettlementStatus_Impl		2. Realization status for the collateral should be "REALIZATION_COMPLETE" Possible values of Realization status are : REALIZATION_IN_PROGRESS and REALIZATION_COMPLETE.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	on-all logic when a C-					Transition to completed status should fail if above validations fail.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	status.					
CaseTypeExitStatusAlgorithmSpot	The purp-	void setCase (ToDoCase toDoCase); void	com.splwg.ccb.domain.collection.caseType.special-	com.splwg.ccb.domain.collection.caseType.special-	Initiate LMI Process: C1-INI-	Parameters to the algorithm must be as follows: • For Initiate LMI Options: 1) "Initiate LMI with highest

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	o-se of th-e al-g-or-it-h-m	setNex-tCaseSt-atus (CaseSt-atus case-Status);	isedCol-lections.AssetRepo.InitiateLMIP	isedCol-lections.AssetRepo.InitiateLMIP_Impl	TL-MI	insured amount" use HIGH_ INSUR_AMT 2) "Initiate LMI from a specific insurer first" use SPEC_ INSURER • For No LMI Option 1)"Mark primary account for strategy review"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to perform additio-					use PRIMARY 2)"Mark all accounts for strategy review" use ALL 3)"No Action" use NA

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the current status to the					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot	The purpose of	void setCase(ToDoCase toDoCase); void setNextCaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.CloseTodo	com.splwg.cb.domain.collection.caseType.specialisedCollections.CloseTodo_Impl	Close To do's Algorithm: C1-CL-ST-	This process will close all To-Do's of specific To-do types associated with the case. Up to 5 To-DO types can be given to this algorithm to close.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the algorithms spot	(CaseStatus caseStatus);			ODO	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	d-di-ti-o-n-al lo-gi-c w-h-e-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the next status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpo-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	Update Collateral Property: C1-UPCO-	This algorithm will perform foolowing operations: 1)if the value of updateCollateralProperty soft parameter is "SET" and type of possession is "Warrant" then Fetch the col-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	se of th- e al- g- or- it-	case- Status) Bool getShoul- dAutoTr- ansition () String getNex- tCaseSt- atus() String getNex-	setRepo.UpdateCollateralProperty	setRepo.UpdateCollateralProperty_ Impl	LP- RO- P	lateral for which case is created and update the IS_LEGAL_SW= "Y" and populate the case_id on this collateral. 2)if the value of updateCol- lateralProperty soft parameter is "RESET" then Fetch the col-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to execute the	tTransCondition()				lateral for which case is created and update the IS_LEGAL_SW= "N" and IS_ REPO_SW= "N". nullify the case_id on this collateral

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ospecific status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specific sam-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lealgorithmcreatesT-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ksthe Case to it as FK Ch-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection	com.splwg.cb.domain.collection.caseType.specialisedCollection	Update Collateral Status in the	Update Collateral Status in the host

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()	s.As-setRepo.Up-dateCollateralStatusInTheHost	s.As-setRepo.Up-dateCollateralStatusInTheHost_Impl	Ho-st: C1-UP-CO-LL-ST-S	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pot is to execut-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
PtpActiveForNgpAlgorithmSpot	This algorithm	void setPromiseToPay (PromiseToPay promiseToPay); Pay-	com.splwg.cb.domain.customerinfo.paymentPlan.CollectionPTPActiveForNgpAlgorithm	com.splwg.cb.domain.customerinfo.paymentPlan.CollectionPTPActiveForNgpAlgorithm_Impl	PTP Active Algorithm: C1-PTPACTI-	This algorithm is used to perform additional processing when the status of a PTP becomes Active. Customer Contacts can be generated via

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spot is used for p-	mentPlanStatusLookup getPaymentPlanStatus();			VE	this algorithm.Contact Class, method and type have to be specified. Following parameters used to perform processing--- 1) contactTypeForLetter -- Contact

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ional processing like					Type for Letter. 2)contactClassForLetter -- Contact Class for Letter. 3)contactMethodForLetter -- Contact Method for Letter. (Value should be -- OTBL(Outbound Letter))

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e.					4)contactTypeForSMS--- Contact Type for SMS. 5)contactClassForSMS--- Contact Class for SMS. 6)contactMethodForSMS--- Contact Method for SMS. (Value

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						should be-- OTBS (Outbound Short Message Service))
PtpKeptForNgpAlgorithmSpot	This algorit-	void setPromiseToPay (PromiseToPay prom-	com.splwg.cb.domain.customerinfo.paymentPlan.CollectionPTPKeptForNgpAlgorithm	com.splwg.cb.domain.customerinfo.paymentPlan.CollectionPTPKeptForNgpAlgorithm_	PTP Active Algorithm: C1-	This algorithm is used to perform additional processing when the status of a PTP becomes Kept.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	hms-pot is used for	iseToPay); PaymentPlanStatusLookup getPaymentPlanStatus();		Impl	PT-PK-EP-T	Customer Contacts can be generated via this algorithm. Contact Class, method and type have to be specified. Following parameters used to perform pro-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	d-di-ti-o-n-al pr-o-c-e-ss-in-g					cessing--- 1) con-tactTypeForLet-ter -- Contact Type for Letter. 2)con-tactClassForLet-ter -- Contact Class for Letter. 3)con-tactMeth-odForLetter --

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Contact Method for Letter. (Value should be -- OTBL(Outbound Letter)) 4)contactTypeForSMS--- Contact Type for SMS. 5)contactClassForSMS--- Contact Class for SMS.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						6)contactMethodForSMS---Contact Method for SMS. (Value should be--OTBS (Outbound Short Message Service))
PtpBrokenForNgpAlgorithmSpot	T-	void setPromiseToPa-	com.splwg.ccb.domain.-	com.splwg.ccb.domain.-	PTP	This algorithm is used to perform

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	his Algorithm spot	y(PromiseToPay promiseToPay); PaymentPlanStatusLookup getPaymentPla-	customer-info.paymentPlan.CollectionPTPBrokenForNgpAlgorithm	customer-info.paymentPlan.CollectionPTPBrokenForNgpAlgorithm_Impl	Broken Algorithm: C1-BRKP-TP-NG-P	additional processing when the status of a PTP is set to Broken. Customer Contacts can be generated via this algorithm. Following parameters used to

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	sused for perform-	nStatus ();				perform processing--- 1) contactTypeForLetter -- Contact Type for Letter. 2)contactClassForLetter -- Contact Class for Letter. 3)contactMeth-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	singwhenPTPisbro-					odForLetter -- Contact Method for Letter. (Value should be -- OTBL(Outbound Letter)) 4)contactTypeForSMS--- Contact Type for SMS. 5)contactClassForSMS--- Contact

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Class for SMS. 6)contactMethodForSMS--- Contact Method for SMS. (Value should be-- OTBS (Outbound Short Message Service))

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
RuleFactsPopulationAlgorithmSpot	This Algorithm spot	void setInputKeyValue1 (String inputKayValue1); void setInputKeyV-	com.splwg.ccb.domain.collection.RuleFactsPopulation	com.splwg.ccb.domain.collection.RuleFactsPopulation_Impl	Rule facts populating algorithm: C1-	This algorithm is used to populate the facts required for rule engine. Input Key Input Key 1 to 5 represent

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is used for popul- at-	alue2 (String inputKay- Value2); void setIn- putKeyV- alue3 (String inputKay-			BR- LS- R	primary key of BO(used in Input BO name 1 - 5) Note: Cur- rently you can use only Input key 1,2 and 3 and Input BO 1,2 and 3 Valid values in

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	required for rule engin-	Value3); void setInputKeyValue4 (String inputKayValue4); void setInputKeyV-				Input key and Input BO Input key 1 (Mandatory) Account Id (No other input value allowed) Input key 2 Party UID (No other input

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		alue5 (String inputKayValue5); void setRuleId(String ruleId); void setRuleSetExecution				value allowed) Input key 3 Main Customer PER_ID for given account ID (No other input value allowed) Input key 4 NA Input key 5 NA Input BO name 1 (Mandatory) C1-ACCT-EXTN

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(Bool isRuleSet); void setRuleEffectiveDate (String ruleEffectiveDate); void				Input BO name 2 BO having primary key as input key 2 i.e. Party UID Input BO name 3 BO having primary key as input key 3 i.e.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setFactDetails (CollectionsFactDetailsLoader collectionsFactDetailsLoa-				Per ID Input BO name 4 NA Input BO name 5 NA

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		RuleFactParameters getRuleFactParameters(); String getGivenFactCodes();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpo-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.AssociateDelinquentAccount	com.splwg.cb.domain.collection.caseType.specialisedCollections.AssociateDelinquentAccount_Impl	Borrower Centric Case Lifecycle-C1-	Associate new delinquent account of the customer to the case.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	se of th- e al- g- or- it-	case- Status) Bool getShoul- dAutoTr- ansition () String getNex- tCaseSt- atus() String getNex-			AS- SO- DE- LA- C	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to execute the	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ospecific status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specific sam-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lealgorithmcreatesT-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ksthe Case to it as FK Ch-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
PreprocessBusinessObjectRequestAlgorithmSpot			com.splwg.cb.domain.collection.addressPersonCollectionAddressPreProcess	com.splwg.cb.domain.collection.addressPersonCollectionAddressPostProcess_Impl	Update Collection Addresson	This is a reference implementation of Post processing Algo. Customization team can utilize this hook.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					BorrowerPanel-C1-PERADD-PP	
Pre-pro-			com.s-plwg.c-	com.s-plwg.c-		Contact Point

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
cessBusinessObjectRequestAlgorithmSpot			cb.domain.collection.address.ContactPreferencePreProcess	cb.domain.collection.address.ContactPreferencePreProcess_Impl	UpdateCollectionContactPoint-C1-PC-	PreProcessing algorithm. Attached on BO pre processing spot. This is a hook provided to customization. This can be utilized to validate the contact point data.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ONTPRE	
CaseTypeEnterStatusAlgorithmSpot	The	void setCase(ToDoCase toDoCase) void setCaseOri-	com.splwg.cb.domain.collection.caseType.specialisedCol-	com.splwg.cb.domain.collection.caseType.specialisedCol-	Check if Special Case Alr-	Check if any active case is present of a given case category or case type on the customer Processing

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	purpose of the	ginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition () String getNex-	lection-s.bankruptcy.CheckBankruptcyCaseExist	lection-s.bankruptcy.CheckBankruptcyCaseExist_Impl	eady Exist on the Customer Enter Pro-	steps are as below 1. If only Case Category is specified check if any active case is running on the customer whose a. Case category is same as the parameter set for the algorithm

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ithms-pot is to e-	tCaseStatus() String getNextTransCondition()				2. If Case Type is specified check if any active case is running on the customer whose a. Case type is same as the parameter set for the algorithm 3. If yes validation should fail

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	when Case is m-					4. If Consider Enterprise Id = Y perform the check for all the parties with same Enterprise Id. Consider Enterprise Id value should be "YES" or "NO"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	a-m-pl-e al-g-or-it-h-m cr-e-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-	Pull all the non delinquent accou-	Processing steps are as below • Pull all Not in Collections accounts into OB Collections (from OBP) where the associated customer is one of

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()	s.bank-ruptcy.Bank-ruptcyPullNonDelinquentAcc	s.bank-ruptcy.Bank-ruptcyPullNonDelinquentAcc_Impl	nts of the cus-tom-er into col-lec-tion-s - Ent-er	the borrower. • If Account Rela-tionships = MC consider only the accounts where the customer is primary owner. If Account Rela-tionships = FO consider all accounts where the customer is a

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				financial owner. If Account Rela-tionship = All consider all accounts where the customer is a financial or non financial owner. • If Consider Enterprise Id = Yes; Determine the Enterprise Id corresponding

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					the party id; then determine the party id corresponding to OBP host and then proceed to pull the accounts. Possible Values of Account Relationships MC, FO, ALL

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					Possible Values fro Consider Enterprise Id Yes/No

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Associate all accounts to the case	Associate all accounts to the case where customer is a primary borrower For the primary customer associated with the case • Get all accounts where

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ose of the alg-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.bankruptcy.BankruptcyAssociateAcc	s.bankruptcy.BankruptcyAssociateAcc_Impl	where customer is a primary borrower-Ent-	this customer is primary owner and the accounts are In Collections. (Fetch accounts based on Enterprise Id if Consider Enterprise ID = Y) • Shortlist the accounts that are not yet associated with the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				case. • Associate the shortlisted accounts with the case Consider Enter-prise Id value should be "YES" or "NO"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into sp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Exclude all the associated accou-	For all the accounts associated with the case • Call the Dialer Exclusion Service to exclude the accounts from feed to Dialer

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ose of th-e alg-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()	s.bank-ruptcy.BankruptcyExcludeAccDlr	s.bank-ruptcy.BankruptcyExcludeAccDlr_ Impl	nts fro-m Dia-ler-Ent-er Pro-ces-sin-g: C1-Ex-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Initiate Collateral Valuation	For each collateral on the associated account if last valuation was done 'X' days before than create a Collateral Valuation Task. Enter the Collateral Code; Col-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.bankruptcy.BankruptcyInitiateCollateralValuation	s.bankruptcy.BankruptcyInitiateCollateralValuation_Impl	for all collaterals whose last valuation wa-	lateral Type and Collateral Description as Remarks Exclude Collaterals with Collateral Types specified in parameter. Also Exclude Collaterals that have been already Repos-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()			Pro-ces-sin-g: C1-Ini-Clt-Val	sessed or Sold. Values of Val-idation Date: POSTING DATE, SYSTEM DATE

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into sp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm	void setCase(ToDoCase toDoCase); Bool getShouldAutoTr-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Monitor if any of the associated	If any of the associated account has delinquency Start Date = Today's posting date Create Bankruptcy Notification as: 'Account

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p- e is u- s- e- d to p- er- fo- r-	ansition (); CaseSta- tus getNex- tCaseSt- atus(); String getNex- tTransC-	s.bank- ruptcy.Bank- ruptcyMonitorChargeOffDelinquency	s.bank- ruptcy.Bank- ruptcyMonitorChargeOffDelinquency_ Impl	acc- ou- nt ne- ed to be cha- rge- d off an- d	<Account Num- ber> has become Delin- quent' Set Display Date of the case to cur- rent business date. Monitor Charge Off: If any of the asso- ciated account

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-	ondition();			C1-MTRCRGDQY	has DPD= Charge Off Threshold Create Bankruptcy Notification as 'Account <Account Number> can be Charged Off' Set Display Date of the case to cur-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						rent business date. If Secured Accounts = Yes than associated accounts with Secured Switch = Y should also be considered. Monitor Delinquency = "Y" or "N" ,Monitor Charge Off = "Y"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						or "N" ,Secured Accounts = "Y" or "N" Values of Validation Date: POSTING DATE, SYSTEM DATE
CaseTypeAutoTransitionAlgorithmSpot	Thi-	void setCase	com.s-plwg.c-cb.-	com.s-plwg.c-cb.-	Notify the	If 341 Hearing

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	s algorithm type is u-	(ToDoCase toDoCase); Bool getShouldAutoTransition (); CaseSta-	domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitor341Hearing	domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitor341Hearing_Impl	Bankruptcy Specialist on Hearing Dates-	Date has been captured and is in future Create a notification for the Bankruptcy Specialist when the 341 Hearing date has been passed. i.e. when Business Date = 341 Hear-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to perform auto transit-	tus getNextCaseStatus(); String getNextTransCondition();			: C1-MT-R3-41-HR-G	ing Date + 1 Notification: "Capture details of 341 Hearing" Set Display Date of the case to current Business Date If Objection Hearing Date has been captured

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						and is in future Create a notification for the Bankruptcy Specialist when the Objection Hearing date has been passed. i.e. when Business Date = Objection Hearing Date + 1 Notification: "Capture details

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						of Objection Hearing for Debtors Proposed Plan" Set Display Date of the case to current Business Date Values of Validation Date: POSTING DATE, SYSTEM

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						DATE
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type	void setCase(ToDoCase toDoCase); Bool getShouldAutoTr-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Monitor if the payment plan on	If for any of the associated account on the case the days since the last PTP Broken reaches X days a notification should be created on the case. The PTP Type

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pe is used to perfor-	ansition (); CaseStatus getNextCaseStatus(); String getNextTransC-	s.bankruptcy.BankruptcyMonitorPaymentPlan	s.bankruptcy.BankruptcyMonitorPaymentPlan_Impl	any of the associated accounts is Broken	specified in the parameter should be considered Notification: <PTP Type> broken for account <Account Number>. Days since plan broken <Days Since PTP Broken>.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-	ondition ();			g: C1-MT-RP-YM-PL-N	Set Display Date of the case to current business date. Values of Validation Date: POSTING DATE, SYSTEM DATE

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-	Notify the Bankruptcy Specialist if	Notify the Bankruptcy Specialist if the Liquidation reaches a specific status. If for any of the associated account if the liquidation case reaches a specific status than

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.bank-ruptcy.Bank-ruptcyMonitorAssetLiquidation	s.bank-ruptcy.Bank-ruptcyMonitorAssetLiquidation_Impl	the Liquid-ation reaches a spe-cific stat-us-	create a notification for the Bankruptcy Specialist. Notification: "Liquidation for Account <Account Number>; Collateral <Collateral Code> has reached status <Case Status>

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNextTransCondition()				Set Display Date of the Bankruptcy Case to Business Date

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm ty-	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Notify the Bankruptcy Specialist on	If for any of the associated account on the case if the RFS Hearing Date is reached Create Notification: "Capture details for RFS Hearing

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p- e is u- s- e- d to p- er- fo- r-	(); CaseSta- tus getNex- tCaseSt- atus(); String getNex- tTransC- ondition	s.bank- ruptcy.Bank- ruptcyMonitorHearingDate	s.bank- ruptcy.Bank- ruptcyMonitorHearingDate_Impl	RF- S He- ari- ng Dat- e- Mo- nit- ori- ng: C1- MT-	for Account <Account Num- ber> When Business date = Hearing Date + 1 Set Display Date of the case to cur- rent Business Date Values of Val-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-	();				idation Date: POSTING DATE, SYSTEM DATE

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithms	void setActionEntity(String actionEntity); void setActionSourceId	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Determine in which status the case	Bankruptcy Chapter Field should be passed as a Filing Information Chapter(FC) or Converted to Chapter(CC) as an input parameter

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot d-e-ci-d-es in w-hi-ch st-	(String actionSourceId); void setActionSourceStatusCode (String actionSourceStatusCd);	s.bank-ruptcy.DetermineBank-ruptcyTreatment	s.bank-ruptcy.DetermineBank-ruptcyTreatment_Impl	sho-uld pro-cee-d for Ba-nkr-upt-cy Tre-atm-en-	If Bankruptcy Chapter = Chapter 7 Then Transition to Manage Chapter 7 Bank-ruptcy Status If Bankruptcy Chapter = Chapter 13 Then Transition

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	has to be made base-	void setActionId (String actionId); void setActionType (ActionType actionType);				to Manage Chapter 13 Bankruptcy Status If Bankruptcy Chapter = Chapter other than 7 or 13 Then Transition to Other Bankruptcy Status Bankruptcy

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setResultType (ResultType resultType); boolean getIsProcessingComplete();				Chapter Field = "FC" or "CC Where "FC" = Filing Chapter and "CC"=Convert to chapter

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithmsp-	void setActionEntity(String actionEntity); void setActionSourceId	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bank-	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bank-	Validate if appropriate Case Details hav-	Validate if the Dynamic Panel Fields mentioned for the corresponding Dynamic panels have some val-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	otdecidesinwhichstat-	(String actionSourceId); void setActionSourceStatusCode (String actionSourceStat-	ruptcy.ValidateBankruptcyCaseData	ruptcy.ValidateBankruptcyCaseData_Impl	ebeenenteredbytheuser-Post Pro-	ues for the case. If yes the Follow Up is saved successfully. If no system should throw an error message

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to be made based on	usCd); void setActionId (String actionId); void setActionType (ActionT-				for the first blank field that it will encounter. Error Message: "<Field Name> cannot be blank"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ype actionType); void setResultType (ResultType resultType);				Possible values for Panel Names and Panel fields belonging to that Panel are as follows: Panel Name :

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();				bankruptcyTrusteeInfoPanel Corresponding Panel Fields: ENTITY_NAME,PHONE,EMAIL,FAX_NUMBER,CONTACT_POINT_

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						NAME,CONTACT_POINT_PHONE_NUM,CONTACT_POINT_EMAIL,CONTAC-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Panel Name : bankruptcyProcessingInfoPanel Corresponding Panel Fields : HEARING_ DATE,HEARING_ LOCATION,LEN-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						DER_COLL_VAL_DATE,LENDER_COLL_VAL,DISCHARGE_DATE,DISMISSED_DATE,CHAPTER_

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Panel Name : bankruptcyDebtorAttorneyPanel Corresponding Panel Fields : FIRM_ NAME,PHONE,-ENTITY_ NAME,DEBTO-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						R_ADDRESS Panel Name : bankruptcyFiling-InfoPanel Corresponding Panel Fields : DATE_OF_ BNKPT_CASE_ FILE,BNKPT_ CASE_ NUM,COURT,C-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						HAPTER Panel Name : bankruptcyConfirmPlanInformationPanel Corresponding Panel Fields : RECEIVE_DT,TOTAL_AMMOUNT,LAST_PAYMENT_DT

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Panel Name : bankruptcyDebtorProposedPlanInfoPanel Corresponding Panel Fields : RECEIVE_DT,TOTAL_AMMOUNT,LAST_PAYMENT_DT,OBJECTION_DATE,OBJECTI-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						ON_ OUTCOME,HEARING_DATE Panel Name : bankruptcyLegalCounselInfoPanel Corresponding Panel Fields : ASSIGNED_ DATE,COUNSEL_ NAME,CONTAC-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						T_POINT_ NAME,EMAIL,PHONE,ALTERNATE_PHONE
BusinessObjectEnterStatusAlgorithmSpot			com.splwg.cb.domain.collection.caseType.special-	com.splwg.cb.domain.collection.caseType.special-	NotifyBankruptcySp-	Create Notification Notification: <PTP Type> Kept for account <Account Number>. Set Display Date of the case to cur-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
			isedCollection-s.bankruptcy.BankruptcyNotifyPaymentPlanKept	isedCollection-s.bankruptcy.BankruptcyNotifyPaymentPlanKept_Impl	ecialist when a Payment Plan status	rent business date.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					rocessing C1-NTPYMPLNK	
ToDoTypeToDoPostProcessAlgorithmSpot	T-		com.splwg.ccb.-	com.splwg.ccb.-	Notify Ba-	Create Notification Notification:

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	his Algorithm spot		domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyNotifyTaskCompletion	domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyNotifyTaskCompletion_Impl	nkruptcy Specialist of Task Completion-	<Task Id> - <Task Name> complete for <Account Number>. Set Display Date of the case to current business date. Notification should be created on the latest case associated

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	sused fornotifying				essing C1-NT-FT-SK-CM-P	on the Account

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	Is-o for al-lo-c-at-in-g ta-sk to					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ToDoTypeToDoPostProcessAlgorithmSpot	This Algorithms		com.splwg.cb.domain.collection.vendor.VendorManagementAutomaticTaskAllocation	com.splwg.cb.domain.collection.vendor.VendorManagementAutomaticTaskAllocation_Impl	Vendor Management - Automatic	On creation of task check if task is already allocated to a member. If Yes no action required. If No allocate the case to the member with lowest number of tasks of that task type in

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pot is used for notify-				Allocation of tasks to Vendors - TODO	the queue.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ion and also for allocat-				KV-ND-R	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Hardship - Associate Accounts of Mai-	This algorithm associates the Party on whom the hardship case is created.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-o-se of th-e al-g-	(CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-tCaseSt-atus()	s.fin-ancialHardship.HardshipAssociation	s.fin-ancialHard-ship.HardshipAssociation_Impl	n Cu-sto-me-r - Ent-er Pro-ces-sin-g C1-HA-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into spp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithmty-	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-		

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p- e is u- s- e- d to p- er- fo- r-	(); CaseSta- tus getNex- tCaseSt- atus(); String getNex- tTransC- ondition	s.bank- ruptcy.Bank- ruptcyMonitorHearingDate	s.bank- ruptcy.Bank- ruptcyMonitorHearingDate_Impl		

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-	();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithmty-	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition	com.s-plwg.c-cb.-domain.-col-lec-tion.scra.al-gorithm.ActiveServiceAlgorithm	com.s-plwg.c-cb.-domain.-col-lec-tion.scra.al-gorithm.ActiveServiceAlgorithm_Impl	This algorithm	This algorithm will Transit the case to 'Suspend Status' if the customer is in Active Service or dependent of a person in Active Service.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-e is u-s-e-d to p-er-fo-r-	(); CaseStatus getNextCaseStatus(); String getNextTransCondition			will Transit the case to 'Su-	Validate against all Financial Owners parameter will decide if check has to be done for main customer or all financial owners. If Validate against all Financial Owners parameter value is Y, algorithm

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-	();			spend Status' if the cus-	will check active service member against all financial owners.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ice or dependent of a per-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Validate against all	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Financial Owners parameter-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					k has to be done for main cus-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ners parameter value is Y, algorit-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code - C1-AC-TM-EM-CH-K	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
GenericEventHostUpdateAlgorithmSpot	This is generic alg-	void setPerson(Person person); void setToDoCase (ToDoCase toDoCase);	com.splwg.cb.domain.collection.loan.UpdateDisputeFlagAlgorithm	com.splwg.cb.domain.collection.loan.UpdateDisputeFlagAlgorithm_Impl	Generic Algorithm to update host flag	Generic Algorithm to update host flag through event manager

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithms pot which ca-	void setAccount (Account account);			through event manager Code - C1- EV-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	te Generic Algorithm of				TH-ST-UP-D	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithms	void setActionEntity(String actionEntity); void setActionSour-	com.splwg.cb.domain.collection.algorithms.ScheduleCallPostProcessingAlgorithm	com.splwg.cb.domain.collection.algorithms.ScheduleCallPostProcessingAlgorithm_Impl	This algorithm	This algorithm is used to fulfil request by customer to collector for calling at specific time. - The Call Back Time will get saved as the Next Action Time on the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot d-e-ci-d-es in w-hi-ch st-	ceId (String actionSo-urceId); void setAc-tionSour-ceStatus-Code (String actionSo-			is use-d to ful-fil req-ues-	case. If 'NA' is selected the value will go as blank. - If the Next Action Date is same as Current date and Online Dialer Inclusion = 'Yes' then add/update the record in the Dialer extract

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	has to be made base-	urceStatusCd); void setActionId (String actionId); void setActionType			o collector for calling	using the Dialer Inclusion Service. The Dialer Extract Status will be set as 10.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(ActionType actionType); void setResultType (ResultType resultType);			- The Call Back	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();			Time will get saved as the Ne-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					n the case. If 'N-A' is select-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					- If the Next Action Date is sa-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					measCurrentdateandOnline	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					dd/-update the record in the Dialer extr-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code - C1-SC-HC-AL-L	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusValidationAlgorithmSpot	The purpose of the	void setCase (ToDoCase toDoCase); void setOriginalCaseStatus (CaseSt-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	If DL Template Co-	If DL Template Code has been mentioned validate if Demand Letter has been sent in last X days. If AL Template Code has been mentioned validate if Accel-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	algorithms spot is to	atus caseOriginalStatus);	setRepo.algorithms.ValidateDemandLetterandAccelerationLetter	setRepo.algorithms.ValidateDemandLetterandAccelerationLetter_Impl	de has been mentioned	eration Letter has been sent in last X days. If X Days is not specified just check if the letters have been sent on the account. Checks will be done for all asso-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ion logic before Case is				and Letter has been	ciated accounts unless 'Only Primary Account = Yes' in which case the check will be only on primary associated account.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					If AL Template Co-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					de has been mentioned	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					eration Letter has been sent in	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					If X Days is not specified	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					just check if the letters have be-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Checks will be done for all	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					associated accounts unless 'Only Pri-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					case the check will be only on prima-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code - C1-VALID-DL-AL	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	If any of the custom-	If any of the customers associated with the repossession case satisfy below criteria block repossession initiation. The customer is a Service Member and

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-	setRepo.algorithms.ActiveMilitaryServiceCheckonAssociatedCustomers	setRepo.algorithms.ActiveMilitaryServiceCheckonAssociatedCustomers_Impl	ers associated wit-	The customer has not waived his SCRA Protection and (The customer is in Active Service or the number of days since the end date of customers last active service < X days or the service member is

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inessl ogic when Case	tTransCondition()			ase satisfy below criteri-	missing in action) Error Message: "Repossession case cannot be initiated. SCRA checks failed."

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					The customer is a Ser-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					vice Member and Th-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					e customer has not waived his	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					RA Protection and (The cus-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					tomer is in Active Service or	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					er of days since the end dat-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					the service member is missing in acti-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Error Message: "Repossession	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					case cannot be initiated. SCRA checks fail-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code - C1-BL-OC-KR-EP-O	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition();	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	If DL Template	If DL Template Code has been mentioned validate if Demand letter has been sent and current date > Demand Letter Expiry Date. If AL Template Code has been

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is used to perform a-	CaseStatus getNextCaseStatus(); String getNextTransCondition();	setRepo.algorithms.MonitorDemandLetterandAccelerationLetterExpiry	setRepo.algorithms.MonitorDemandLetterandAccelerationLetterExpiry_Impl	Code has been	mentioned validate if Acceleration letter has been sent and the current date > Acceleration letter Expiry Date. If 'Only Primary Account' = Yes then the above checks need to

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ssing for a Case.				ate if Demand let-	be done only on Primary account else the checks should be done on all associated accounts. If both are true transition the case to 'Repossession Referred' Status

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Letter Expiry Date.	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					If AL Template Code	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					has been mentioned val-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					n letter has been sent	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					If 'Only Primary Accou-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					nt' = Yes then the above che-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ly on Primary account else the	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					If both are true transition the	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					case to 'Repossession Referred' Status	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code - C1-MNTR-DL-AL	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	If the Auto-	If the Auto-Approval Rule returns true the case will be transitioned to the Approved status. If the Auto Approval Rule returns false the case will remain in the Repos-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-	setRepo.algorithms.AutoApprovalCheckforRepossession	setRepo.algorithms.AutoApprovalCheckforRepossession_Impl	Approval Rul-	session Referred Status and a Task is created for the given Task Type and is assigned to the supervisor of the queue. Below facts are used : Collateral Type Collateral Category

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	in-ess logi-c w-h-e-n C-a-se	tTransC-ondition ()			e the cas-e	Repossession Reason Outstanding Amount Overdue Amount Days Past Due Last Payment Date Last Payment Amount Estimated Real-ization Amount Deficiency Bal-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					tatus. If the Aut-	ance Number of accounts associated with the collateral

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					o Approval Rule	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					lse the case will re-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					atus and a Task is	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Below facts are used : Col-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					lateral Type Collateral	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Category Repossession Reaso-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					n Outstanding Amount Overd-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ue Amount Days Past Due Las-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					t Payment Date Last Payment	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Amount Estimated Realization Amou-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					nt Deficiency Balance Number of acc-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ounts associated with the collateral	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code - C1-REPO-APRV	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithm spp-	void setActionEntity(String actionEntity); void setActionSourceId	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	Transition the case to given	Transition the case to given Case Status if Case Status is configured. Close the Approval Task Type present on the case if approval task type is configured. Copy the com-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	otdecidesinwhichstat-	(String actionSourceId); void setActionSourceStatusCode (String actionSourceStat-	setRepo.algorithms.Repossessionsessionapprovalresultpostprocessingalgorithm	setRepo.algorithms.Repossessionsessionapprovalresultpostprocessingalgorithm_Impl	Status if Case Status is configure-	ments in the result to the Approver remarks field

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to be made based on	usCd); void setActionId (String actionId); void setActionType (ActionT-			Close the Approval Task Type	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ype actionType); void setResultType (ResultType resultType);			present on the case if approval task typ-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();			Copy the comments in the result to the	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Approver remarks field Code-C1-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					RA-PR-VR-SL-T	
CaseTypeEnterStatusAlgorithmSpot	The p-	void setCase (ToDoCase toDoCase) void	com.splwg.ccb.domain.collection.caseType.spe-	com.splwg.ccb.domain.collection.case-	For each of the	For each of the accounts associated to the repossession case send the Redemption letter (create customer contact of

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	urpose of the algorit-	setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition ()	cialisedCollections.AssetRepo.algorithms.AutomaticSendingofRedemptionLetters	Type.specialisedCollections.AssetRepo.algorithms.AutomaticSendingofRedemptionLetters_Impl	accounts associated to the reposs-	given template code) If 'Only Primary Account = Yes' then send letter only on the primary account.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	t is to e- x- e- c- ut- e th- e b-	String getNext- tCaseSt- atus() String getNext- tTransC- ondition ()			the Re- de- mpt- ion let- ter (cr- eat- e cus-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	into specific status.				If 'Only Primary Account = Yes'	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					then send letter only on the primary	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code - C1-RE-DE-MP-LT-R	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	Generate and send the email	Generate and send the email to the email id of the contact person associated to the service type mentioned in the parameter. Email of specified template code will be sent. The algorithm

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-	setRepo.algorithms.RepossessionAssignmentAlert	setRepo.algorithms.RepossessionAssignmentAlert_Impl	to the email id of the contact per-	will generate the contact as well as initiate contact processing

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwheneCase	tTransCondition()			etypemementionedinthe parameter	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Email of specified template code	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					will be sent. The algorithm will gener-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ate the contact as well as initiate contact pro-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code - C1-REPOASAL	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
LetterTemplateLetterExtractCollectionAlgorithmSpot	Extract all the Col-	void setCustomerContact (CustomerContact customerContact); Letter-	com.sapl wg.cb.domain.collection.caseType.specialisedCollections.As-	com.sapl wg.cb.domain.collection.caseType.specialisedCollections.As-	Extract all the Collateral, Accou-	Extract all the Collateral, Account and Customer Information and send it to Alert Module. The contact person details of the Vendor will also be sent to the Alert Module to generate the alert.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lateral, Account	TemplateInfoBean getLetterTemplateInfo(); ReportDefinition getReportDefi-	setRepo.algorithms.ExtractRepossessionAssignmentAlgorithm	setRepo.algorithms.ExtractRepossessionAssignmentAlgorithm_Impl	nt and Customer Information	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ation and send	nition();			The contact person details of the Ve-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The contact person will also be sent to the Alert Modul-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	second details of the Vertex				Code-C1-REPEMTEMP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	base-ent to the Alert Mo-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
GenericAlgorithmSpot	This is generic	void setPerson(Person person); void setToDoCase (ToDoCase	com.splwg.cb.domain.collection.dmdc.VerifyDMDCDetailsAlgorithm	com.splwg.cb.domain.collection.dmdc.VerifyDMDCDetailsAlgorithm_Impl	This algorithm is used to check	This algorithm is used to check whether SCRA verification request should call to DMDC or not based on number of days passed.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	algorithmspotwhi-	toDoCase); void setAccount (Account accounnt); Bool getDMDCVeri-			whether SCRA verification request	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ogeneratedGenericAl-	ficationRequired();			on number of days passed. Code	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					-C1-DM-DC-REQ	
CaseTypeEnterStatusAlgorithmSpot	The	void setCase (ToDoCase toDoCase)	com.splwg.cb.domain.collection.case-	com.splwg.cb.domain.collection.case-	If	If Repossession Reason <> Bankruptcy For each customer associated with the case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	purpose of the algor-	void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition	Type.specialisedCollections.AssetRepo.ChkBkpcyOnAssociateCust	Type.specialisedCollection-s.AssetRepo.ChkBkpcyOnAssociateCust_Impl	Repossession Reason	Check if the Bankruptcy_ Switch = Y. If yes Case Creation will be rolled back and below error message will be displayed. "One or more of the collateral owners have

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	s-p-ot is to e-x-e-c-ut-e th-	() String getNex-tCaseSt-atus() String getNex-tTransC-ondition ()			Ba-nkr-upt-cy For eac-h cus-	claimed Bank-ruptcy. Repos-session process should be ini-tiated from Bank-ruptcy process"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	moved into specific st-				tomer associated with the	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Check if the Bankruptcy_	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Switch = Y. If yes Case Creatio-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					d below error message will be	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					"One or more of the collat-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					eral owners have claimed Bankr-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					s should be initiated from Bankruptcy	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code-C1-CH-KB-KP-TC-Y	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.AssetRepo.AssociateCustAssRepo	com.splwg.cb.domain.collection.caseType.specialisedCollections.As-	Associate all financial owners on the	Associate all financial owners on the associated accounts to the Repossession case.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-		setRepo.AssociateCustAssRepo_Impl	associated accounts to the Repossessio-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inessl-ogic when Case	tTransC-ondition()			Code-C1-AS-SO-CU-ST	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusValidationAlgorithmSpot	The purpose of	void setCase(ToDoCase toDoCase); void setOriginalCaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.AssetRepo.ValidateCollateral	com.splwg.cb.domain.collection.caseType.specialisedCollections.AssetRepo.ValidateCollateral_	The input collateral is associated with the account on which the repossession case is being created. The collateral belongs to the collateral type and collateral cat-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the algorithm spot	(CaseStatus caseOriginalStatus);		Impl	lateral is associated	egory specified in the parameters. If collateral type and collateral category are not mentioned no validation will be done. The collateral status is not 'Sold'. Date of Sale is blank.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	hevalidationlogicb-				unt on which the reposs-	There is no repossession case active on the collateral (IS_REPO_SW = N)

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					The collateral bel-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ongs to the collateral	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					l category specified in the	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					not mentioned no validation will be	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					The collateral status is not	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					'Sold'. Date of Sale is blank. There is no	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					repossession case active on the collatera-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					W = N) Code - C1-VALD-COLL	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithm sp-	void setActionEntity(String actionEntity); void setActionSourceId	com.splwg.ccb.domain.collection.caseType.specialisedCollections.As-	com.splwg.ccb.domain.collection.caseType.specialisedCollections.As-	If Repossession Reason = "Voluntary Repossession" transition to 'Repossession In Progress - Voluntary Surrender' else transition to 'Repossession in Progress"	If Repossession Reason = "Voluntary Repossession" transition to 'Repossession In Progress - Voluntary Surrender' else transition to 'Repossession in Progress"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	otdecidesinwhichstat-	(String actionSourceId); void setActionSourceStatusCode (String actionSourceStat-	setRepo.RepossessionTransition	setRepo.RepossessionTransition_Impl	tary Repossession" transition to 'Repos-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to be made based on	usCd); void setActionId (String actionId); void setActionType (ActionT-			Surrender' else transition to 'Repossessio-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ype actionType); void setResultType (ResultType resultType);			n in Progress" Code - C1-RSTU-PC-MP-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();			L	
ToDoTypeToDoPostProcessAlgorithmSpot		void setOldToDoEntryDTO (ToDoEntry_DTO	com.splwg.ccb.domain.collection.caseType.spe-	com.splwg.ccb.domain.collection.caseType.spe-	Create	Create Notification Notification: <Task Id> - <Task Name> complete for <Collateral

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		oldDTO); void setNewToDoEntry (ToDoEntry newToDoEntry);	cialisedCollections.AssetRepo.NotifyOnTaskCompletion	cialisedCollections.AssetRepo.NotifyOnTaskCompletion_Impl	Notification Notification:	Code> <Collateral Description>. Set Display Date of the case to current business date. Notification should be created on the case associated to the task.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					<Task Id> - <Task Nam-	This algorithm can be attached to any case level task on the Repossession case to alert the repossession specialist.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					r <Collateral Code> <C-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Set Display Date of the case to	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					current business date. Notific-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ation should be created on the cas-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					iated to the task. This algorithm can	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					be attached to any case level task on	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					se to alert the repossession specialist.	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code-C1-NOTRST-SK	
CaseTypeAutoTransitionAlgorithmSpot	This	void setCase(ToDoCase	com.splwg.cb.domain.-	com.splwg.cb.domain.-	When the	When the outstanding amount of all the associated accounts becomes zero

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	algorithm type is us-	toDoCase); Bool getShouldAutoTransition(); CaseStatus getNex-	collection.caseType.specialisedCollections.AssetRepo.MonitorForRedemptionProc	collection.caseType.specialisedCollections.AssetRepo.MonitorForRedemptionProc_Impl	outstanding amount of all the associ-	move the case to Closed Status.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	perform auto transit-	tCaseStatus(); String getNextTransCondition();			unts becomes zero move the case to Clo-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Code - C1-REDE-PROC	
CaseTypeAutoTransitionAl-	T-	void setCase	com.s-plwg.c-	com.s-plwg.c-	Wh-	When Repo Title Received Date and Vehicle at

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
gorithmSpot	his algorithm type is	(ToDoCase toDoCase); Bool getShouldAutoTransition (); CaseSta-	cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-s.As-setRepo.Mon-itorForLiquidationSetUpComplete	cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-s.As-setRepo.Mon-itorForLiquidationSetUpComplete_	en Repo Titl-e Re-cei-ved Dat-e an-d	Sale Location Date is available the case is moved to the next status.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	s-ed to per-for-m a-ut-o tr-	tus getNex-tCaseSt-atus(); String getNex-tTransC-ondition ();			hicl-e at Sal-e Loc-atio-n Dat-e is ava-ilab-le the	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	a Case.				Code-C1-LIQ-SE-TC-MP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus(CaseStatus	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-s.As-	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-s.As-	Create a Task of	Create a Task of given Task Type and assign it to the queue code specified in the parameter. Additionally assign the task to the vendor defined against the service type for the case. If the vendor is

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-	setRepo.AutoTaskCreationForVendor	setRepo.AutoTaskCreationForVendor_Impl	given Task Type	not allocated to the Queue code or if there is no vendor assigned to the service type in the case give error message "Task cannot be allocated for service type: <Service Type>. Please contact

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	in-ess log-ic w-h-en C-a-se	tTransC-ondition ()			the qu-eu-e cod-e spe-	system admin-istrator." Case Transition will be rolled back in this case.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Additionally assign the	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					task to the vendor defined	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					for the case. If the vendor is	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					not allocated to the Queue	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					is no vendor assigned to the ser-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					"Task cannot be allocated for ser-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					vice type: <Service Type>. Please	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Case Transition will be rolled back in this	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					case. Code-C1-AUTO-TASK-C	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithm spp-	void setActionEntity(String actionEntity); void setActionSourceId	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.ValidateRepoCaseData	com.splwg.ccb.domain.collection.caseType.specialisedCollections.As-	Validate if the Dynamic Panel Data Elements and Case Characteristics mentioned in the parameters have some values for the case. If yes the Follow Up is saved successfully and	Validate if the Dynamic Pa-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	otdecidesinwhichstat-	(String actionSourceId); void setActionSourceStatusCode (String actionSourceStat-		setRepo.ValidateRepoCaseData_Impl	nel Data Elements and	case is transitioned to the previous case status. If no system should throw an error message for the first blank field that it will encounter. Error Message: "<Field Name>

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to be made based on	usCd); void setActionId (String actionId); void setActionType (ActionT-			oned in the parameters hav-	cannot be blank"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ype actionType); void setResultType (ResultType resultType);			If yes the FollowUp is sav-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();			ed successfully and case is tra-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ase status. If no system should throw	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					an error message for the first blank fiel-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Error Message: "<FieldName> can-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					not be blank" Code - C1-VALD-ATAP-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					R	
CaseTypeAutoTransitionAlgorithmSpot	This algorithmty-	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	When the redemption clear date is	When the redemption clear date is reached transition the case to the Liquidation Setup Status.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-e is u-s-e-d to p-er-fo-r-	(); CaseStatus getNextCaseStatus(); String getNextTransCondition	s.AssetRepo.MonitorRedemptionClearDate	s.AssetRepo.MonitorRedemptionClearDate_Impl	reached transition the case to the Liquidatio-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-	();			Code-C1-RE-DC-LR-DT	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Set CII = X based on Chapter ent-	Set CII = X based on Chapter entered in Filing Information for all customers associated to the case.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.bankruptcy.Metro2ConsumerInformationIndicator	s.bankruptcy.Metro2ConsumerInformationIndicator_Impl	ered in Filing Information for all customers	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()			Code - C1-CONI-NF-OI-N	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	If any associated secured acc-	If any associated secured account without confirmed plan on it report CII = Q Else Report CII = G for Chapter 12 Report CII = H for Chapter 13.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.bankruptcy.Metro2ConsumerInfoIndiChap13PostDis	s.bankruptcy.Metro2ConsumerInfoIndiChap13PostDis_Impl	ount without confirmed plan on it	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()			Else Report CII = G for Chapter 12	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Report CII = H for Chapter 13. Code	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					- C1-CII-PS-TDI-S	
CaseTypeEnter-StatusAlgorithmSpot		void setCase (ToDoCase toDoCase);	com.splwg.cb.domain.collection.tasks.algo.Auto-	com.splwg.cb.domain.collection.tasks.algo.Auto-	If case	If case level task create a task on the case id. If account level task create a task each on all the accounts

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The purpose	void setCaseOriginalStatus (CaseStatus caseStatus); Bool getShouldAutoTransition	maticTaskCreatiomn	maticTaskCreatiomn_Impl	level task create a task on	associated on the case. If customer level task create a task each on all the customers associated on the case.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	f th- e al- g- or- it- h-	(); String getNex- tCaseSt- atus(); String getNex- tTransC- ondition ();			id. If acc- ou- nt lev- el tas- k cre-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	execute the bu-				ed on the case. If customer level task	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	fic st- at- u- s. T- h- e s- p-				Ca- se Ty- pe – Ent- er Sta- tus - C1-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e-cific s-a-m-ple al-g-or-it-				CR-EA-TT-AS-K	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	testToDoentryand					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ristic					
CaseTypeExitStatusValidationAlgorithmSpot	The purpo-	void setCase (ToDoCase toDoCase); void setPreviousCaseStatus	com.s-plwg.c-cb.-domain.-col-lection.tasks.algo.ValidateTaskCompletion	com.s-plwg.c-cb.-domain.-col-lection.tasks.algo.ValidateTaskCompletion_Impl	Validate if giv-	Validate if given tasks have been completed before exiting the status. For case level tasks check if any open tasks on the case id.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	se of the algorithms-	(CaseStatus case-Status);			en tasks have been co-	For account level tasks check if any open tasks on the accounts associated with the case. For customer level tasks check if any open tasks on the customers associated with the case.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	execute businesse				exiting the status. For case lev	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	a specific status.				el tasks check if any open	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					case id. For account level	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					tasks check if any open tasks	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					sociated with the case. For custom-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					er lev-el tas-ks che-ck if any op-en tas-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					customers associated with the case.	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Case Type – Exit Validation C1-VALT-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					SKEX	
CaseTypeEnterStatusValidationAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase); void setOriginalCaseStatus	com.splwg.cb.domain.collection.tasks.algo.ValidateTaskCompletionClosure	com.splwg.cb.domain.collection.tasks.algo.ValidateTaskCompletionClosure_Impl	Validate if given tasks	Validate if given tasks have been completed before entering the status For case level tasks check if any open tasks on the case id. For account level tasks check

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithms spp-	(CaseStatus caseOriginalStatus);			have been completed before	if any open tasks on the accounts associated with the case.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	cute the validation lo-				For case level tasks check if	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	us.				any open tasks on the case id. For	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					account level tasks check if any op-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					asks on the accounts associated with	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ALTASKCM	
CaseTypeEnterStatusAlgorithmSpot	Th-	void setCase (ToDoCase toDoCase); void setCase-	com.s- plwg.c- cb.- domain.- col- lec- tion.case- Type.spe- cial-	com.s- plwg.c- cb.- domain.- col- lec- tion.case- Type.spe- cial-	Set the DPD and Out-	On creation of a case the algorithm will Set DPD and Outstanding amount to all associated accounts

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e p- ur- p- o- se of th-	Ori- gin- alStatus (CaseSt- atus case- Status); Bool getShoul- dAutoTr- ansition (); String	isedCol- lec- tion- s.bank- ruptcy.SetDPDOutstandingAmount	isedCol- lec- tion- s.bank- ruptcy.SetDPDOutstandingAmount_ Impl	sta- ndi- ng am- ou- nt to all ass- oci- ate- d acc-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lgorithmspot is	getNextCaseStatus(); String getNextTransCondition();			he status_EnterStatus-C1-SE-TD-PD	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	sslogicwhenCa-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase (ToDoCase toDoCase); void setCaseOriginalStatus	com.s-plwg.c-cb.-domain.-collection.caseType.earlyCollections.CreditGrantorCannotLocateConsumer	com.s-plwg.c-cb.-domain.-collection.caseType.earlyCollections.CreditGrantorCannotLocateConsumer_	Automatically set for all borrowers	Automatically set for all borrowers the account the CII Code in skip tracing status on entering a case status

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ose of the alg-	(CaseStatus caseStatus); Bool getShouldAutoTransition(); String getNextCaseStatus();		Impl	the account the CII Code in skip tra-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot is to e-x-e-c-ut-	String getNex-tTransC-ondition ();			tus Ent-er Pro-ces-sin-g. C1-CN-LR-EM	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into sp-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specifics-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	amplealgorithmcre-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	yandlinksthe Caseto					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithms	void setActionEntity(String actionEntity); void setActionSourceId	com.splwg.cb.domain.collection.caseType.earlyCollections.ConsumerNowLocated	com.splwg.cb.domain.collection.caseType.earlyCollections.ConsumerNowLocated_Impl	This algorithm will set the given	This algorithm will set the given CII Code for the party id provided as result characteristics

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot d-e-ci-d-es in w-hi-ch st-	(String actionSourceId); void setActionSourceStatusCode (String actionSourceStatusCd);			CII Code for the party id provided as	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	has to be made base-	void setActionId (String actionId); void setActionType (ActionType actionType);			Result type post processing algo	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setResultType(ResultType resultType); boolean getIsProcessingComplete();			C1-CG-CL-C	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpo-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-s.As-	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCol-lec-tion-s.As-	Metro 2 Reporting – Account Status Co-	This algorithm is used for Metro 2 Reporting – Account Status Code

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	se of th- e al- g- or- it-	case- Status) Bool getShoul- dAutoTr- ansition () String getNex- tCaseSt- atus() String getNex-	setRepo.al- gorithms.Metro2AcctStatuscodeEn- terProcessingAlgo	setRepo.al- gorithms.Metro2AcctStatuscodeEn- terProcessingAlgo_Impl	de C1- AS- CR- EP- O	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to execute the	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ospecific status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The specific sam-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lealgorithmcreatesT-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ksthe Case to it as FKCh-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithms	void setActionEntity(String actionEntity); void setActionSour-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Metro 2 Reporting – Account Sta-	This algorithm is used for Metro 2 Reporting – Account Status Code post Liquidation

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot d-e-ci-d-es in w-hi-ch st-	ceId (String actionSourceId); void setActionSourceStatusCode (String actionSo-	s.AssetRepo.algorithms.Metro2AcctStatusCodePostLiquidationPostProcessing	s.AssetRepo.algorithms.Metro2AcctStatusCodePostLiquidationPostProcessing_Impl	tus Code post Liquidatio-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	has to be made base-	urceStatusCd); void setActionId (String actionId); void setActionType			C1-AS-CLI-QUA	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(ActionType actionType); void setResultType (ResultType resultType);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();				
ResultTypePostProcessingAlgorithmSpot	This Al-	void setActionEntity(String	com.splwg.cb.domain.collection.case-	com.splwg.cb.domain.collection.case-	Metro 2 Reporting	This algorithm is used for Metro 2 Reporting - Compliance condition code

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	g-orith-m s-pot d-e-ci-	actionEntity); void setActionSourceId (String actionSourceId); void	Type.earlyCollection-s.Metro2ComplianceCodePostProcessingAlgo	Type.earlyCollection-s.Metro2ComplianceCodePostProcessingAlgo_Impl	-Compliance condition cod-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	which status transition	setActionSourceStatusCode (String actionSourceStatusCd); void setActionId			C1-COMC-ODE	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ing of result.	(String actionId); void setActionType (ActionType actionType); void				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary

		setResultType(ResultType resultType);
		
		boolean getIsProcessingComplete();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition();	com.splwg.cb.domain.collection.caseType.earlyCollections.UpdateDisputeMonitor	com.splwg.cb.domain.collection.caseType.earlyCollections.UpdateDisputeMonitor_Impl	Monitoring Algo For Dispute Resol-	This algorithm is a Monitoring Algo For Dispute Resolved.Used for updating DisputeFlag to 'N'

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is used to perform a-	CaseStatus getNextCaseStatus(); String getNextTransCondition();			ved C1-DI-SM-ON	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ssing for a Case.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithm sp-	void setActionEntity(String actionEntity); void setActionSourceId	com.splwg.ccb.domain.collection.caseType.earlyCollections.CaseCreationonFollowupPostProcessingAlgo	com.splwg.ccb.domain.collection.caseType.earlyCollections.CaseCreationonFollowupPostProcessingAlgo_Impl	Create Required Case on Follow Up Result	Create Required Case on Follow Up If Account Level Case Type creates case on account, If Customer level Case Type creates case on the main customer of the account.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	otdecidesinwhichstat-	(String actionSourceId); void setActionSourceStatusCode (String actionSourceStat-			Post processing Algorithm C1-CRET-CS-	Queue to which the case should be allocated if provided else the case should remain unallocated with Re-Allocation Switch as Y

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to be made based on	usCd); void setActionId (String actionId); void setActionType (ActionT-			FL	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ype actionType); void setResultType (ResultType resultType);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();				
ResultTypePostProcessingAlgorithmSpot	This Alg-	void setActionEntity(String actionEntity);	com.splwg.cb.domain.collection.caseType.earlyCol-	com.splwg.cb.domain.collection.caseType.earlyCol-	Generic Result Po-	Generic Result

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithmspotdecid-	void setActionSourceId (String actionSourceId); void setActionSour-	lection-s.CaseTransition-andTraskCreationPostProcessingAlgo	lection-s.CaseTransition-andTraskCreationPostProcessingAlgo_Impl	st Processing Algorithm for Case Tra-	Post Processing Algorithm for Case Transition and Task Creation Transition the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ich status transition h-	ceStatusCode (String actionSourceStatusCd); void setActionId (String actionI-			Creation Result Post processing Algorit-	case to given Case Status if Case Status is configured and the current status is present in one of the Valid Cur-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of result.	d); void setActionType (ActionType actionType); void setResultType			hm C1-CTRANT-CRE	Display an error 'The selected result <Result Type> is not allowed in current Status.' If the current status is not present in

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(ResultType resultType); boolean getIsProcessingComplete();				• Create Task of given Task Type and assign it to the give Task Queue if Task Type is configured.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						• Map the created task with the Follow up Id of the Follow Up that created the task. • Set Re-Allocation Switch = Y if Re-Allocate =

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Y • Copy the common characteristics of result into the case. (here the char codes need to be maintained at both the result

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						type and case type level) • Task Creation Logic: If Task For = 'Account' Create Task on the primary associated account

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						on the case If Task For = Customer Create Task on the primary associated Customer of the case If Task For = Case Create Task on

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						the case If Task For = Admin Create Admin level Task Note: Task For is the mandatory characteristic at Task Level. Task For = Cus-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						tomer is an invalid configuration for Account level Case and vice versa. Possible Values of Re-Allocate Switch and Copy Characteristics

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						to Case are : Y/N "Event Name" and "Action Flag" fields are introduced to update Cease_Desist\Contact_Alternate\Dispute Flags, where:-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						"Event Name" will be provided depending on the FLAG which you need to update. So, it can have one of the values:- Event Name :- "CEASE_

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						DESIST" Event Name :- "CONTACT_ ALT" Event Name :- "DISPUTE" And "Action Flag" value will be SET\RESET. To set Cease_

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Desist\Contact_Alternate\Dispute Flags to "Y", provide Action Flag :- "SET". To set Cease_Desist\Contact_Alternate\Dispute Flags "N", provide Action Flag :- "RESET".

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.earlyCollections.CopyCharacteristicsOnCaseCreate	com.splwg.cb.domain.collection.caseType.earlyCollections.CopyCharacteristicsOnCaseCreate_Impl	Copy Case Characteristics Algorithm Ca-	Copy Characteristics Algorithm to copy the Characteristics of recently closed case of a particular Case Category to newly created Case of the same Case Category, when "CONTACT_

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex			seTypeEnterStatusAlgorithmC1-COPY-	ALT_SW" in CI_ACCT_EXTN table is set to "Y".

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwheneCase	tTransCondition()			CHAR	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithmty-	void setCase (ToDoCase toDoCase); Bool getShouldAutoTr-	com.splwg.ccb.domain.collection.caseType.earlyCollections.DetermineContactIntensity	com.splwg.ccb.domain.collection.caseType.earlyCollections.DetermineContactIntensity_Impl	Determine Contact Intensity an-	"Determine Contact Intensity and Contact Intensity Review Date • If case is not on Hold • And Business Date >= Contact Intensity Review

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pe is used to perfor-	ansition (); CaseStatus getNextCaseStatus(); String getNextTransC-			d Contact Intensity Review Date Ca-	Date or Contact Intensity Review Date is Blank • Call Rule Specified in the parameter • Set Contact Intensity and Contact Intensity Review Date Validation Date Can be

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-	ondition ();			se Type Auto Transition Algo C1-EC-DC-	POSTINGDATE or SYSTEMDATE"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					I	
ResultTypePostProcessingAlgorithmSpot	This Algorithms	void setActionEntity(String actionEntity); void setActionSour-	com.splwg.cb.domain.collection.caseType.earlyCollections.HoldCasePostProcessingAlgo	com.splwg.cb.domain.collection.caseType.earlyCollections.HoldCasePostProcessingAlgo_Impl	Hold Case for Da-	Hold Case for Days as provided in Characteristic Type provided in Hold Period or if that is blank Hold Period should be referred from Hold Period parameter.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot d-e-ci-d-es in w-hi-ch st-	ceId (String actionSo-urceId); void setAc-tionSour-ceStatus-Code (String actionSo-			ys as pro-vid-ed in Ch-ara-	And Hold Reason should be set as provided in char-acteristic type provided in Hold Reason or if that is blank Hold Reason should be referred from Hold Reason parameter. Validation Date

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	has to be made base-	urceStatusCd); void setActionId (String actionId); void setActionType			rovided in Hold Period or if	Can be POSTINGDATE or SYSTEMDATE

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(ActionType actionType); void setResultType (ResultType resultType);			Period parameter. And Hold	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();			Reason should be set as provid-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					characteristic type provided	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Reason should be referred from Hol-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Result Type Post Processing Alg-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					o C1-HOLD-CASE	
CaseTypeAutoTransitionAlgorithmSpot	This alg-	void setCase (ToDoCase toDoCas-	com.s-plwg.c-cb.-domain.-col-lec-tion.case-	com.s-plwg.c-cb.-domain.-col-lec-tion.case-	Tra-nsit-ion to con-tact stat-	Transition to contact state if First Contact Date has reached

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithm type is used	e); Bool getShouldAutoTransition(); CaseStatus getNextCaseSt-	Type.earlyCollections.InitiateContact	Type.earlyCollections.InitiateContact_Impl	e if First Contact Date has reached an-	If First Contact Date has reached (based on the parameters below) Or Account is Direct Debit and

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	r-ma-uto tra-n-sitio-n pro-	atus(); String getNextTransCondition();			d set the Re All-oca-tion Sw-itch Ca-se Ty-pe Aut-o	Immediate Transition if Direct Debit = Yes/No Transition to Contact RM status if Relationship Manager exists and Contact RM status has been specified

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Transition Algo C1-EC-IC	Transition to Contact Alternate status if Contact Alternate Flag = Y and Contact Alternate Status has been specified Else

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Transition to Contact Status Set Re-Allocation Switch = Y for the case post case transition Possible Values

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						First Contact Calculation Parameter: DPD, DIA, Days Since Case Start Immediate Transition if Direct Debit: Y,N Validation Date : POSTINGDATE,

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						SYSTEMDATE
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.earlyCollections.InitiateSkipTracing	com.splwg.cb.domain.collection.caseType.earlyCollections.InitiateSkipTracing_Impl	Transition to skip tacing status	If no contact points exists then move the case to Skip Tracing status Check if one of the Contact Points as specified in the parameters exists for

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			if no telephone number exists for	any of the account holder. If no contact point exists than move the case to Skip Tracing Status. Set Re-Allocation Switch = Y for the case post case transition.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()			CaseType-EnterStatusAlgoC1-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ECLSTNTN	
CaseTypeAutoTransitionAlgorithmSpot	This algorit-	void setCase (ToDoCase toDoCase);	com.splwg.cb.domain.collection.caseType.earlyCollec-	com.splwg.cb.domain.collection.caseType.earlyCollec-	Transition to skip taci-	"Transition to skip review if 'X' number of consecutive failed contacts • If last X number of consecutive

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	h- ty- p- e is u- s- e- d to p-	Bool getShoul- dAutoTr- ansition (); CaseSta- tus getNex- tCaseSt- atus();	tion- s.Ini- tiateSkipTracingInvalidTelNumber	tion- s.Ini- tiateSkipTracingInvalidTelNumber_ Impl	ng stat- us if 'X' nu- mb- er of con- sec- utiv- e	contacts has been unsuc- cessful, trans- ition to Skip Tracing Status. Logic for con- sidering unsuc- cessful contacts: If last X con- secutive con- tacts with given contact methods

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	transitionprocessing	String getNextTransCondition();			Case Type - Auto Transition C1-EC-	have Authentication Status other than 'Green' Set Re-Allocation Switch = Y for the case post case transition Possible Values for Validation Date are POSTINGDATE

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					IST-ITN	and SYSTEMDATE"
CaseTypeAutoTransitionAlgorithmSpot	This algorithm	void setCase(ToDoCase toDoCase); Bool getShoul-	com.splwg.cb.domain.collection.caseType.earlyCollections.ParkSmallBalanceAccounts	com.splwg.cb.domain.collection.caseType.earlyCollections.ParkSmallBalanceAccounts_Impl	Park accounts with small balances to a	Park accounts with small balances to a separate status so that no contacts are initiated for the account .

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	type is used to perfo-	dAutoTransition(); CaseStatus getNextCaseStatus(); String getNex-			separate status Case Type - Auto Transition C1-	If Net Arrear Amount <= Small Balance Threshold And Net Arrear Amount > 0 Then transition to small balance

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ion processing for a C-	tTransCondition ();			EC-PS-BA	status. Net Arrear Amount = (Overdue Amount - Unclear Amount) If Use Overdue Amount = Yes than use Overdue Amount

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						instead of Net Arrear Amount in the calculations. Set Re-Allocation Switch = Y for the case post case transition. Possible Values :

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Use Overdue Amount : Y,N
ResultTypePostProcessingAlgorithmSpot	This Alg-	void setActionEntity(String actionEntity);	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.earlyCol-	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.earlyCol-	Re-su-me Col-lec-tion-	Resume Collections Transition the case to

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithms-pot-deci-d-	void setActionSourceId (String actionSourceId); void setActionSour-	lection-s.ResumeCollectionsPostProcessingAlgo	lection-s.ResumeCollectionsPostProcessingAlgo_Impl	s Transits the case to Contact stat-	Contact RM Status if RM exists and Contact RM status has been configured Contact Alternate Status If Contact Alternate Flag = Y

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ich status transition h-	ceStatusCode (String actionSourceStatusCd); void setActionId (String actionI-			us ResultType Post Processing Alg-	Else Contact Status Set Re-Allocation Switch = Yes if Re-Allocate = Y Re-Allocate can

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of result.	d); void setActionType (ActionType actionType); void setResultType			o C1-RESCOLL	be Y/N

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(ResultType resultType); boolean getIsProcessingComplete();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition();	com.splwg.ccb.domain.collection.caseType.earlyCollections.ResumeContactFromUnderResolution	com.splwg.ccb.domain.collection.caseType.earlyCollections.ResumeContactFromUnderResolution_Impl	ResumeContactFromUnderRe-	Resume Contact From Under Resolution Status • If there is no more active PTP on the account and • If the Net Arrear Amount > 0

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is used to perfoma-	CaseStatus getNextCaseStatus(); String getNextTransCondition();			solution Status Move case to contact stat-	Than transition the case to Contact RM Status if RM exists and Contact RM status has been configured Contact Altern-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ssing for a Case.				us if the Net Arrear Amount is greater	ate Status If Contact Alternate Flag = Y Else Contact Status Set Re-Allocation Switch = Y for the case post case transition

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Case Type - Auto Transition C1-EC-	If Use Overdue Amount = Yes than use Overdue Amount instead of Net Arrear Amount in the calculations. Net Arrear Amount = (Overdue Amount -

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					RC-FU-R	Unclear Amount) Use Overdue Amount can be Y/N or Yes/No
CaseTypeAutoTransitionAlgorithmSpot	This	void setCase	com.splwg.cb.domain.	com.splwg.cb.domain.	This algorithm	This algorithm is

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	s algorithm type is u-	(ToDoCase toDoCase); Bool getShouldAutoTransition (); CaseSta-	collection.caseType.earlyCollections.ResumeContactfromSmallBalance	collection.caseType.earlyCollections.ResumeContactfromSmallBalance_Impl	is used to resume contact from small balance status. If Net Arrear Amount > Small	used to resume contact from small balance status.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to perform auto transit-	tus getNextCaseStatus(); String getNextTransCondition();			Case Type - Auto Transition C1-EC-RC-	Balance Threshold Then transition the case to Contact RM Status if RM exists and Contact RM status has been configured

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					SB	Contact Alternate Status If Contact Alternate Flag = Y Else Contact Status Set Re-Allocation Switch = Y for the case post

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						case transition If Use Overdue Amount = Yes than use Overdue Amount instead of Net Arrear Amount in the calculations. Net Arrear

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Amount = (Overdue Amount - Unclear Amount) Possible Value: Overdue Amount : Y,N

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition ();	com.splwg.cb.domain.collection.caseType.earlyCollections.ScheduleContact	com.splwg.cb.domain.collection.caseType.earlyCollections.ScheduleContact_Impl	This algorithm will Schedule Contact for	Schedule Contact for the case as per intensity • If case is not on Hold • And Display

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is used to perform a-	CaseStatus getNextCaseStatus(); String getNextTransCondition();			the case as per the given intensity Case Type	Date <= Business Date or Display Date is Blank • Set Display Date = Max((Last Successful Contact Date + Contact Intensity), Business Date)

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ssing for a Case.				- Auto Transition C1-EC-SC	Consider Contact Intensity from Algorithm parameter if specified else picks up Contact Intensity from case level field. Logic for con-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						sidering successful contacts: Last contact with given contact methods that have Authentication Status = Green Validation Date can be

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						POSTINGDATE or SYSTEMDATE
ResultTypePostProcessingAlgorithmSpot	This Algorit-	void setActionEntity(String actionEntity); void	com.splwg.ccb.domain.collection.caseType.earlyCollection-	com.splwg.ccb.domain.collection.caseType.earlyCollection-	This algorithm will transf-	Supervisor Referral Algorithm • If case is present in one of the status's spe-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	hmspotdecidesinw-	setActionSourceId (String actionSourceId); void setActionSourceStatusCode	s.SupervisorReferralPostProcessingAlgo	s.SupervisorReferralPostProcessingAlgo_Impl	er the case to the given status if the cur-	cified in 'Valid Current Status' than Proceed with further actions Else Display an error 'The selected result <Result

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	transitionhastobem-	(String actionSourceStatusCd); void setActionId (String actionId);			present in Valid Current status list Result	Type> is not allowed in current Status.' And don't proceed with further actions. • Transition the case to given Case Status

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setActionType (ActionType actionType); void setResultType (ResultType			Type Post Processing Algo C1-EC-RT-	• Create Task of given Task Type and assign it to the Supervisor Queue (Queue of Task) of the Case Queue • Map the created task with the Follow up Id of the Follow Up that created the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		resultType); boolean getIsProcessingComplete();			S	task. • Set Re-Allocation Switch = Y if Re-Allocate = Y Re-Allocate can be Y/N

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition();	com.splwg.cb.domain.collection.caseType.earlyCollections.TransitionToSuspendedStatus	com.splwg.cb.domain.collection.caseType.earlyCollections.TransitionToSuspendedStatus_Impl	Transition to suspended status if the account	If the Account has one of the Account Risk Indicators specified in the parameter Transition to Suspended status. Create a task if Task Type has

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is used to perform a-	CaseStatus getNextCaseStatus(); String getNextTransCondition();			has one of the warning indicator set Case Type -	been mentioned and assign it to the Specified Queue Set Re-Allocation Switch = Y for the case post case transition. Set Suspend Reason = Risk Indicator

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ssing for a Case.				Auto Transition C1-ECTT- SS	Exit. If either of the financial owners have one of the Party Indicators mentioned in the parameter than transitionto Suspended status. Create a task if Task Type has

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						been mentioned and assign it to the Specified Queue Set Re-Allocation Switch = Y for the case post case transition. Set Suspend Reason = Risk Indicator

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Exit. If there is at least one financial owner with no Risk indicators mentioned in the parameter 'Party Risk Indicators – Contact Alternate' than transition the case to the Contact

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Alternate Status. If case already in Contact Alternate Status don't initiate the transition but perform the other activities. Create a task if Task Type has been mentioned and assign it to the Specified

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Queue Set Re-Allocation Switch = Y for the case post case transition. Set Alternate Contact Flag = Y Set Alternate Contact Reason = Risk Indicator Exit.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition();	com.splwg.cb.domain.collection.caseType.earlyCollections.TransitionToUnderResolutionStatus	com.splwg.cb.domain.collection.caseType.earlyCollections.TransitionToUnderResolutionStatus_Impl	Transition to under resolution status if	Transition to under resolution status if Net Arrear Amount <=0 • Transition the case to Under Resolution Status if Net

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is used to perform a-	CaseStatus getNextCaseStatus(); String getNextTransCondition();			Net Arrear Amount <= 0 Case Type -	Arrear Amount <= 0 or PTP is running on the account. • Set Re-Allocation Switch = Y for the case post case transition Net Arrear Amount = (Overdue Amount -

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ssing for a Case.				Auto Transition C1-ECTT-URS	Unclear Amount) If Use Overdue Amount = Yes than use Overdue Amount instead of Net Arrear Amount in the calculations.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Possible values: Use Overdue Amount: Y,N
CaseTypeAutoTransitionAlgorithmSpot	This algor-	void setCase (ToDoCase toDoCase);	com.splwg.ccb.domain.collection.caseType.earlyCol-	com.splwg.ccb.domain.collection.caseType.earlyCol-	The algorithm will hol-	Check if the contact cap has reached for the case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ithmtypeisusedto	Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus();	lections.ValidateContactCap	lections.ValidateContactCap_Impl	dthecasewhenthecontactcapisreache-	If case is not already on Hold and Display Date <= Business Date And the number of successful contacts linked to the case in last X number of

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to transition process in-	String getNextTransCondition();			Case Type - Auto Transition C1-EC-	days >= Contact Cap Hold the case for Y number of days with the given Hold Reason.. Logic for considering suc-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					VCC	cessful contacts: All contacts with given contact methods that have Authentication Status = Green Possible Values for Validation Date

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						{SYSTEMDATE,POSTINGDATE}
VendorServiceTypeAllocationAlgorithmSpot	This algorithm ty-	void setVendorServiceType (VendorServiceType vendorServiceTy-	com.splwg.cb.domain.collection.caseType.specialisedCollec-	com.splwg.cb.domain.collection.caseType.specialisedCollec-	Legal vendor Allocation - C1-LGLV-	Legal vendor Allocation

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pe is used to perform	pe); void setCase (ToDoCase toDoCase);	tions.legal.LspVendorAllocationAlgorithm	tions.legal.LspVendorAllocationAlgorithm_Impl	NDRAL	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	cation.					
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase (ToDoCase toDoCase) void setCase-	com.splwg.cb.domain.collection.caseType.special-	com.splwg.cb.domain.collection.caseType.special-	This algorithm will val-	This algorithm will validate to ensure the extended hardship expiry date is greater than current hardship expiry date.Poss-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p- o- se of th- e al- g- or- it- h-	Ori- gin- alStatus (CaseSt- atus case- Status) Bool getShoul- dAutoTr- ansition () String	isedCol- lec- tion- s.fin- ancialHard- ship.ValidateExtendedExpiryDate	isedCol- lec- tion- s.fin- ancialHard- ship.ValidateExtendedExpiryDate_ Impl	idat- e to ens- ure the ext- en- de- d har- dsh- ip	ible Values for Hardship Expiry Date Characteristic Type Code : C1-EXPDT and Extended Expiry Date Characteristic Type Code : C1-EXTEX

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	s to e- x- e- c- ut- e th- e b- u-	getNextCaseStatus() String getNextTransCondition()			tha- n cur- ren- t har- dsh- ip exp- iry dat- e - C1-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	specific status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus(CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	This algorithm will invoke the hostservic-	This algorithm will invoke the host service to extend the hardship expiry date.Possible Values for Extended Expiry Date Char Type : C1-EXPDT and Exception Transition Condition : EXP

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e algorithm spot is	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-	ancialHardship.ExtendExpiryDate	ancialHardship.ExtendExpiryDate_Impl	e to extend the hardship expiry date - C1-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	This algorithm will store Case	This algorithm will store Case Status Update Date/Time for current status into the element specified by xpath in algorithm soft parameter.Possible Values for Excep-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.CaptureEnterStausUpdateDateTime	s.financialHardship.CaptureEnterStausUpdateDateTime_Impl	Status Update Date/Time for current	tion Transition Condition : EXP and Xpath to Date Element : /applicationForm/-main/reliefEffectiveDt

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()			ed by xpath in algorithm soft parameter -	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.PerActContractVCreationAlgo	com.splwg.cb.domain.collection.caseType.PerActContractVCreationAlgo_Impl	This algorithm will retrieve the main Customer and its associated Accounts as well as joint account holders from FLEXCUBE. It will filter out	This algorithm will retrieve the main Customer and its associated Accounts as well as joint account holders from FLEXCUBE. It will filter out

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			e the main Customer and	accounts that are not relevant for Hardship application (based on algorithm soft parameter) and proceed to create the "V" . Possible Values for Feeder Person Id : <Any person id>

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the busi-ness logi-	String getNextTransCondition()			well as joint account holders fro-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					It will filter out accounts that are	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					not relevant for Hardship application	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					nd proceed to create the "V" - C1-RE-TV-FL-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.CreateToDo	com.splwg.cb.domain.collection.caseType.CreateToDo_Impl	This common algorithm creates a	This common algorithm creates a To Do using the values from algorithm parameters. Possible Values for To Do Type,Message Category,Message Num-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			To Do using the values from algorithm par-	ber- ,Characteristic Type For Log Entry and Exception Transition Condition

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.roso.CheckCustomerEligiblity	com.splwg.cb.domain.collection.caseType.specialisedCollections.roso.CheckCustomerEligiblity_	Check customer eligibility - C1-CHKC-	This process will check warning indicators for a customer. This

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()		Impl	USST	check will be done by a call to rule engine for each customer. Processing logic will be as below

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				Primary entity for the case is account. Based on ownership type parameter

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						for the process, system should consider the customers for eligibility check • If ownership type parameter

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						is set to "financial owner" o Get all financially responsible customers for the account

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						o For each customer, system should call the rule engine to check for customer eligibility • If ownership

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						type parameter is set to "primary" o System should call the rule engine to check for primary customers eli-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						gibility Customers' facts should be used for rule engine decision. For each call

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						• Rule will return output as "Validation Status". Possible values can be "Success" OR "Failure" • If validation

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						status = Failed, process should return result as validation failed. o Check Validation failure option § Validation failure

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						option = Fail case creation/transition. Case should not get created or should not transition status. § Val-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						idation failure option = Transition status. Case status should be transitioned to the specified status. Set given char value for

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						the given char type (as defined in parameters). • If validation status = Success, process should return result as validation successful.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Parameters - • Ownership Type - Ownership type can be FINANCIAL_OWNER(Financial Owner) or PRIMARY (Primary Owner). • Rule ID -

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Defined rule id to check customer eligibility. Rule should return output validation status in fact 'SuccessOrFailure', which can have value true or

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						false. • Validation Failure Option - This option is use to determine action to be taken in case of

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						validation failure. Permissible values are FAIL_CASE_CREATION(fail case creation) and TRANSITION_STATUS(transition status).

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						• Validation Failure Transition Status - Case transition status in case of validation failure. • Cancel Reason Char Type - Characteristic type to set as case characteristic if validation failure

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						option is transition status. • Cancel Reason Char Value - Characteristic value for the defined characteristic type.
CaseTypeEnterStatusAlgorithmSpot		void setCase (ToDoC-	com.splwg.ccb.domain.-	com.splwg.ccb.-	Check cus-	This process will

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The purpose of the	ase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool	collection.caseType.specialisedCollections.roso.CheckCustomerContact	domain.collection.caseType.specialisedCollections.roso.CheckCustomerContact_Impl	tomer contacts for the account - C1-CHKC-	check number of "successful" contacts with the customer during specified days inclusive of current date. Number of days will be set as parameter for the process. Contact

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lgorithms pot is to ex-	getShouldAutoTransition() String getNextCaseStatus() String getNextTransCondition()				ed with the account and any customer on the account and should belong to one of the five contact class defined as parameters to this process. If number of con-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	gic when Case is moov-					tacts is less than a minimum specified threshold, process should return result as validation failed. This will • Check Validation failure option o Validation failure option = Fail

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						case creation/transition. Case should not get created or should not transition status o Validation failure option = Transition status. Case status should be transitioned to the specified

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						status. Set given char value for the given char type (as defined in parameters) • If validation status = Success, process should return result as validation successful.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	Capture Hardship Approval Date - C1-HA-	This algorithm will store Case Status Update Date/Time for current status into the element specified by xpath in algorithm soft parameter.Possible Values for Xpathto Date Ele-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-	ancialHardship.CaptureHardshipApprovalDate	ancialHardship.CaptureHardshipApprovalDate_Impl	RAP-DT	ment and Exception Transition Condition

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusValidationAlgorithmSpot	The purpose of	void setCase (ToDoCase toDoCase); void setOriginalCaseStatus	com.splwg.ccb.domain.collection.caseCreation.AdapterTest	com.splwg.ccb.domain.collection.caseCreation.AdapterTest_Impl	Algorith that will interface with Rul-	Algorith that will interface with Rule Engine.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the algorithm spot	(CaseStatus caseOriginalStatus);			e Engine - C1-RULE-ADAPT	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	hevalidationlogicb-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Check target account eligibility - C1-	o System should call the rule engine for eligibility check. Output of rule engine will be "Validation Status" o If validation

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.roso.CheckTargetAccountEligiblity	s.roso.CheckTargetAccountEligiblity_Impl	CHKTRGT	status is "Success" , § Set set-off status as "Pending" § Compute maximum amount allowed to Debit = Target account Balance - Minimum residual

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				amount § Clear the values in the "Exclude Target Till Date" and "Exclude Reason" fields, if populated o If validation status is "Fail", § Set set-off

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						status for target account as "Not eligible" § "Exclude Reason" should be set as "Not Eligible" § Get offset days for exclude reason from the look-up § Set "Exclude

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						target till" date for the target account to current business days + offset day. § If no offset days are returned, "Exclude target till" date should not be updated • Once all target

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						accounts have been processed and for this case, if none of the target accounts has set-off status as "Pending". o Case should be created and transitioned to the status specified in parameters.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						o Set given char value for the given char type (as defined in parameters)
CaseTypeEnterStatusAlgorithmSpot	The p-	void setCase (ToDoCase toDoCase) void	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-	Ap-pro-val che-ck for set-	This process will check if approval is required for a set-off trans-action. Approval will be

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	urpose of the algorit-	setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition ()	cialisedCollections.roso.RosoApprovalCheck	cialisedCollections.roso.RosoApprovalCheck_Impl	off transaction - C1-ROSOAPPR	required if • Asset classification = Value set as parameter for the process • Accrual status = Value set as parameter for the process • Sum of Debit Amounts for all target accounts

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	t is to e- x- e- c- ut- e th- e b-	String getNextCaseStatus() String getNextTransCondition()				>= Specified threshold Based on whether approval is required or not, transition the case to a status as set in the parameters of the process.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	into specific status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.s-plwg.c-cb.-domain.-collection.caseType.specialisedCollections.roso.FetchTargetAccounts	com.s-plwg.c-cb.-domain.-collection.caseType.specialisedCollections.roso.FetchTargetAccounts_Impl	Get target accounts - C1-GETT-RGT	This algorithm gets all savings accounts and term deposit accounts having the same set of owners as owners of the delin-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e algorithms spot is	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-				quent account.Processing logic for this will be as below • Get all savings accounts and term deposit

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inessl ogic when Case	tTransCondition()				accounts having the same set of owners as owners of the delinquent account (i.e. primary account associated with

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						the case). Ownership types can however be different. • "Same owners" - indicates that all owners of delinquent accounts are

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						one the savings account / term deposit and there is no additional owner. • If no such accounts are found, Case should be cre-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						ated and transitioned to the status specified in parameters. Set given char value for the given char type (as defined in parameters).

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Parameters - • Validation Failure Transition Status - Case transition status in case of validation failure. • Cancel

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Reason Char Type - Characteristic type to set as case characteristic if validation failure option is transition status. • Cancel Reason Char

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Value - Characteristic value for the defined characteristic type. • Casa Account Type Identifier List - Comma separated savings account(CASA)

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						identifiers. • Td Account Type Identifier List - Comma separated term Deposit account (TD) identifiers. • Casa Account Exclude Status List - Comma separated savings

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						account(CASA) status to be excluded while fetching account data from host. • Td Account Exclude Status List - Comma separated term Deposit account (TD) status to be

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						excluded while fetching account data from host. • Exclude Blocked Td Account - Flag to exclude blocked Term Deposit account (Y or N). • Exclude Blocked Deposit

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						- Flag to exclude blocked Deposit (Y or N).
CaseTypeEnterStatusAlgorithmSpot	The purp-	void setCase (ToDoCase toDoCase) void setCaseOri-	com.splwg.ccb.domain.collection.caseType.specialisedCol-	com.splwg.ccb.domain.collection.caseType.specialisedCol-	Update status of relief to	Update status of relief to Expired in Hardship details table. Possible values for Hardship Expire Status.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	o-se of th-e al-g-or-it-h-m	gin-alStatus (CaseSt-atus case-Status) Bool getShoul-dAutoTr-ansition () String getNex-	lec-tion-s.fin-ancialHard-ship.UpdateHardshipStatusToExpire	lec-tion-s.fin-ancialHard-ship.Up-dateHardshipStatusToExpire_Impl	Ex-pir-ed in Har-dsh-ip - C1-UP-DH-DS-TA-T	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	oexecutethebusi-	tCaseStatus() String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ific status.					
CaseTypeEnterStatusAlgorithmSpot	The	void setCase (ToDoCase toDoCase)	com.splwg.cb.domain.collection.batch.al-	com.splwg.cb.domain.collection.batch.al-	CSAMCasent-	Cross Strategy Action Matrix Algorithm Type is used by Strategy Monitor and case asso-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	purpose of the algor-	void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition	gorithm.CSAMCaseEnterStatusAlgorithm	gorithm.CSAMCaseEnterStatusAlgorithm_Impl	er status algorithm type - C1-CSAM-ENTS-	ciation process in order to take actions on existing strategies and recommended strategies based on CSAM Matrix. This algorithm type is used to attach algorithms to the case enter status of case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spot is to execute th-	() String getNextCaseStatus() String getNextTransCondition()				type. Parameter: Check Status- It checks the status with which the matrix has to be dealt with. Possible values are "Y" or "N"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	moved into specific st-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot	The purpose of	void setCase (ToDoCase toDoCase); void setNextCaseStatus	com.splwg.cb.domain.collection.caseType.GenericToDoCompletionForCase	com.splwg.cb.domain.collection.caseType.GenericToDoCompletionForCase_Impl	To Do Completion for case - C1-T-O-	This common algorithm will complete all To Do's with Drill Keys = Current Case Id and To Do's To Do Type is not excluded from auto completion

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the algorithms spot	(CaseStatus caseStatus);			DO END	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	d-di-ti-o-n-al lo-gi-c w-h-e-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the next status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	Update Marketing Consent flag - C1-MK-	This is a generic algorithm that will make a service call to Host to update the Marketing Consent flag.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-	ancialHardship.UpdateMarketingConsentFlag	ancialHardship.UpdateMarketingConsentFlag_Impl	T-FLG	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Check Default Notice for Voluntary possession	Check Default Notice for Voluntary possession. Possible value for Check Expiry Status.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.AssetRepo.CheckDefaultNoticeForVoluntaryPossession	s.AssetRepo.CheckDefaultNoticeForVoluntaryPossession_Impl	sion - C1-CHKDFLT	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Update Financial Hardship flag - C1-	This algorithm will make a service call to Host to update the Financial Hardships flag for Primary Customer and corresponding joint account holders

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.UpdateFinancialHardshipFlag	s.financialHardship.UpdateFinancialHardshipFlag_Impl	FNHRDFLG	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithms	void setActionEntity(String actionEntity); void setActionSour-	com.splwg.ccb.domain.collection.actionObject.actionType.ResultTypeCaseTransitionAlgo	com.splwg.ccb.domain.collection.actionObject.actionType.ResultTypeCaseTransitionAlgo_Impl	Result Type Case Transition Algorithm	If specified on the Result Type, this algorithm will be invoked when the corresponding result is recorded for a Case (Action/Result UI).

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot d-e-ci-d-es in w-hi-ch st-	ceId (String actionSo-urceId); void setAc-tionSour-ceStatus-Code (String actionSo-			- C1-RT-CT	This can be used to transition the case from the current status to the next possible status as follows, - This algorithm has a parameter Output Status i.e. next possible

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	has to be made base-	urceStatusCd); void setActionId (String actionId); void setActionType				status, so for case transition, it will be checked whether Output Status is one of the next possible status. If YES, it will transition the case to that status. - This algorithm has a parameter

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(ActionType actionType); void setResultType (ResultType resultType);				Input Status, which will be checked against the current status of the Case. This is an optional parameter. If specified, Case transition will happen only when the current status of the case matches

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();				
CaseTypeExitStatusAlgorithmSpot	The	void setCase (ToDoCase toDoCas-	com.splwg.cb.domain.collection.case-	com.splwg.cb.domain.collection.case-	Algorithm to see if	The algorithm sees if the case is running in the child case category before closing the case from the parent

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	purpose of the algor-	e); void setNextCaseStatus (CaseStatus caseStatus);	Type.specialisedCollections.legal.CheckActiveArsCase	Type.specialisedCollections.legal.CheckActiveArsCase_Impl	case is running before closing - C1-CHKC-	case category

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spot is to perform					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	transitions out of the c-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Check Deceased status for the customer -	For the customer for whom the deceased case is being initiated check if Deceased warning indicator is already set OR An active deceased case is present

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.deceased.CheckDeceasedStatusForCustomer	s.deceased.CheckDeceasedStatusForCustomer_Impl	C1-CHKD-CD	If either of above is true, case creation should fail.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Associated accounts with deceas-	For the primary customer associated with the case Get all accounts where this customer is primary owner and the accounts are in collections Associated those accounts

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm sp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.deceased.AssociatedAccWithDeceasedCustomer	s.deceased.AssociatedAccWithDeceasedCustomer_Impl	ed customer case - C1-DC-DA-CC-TS	with the case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.roso.ExecuteFundTransfer	com.splwg.cb.domain.collection.caseType.specialisedCollections.roso.ExecuteFundTransfer_Impl	Execute Fund Transfer - C1-FUNDTR-	This process will execute the fund transfer. This should follow below steps for "each" target account where

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			FR	debit amount specified is > 0 and set-off status = "Pending" • Execute a payment transfer transaction from Target account to the delinquent

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				account. • If transaction is successful, set set-off status = "Success" for this target account • If transaction is not successful, set set-off status

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						= "Fail" for this target account For any target account where set-off status was "Pending", but were not considered for set-off (because

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						debit amount was specified as zero) update set-off status to "Cancelled" Once all target accounts have been processed, check if at least one payment

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						transfer has status as "Success". • If yes, transition the case to status as set in the parameter "Execution Success Status" • If no, transition the case to

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						status as set in the parameter "Execution Failure Status". Set the char value for the char type as specified in the parameters. Parameters -

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Execution Success Status - Case trasition status if fund tranfer is successful. Execution Failure Status - Case trasition status if fund tranfer fails. Cancel Reason Char Type -

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Characteristic type to set as case characteristic if fund transfer fails. Cancel Reason Char Value - Characteristic value for the defined characteristic type. Successful Fund Transfer Trans-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						action Status - Trasaction status code to identify successful fund transfer.This values is returned from host service.
CaseTypeEnterStatusAlgorithmSpot		void setCase (ToDoC-	com.splwg.ccb.domain.-	com.splwg.ccb.domain.-	Algorithm	Algorithm to save the case status in CI_

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	The purpose of the	ase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool	collection.caseType.specialisedCollections.legal.SavePrevioustStatus	collection.caseType.specialisedCollections.legal.SavePrevioustStatus_Impl	to save previous state's status code - C1-	LSP_DTLS table from where it has come to the current status. This algorithm is must when we are using C1-RESSTATUS. C1-RESSTATUS transition the case to the status which is

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	lgorithms-pot is to ex-	getShouldAutoTransition() String getNextCaseStatus() String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	gicwhenCaseismov-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
PreprocessBusinessObjectRequestAlgorithmSpot		void setAction(BusinessObjectActionLookup boAction); void setBusi-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Attach case type from feature	Attach case type from feature config attach to BO.Possible value for Hardship Case Type Feature Config : C1_HSCATY_FC

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObject(BusinessObject bo); void setRequest(BusinessObjectInstance boRequest);	s.financialHardship.AssignCaseTypeFromFeatureConfig	s.financialHardship.AssignCaseTypeFromFeatureConfig_Impl	config attach to BO-C1-ATCH-CS	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	Update Collection Warning Indicator - C1-	This is a generic algorithm that will make a service call to the Host to update Party level warning indicators for the Main Customer. It has following parameters: 1. Warning Indicator Type.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-	ancialHardship.UpdateCollectionPartyWarningIndicator	ancialHardship.UpdateCollectionPartyWarningIndicator_Impl	UPDWRIND	2. Warning Indicator Value 3. Rule Type Code 4. Collection Column To Be Updated 5. Set In Collections On Related Accounts 6. Exception Transition Condi-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				tion

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithms	void setActionEntity(String actionEntity); void setActionSour-	com.splwg.cb.domain.collection.interaction.CreateCustomerContactAlgo	com.splwg.cb.domain.collection.interaction.CreateCustomerContactAlgo_Impl	Create Customer Contact for Resultype Algo.Possible values for Customer Contact Class,Customer Contact Type and Preferred Contact Method	Create Customer Contact for Re-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot d-e-ci-d-es in w-hi-ch st-	ceId (String actionSourceId); void setActionSourceStatusCode (String actionSo-			sul-typ-e Algo - C1-CR-EA-TC-C	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	has to be made base-	urceStatusCd); void setActionId (String actionId); void setActionType				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(ActionType actionType); void setResultType (ResultType resultType);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();				
CaseTypeExitStatusAlgorithmSpot	The	void setCase (ToDoCase toDoCas-	com.splwg.ccb.domain.collection.case-	com.splwg.ccb.domain.collection.case-	Initiate LMI Process -	Parameters to the algorithm must be as follows: • For Initiate LMI Options:

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	purpose of the algor-	e); void setNextCaseStatus (CaseStatus caseStatus);	Type.special-isedCollections.AssetRepo.InitiateLMIP	Type.special-isedCollections.AssetRepo.InitiateLMIP_impl	C1-INITL-MI-P	1) "Initiate LMI with highest insured amount" use HIGH_INSUR_AMT 2) "Initiate LMI from a specific insurer first" use SPEC_INSURER • For No LMI Option 1)"Mark primary

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spot is to perform					account for strategy review" use PRIMARY 2)"Mark all accounts for strategy review" use ALL 3)"No Action" use NA

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	transitions out of the c-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Hardship Entity Association to no-	This algorithm associates all the accounts nominated for hardship and also associates related financial Owners of the accounts selected.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.HardshipEntityAssociation	s.financialHardship.HardshipEntityAssociation_Impl	minated accounts and financial owner-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Assign Applicable Relief Type-	This algorithm will invoke Rules Engine to determine Applicable Relief Type(s) for each nominated Account.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.AssignApplicableReliefTypes	s.financialHardship.AssignApplicableReliefTypes_Impl	C1-RELIF-TYP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.CalculateCaseStatusExpiryDate	com.splwg.cb.domain.collection.caseType.CalculateCaseStatusExpiryDate_Impl	Calculate an expiry date when ent-	This algorithm type accepts a parameter for a characteristic type which will be used to create a Case Characteristic which contains a date that is equal to case status change plus

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			ering case status - C1-CSEXPDT	Number of Days parameter value

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.TransitionToPreviousStatusPriorToException	com.splwg.cb.domain.collection.caseType.TransitionToPreviousStatusPriorToException_Impl	Transition to previous status prior	This common algorithm will determine the previous status of the Exception status and transition to that state.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithms spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			to Exception- C1-PRE-V-EXP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.CustomerContact	com.splwg.cb.domain.collection.caseType.CustomerContact_Impl	CreateCustomerContact - C1-CUST-	This common algorithm creates a customer contact for the given customer contact type.Possible values are Customer Class,Customer Contact Type,Char Type

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			CONT	Cust Cont Log Entry,X Path Completion Flag,Transition Condition and Contact Method

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the busi-ness logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm ty-	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition ();	com.splwg.cb.domain.collection.caseType.TransitionToNextDaysOnBeforeExpiry	com.splwg.cb.domain.collection.caseType.TransitionToNextDaysOnBeforeExpiry_Impl	Transition to Next Status x days	Transition to Next Status x days before expiry.Possible values are Days Before Expiry,Xpath to Expiry Date,Next Status and Next Transition Condition

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pe is used to perfor-	CaseStatus getNextCaseStatus(); String getNextTransCondition();			before expiry - C1-NXT-B-X-DY	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	ValidateHardshipExpiryDate - C1-VA-	This validates the Hardship Expiry Date.It validates if the expiry date is greater than Posting date and the Allowed Minimum maturity date.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.ValidateHardshipExpiryDate	s.financialHardship.ValidateHardshipExpiryDate_Impl	L-FH-EXP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
LeavePlanAlgorithmSpot		void setEmailId (String emailId); void setSubject (String subject); void	com.splwg.cb.domain.collection.leavePlan.LeavePlanEmailNotification	com.splwg.cb.domain.collection.leavePlan.LeavePlanEmailNotification_Impl	LeavePlan Email Notification - C1-US-	This will be used to notify with email to User and it's supervisor for updation of Leave application details

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setTemplate (String template); void setLeaveId (String leaveId);			RMAILNT	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.UpdateAccountInCollectionFlag	com.splwg.cb.domain.collection.caseType.specialisedCollections.UpdateAccountInCollectionFlag_	Update Account in collections flag -	Get all accounts for the customer from the host. Relationship type to be considered will be primary or financial ownership based on parameter set for the process. For the accounts retrieved, check

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-		Impl	C1-ACTIN-COL	if the account is setup in collections i.e. an active contract is present for the account • If no, set in-collections flag to "N" for the account • If yes. No updates should be done

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection	com.splwg.cb.domain.collection.caseType.specialisedCollection	Associate related entities with the case. Possible values are Customer Association and Account Association	Associate related entities with the cas-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.AssetRepo.CaseAssociationForAssetRepossessionCase	s.AssetRepo.CaseAssociationForAssetRepossessionCase_Impl	e - C1-ARSENTITY	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase); void setNextCaseStatus (CaseSt-	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCollections.AssetRepo.Initiate_LMIProcess	com.s-plwg.c-cb.-domain.-col-lec-tion.case-Type.spe-cial-isedCollections.AssetRepo.Initiate_LMIProcess_Impl	Initiate LMI - C1-INI-TL-MI-S	Initiate LMI. Possible values are No L M I Option,Lmi Insurer Code,Initiate L M I Options,Lmi Case Type and Balance Threshold

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithms-spotis	atus case-Status);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	onallogicwheenaC-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	status.					
CollectionClassOverdueMonitorRuleAlgorithmSpot		void setAccount (Account account); Bool getIsProcess-	com.s-plwg.c-cb.-domain.-col-lec-tion.batch.al-gorith-m.Col-	com.s-plwg.c-cb.-domain.-col-lec-tion.batch.al-gorith-m.Col-	NGP Collection case cre-	This is overdue monitor Rule algorithm used for NGP Collection Case creation. It will be invoked through the over due monitor batch

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ingComplete();	lectionCaseCreationOverdueMonitorRuleAlgo	lectionCaseCreationOverdueMonitorRuleAlgo_Impl	ation algorithm - C1-COLL-CASE	process C1-ADMOV.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	CaseEnterStatusContractStopAlgoComp	CaseEnterStatusContractStopAlgoComp_Impl	StopContractAlgorithm-C1-CONT-ST-	This algorithm will stop the contract linked to case in the CI_CASE_PARTY table.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-			OP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Check for existing Hardship - C1-CHKH-	Before creating case in Pending state,This Algorithm checks,if there is any active case of Hardship case type (By Retrieveing case type code from feature con-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.CheckExistingHardship	s.financialHardship.CheckExistingHardship_Impl	RDSHP	figuration). If yes It Display message ' Party is already in Hardship' If no,It will proceed with case creation. This checks for an existing Hardship application for the party.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CollectionActionAlgorithmSpot		void setPersionID (String personId); void setContactType (String conType);	com.splwg.ccb.domain.collection.contacthistory.ContactProcessing	com.splwg.ccb.domain.collection.contacthistory.ContactProcessing_Impl	Algorithm for contact processing - C1-CN-	Algorithm for contact processing.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setContactID (String contID);			TC-T	
ValidateBusinessObjectAlgorithmSpot		void setMaintenanceObject	com.splwg.ccb.domain.collection.caseType.spe-	com.splwg.ccb.domain.collection.caseType.spe-	check application expiry	check application expiry date with allowed minimum date of nominated account and posting date

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(MaintenanceObject mo); void setBusinessObject (BusinessObject bo);	cialisedCollections.financialHardship.CheckApplicationExpiryDate	cialisedCollections.financialHardship.CheckApplicationExpiryDate_Impl	date - C1-CHKEXP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setEntityId (EntityId id); void setAction(BusinessObjectActionLookup				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boAction); void setBusinessObjectKey (BusinessObjectInstanceKey boKey);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setOriginalBusinessObject (BusinessObjectInstance boRequest); void				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setNewBusinessObject (BusinessObjectInstance boRequest);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.correspondence.CustomerContactCreation	com.splwg.cb.domain.collection.correspondence.CustomerContactCreation_Impl	New Customer Contact Creation Alg-	This AlgorithmType is used to create Customer Contact on the basis of Customer Contact class,Customer Contact Type and Preferred Contact Method on a Customer Level case or an Account Level

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-			orithm-C1-CCCREATE	case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot	The purpose of	void setCase (ToDoCase toDoCase); void setNextCaseStatus	com.splwg.cb.domain.collection.caseType.RemoveCaseCharacteristic	com.splwg.cb.domain.collection.caseType.RemoveCaseCharacteristic_Impl	Removes a case characteristic	This algorithm type removes a case characteristic with char type = parameter 10 value.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the algorithm spot	(CaseStatus caseStatus);			on case status exit - C1-REMCSCH	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	d-di-ti-o-n-al lo-gi-c w-h-e-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the next status.					
CaseTypeAutoTransitionAl-	T-	void setCase	com.splwg.ccb.-	com.splwg.ccb.-	Transit-	This algorithm type transitions

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
gorithmSpot	his algorithm type is	(ToDoCase toDoCase); Bool getShouldAutoTransition (); CaseStatus getNex-	domain.- col- lec- tion.case- Type.TransitionToNextStatusOnDate	domain.- col- lec- tion.case- Type.Trans- itionToNextStatusOnDate_Impl	ion cas- e on a dat- e on a cas- e cha- rac-	the case on a date stored on a case characteristic (char type = parameter 10 value). If the case characteristic is not found, the case will be transitioned on the current date. This algorithm

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	s-ed to per-for-m a-ut-o tr-	tCaseSt-atus(); String getNex-tTransC-ondition ();			c - C1-TR-AN-SD-T	tus or Next Trans-ition Condition to determine the next status

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	a Case.					
CaseTypeEnterStatusAlgorithmSpot	The	void setCase (ToDoCase toDoCase) void	com.splwg.cb.domain.collection.caseType.SetAc	com.splwg.cb.domain.collection.caseType.SetAc	Set Account Nxt Credit	This algorithm sets the accounts next credit review date to current date

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	urpose of the algorit-	setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition ()	countNextCreditReviewDateToCurrentDate	countNextCreditReviewDateToCurrentDate_Impl	Review Date to current date - C1-NXTR-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	t is to e- x- e- c- ut- e th- e b-	String getNextCaseStatus() String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	into specific status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseSta-	com.splwg.ccb.domain.collection.caseType.TransitionDefaultNextStatus	com.splwg.ccb.domain.collection.caseType.TransitionDefaultNextStatus_Impl	Transition to Default Next Status - C1-TR-	This is a common algorithm that will automatically transition the case to the next status. Following are the parameters : 1. Next Status - The next status to which the case will be transitioned.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is used to perform a-	tus getNextCaseStatus(); String getNextTransCondition();			A-N-ST-AT	2. Next Transition Condition - Mention the transition condition for the next status.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ssing for a Case.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	Mark accounts for strategy review - C1-	This algorithm will mark all accounts that are "in-collections" for the customer in hardship for review.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-	ancialHardship.MarkAccountsForStrategyReview	ancialHardship.MarkAccountsForStrategyReview_Impl	REVIWACT	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition();	com.splwg.cb.domain.collection.caseType.WaitTimeOut	com.splwg.cb.domain.collection.caseType.WaitTimeOut_Impl	WaitTimeOut (in days) - C1-WAITDA-	This algorithm times out when the Case has been on the state for too long.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-e is u-s-e-d to p-er-fo-r-	CaseStatus getNextCaseStatus(); String getNextTransCondition();			YS	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Validate Hardship Application inputs	This algorithm validates that all the mandatory fields on the Hardship Application Form are populated.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.ValidateHardshipApplicationInputs	s.financialHardship.ValidateHardshipApplicationInputs_Impl	-C1-V-F-H-APP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Check for Operational Relief Types-	This algorithm checks if any of the identified stp relief types need to be operational

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.CheckForOperationalReliefTypes	s.financialHardship.CheckForOperationalReliefTypes_Impl	C1-O-P-RT	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	AutoApprovalCheck-C1-FH-AU-	This algorithm invokes an Application service which in turn invokes host service which determines if the Hardship application can be auto-approved

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.AutoApprovalCheck	s.financialHardship.AutoApprovalCheck_Impl	TOAP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Apply Hardship Relief Types for acc-	This algorithm applies hardship relief types for the accounts in the host.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.ApplyHardshipReliefTypes	s.financialHardship.ApplyHardshipReliefTypes_Impl	ounts in Host - C1-F-H-EV-AL	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Update Party Warning Indicator - C1-	This is a generic algorithm that will make a service call to Host to update Party level warning indicators for Main Customer If a Rule Type Code is populated, it will first invoke the rule to

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.UpdatePartyWarningIndicator	s.financialHardship.UpdatePartyWarningIndicator_Impl	UPD-PRTW-I	determine if the Warning Indicator should be updated.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithmty-	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition();	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Transition to Next status when all reli-	This is algorithm that will transition the case to the next status when all reliefs have been applied

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pe is used to perfor-	CaseStatus getNextCaseStatus(); String getNextTransCondition();	s.financialHardship.TransitionToNextStatusWhenAllReliefsApplied	s.financialHardship.TransitionToNextStatusWhenAllReliefsApplied_Impl	efs are app - C1-RAPP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
InstallationEntityActivityPopulationAlgorithmSpot		void setEntityType (EntityTypeFlagLookup entityType); void	com.splwg.cb.domain.collection.caseCreation.PopulateAccountActivityAlgo	com.splwg.cb.domain.collection.caseCreation.PopulateAccountActivityAlgo_Impl	Collection Entity Activity Populatio-	This sample algorithm is called from various entities classes for population of Account Activity. The algorithm takes following input parameters:

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setEntityId(String entityId); void setModeOfOperation (String modeOfOperation);			n - C1-ENTACT-POP	1)EntityType : Person/Account for which activity is getting created (e.g. Case can be created on Person as well as Account) 2) EntityId : Person/Account Id 3) ModeOfOperation: Add/Update/De-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setActivityEntitytId(String ActivityEntitytId); void setActivityEntitytType (String				lete/Cancel 4) HostEntitytId: Activity Entity Id (e.g PTP/CC/Follow-up/Case Id) 5)HostEntitytName: PTP/CC/FOLLOWUP/CASE

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ActivityEntitytType); void setActivityEntitytStatus (String ActivityEntitytStatus); void				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setActivityType (ActivityTypeCdLookup activityType); void setActivityDateTime (DateTime				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		activityDateTime); void setMessageFields(String messageFields);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	com.splwg.cb.domain.collection.caseType.specialisedCollections.fin-	Cancel Hardship Application - C1-CXLF-	This algorithm will make a service call to host to cancel an active Hardship Application.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-	ancialHardship.CancelHardshipApplication	ancialHardship.CancelHardshipApplication_Impl	H	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus(CaseSt-	com.splwg.cb.domain.collection.caseType.rightofSetOff.PerformPaymentXferForROSO	com.splwg.cb.domain.collection.caseType.rightofSetOff.PerformPaymentXferForROSO_Impl	Perform Payment Transfer for ROSO	This Algorithm Type will call a web service which calls Oracle NGP Core Banking to perform a payment transfer between an eligible delinquent Account and eligible Target

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	atus caseStatus) Bool getShouldAutoTransition () String getNextCaseStatus() String			- C1-RO-SO-PM-TX-R	Account(s).

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.rightofSetOff.ValidateRosoInputs	com.splwg.cb.domain.collection.caseType.rightofSetOff.ValidateRosoInputs_Impl	Validate ROSO Target Account inputs	This Algorithm Type will validate the user inputs entered into the Target Account dynamic panel to ensure they comply with the business rules. If the inputs are not valid, the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			- C1-R-S-VALIN	Case will transition back to the previous status and prompt the user to re-enter the inputs.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.AdhocCollectionEntityCreation	com.splwg.cb.domain.collection.caseType.AdhocCollectionEntityCreation_Impl	Create RMB Entities from Host Dat-	Create RMB Entities such as Person, Account ,Account Person, PartyCollect etc from Host Data. Input parameters : 0) Source Host Id : Host Iden-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			a - C1-VCREATE	tifier Value e.g. NGP -- Removed in R2.2- Host Id will come from UI b) Inapplicable Statuses : Comma separated Host System Statuses for Account (host_sys_acct_stat_

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the busi-ness logi-	String getNextTransCondition()				flg) c) Exclude Accrual Status Flag: Comma separated Accrual Statuses for Account (accrl_stat_flg) d) Exclude Asset Class Code: Comma separated Asset Class Codes for

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Account (asst_class_cd) e) Exclude User Defined Acct Status: Comma separated User Defined Account Status (usr_def_acct_stat_flg) f) Exclude Offer Id: Comma separated Offer Id (offer_id)

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
PostProcessBusinessObjectAlgorithmSpot		void setMaintenanceObject (MaintenanceObject mo); void setBusi-	com.splwg.ccb.domain.collection.PopulateAccountactivityForNote	com.splwg.ccb.domain.collection.PopulateAccountactivityForNote_Impl	Populate Activity Table For Notes Creatio-	Populate Activity Table For Notes Creation

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObject(BusinessObject bo); void setEntityId (EntityId id); void setAction(Busi-			n - C1-NT-AC-TVI-TY	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObjectActionLookup boAction); void setBusinessObjectKey (BusinessObjectIn-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		stanceKey boKey); void setOriginalBusinessObject (BusinessObjectInstance				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boRequest);				
PreprocessBusinessObjectRequestAlgorithmSpot		void setAction(BusinessObjectActionLookup boAction); void	com.splwg.ccb.domain.collection.suspendActivity.SuspendActivityPreProcessing	com.splwg.ccb.domain.collection.suspendActivity.SuspendActivityPreProcessing_Impl_Impl	SuspendActivityforAccount	Suspend Activity for Account Pre Processing

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setBusinessObject (BusinessObject bo); void setRequest(BusinessObjectInstance boRequ-			PreProcessing - C1-SP-AT-AC-PRE	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		est);				
TreatmentActivityMonitorAlgorithmSpot		void setCaseId (ToDoCase_Id caseId);	com.splwg.ccb.domain.collection.batch.algorithm.TreatmentActivityMonitorAlgoComp	com.splwg.ccb.domain.collection.batch.algorithm.TreatmentActivityMonitorAlgoComp_Impl	SampleTAMAlgorithmType-C1-	This algorithm will update account and TAM review date for case.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					TAMALG	
CaseTypeExitStatusAlgorithmSpot	The purpo-	void setCase (ToDoCase toDoCase); void setNex-	com.splwg.ccb.domain.collection.caseType.specialisedCol-	com.splwg.ccb.domain.collection.caseType.specialisedCol-	Cancel Approval Request -	This algorithm will cancel all pending approval requests for the case. Example for parametervalues for legal Pro-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	seofthealgorithms-	tCaseStatus (CaseStatus caseStatus);	lections.CancelApprovalReqAlgo	lections.CancelApprovalReqAlgo_Impl	C1-CANAPPR	cess: Composite Name:- com.ofss.fc.- workflow.- process.LegalProcessForApproval Instance Title:- LEGAL_CASE_ Value of the above parmeters

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	perform additio-					are depends upon the SOA approval work flow

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the current status to the					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollections.UpdateCustomerInCollFlg	com.splwg.ccb.domain.collection.caseType.specialisedCollections.UpdateCustomerInCollFlg_Impl	Update Customer in collections flag	This algorithm will set or reset in-collections flag for the customer in core banking While setting the Flag • Skip the pro-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			-C1-CUSINCOL	cess if flag is already set • If not, set the in-collection flag for the customer to "Y" in the host. While resetting the Flag, • Skip the process if flag is already reset • System should

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				check for all cases to which the customer is associated (as primary or secondary entity). • If any of the cases for the customer belong to a specific category, then system should not reset the flag.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						List of categories to be checked will be set as parameters to this process. • If none of the cases for the customer are from those categories, set the in-collection

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.UpdateReviewDate	com.splwg.cb.domain.collection.caseType.specialisedCollections.UpdateReviewDate_Impl	Update Review Date for associated	For all accounts associated with the case this process will update the review date. Below parameters should be available for the process • Update Type o Set Review Date - This will

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-			accounts-C1-UPDRVWDT	set the Review Date for the account o Remove Review Date - This will remove the Review date from the account • Days Offset - Applicable only of Update Type = Set. System

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inessl ogic when Case	tTransCondition()				will set the review date as Current business days + Offset days. • Override Flag o Yes - System will update existing account review date i.e. in case a review date is already present, system

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						will override the same o No: System will not update existing account review date i.e. in case a review date is already present, system will not override the same

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.SetDisplayDate	com.splwg.cb.domain.collection.caseType.specialisedCollections.SetDisplayDate_Impl	Set Display Date - C1-SETD-SPDT	This process will update the display date for the account. New display date will be computed as = Current display date + offset days If a display date is already present on the account, it

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-				should be updated only if new display date is < existing display date

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.UpdateWarningIndicator	com.splwg.cb.domain.collection.caseType.specialisedCollections.UpdateWarningIndicator_Impl	Update warning indicator for the cus-	This process will update the warning indicator for the customer • Update activity i.e. set or remove the warning will also be defined as parameter to this process

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			tomer - C1-UPD-WARN	• Warning indicator to be set or removed will be set as parameter to this process • Additionally process will have a parameter to define if update needs to be done only for the customer asso-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the busi-ness logi-	String getNextTransCondition()				ciated as primary entity or for all customers associated to the case Call the service form host to update the warning indicator. Please give following values for the below parameters:

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Association Type : P (Primary) and A (Primary and Secondary) Update Type : S (Set) and R (Remove)
CaseTypeAutoTransitionAlgorithmSpot	This	void setCase (ToDoCase	com.splwg.ccb.domain.col-	com.splwg.ccb.domain.-	Transition to	Transition the case to default next status after specific days.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	algorithm type is us-	toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String	lection.caseType.specialisedCollections.DefaultNextStatusAutoTransition	collection.caseType.specialisedCollections.DefaultNextStatusAutoTransition_Impl	Default next status after N Days - C1-TRND-	Days will be set as parameter for the process. Case should transition to Default next status if, difference in current date and date of entry into current status is >= specified number of days

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	perform auto transit-	getNextTransCondition();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.roso.AccountExclusion	com.splwg.cb.domain.collection.caseType.specialisedCollections.roso.AccountExclusion_Impl	Check current cases on account	System should maintain a lookup with list of case categories for set-off exclusion.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			for exclusion - C1-EXCL-CASE	Processing logic should be as below • Get all active cases for the account. Account can be primary or secondary entity for

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				that case. • Get case categories for all these cases • If the case category for any of the cases is from the exclusion list, validation should fail.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						• Check Validation failure option - Validation failure option = Fail case creation/transition. Case should not get created or should not

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						transition status - Validation failure option = Transition status. Case status should be transitioned to the specified status. Set given

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						char value for the given char type (as defined in parameters) • If the case category for any of the cases is not from the exclusion list, validation is successful and

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						process should move to next step. Parameters - Cancel Reason Char Type: Characteristic type to set as case characteristic if validation failure option is transition status.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Cancel Reason Char Value: Characteristic value for the defined characteristic type. Validation Failure Transition Status: Case transition status in case of validation failure. Validation Fail-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						ure Option: This option is use to determine action to be taken in case of validation failure. Permissible values are FAIL_CASE_CREATION(fail case creation) and

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						TRANSITION_STATUS(transition status).
CaseTypeEnterStatusAlgorithmSpot	The pur-	void setCase (ToDoCase toDoCase) void setCase-	com.splwg.cb.domain.collection.caseType.special-	com.splwg.cb.domain.collection.caseType.special-	Set exclusion date for deli-	This process will set set-off exclusion date for the delinquent account. Pro-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p- o- se of th- e al- g- or- it- h-	Ori- gin- alStatus (CaseSt- atus case- Status) Bool getShoul- dAutoTr- ansition () String	isedCol- lec- tion- s.roso.UpdateSetoffExclusionDate	isedCol- lec- tion- s.roso.UpdateSetoffExclusionDate_ Impl	nq- ue- nt acc- ou- nt - C1- EX- CL- RO- SO	cessing will be driven by parameters set for the process. Set-off Exclusion date should be updated only if current exclusion date is <= business date. Else,

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	s to execute the bu-	getNextCaseStatus() String getNextTransCondition()				skip all below processing If Cancel Reason char type parameters is not bank • Get the value for the specified char type • This char type should be

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	specific status.					used to get the offset days from the Lookup for set-off exclusion days • Set-off exclusion date should be set as current business days + offset days. • If mapping for

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						the reason is not found, default value for offset days should be used. If Cancel Reason char type parameters is blank but Reason code is provided • Get the cor-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						responding offset days from the lookup for the Reason code • Set-off exclusion date should be set as current business days + offset days. • If mapping for the reason is not found, default value for offset

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						days should be used. Parameters - • Default Offset - Number of days to add to the set-off exclusion date. • Reason Code - Code to fetch offset days from lookup. • Cancel

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Reason - Characteristic type code to fetch offset days from lookup.
CaseTypeEnterStatusAlgorithmSpot	The p-	void setCase (ToDoCase toDoCase) void	com.splwg.cb.domain.collection.caseType.spe-	com.splwg.cb.domain.collection.caseType.spe-	Cancel Set-off - C1-CA-	This algorithm will update the set-off status as "Cancelled" for target accounts associated to the case and having set-off status as

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	urpose of the algorit-	setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition ()	cialisedCollections.roso.CancelSetoff	cialisedCollections.roso.CancelSetoff_Impl	NROS-O	"Pending".

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	t is to e- x- e- c- ut- e th- e b-	String getNextCaseStatus() String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	intos-pecific status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.roso.CompleteSetoff	com.splwg.cb.domain.collection.caseType.specialisedCollections.roso.CompleteSetoff_Impl	Complete Set-off - C1-COMPRO-SO	This algorithm transitions the case to complete. Processing Logic will be as below • Validate that at least one of the target account has set-off status = "Success" and

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-				Reversed Flag = "N". • If above validation fails transition to complete should not be allowed and Todo of given Todo Type should be created.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.roso.ReverseSetoff	com.splwg.cb.domain.collection.caseType.specialisedCollections.roso.ReverseSetoff_Impl	Reverse Set-off - C1-REVR-OS-O	This algorithm transitions the case to Reversed status. Processing Logic will be as below • Validate below for each target account o Set-off

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition () String getNextCaseStatus()				status is not "Success" or If set-off status is "Success" then Reversed Flag should be "Y". o There should be at least one account with

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				Reversed Flag as "Y". • If above validation fails, transition to this status should not be allowed.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Customer Association for Asset	This algorithm will perform following actions: • It will get all customers who are the owners for the selected collateral • It will Associate these customers with the case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.AssetRepo.CustomerAssociationForAssetRepossessionCase	s.AssetRepo.CustomerAssociationForAssetRepossessionCase_Impl	repossession case - C1-ARSCUSTS	Scope of this association is limited to customers already in collections. This process will not check for any customers not in collections. This algorithm doesn't have any soft parameter.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Update Collateral Status in the hos-	This process will update the collateral status in the host. The value of status to be set will be passed as parameter to the process. If the update fails for any reason,

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.AssetRepo.UpdateCollateralStatusInTheHost	s.AssetRepo.UpdateCollateralStatusInTheHost_Impl	t-C1-UPCOLLST-X	system should create a To Do. Message for the To Do should be configured based on type of update which failed. To Do should be assigned to the To Do Role set as parameter to this process. If

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				the parameter is left blank, To Do should be assigned to the default role. Collateral status can be one of the following: 1) REALIZATION_IN_PROGRESS 2)RELIZATION_COMPLETE

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Algorithm type for update case id	Algorithm type for update case id for Insurance

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.AssetRepo.UpdateInsuranceCaseDetails	s.AssetRepo.UpdateInsuranceCaseDetails_Impl	for Insurance-C1-UPCASFINS	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.actionObject.actionHistory.CaseCreationOnEnterAlgo	com.splwg.ccb.domain.collection.actionObject.actionHistory.CaseCreationOnEnterAlgo_Impl	Case Creation on enter processing -	This Algorithm will create a new case for the given Case Type on enter processing. This algorithm accepts Case type as a string, which is required to create a case.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			C1-CC-OE-NT-ER	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot	The purpose of	void setCase (ToDoCase toDoCase); void setNextCaseStatus	com.splwg.ccb.domain.collection.actionObject.actionHistory.CaseCreationOnExitAlgo	com.splwg.ccb.domain.collection.actionObject.actionHistory.CaseCreationOnExitAlgo_Impl	Collection - Case Creation On Exit of	This algorithm will create a case on the exit processing of the status. This algorihtm will create a case for the account in context and the provided Case type soft para-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the algorithm spot	(CaseStatus caseStatus);			Status-C1-CCOE	meter.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	d-di-ti-o-n-al lo-gi-c w-h-e-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the next status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ValidateBusinessObjectAlgorithmSpot		void setMaintenanceObject (MaintenanceObject mo); void setBusi-	com.splwg.ccb.domain.collection.actionObject.actionCategory.ActionCategoryValidation	com.splwg.ccb.domain.collection.actionObject.actionCategory.ActionCategoryValidation_Impl	Action category Validation algorithm - C1-	Action category Validation algorithm.This algorithm checks that there should be atleast on action category entity on it.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObject(BusinessObject bo); void setEntityId (EntityId id); void setAction(Busi-			AC-TC-AT	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObjectActionLookup boAction); void setBusinessObjectKey (BusinessOb-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		jectInstanceKey boKey); void setOriginalBusinessObject (BusinessObjectIn-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		stance boRequest); void setNewBusinessObject (BusinessObjectInstance boRequ-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		est);				
ValidateBusinessObjectAlgorithmSpot		void setMaintenanceObject (MaintenanceObject mo);	com.s-plwg.c-cb.-domain.-col-lec-tion.ac-tionOb-ject.ac-tionType.Ac-tionTypeResultTypeValidation	com.s-plwg.c-cb.-domain.-col-lec-tion.ac-tionOb-ject.ac-tionType.Ac-tionTypeResultTypeValidation_Impl	Action Type Algorithm Type - C1-AC-	Action Type Algorithm Type vvv

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setBusinessObject (BusinessObject bo); void setEntityId (EntityId			TTYP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		id); void setAction(BusinessObjectActionLookup boAction); void setBusi-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObjectKey (BusinessObjectInstanceKey boKey); void setOriginalBusinessOb-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ject(BusinessObjectInstance boRequest); void setNewBusinessObject (BusinessOb-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		jectInstanceboRequest);				
ValidateBusinessObjectAlgorithmSpot		void setMaintenanceObject (Main-	com.splwg.ccb.domain.collection.actionObject.case-	com.splwg.ccb.domain.collection.actionObject.case-	CaseTypeStatusMappi-	Case Type Status Mapping Algorithm Type

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		tenanceObject mo); void setBusinessObject (BusinessObject bo); void	TypeMapping.CaseTypeMappingValidation	TypeMapping.CaseTypeMappingValidation_Impl	ng Algorithm Type - C1-CASE-TYMP	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setEntityId (EntityId id); void setAction(BusinessObjectActionLookup boAc-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		tion); void setBusinessObjectKey (BusinessObjectInstanceKey boKey); void				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setOriginalBusinessObject (BusinessObjectInstanceboRequest); void setNew-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		BusinessObject (BusinessObjectInstanceboRequest);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CollectionClosingAlgorithmSpot		void setServiceAgreementId (String said);	com.splwg.cb.domain.collection.caseCreation.CollectionClosingAlgo	com.splwg.cb.domain.collection.caseCreation.CollectionClosingAlgo_Impl	Collection - Close Processing Algorithm	This algorithm will perform processing done when a Pending Stop Contract is picked up by the Overdue Monitor (collection is to be closed for an account).

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					- C1-CC-AL-G	- It will update the financial balance of the Contract to zero through an adjustment. - Check if there is one or more active promise to pay for the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						account, if it does it will update the promise to pay status to cancelled and provides the cancel reason - If it is required to close any cases, then it will check if the case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						has a next status in a final status and if it does will transition to that state. If the case has multiple next statuses which are final statuses, then it will use the default final

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						The following parameters are available and are required: - Adjustment Type used for the adjustment created by this algorithm. - Cancelation Reason Code

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						used while canceling Active PTPs - Is Closing Required Flag to specify if the cases associated have to be closed. If this flag is Y but one or more cases cannot be closed the algorithm will

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						generate an error. - Final Default Case Status - If the case to be closed has multiple next statuses that are final and the status specified in this parameter is one of those final statuses,

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						the case will be moved to the status specified in this parameter.
ValidateBusinessObjectAlgorithmSpot		void setMaintenanceObject	com.splwg.cb.domain.collection.caseGroup.CaseGroupVal-	com.splwg.cb.domain.collection.caseGroup.CaseGroupVal-	Case Group add val-	Case Group add validation algorithm

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(MaintenanceObject mo); void setBusinessObject (BusinessObject bo);	idationAlgorithm	idationAlgorithm_Impl	idationalgorithm-C1-CG-VAL	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setEntityId (EntityId id); void setAction(BusinessObjectActionLookup				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boAction); void setBusinessObjectKey (BusinessObjectInstanceKey boKey);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setOriginalBusinessObject (BusinessObjectInstance boRequest); void				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setNewBusinessObject (BusinessObjectInstanceboRequest);				
GetStrategyAlgorithmSpot		void setSer-	com.splwg.ccb.-	com.splwg.ccb.-	Collec-	This algorithm calls the Rules

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		viceAgreementId (String sald); void setCaseId(String caseId); String getNewCaseType();	domain.collection.caseCreation.GetStrategyAlgo	domain.collection.caseCreation.GetStrategyAlgo_Impl	tion - GetStrategyAlgorithm - C1-CO-LG-	Engine to determine a collection strategy. It is invoked by the Collection Class Overdue Rule - Overdue Monitor Rule The following parameters are passed to the Rules Engine : - Rule Type

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						(defined in the input parameter) - Case Type (if any) - Days Past Due - Overdue Amount - Collection Type

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CollectionClassOverdueMonitorRuleAlgorithmSpot		void setAccount (Account account); Bool getIsProcessingComplete();	com.splwg.ccb.domain.collection.caseCreation.CaseOverdueMonitorRuleAlgo	com.splwg.ccb.domain.collection.caseCreation.CaseOverdueMonitorRuleAlgo_Impl	Create/-Move Collection Strategy Cases for	This overdue monitor rule algorithm is used to determine the appropriate case type to be used to create a case for an account in

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					Account - C1-COLMR	collections. It is also responsible for creating the case or for case movement. It will first check for the Collection events (contracts) that are

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						under the account. For Active Contracts it will call the Collection - Get Strategy Algorithm, to determine which Case Type

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						should be used before creating a case. If one or more cases already exist for the Contract they may get closed and new cases created (case movement) if Col-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						n be closed. This algorithm also consider the feature configuration 'C1-NMCSTY' to determine the cases that should not be moved.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						For Pending Stop Contracts it will call the Collection - Close Processing Algorithm to move the Contract into a closed status. May also close the Cases

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						attached to the contract and reduce the overdue amount on the contract to zero. All other SA statuses are ignored by this algorithm. Notes on the algorithm para-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						meters - Final Default Case Status - If the case to be closed have multiple next statuses that are final and the status specified in this parameter is one of those final statuses,

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						the case will be moved to the status specified in this parameter. - Is Closing required - Flag indicate whether case closing is required or not (Y/N) - Collection Closing Algorithm -

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						This is the algorithm code for Collection - Close Processing Algorithm - Get Strategy Algorithm - This is the algorithm code for Collection - Get Strategy Algorithm

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
AuditBusinessObjectAlgorithmSpot		void setMaintenanceObject (MaintenanceObject mo); void	com.splwg.cb.domain.collection.caseCreation.CaseSaUpdateBoAuditAlgo	com.splwg.cb.domain.collection.caseCreation.CaseSaUpdateBoAuditAlgo_Impl	Collection - Case SA Update for Manu-	This Algorithm will update Case SA table for Manual Case Creation

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setBusinessObject (BusinessObject bo); void setEntityId (EntityId id);			al Creation - C1-CS-AU-PD	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setAction(BusinessObjectActionLookup boAction); void setBusi-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObjectKey (BusinessObjectInstanceKey boKey); void setOriginalBusi-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObject(BusinessObjectInstance boRequest); void setNewBusinessObject (Busi-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObjectInstance boRequest); void setChangedValues (SchemaInstanceChanges				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		changes);				
InstallationPayPlanAddlGraceDaysAlgorithmSpot		BigInteger getPayPlanAdditionalGraceDays(); void setPaymentPla-	com.splwg.ccb.domain.collection.payPlan.AdditionalGraceDaysCalculationAlgorithm	com.splwg.ccb.domain.collection.payPlan.AdditionalGraceDaysCalculationAlgorithm_Impl	Promise to Pay - Additional Gra-	This sample algorithm is called by the Promise to Pay Monitor; it takes the output, which represents additional grace days that should be added to a prom-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		n(PaymentPlan paymentPlan); void setAccount (Account account);			ce Days Sample Algo - C1-PP-AD-DL-GR-	ise to pay's scheduled payment date. The algorithm takes the input parameter value and passes it back to the Promise to Pay Monitor as additional grace days.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
InstallationPtpThresholdPcForNgpAlgorithmSpot		void setPromiseToPay (PromiseToPay prom-	com.splwg.ccb.domain.collection.payPlan.PaymentThresholdPercentageCalculationAlgorithm	com.splwg.ccb.domain.collection.payPlan.PaymentThresholdPercentageCalculationAlgorithm_Impl	Promise to Pay Threshold Percentage -	This algorithm is called by the Pay Plan Monitor when an expected scheduled payment is not fully met. At this point the promise to pay

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		iseToPay); void setAccount (Account account); void setTotalPaidA-			C1-PP-TH-RESH	has been marked to be broken. It receives the following inputs from the pay plan monitor - Promise to Pay ID - Total Amount Paid towards the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		mount (BigDecimal totalPaidAmount); void setProcessDate(Date processDat-				promise to pay - Date (Business Date - Grace Days) - Array of Promise to Pay Scheduled Payments balance The algorithm will check if the Total Amount Paid is within the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		e); void setLastPaidScheduleDate(Date lastPaidScheduleDate); void setLastP-				threshold percentage (input parameter) of the Total Scheduled Payments expected. If the payments are within the threshold, then the algorithms returns a value of "Y" indicating the promise to

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		aidScheduleBalance (BigDecimal lastPaidScheduleBalance); void setGraceDays				pay that was set to be broken should be overridden and remain active/kept Else if the total payments are not within the threshold, then the algorithm returns a value of "N" indicating

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(BigInteger graceDays); String getOverrideflag (); void setValidStartDate(Date val-				the promise to pay should be set to broken.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		idationStartDate); void setValidEndDate(Date validationEndDate);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithm spp-	void setActionEntity(String actionEntity); void setActionSourceId (String actionSo-	com.splwg.ccb.domain.collection.actionObject.actionType.ResultTypePostProcCaseTransAlgo	com.splwg.ccb.domain.collection.actionObject.actionType.ResultTypePostProcCaseTransAlgo_Impl	Result type Post Processing Case Tra-	If specified on the Result Type, this algorithm will be invoked when the corresponding result is recorded for a Case (Action/Result UI). This can be used to transiton

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	otd-e-ci-d-es in w-hi-ch st-at-	urceId); void setAc-tionSour-ceStatus-Code (String actionSo-urceStat-usCd); void setAc-			nsit-ion Alg-o - C1-RT-PC-C	the case from the current status to the next possible status as follow, - This algorithm has a parameter Output Status i.e. next possible status, so for case transition it will be checked whether Output

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	to be made based on	tionId (String actionId); void setActionType (ActionType actionType); void setRes-				Status is one of the next possible status, if YES, it will transiiton the case to that status - This algorihtm has a parameter Input Status, which will be checked against the current status of the Case. This

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ultType (ResultType resultType); boolean getIsProcessingComplete();				is an optional parameter. If specified the Case transition will happen only when the current status of the case matches with this parameter.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
BusinessObjectEnterStatusAlgorithmSpot			com.splwg.cb.domain.collection.inboundCustomer.CreateEntityAlgo	com.splwg.cb.domain.collection.inboundCustomer.CreateEntityAlgo_Impl	Inbound Customer algorithm - C1-IN-	This algorithm will create the Person,Account,SA, SAcollection object and Adjustment from FACT clob. This is a Business Object Status Enter algorithm.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					CUST	The algorithm perform the following actions - Retrieve the XML message containing the customer information, which stored on the FACT MO. - Read the XML

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						and determine if the action is to add a new customer or update an existing customer. - It may create a combination of Person, Account, Contract, Contract, or Adjustment, depending

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						on what was contained in the XML. - If all objects are created successfully it will transition the lifecycle to the 'Created' status - Else if any of the objects experienced and error

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						while processing it will transition the lifecycle to the "Rejected" status. It has two parameters , both optional. - Account Id Type identifies the Account Identifier Type used to locate the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						account in ORMB. - Person Id Type identifies the Person Identifier Type used to locate the person in ORMB.
ResultTypePreProcessingAl-			com.splwg.cb.domain.col-	com.splwg.cb.domain.col-	ResultType	Result Type Preprocessing Algorihtm Type

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
gorithmSpot			lection.actionObject.actionType.ResultTypePreProcAlgo	lection.actionObject.actionType.ResultTypePreProcAlgo_Impl	Preprocessing Algorithm Type-C1-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithms	void setActionEntity(String actionEntity); void setActionSourceId	com.splwg.ccb.domain.collection.actionObject.actionType.ResultTypePostProcAlgo	com.splwg.ccb.domain.collection.actionObject.actionType.ResultTypePostProcAlgo_Impl	ResultTypePostprocessing Alg-	Result Type Post-processing Algorithm Type

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot d-e-ci-d-es in w-hi-ch st-	(String actionSourceId); void setActionSourceStatusCode (String actionSourceStatusCd);			orit-hm Ty-pe - C1-RS-TP-OS-T	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	has to be made base-	void setActionId (String actionId); void setActionType (ActionType actionType);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary

		void setResultType(ResultType resultType);
		
		boolean getIsProcessingComplete();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
AdhocCharacteristicValueValidationAlgorithmSpot	This algorith-	void setFormatOnly (boolean formatOnly); void setCharacteristicType (CharacteristicType	com.splwg.cb.domain.collection.caseType.specialisedCollections.Cus-	com.splwg.cb.domain.collection.caseType.specialisedCollections.Cus-	Characteristic Type :Validate Date Fie-	Custom Date validation This algorithm is used to validate that an ad hoc characteristic value is a date or a date/time. The Parameters From Date and

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	msp-ot is inv-ok-	type); void setAd-hocValu-e(String value); String getRe-	tomAdhocDateValidationAlgComp	tomAdhocDateValidationAlgComp_Impl	ld (Cu-sto-m) - C1-AD-HD-AT-E	To Date are both optional. The algorithm will check that the date is later than the From Date (if entered) and/or earlier than the To Date (if entered). If

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ristic adhoc value	formattedValue(); boolean isValidAdhoc();				ey must be in the format YYYYMMDD. These parameters are ignored if the characteristic value is a date/-time field. The various Date Format

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	1) validate that th-					parameters are used to control the format in which the date/-time is entered by a user. You must supply at least one format in parameter 3. The other parameters exist in case you allow

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	evalue is correct 2)					be used. Examples of date formats include: YYYYMMDD, DD/MM/YYYY, DD-MM-YYYY, MM/DD/YYYY, YYYY-MM-DD, etc. However, only three types of date/time

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	possibly perform are-					Regardless of the format entered by the user, the date is stored in the format defined by parameter 3. We strongly recommend this parameter be set to YYYY-MM-DD for dates and

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	t of the value prior to st-					YYYY-MM-DD-HH:MI:SS for date/time fields as this is how all dates are stored in our system.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
DialerResultsUploadAlgorithmSpot		void setAccountNumber(String accountNumber); void setCustomerNumber	com.splwg.cb.domain.collection.dialerResultUpload.DialerResultUploadAlgo	com.splwg.cb.domain.collection.dialerResultUpload.DialerResultUploadAlgo_Impl	AlgorithmTypeforDialerResultsUpl-	Algorithm Type for Dialer Results Upload

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(String customerNumber);			oad - C1-DLRRSUPLD	
CaseTypeEnterStatusAlgorithmSpot	T-	void setCase (ToDoCase	com.splwg.ccb.domain.-	com.splwg.ccb.domain.-	Algorithm for	This Algorithm is responsible for making a Hardship Case entry on the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	he purpose of the al-	toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShoul-	collection.caseType.specialisedCollections.financialHardship.HardshipCaseCreationActivity	collection.caseType.specialisedCollections.financialHardship.HardshipCaseCreationActivity_Impl	Hardship case creation activity - C1-CR-	Party,when the Hardship case is created.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	rithms spot is to exec	dAutoTransition() String getNextCaseStatus() String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	hen Case is moved in-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot	The purpose of	void setCase (ToDoCase toDoCase); void setNextCaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.AbortApprovalWorkItemsAlgo	com.splwg.cb.domain.collection.caseType.specialisedCollections.AbortApprovalWorkItemsAlgo_	This Algorithm is used to abort Ap-	This algorithm is used to abort approval work item. Input to the algorithm is composite name,instance title and case status exclusion list. If next case status

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the algorithms spot	(CaseStatus caseStatus);		Impl	proval work item - C1-ABORTAPP	is present in case status exclusion list then work item instance is not aborted. caseStatusExclusionList:- comma seperated list of case status for which

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	d-di-ti-o-n-al lo-gi-c w-h-e-					approval work item shouldn't be aborted. Composite Name:- Fully qualified approval class name. Instance Title:- Approval instance work item title prefix.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the next status.					Example - Input parameters and it's applicable value for ROSO Process, Composite Name:- com.ofss.fc.workflow.process.ROSOProcessForApproval

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Instance Title:- ROSO_CASE_ Value of the above parmeters is dependent upon the SOA approval work flow.
CaseTypeEnterStatusAlgorithmSpot	T-	void setCase (ToDoCase	com.splwg.cb.domain.-	com.splwg.cb.domain.-	Cancel Pro-	This algorithm will cancel all pending approval

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	he purpose of the al-	toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShoul-	collection.caseType.specialisedCollections.financialHardship.CancelFinancialHardshipApprovalReqAlgo	collection.caseType.specialisedCollections.financialHardship.CancelFinancialHardshipApprovalReqAlgo_Impl	cessApprovalRequest:-FinancialHardsh-	requests for the case. Example for parametervalues for hardship Process: Composite Name:- com.ofss.fc.-workflow.-process.Fin-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	rithms-pot is to exec-	dAutoTransition() String getNextCaseStatus() String getNextTransCondition()			NFHAPPR	ancialHardshipProcessForApproval Instance Title:- FINANCIAL_HARDSHIP_CASE_ Value of the above parmeters depends upon the SOA approval work flow.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	hen Case is moved in-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Hardship Characteristic Association	Hardship Characteristic Association

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.HardshipCharAssociation	s.financialHardship.HardshipCharAssociation_Impl	-C1-FH-CHAR-ASC	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
PrePopulatedRuleFactsAlgorithmSpot		List<RuleFactInfoDto> getPrePopulatedFacts();	com.splwg.ccb.domain.collection.PrePopulatedSystemFacts	com.splwg.ccb.domain.collection.PrePopulatedSystemFacts_Impl	Pre-Populated system facts to be use-	This algoritm is used to populate input system fact for Rule.It used as an input to RuleFactPopulation algorithm. System Facts populated through this algoritm are

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					d for Rule - C1-PP-SF	SystemDate and PostingDate. This is sample implementation to provide populated facts to RuleFactPopulation algoritm.User can create his own algoritm type based on his requirement

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						(Algoritm Entity must be Rule Execution - Pre Populated Rule Facts).
ToDoTypeToDoPostProcessAlgorithmSpot	This Alg-	void setOldToDoEntryDTO (ToDoEntry_DTO	com.s-plwg.c-cb.-domain.-col-lec-tion.batch.al-gorith-	com.s-plwg.c-cb.-domain.-col-lec-tion.batch.al-gorith-	Assign Batch level	Assign Batch level TODOs (task) to a queue.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithm spot is used	oldDTO); void setNewToDoEntry (ToDoEntry newToDoEntry);	m.AssignTaskToQueueAlgorithm	m.AssignTaskToQueueAlgorithm_Impl	TODOs (task) to a queue. - C1-ASGNTA-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	fying task completion					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	or.					
AdhocCharacteristicValueValidationAlgorithmSpot	This alg-	void setFormatOnly (boolean formatOnly); void setCharacteristicType	com.splwg.cb.domain.collection.caseType.specialisedCollec-	com.splwg.cb.domain.collection.caseType.specialisedCollec-	Validate Date Field :Custom - C1-	This algorithm is used to validate that an ad hoc characteristic value is a date or a date/time. The Parameters

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithms pot is in-	(CharacteristicType type); void setAdhocValue(String	tions.ProductAdhocDateValidationAlgComp	tions.ProductAdhocDateValidationAlgComp_Impl	ADHVDTD	From Date and To Date are both optional. The algorithm will check that the date is later than the From Date (if entered) and/or earlier than the To Date (if

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	characteristic add-	String getReformattedValue (); boolean isValidAdhoc				ither value is specified, they must be in the format YYYYMMDD. These parameters are ignored if the characteristic value is a date/-time field. The various

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	1) validate that th-	();				Date Format parameters are used to control the format in which the date/-time is entered by a user. You must supply at least one format in parameter 3. The other parameters exist in

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	evalue is correct 2)					ow multiple date formats to be used. Examples of date formats include: YYYYMMDD, DD/MM/YYYY, DD-MM-YYYY, MM/DD/YYYY, YYYY-MM-DD, etc. However, only three types

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	possibly perform are-					Regardless of the format entered by the user, the date is stored in the format defined by parameter 3. We strongly recommend this parameter be set to YYYY-MM-DD for dates and

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	t of the value prior to st-					YYYY-MM-DD-HH:MI:SS for date/time fields as this is how all dates are stored in our system.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition();	com.splwg.ccb.domain.collection.caseType.specialisedCollection	com.splwg.ccb.domain.collection.caseType.specialisedCollection	Retry Case in Error - C1-RCASEE	This algorithm is plugged-in on auto-transition of error states and attempts to retry validation, completion or wait if the To Do Entry associated is not being worked on. The retry will be performed

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pe is used to perfor-	CaseStatus getNextCaseStatus(); String getNextTransCondition();	s.financialHardship.RetryCaseInErrorForHardshipApp	s.financialHardship.RetryCaseInErrorForHardshipApp_Impl		only until the input Maximum Number of Retries is reached.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ValidateBusinessObjectAlgorithmSpot		void setMaintenanceObject (MaintenanceObject mo);	com.splwg.cb.domain.collection.address.PersonCollectionAddressValidation	com.splwg.cb.domain.collection.address.PersonCollectionAddressValidation_Impl	PersonAddress - Collection - C1-PERA-	This Algorithm is a reference implemenation for consulting. This algorithm will be used for validating Person address as per requirment

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setBusinessObject (BusinessObject bo); void setEntityId (EntityId			DDRC	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		id); void setAction(BusinessObjectActionLookup boAction); void setBusi-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObjectKey (BusinessObjectInstanceKey boKey); void setOriginalBusinessOb-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ject(BusinessObjectInstance boRequest); void setNewBusinessObject (BusinessOb-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		jectInstance boRequest);				
PostProcessBusinessObjectAlgorithmSpot		void setMaintenanceObject (Maintenance	com.splwg.ccb.domain.collection.address.PersonCollectionAddressPostProcess	com.splwg.ccb.domain.collection.address.PersonCollectionAddressPostProcess_	PersonAddressupdate	Person Address update post processing algorithm.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		anceObject mo); void setBusinessObject (BusinessObject bo); void setEntityId		Impl	post processing algorithm. - C1-PA-DD-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(EntityId id); void setAction(BusinessObjectActionLookup boAction); void				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setBusinessObjectKey (BusinessObjectInstanceKey boKey); void setOriginalBusi-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObject(BusinessObjectInstance boRequest);				
ICollectionContactPointPostProcessingSpot		void setOldEntity (ContactPrefCollOldEntit-	com.splwg.ccb.domain.collec-	com.splwg.ccb.domain.collec-	Person Contact	This is a reference implementation of Post processing Algo. Customization team

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		y); void setNewEntity (ContactPrefColl newEntity); y);	tion.address.CollectionContactPointPostProcessingSpot	tion.address.CollectionContactPointPostProcessingSpot_Impl	Point Update - Post Processing - C1-PE-	can utilize this hook.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
PreprocessBusinessObjectRequestAlgorithmSpot		void setAction(BusinessObjectActionLookup boAction); void setBusi-	com.splwg.cb.domain.collection.address.PersonCollectionAddressPreProcess	com.splwg.cb.domain.collection.address.PersonCollectionAddressPreProcess_Impl	PersonAddressUpdate - PreProcessin-	Person Address PreProcessing algorithm. Attached on BO pre processing spot. This is a hook provided to customization. This can be utilized to validate the address data.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObject(BusinessObject bo); void setRequest(BusinessObjectInstance boRequest);			g - C1-PA-DD-PR-E	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.ccb.domain.collection.caseType.UpdateSelfServeFlag	com.splwg.ccb.domain.collection.caseType.UpdateSelfServeFlag_Impl	Update Self Serve Flag Algorithm -	Action -soft parameter mentioned in algorithm type which will update the self_serve flag to Y or N. If Action = Set make Self Serve Flag = Y If Action = Reset make Self Serve Flag = N

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-			C1-SELF-SERVE	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithmSpot	This Algorithms	void setActionEntity(String actionEntity); void setActionSour-	com.splwg.cb.domain.collection.actionObject.actionHistory.FollowUpResultTaskAlgo	com.splwg.cb.domain.collection.actionObject.actionHistory.FollowUpResultTaskAlgo_Impl	Create Task for Self Serve Requ-	This algorithm will be used to Create Task post Follow Up.Possible values are Task For,Task Type and Task Queue

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-ot d-e-ci-d-es in w-hi-ch st-	ceId (String actionSourceId); void setActionSourceStatusCode (String actionSo-			est for As-sis-tan-ce tra-nsa-ctio-n - C1-FL-W-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	has to be made base-	urceStatusCd); void setActionId (String actionId); void setActionType				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		(ActionType actionType); void setResultType (ResultType resultType);				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		boolean getIsProcessingComplete();				
CaseTypeEnterStatusAlgorithmSpot	The	void setCase (ToDoCase toDoCase)	com.splwg.cb.domain.collection.case-	com.splwg.cb.domain.collection.case-	Associate all accou-	For the customer associated with the case • Get all accounts where

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	purpose of the algor-	void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition	Type.specialisedCollections.bankruptcy.BankruptcyAssociateAccounts	Type.specialisedCollections.bankruptcy.BankruptcyAssociateAccounts_Impl	nts to the case where customer is a borrow-	this customer is borrower and the accounts are In Collections. (Fetch accounts based on Enterprise Id if Consider Enterprise ID = Y). • If Account Relationships = MC consider only the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spot is to execute th-	() String getNextCaseStatus() String getNextTransCondition()				accounts where the customer is primary owner. If Account Relationships = FO consider all accounts where the customer is a financial owner. If Account Relationship = All consider all accounts where

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	moved into specific st-					the customer is a financial or non financial owner. • Shortlist the accounts that are not yet associated with the case. Associate the shortlisted accounts with the case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Associate all accounts to the case	For the primary customer associated with the case • Get all accounts where this customer is primary owner and the accounts are In Collections. • Shortlist the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm sp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.bankruptcy.BankruptcyAssociateAcc	s.bankruptcy.BankruptcyAssociateAcc_Impl	where customer is a primary borrower - C1-	accounts that are not yet associated with the case. • Associate the shortlisted accounts with the case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithmty-	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition();	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Notify the Bankruptcy Specialist if	If for any of the associated account if the liquidation case reaches a specific status than create a notification for the Bankruptcy Specialist. Notification: "Liquidation for

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-e is u-s-e-d to p-er-fo-r-	CaseStatus getNextCaseStatus(); String getNextTransCondition();	s.bankruptcy.BankruptcyMonitorAssetLiquidation	s.bankruptcy.BankruptcyMonitorAssetLiquidation_Impl	the Liquidation reaches a specific status -	Account <Account Number> has reached < Case Status>

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm ty-	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition ();	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Monitor if any of the associated acc-	If any of the associated account has delinquency Start Date = Today's posting date Create Bankruptcy Notification as 'Account <Account Number> has

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p- e is u- s- e- d to p- er- fo- r-	CaseSta- tus getNex- tCaseSt- atus(); String getNex- tTransC- ondition ();	s.bank- ruptcy.Bank- ruptcyMonitorDelinquency	s.bank- ruptcy.Bank- ruptcyMonitorDelinquency_Impl	ou- nt bec- om- es deli- nq- ue- nt - C1- MT- RD- LN-	become Delin- quent' Set Display Date of the case to cur- rent business date.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.earlyCollections.UpdateCaseData	com.splwg.cb.domain.collection.caseType.earlyCollections.UpdateCaseData_Impl	Update Case Level Data when a case enters a new status	Set Case Characteristics to specific values provided in algorithm parameters. On entering the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()			a case enters a new status - C1-EC-UP-	value the corresponding characteristic validation algorithm should be triggered. If type is mentioned but value is not than the char type needs to be made blank.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Algorithm type for case list update -	Algorithm type for case list update or insert in CI_LIST_MODE_UPDATE table.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.financialHardship.HardShipCaseListUpdate	s.financialHardship.HardShipCaseListUpdate_Impl	C1-CASE-LIST	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Check if any active case is present of	Processing steps are as below 1. If only Case Category is specified check if any active case is running on the customer whose a. Case category is same as the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.bankruptcy.CheckBankruptcyCaseExist	s.bankruptcy.CheckBankruptcyCaseExist_Impl	a given case category or case type	parameter set for the algorithm 2. If Case Type is specified check if any active case is running on the customer whose a. Case type is same as the parameter set for the algorithm 3. If yes val-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				idation should fail

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusValidationAlgorithmSpot	The purpose of	void setCase (ToDoCase toDoCase); void setOriginalCaseStatus (CaseSt-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Check if any active case is present of	Processing steps are as below 1. If only Case Category is specified check if any active case is running on the customer whose a. Case category is same as the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	the algorithm spot	atus caseOriginalStatus);	s.bankruptcy.CheckBankruptcyCaseExist	s.bankruptcy.CheckBankruptcyCaseExist_Impl	a given case category or case type	parameter set for the algorithm 2. If Case Type is specified check if any active case is running on the customer whose a. Case type is same as the parameter set for the algorithm 3. If yes val-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	hevalidationlogicb-					idation should fail

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
InstallationContactInformationAlgoSpot		void setContactPreference(ContactPref contactPref); boolean getStatus(); void	com.splwg.cb.domain.collection.collectionLandingPage.ContactInformationCallAdviceAlgo	com.splwg.cb.domain.collection.collectionLandingPage.ContactInformationCallAdviceAlgo_Impl	Call Advice - Red/Green logic	Call Advice calculation • Call Advice will be 'Green' if - 'Permission to Call' is Yes And - Current Time is within the State level Acceptable Time Limits And

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setPerson(Person person);			calculation - C1-CALA-DVICE	- Current Time is within the preferred times of the Customer And - Current Date is not within the Do Not Disturb Dates Else it will be 'Red'.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.earlyCollections.CeaseDesistAccountSuspension	com.splwg.cb.domain.collection.caseType.earlyCollections.CeaseDesistAccountSuspension_Impl	This algorithm will transition the case	Additional algorithm in Pending Status: Enter Processing to transition to Suspend Status if Cease and Desist = Y.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex-			status to the Suspension status if Cease	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
TaskCaseValidationAlgorithmSpot		void setTaskCaseMapping (CaseTaskMapping caseTaskMapping); void	com.splwg.cb.domain.collection.caseType.specialisedCollections.AssetRepo.RepoDateValidation	com.splwg.cb.domain.collection.caseType.specialisedCollection-	Task Case Mapping Validation Algorithm	Algorithm will validate Repossession Date cannot be greater than future date for the process field mapped to Task

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setToDoCase (ToDoCase toDoCase); void setToDoEntry (ToDoEntry toDoEntry);		s.AssetRepo.RepoDateValidation_Impl	hm-C1-TC-VAL	Type Code and Case Type Code mentioned in soft parameters. This algorithm will validate the Repossession Date field only if value is already present. Validation Date can be SYSTEM DATE or

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						POSTING DATE
TimeZoneDerivationAlgorithmSpot			com.splwg.cb.domain.collection.batch.algorithm.TimeZoneDerivationAlgorithm	com.splwg.cb.domain.collection.batch.algorithm.TimeZoneDerivationAlgorithm_Impl	Timezone derivation field updat-	This algorithm will update timezone of a person if it is blank

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					e algorithm - C1-TZ-DR-FL-D	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of th-	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus	com.splwg.cb.domain.collection.caseType.specialisedCollections.bank-	com.splwg.cb.domain.collection.caseType.specialisedCollections.bank-	Joint Bankruptcy - Associate other custom-	Associate additional customers specified on the UI that exist in OB Collections. (Assumption - If the party does not exist in OB Collection assumption is the party is pulled in OB Collections from

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ealgorithmspotis	caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNex	ruptcy.JointBnkptcyAssociateCust	ruptcy.JointBnkptcyAssociateCust_Impl	ers to the Bankruptcy case - C1-ASSC-US-	OBP through UI or through pull non delinquent accounts)

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	inesslogicwhenCase	tTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase(ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	com.splwg.ccb.domain.collection.caseType.specialisedCollection-	Metro 2 Reporting	The logic is incorporated for Metro Algorithm only if a Account is close than it should be marked as Close

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spp-	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	s.bankruptcy.Metro2CheckForOpenStatusEnterProcessing	s.bankruptcy.Metro2CheckForOpenStatusEnterProcessing_Impl	- Marking Account as Close - C1-CF-OS-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ute the business logi-	String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
PostProcessBusinessObjectAlgorithmSpot		void setMaintenanceObject (MaintenanceObject mo); void	com.splwg.ccb.domain.collection.vendor.SLAParametersPostProcessAlgo	com.splwg.ccb.domain.collection.vendor.SLAParametersPostProcessAlgo_Impl	SLA Parameters validation algorithm	created for Recovery 2.6.2 release. This algorithm to be called along with CI-SLABO.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setBusinessObject (BusinessObject bo); void setEntityId (EntityId id); void			- C1-SL-AP-AR-AM	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		setAction(BusinessObjectActionLookup boAction); void setBusinessObjectKey (Busi-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		nessObjectInstanceKey boKey); void setOriginalBusinessObject (BusinessOb-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		jectInstance boRequest);				
ResultTypePostProcessingAlgorithmSpot	This Algor-	void setActionEntity(String actionEntity);	com.splwg.ccb.domain.collection.caseType.spe-	com.splwg.ccb.domain.collection.caseType.spe-	Redemption Clear Date	This algorithm is used to calculate the Redemption Clear Date. By Default Redemption Clear Date will be caculated if

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ithmspotdecides	void setActionSourceId (String actionSourceId); void setActionSourceStatus-	cialisedCollections.AssetRepo.algorithms.ReposessionClosureRedemptionClearDateCal	cialisedCollections.AssetRepo.algorithms.ReposessionClosureRedemptionClearDateCal_Impl	ValueDatefieldCalculation - C1-RDEE-	REDEM_CLEAR_DT in CI_REPO_CLOSURE table is null and will be calculated as repossession Date + Redemption Clearing Days. Otherwise, Redemption Clear Date will be shown as per

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	tatus transitionhasto	Code (String actionSourceStatusCd); void setActionId (String actionId);				E table.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		void setActionType (ActionType actionType); void setResultType (Res-				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		ultType resultType); boolean getIsProcessingComplete();				
CaseTypeAutoTransitionAl-	T-	void setCase	com.splwg.ccb.-	com.splwg.-	Check	The algorithm checks if the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
gorithmSpot	his algorithm type is	(ToDoCase toDoCase); Bool getShouldAutoTransition (); CaseStatus getNex-	domain.collection.caseType.specialisedCollections.legal.CheckSubmissionDate	cb.domain.collection.caseType.specialisedCollections.legal.CheckSubmissionDate_Impl	Submission date - CI_CHKSUBDT-1	legal documents have been submittedand if that is the case,it changes the status for the caseId. The algorithm takes the following parameters: 1)Next Status: Specifies the next status for

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	s-ed to per-for-m a-ut-o tr-	tCaseSt-atus(); String getNex-tTransC-ondition ();				case transition if change status is {Y} 2)Change Status {Y,N}: Y defines to change the status,N defines not to change the status.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	a Case.					
BusinessObjectEnterStatusAlgorithmSpot			com.splwg.cb.domain.customerinfo.paymentPlan.PtpLetterSmsGeneration	com.splwg.cb.domain.customerinfo.paymentPlan.PtpLetterSmsGeneration_	algo type to generate	algo type to generate letter and sms on ptp

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
				Impl	letterandsmsonptp-C-1_PT-PL-	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
AdhocCharacteristicValueValidationAlgorithmSpot	This alg-or-	void setFormatOnly (boolean formatOnly); void setCharacteristicType (Char-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	com.splwg.cb.domain.collection.caseType.specialisedCollection-	ResultCharacteristicValueDate	This algorithm is used to validate format enter by user for result characteristics during taking follow up. 1. Validation Date: This Val-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ithmspotinv-	acteristicType type); void setAdhocValue(String	s.AssetRepo.algorithms.Repossessionclosureredemptionclearedate	s.AssetRepo.algorithms.Repossessionclosureredemptionclearedate_Impl	fieldValidation-PASTDATE_VAL	idation Date will validate and compare the date with user provided date. It's value can be SYSTEM DATE or POSTING DATE. This is mandatory para-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ract-eristic adh-oc v-	String getRe-formatt-edValue (); boolean isVal-idAdhoc				meter. 2. The various Date Format parameters are used to control the format in which the date/-time is entered by a user. You must supply at least one format in parameter 3.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	1) validateth-atth-	();				The other parameters exist in case you allow multiple date formats to be used. Examples of date formats include: YYYYMMDD, DD/MM/YYYY, DD-MM-YYYY, MM/DD/YYYY, YYYY-MM-DD,

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	evalue is correct 2)					"Stored Date Format" is a mandatory parameter whereas "Date Format2" is not. "Date Format2" is given for future requirement, if any.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	possibly perfo-rmare-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	t of the value prior to st-					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition();	com.splwg.cb.domain.collection.caseType.CaseAutoTransition	com.splwg.cb.domain.collection.caseType.CaseAutoTransition_Impl	CaseMonitoring-C-SMONITOR	This algorithm determines if a case has been in its current status long enough to be automatically transitioned to another status or some other action needs to

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	p-e is u-s-e-d to p-er-fo-r-	CaseStatus getNextCaseStatus(); String getNextTransCondition();				be taken on case. If the case has been in its current status for more than the given Number of days, it is allowed to do the following activity as par configuration:

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	processing for a Ca-					1. Create a To Do, for a given To Do type. 2. Re-Allocate the case to a different Queue. 3. Set Prompt Days. 4. Transition to another Status. The Number parameters cannot be changed.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						This Algorithm type is hard wired with the product and implementation can only override the 'Program Name' using Feature Config. It is expected that is one time set up per implementation.