Oracle® Banking Platform

Host Extensibility Guide

Release 2.6.2.0.0

E95189-01

May 2018

Oracle Banking Platform Host Extensibility Guide, Release 2.6.2.0.0

E95189-01

Contents

List of Figures

List of Tables

Preface

This guide explains customization and extension of Oracle Banking Platform.

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

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

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/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 Platform Installation Guide - Silent Installation.
- For a comprehensive overview of security, see the Oracle Banking Security Guide.
- For the complete list of licensed products and the third-party licenses included with the license, see the Oracle Banking Licensing Guide.
- For information related to setting up a bank or a branch, and other operational and administrative functions, see the Oracle Banking Administrator's Guide.
- For information on the functionality and features, see the respective Oracle Banking Functional Overview documents.

Conventions

The following text conventions are used in this document:

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

1 About This Guide

This guide is applicable for the following products:

- Oracle Banking Platform (OBP)
- Oracle Banking Enterprise Product Manufacturing (OBEPM)
- Oracle Banking Enterprise Originations (OBEO)
- Oracle Banking Enterprise Collections (OBEC)

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 11.3 Alert Processing Steps
- Chapter 12 Creating New Reports
- Chapter 21 OCH Integration

1.2 Sections Applicable Only for Oracle Banking Enterprise Collections

This following chapters and sections in this guide are applicable only for Oracle Banking Enterprise Collections.

- Chapter 22 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.

2.2.2 Document Scope

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

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

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

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.

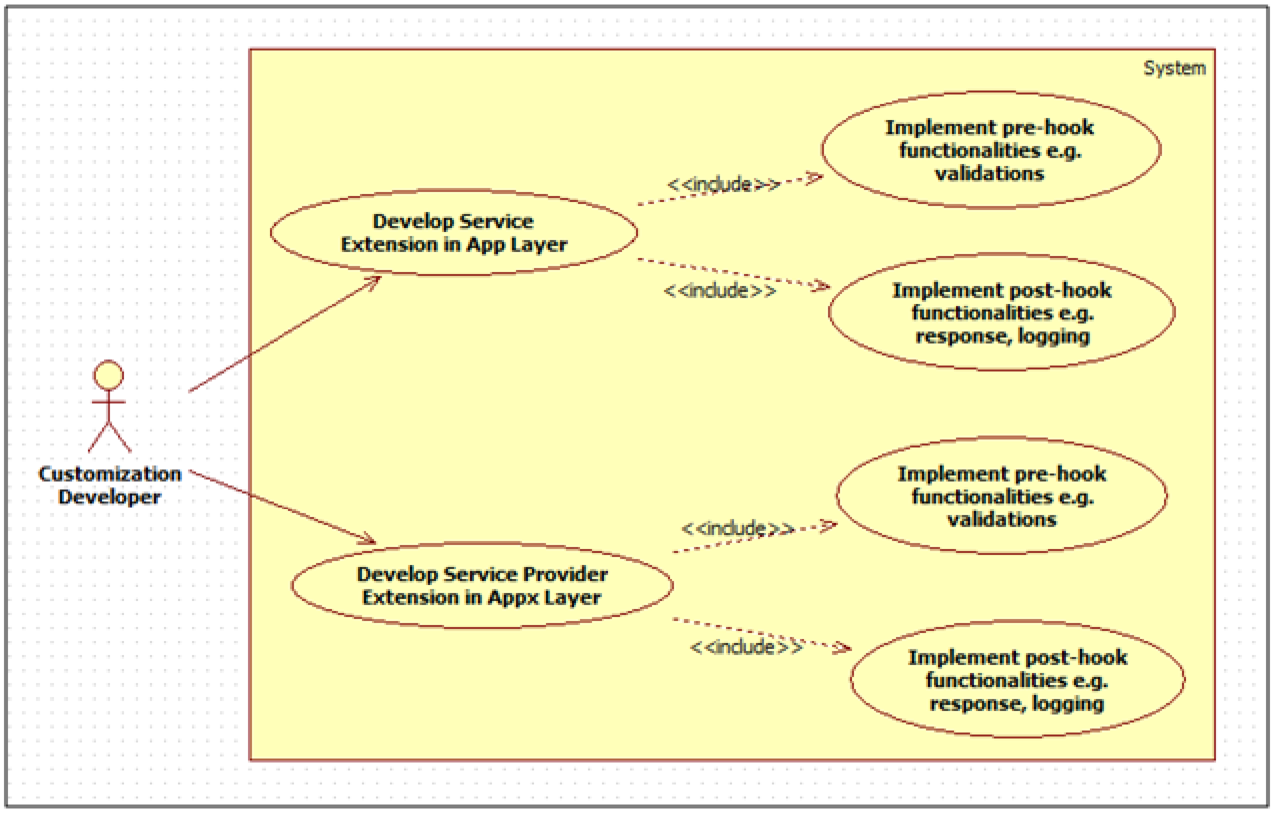
3.1 Extensibility Use Cases

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


System
Customization Developer
Develop Service Extension in App Layer
<<include>>
Implement pre-hook functionalities e.g. validations
<<include>>
Implement post-hook functionalities e.g. response, logging
Develop Service Provider Extension in Appx Layer
<<include>>
Implement pre-hook functionalities e.g. validations
<<include>>
Implement post-hook functionalities e.g. response, logging

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.

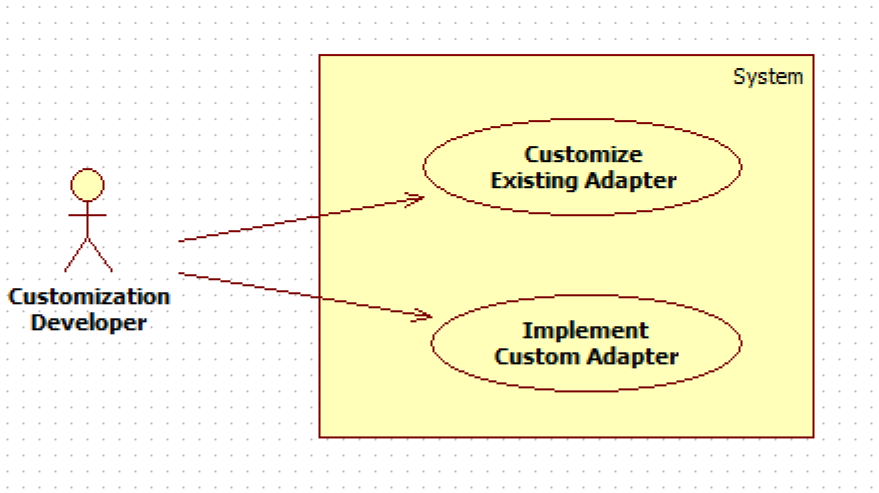
Service Provider Extension

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



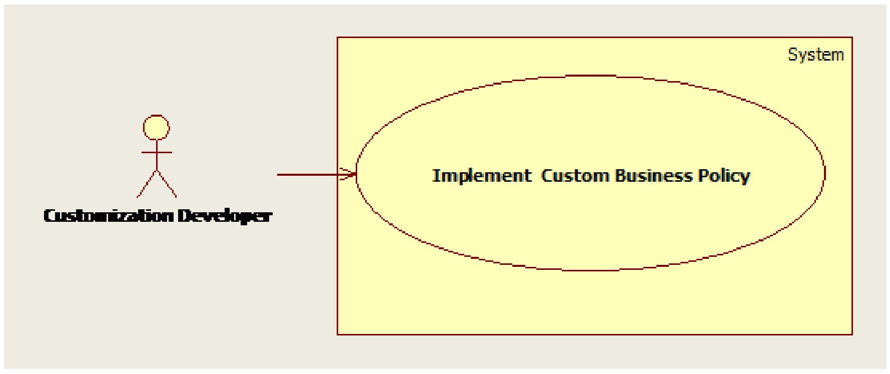
3.1.3 Extending Business Policy

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.

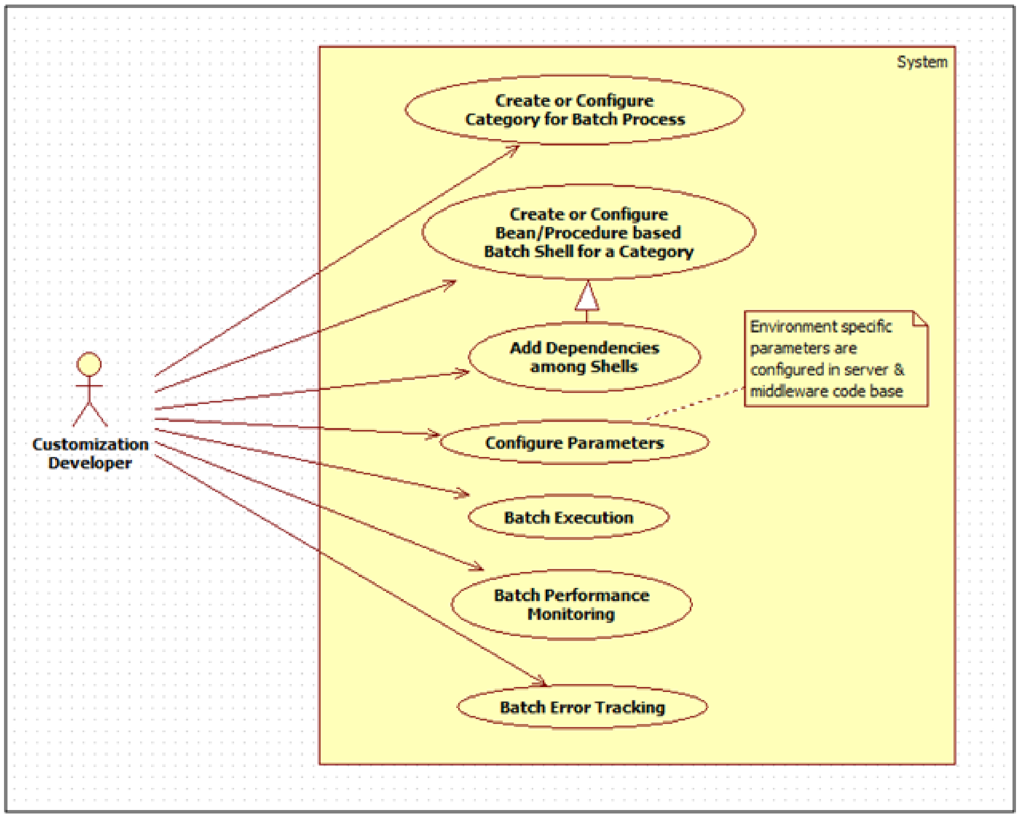
- **Client:** The UI of the screen in which the additional data needs to be captured has to be modified for the additional fields. The view-service linkage also needs to be modified for transferring the additional data.
- **Host:** On the host side, the ORM and POJO for the entity have to be modified 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

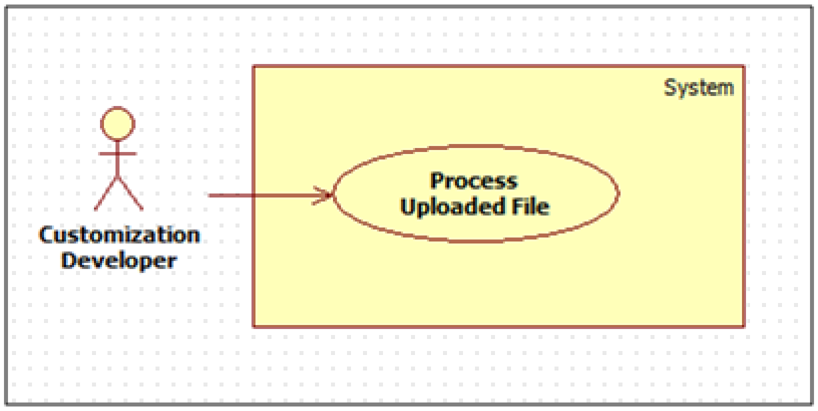

System
Create or Configure Category for Batch Process
Create or Configure Bean/Procedure based Batch Shell for a Category
Add Dependencies among Shells
Environment specific parameters are configured in server & middleware code base
Configure Parameters
Batch Execution
Batch Performance Monitoring
Batch Error Tracking
Customization Developer

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

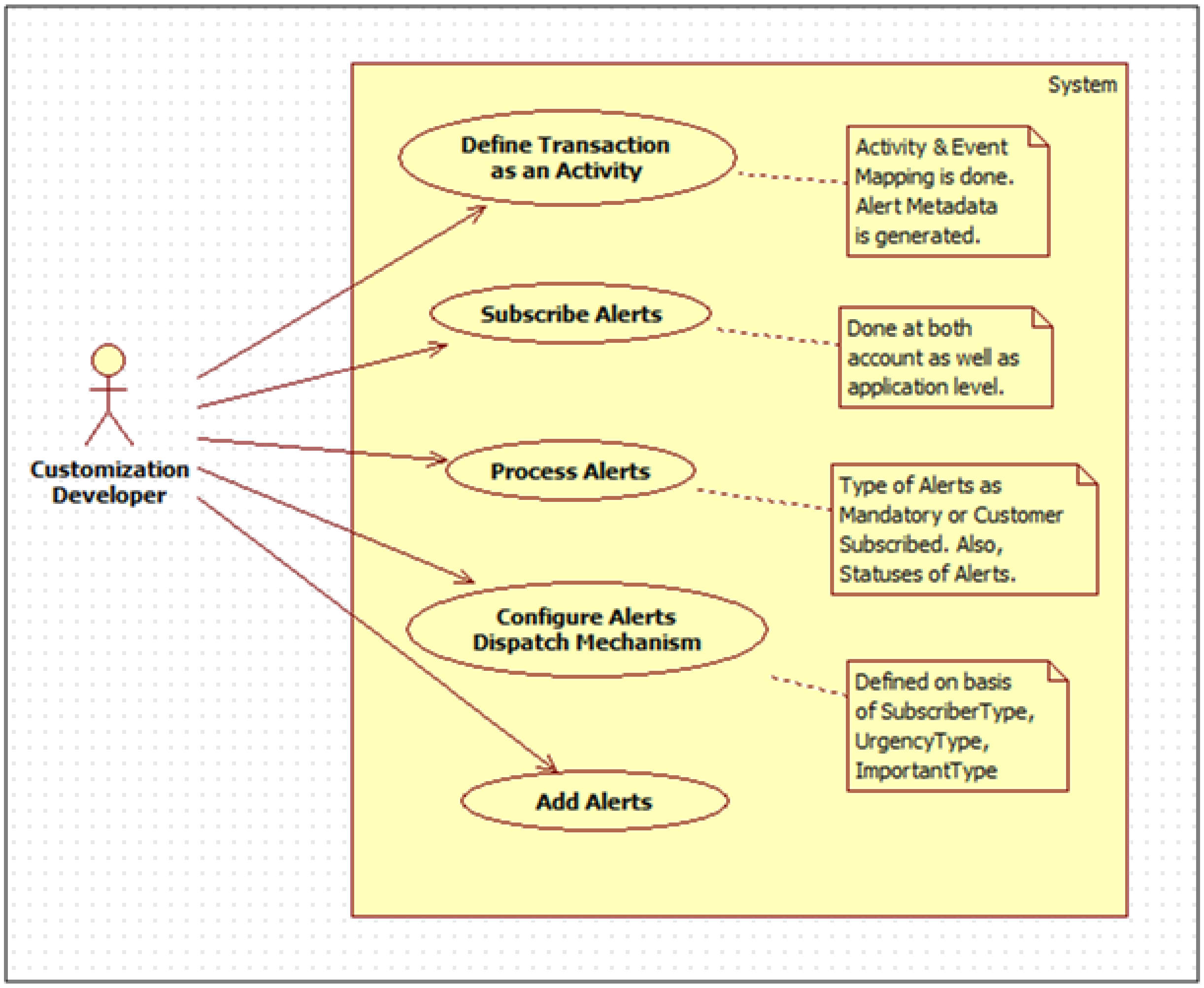


3.1.7 Alert Extension

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

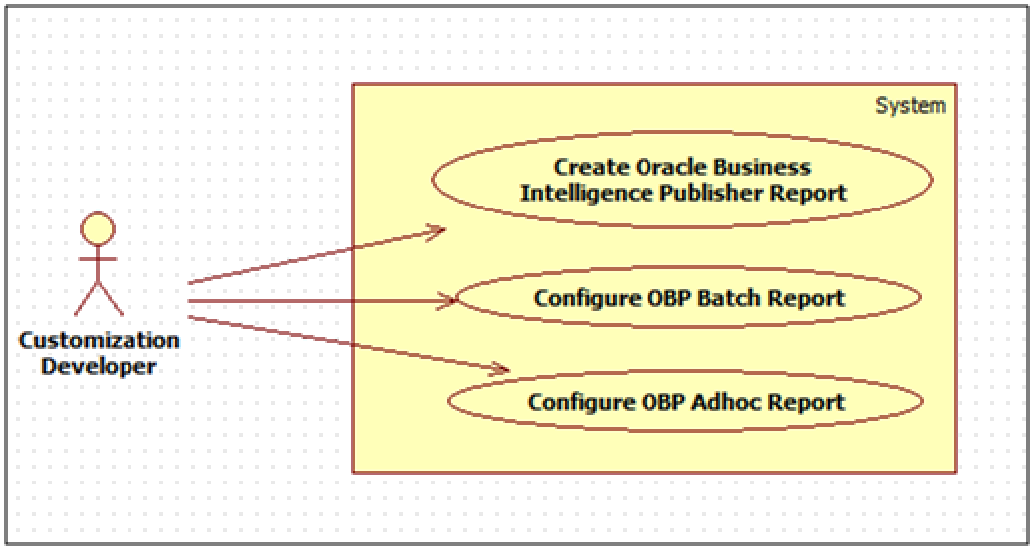

System
Define Transaction as an Activity
Subscribe Alerts
Process Alerts
Configure Alerts Dispatch Mechanism
Add Alerts
Customization Developer
Activity & Event Mapping is done. Alert Metadata is generated.
Done at both account as well as application level.
Type of Alerts as Mandatory or Customer Subscribed. Also, Statuses of Alerts.
Defined on basis of SubscriberType, UrgencyType, ImportantType

3.1.8 Create New Reports

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.

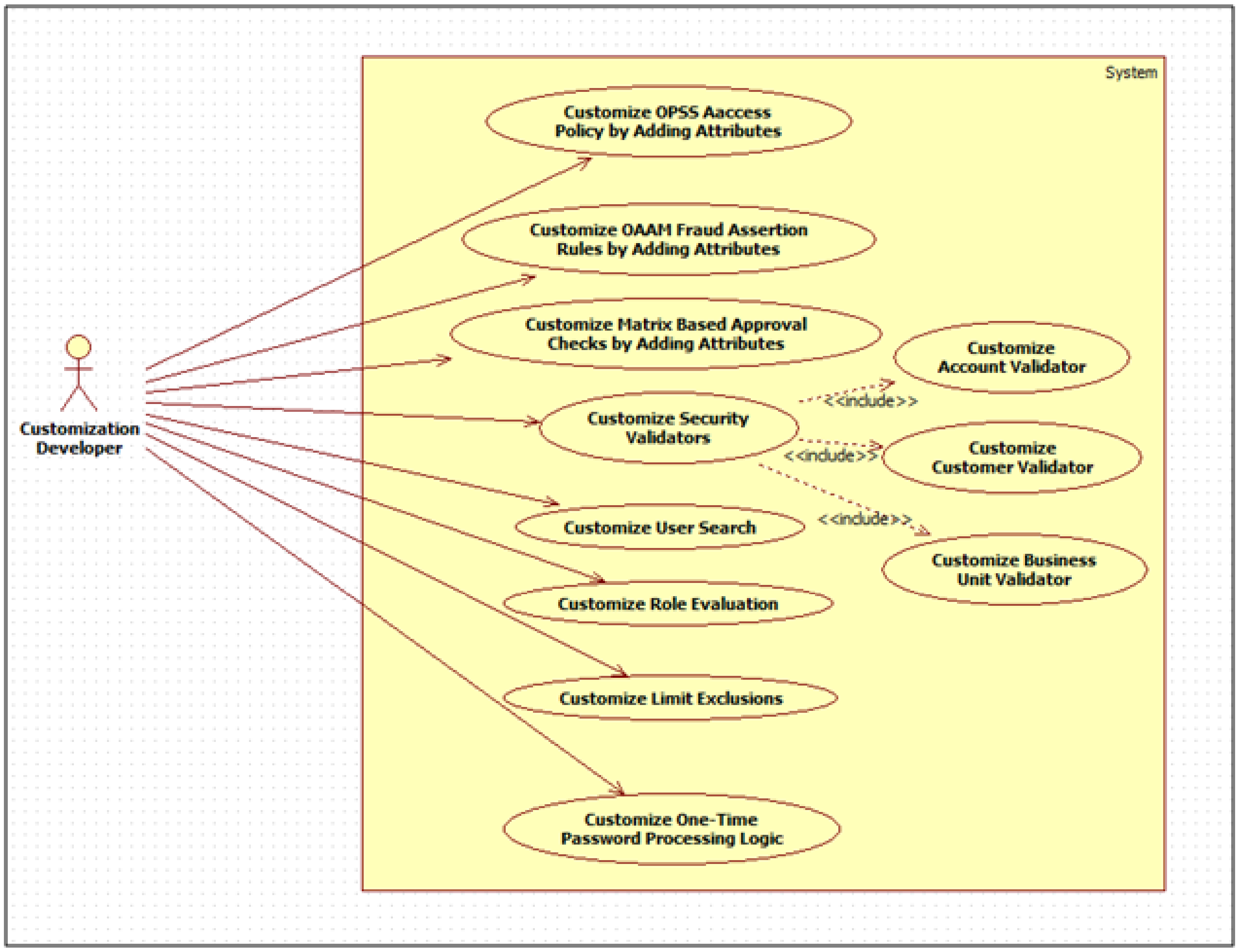
Figure 3–7 Creating New Reports


System
Create Oracle Business Intelligence Publisher Report
Configure OBP Batch Report
Configure OBP Adhoc Report
Customization Developer

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


System
Customization Developer
Customize OPSS Aaccess Policy by Adding Attributes
Customize OAAM Fraud Assertion Rules by Adding Attributes
Customize Matrix Based Approval Checks by Adding Attributes
Customize Security Validators
Customize User Search
Customize Role Evaluation
Customize Limit Exclusions
Customize One-Time Password Processing Logic
Customize Account Validator
Customize Customer Validator
Customize Business Unit Validator
<<include>>
<<include>>
<<include>>

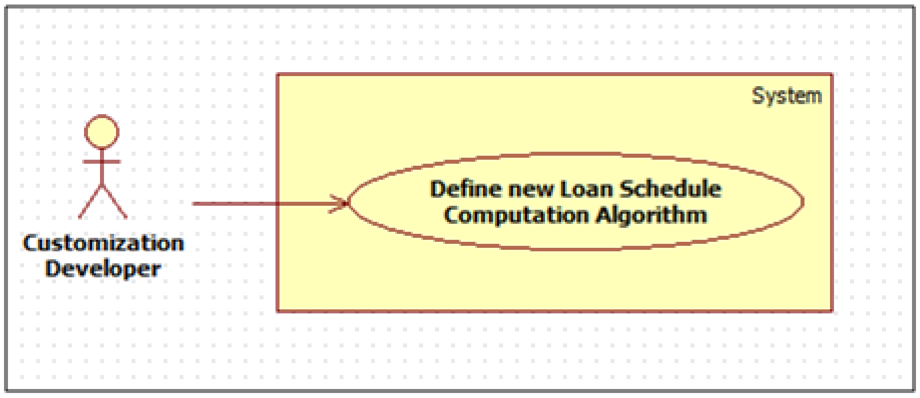
OBP application therefore provides functionality where there is a provision for customizing the security attributes or functions. For example:

- 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.

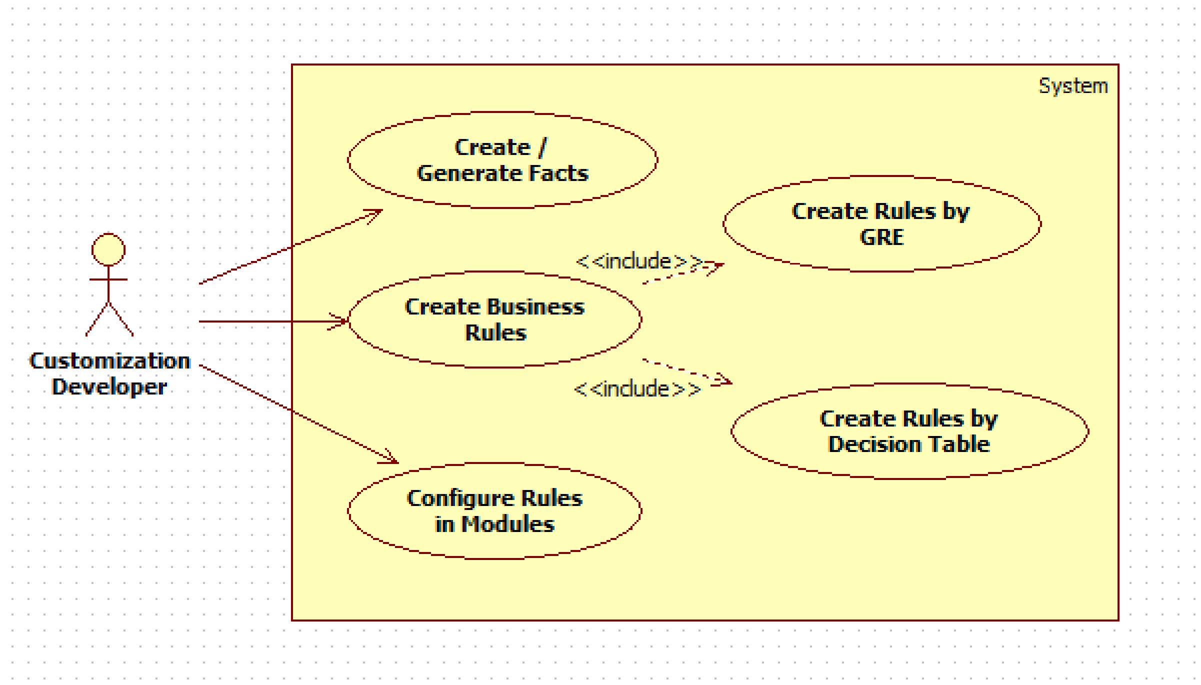
Figure 3–9 Loan Schedule Computation Algorithm



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.

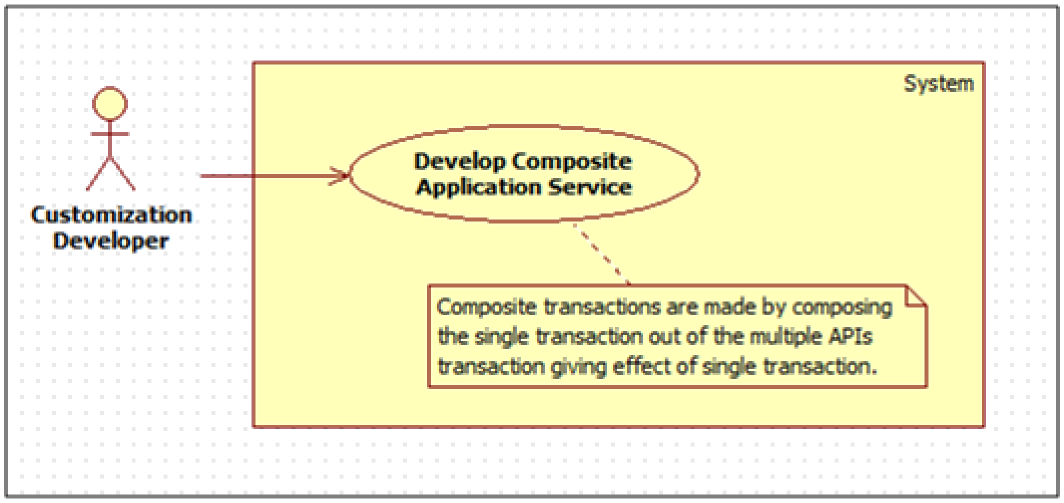
Figure 3–10 Facts and Business Rules



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.

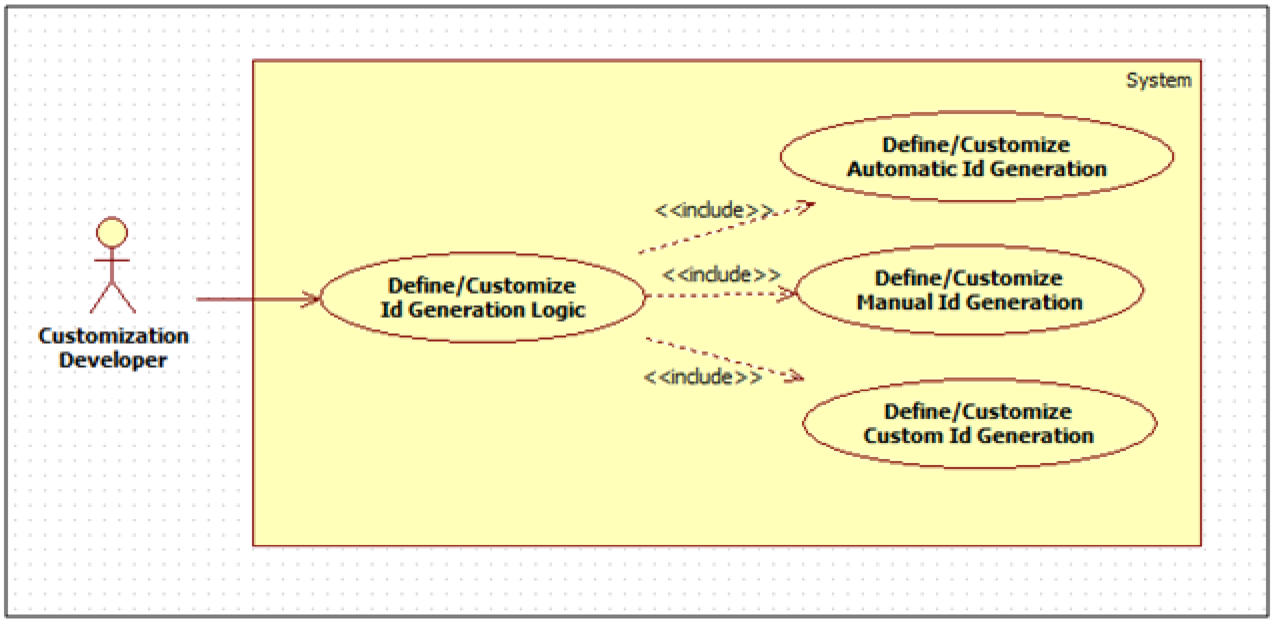
Figure 3–11 Composite Application Service


System
Develop Composite Application Service
Customization Developer
Composite transactions are made by composing the single transaction out of the multiple APIs transaction giving effect of single transaction.

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

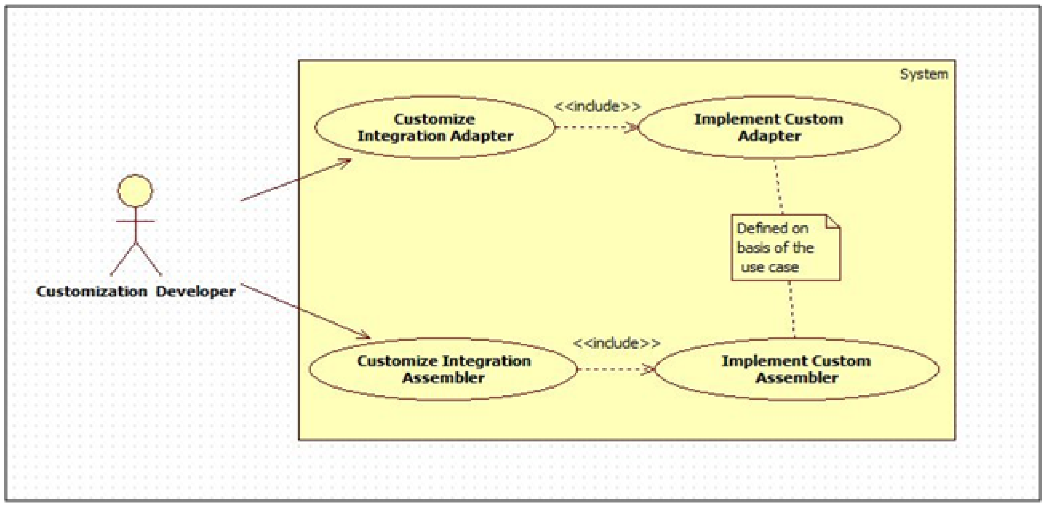

System
Customization Developer
Define/Customize Id Generation Logic
<<include>>
<<include>>
<<include>>
Define/Customize Automatic Id Generation
Define/Customize Manual Id Generation
Define/Customize Custom Id Generation

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


System
Customize Integration Adapter
<<include>>
Implement Custom Adapter
Customization Developer
Defined on basis of the use case
Customize Integration Assembler
<<include>>
Implement Custom Assembler

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.

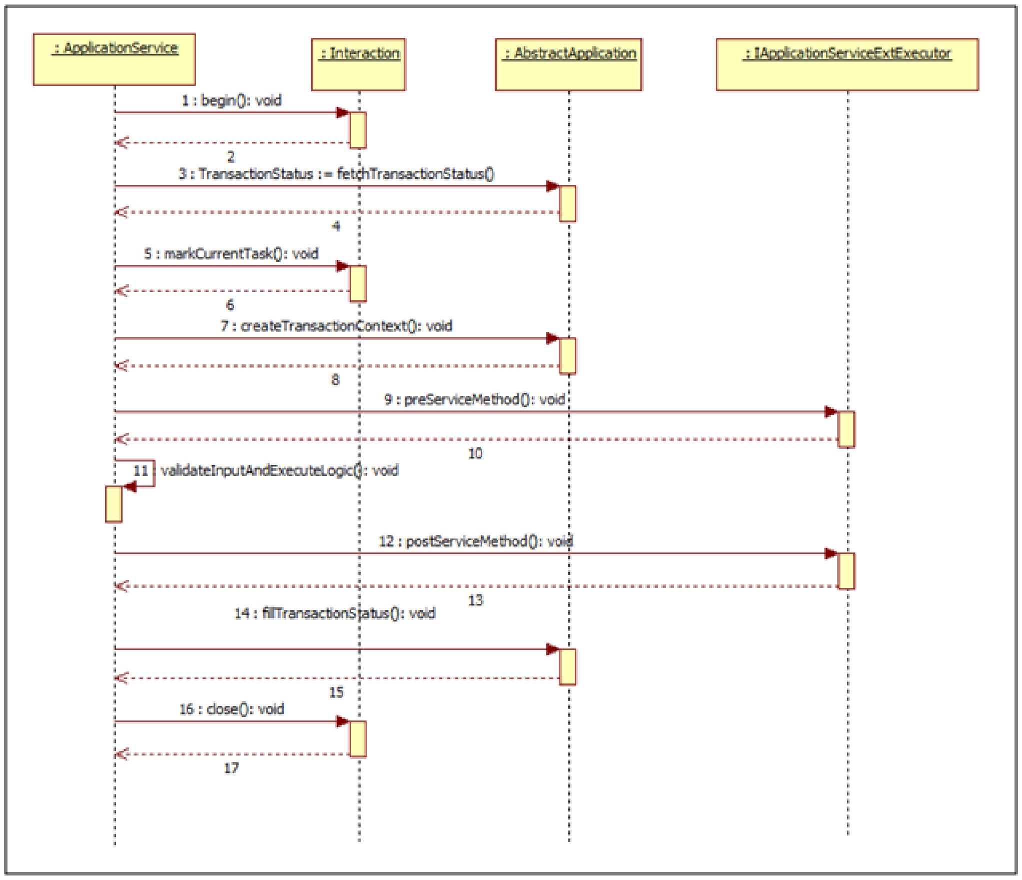
There are few differences in the extensions of the app and appx layer:

- 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


: ApplicationService
: Interaction
: AbstractApplication
: IApplicationServiceExtExecutor
1 : begin(): void
2
3 : TransactionStatus := fetchTransactionStatus()
4
5 : markCurrentTask(): void
6
7 : createTransactionContext(): void
8
9 : preServiceMethod(): void
10
11 : validateInputAndExecuteLogic(): void
12 : postServiceMethod(): void
13
14 : fillTransactionStatus(): void
15
16 : close(): void
17

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.

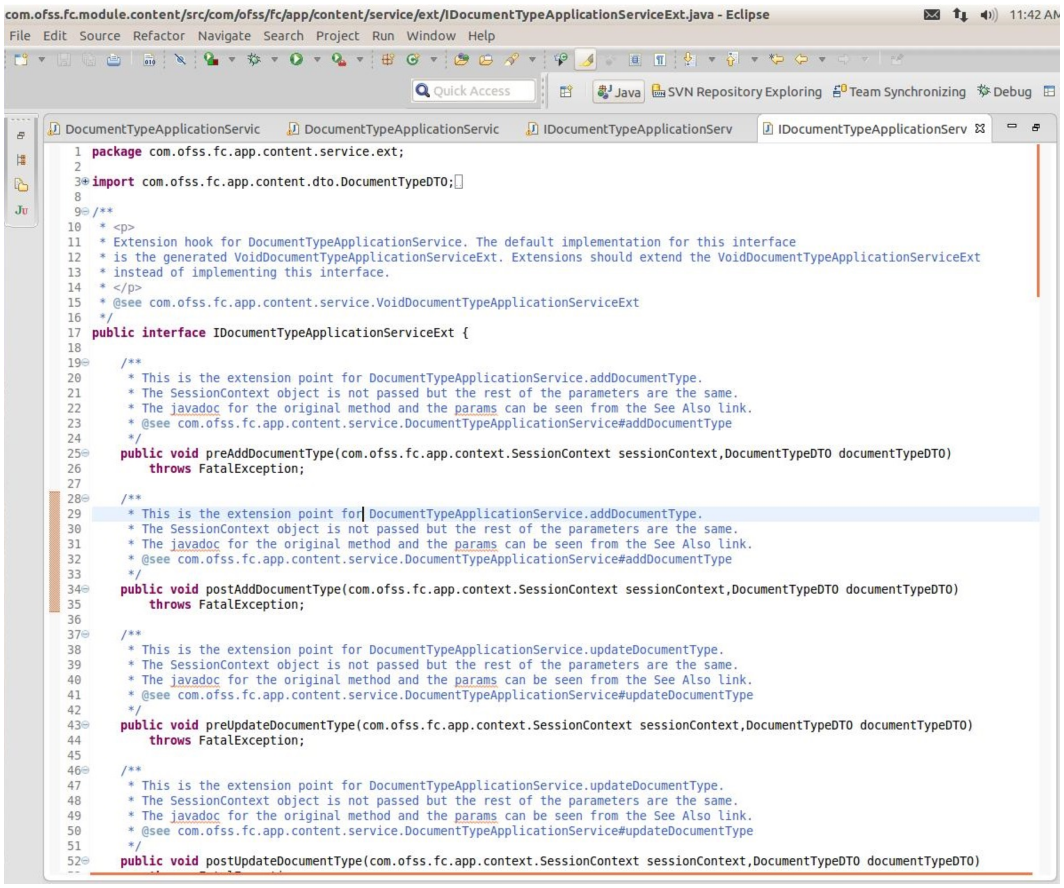
4.1.1 Application Service Extension Interface

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


com.ofss.fc.module.content/src/com/ofss/fc/app/content/service/ext/IDocumentTypeApplicationServiceExt.java - Eclipse
11:42 AM
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
Team Synchronizing
Debug
DocumentTypeApplicationServic
DocumentTypeApplicationServic
IDocumentTypeApplicationServ
IDocumentTypeApplicationServ

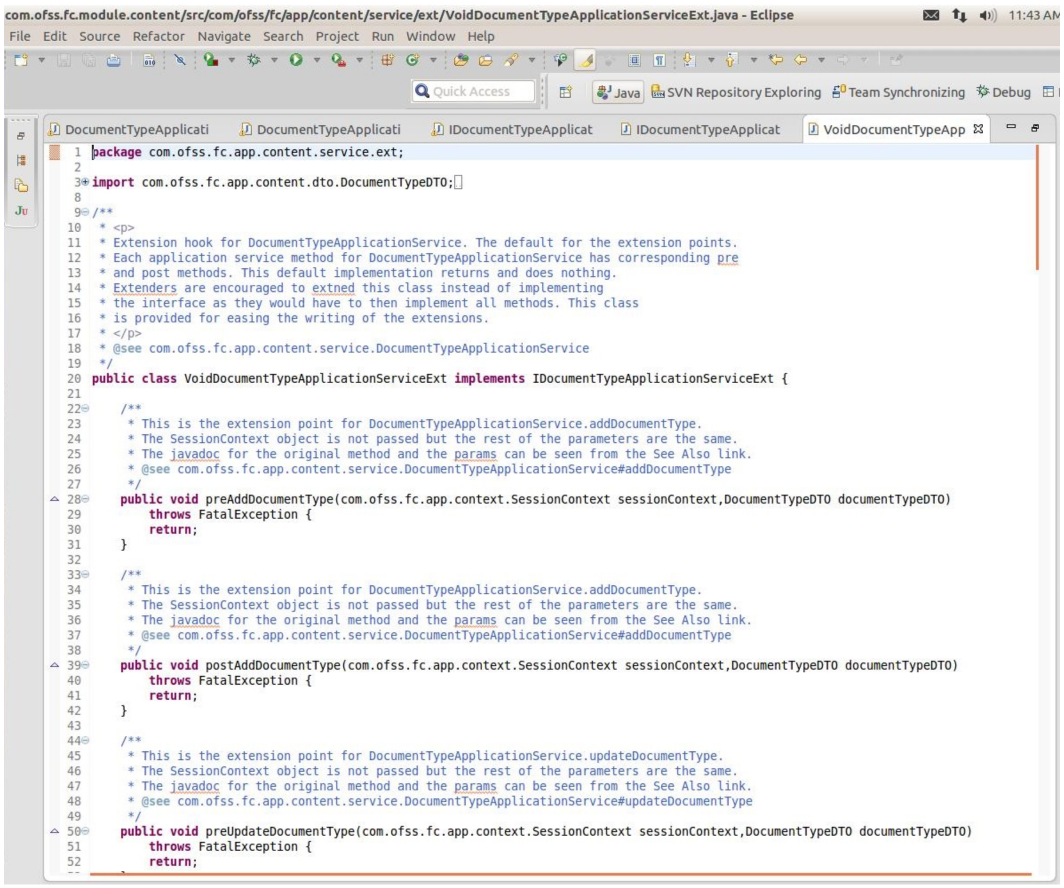
```
package com.ofss.fc.app.content.service.ext;

import com.ofss.fc.app.content.dto.DocumentTypeDTO;

/**
 * <p>
 * Extension hook for DocumentTypeApplicationService. The default implementation for this interface
 * is the generated VoidDocumentTypeApplicationServiceExt. Extensions should extend the VoidDocumentTypeApplicationServiceExt
 * instead of implementing this interface.
 * </p>
 * @see com.ofss.fc.app.content.service.VoidDocumentTypeApplicationServiceExt
 */
public interface IDocumentTypeApplicationServiceExt {

    /**
     * This is the extension point for DocumentTypeApplicationService.addDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#addDocumentType
     */
    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationService.addDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#addDocumentType
     */
    public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationService.updateDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#updateDocumentType
     */
    public void preUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationService.updateDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#updateDocumentType
     */
    public void postUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
```

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


com.ofss.fc.module.content/src/com/ofss/fc/app/content/service/ext/VoidDocumentTypeApplicationServiceExt.java - Eclipse
11:43 AM
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
Team Synchronizing
Debug
DocumentTypeApplicati
DocumentTypeApplicati
IDocumentTypeApplicat
IDocumentTypeApplicat
VoidDocumentTypeApp

```
package com.ofss.fc.app.content.service.ext;

import com.ofss.fc.app.content.dto.DocumentTypeDTO;

/**
 * <p>
 * Extension hook for DocumentTypeApplicationService. The default for the extension points.
 * Each application service method for DocumentTypeApplicationService has corresponding pre
 * and post methods. This default implementation returns and does nothing.
 * Extenders are encouraged to extned this class instead of implementing
 * the interface as they would have to then implement all methods. This class
 * is provided for easing the writing of the extensions.
 * </p>
 * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService
 */
public class VoidDocumentTypeApplicationServiceExt implements IDocumentTypeApplicationServiceExt {

    /**
     * This is the extension point for DocumentTypeApplicationService.addDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#addDocumentType
     */
    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException {
        return;
    }

    /**
     * This is the extension point for DocumentTypeApplicationService.addDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#addDocumentType
     */
    public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException {
        return;
    }

    /**
     * This is the extension point for DocumentTypeApplicationService.updateDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#updateDocumentType
     */
    public void preUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException {
        return;
```

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

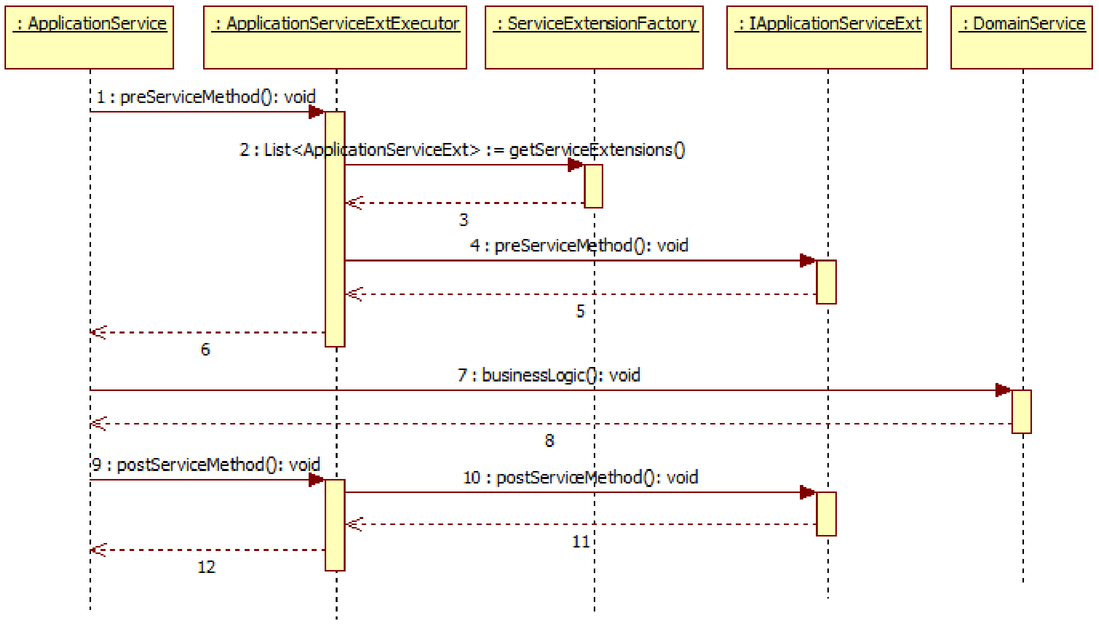
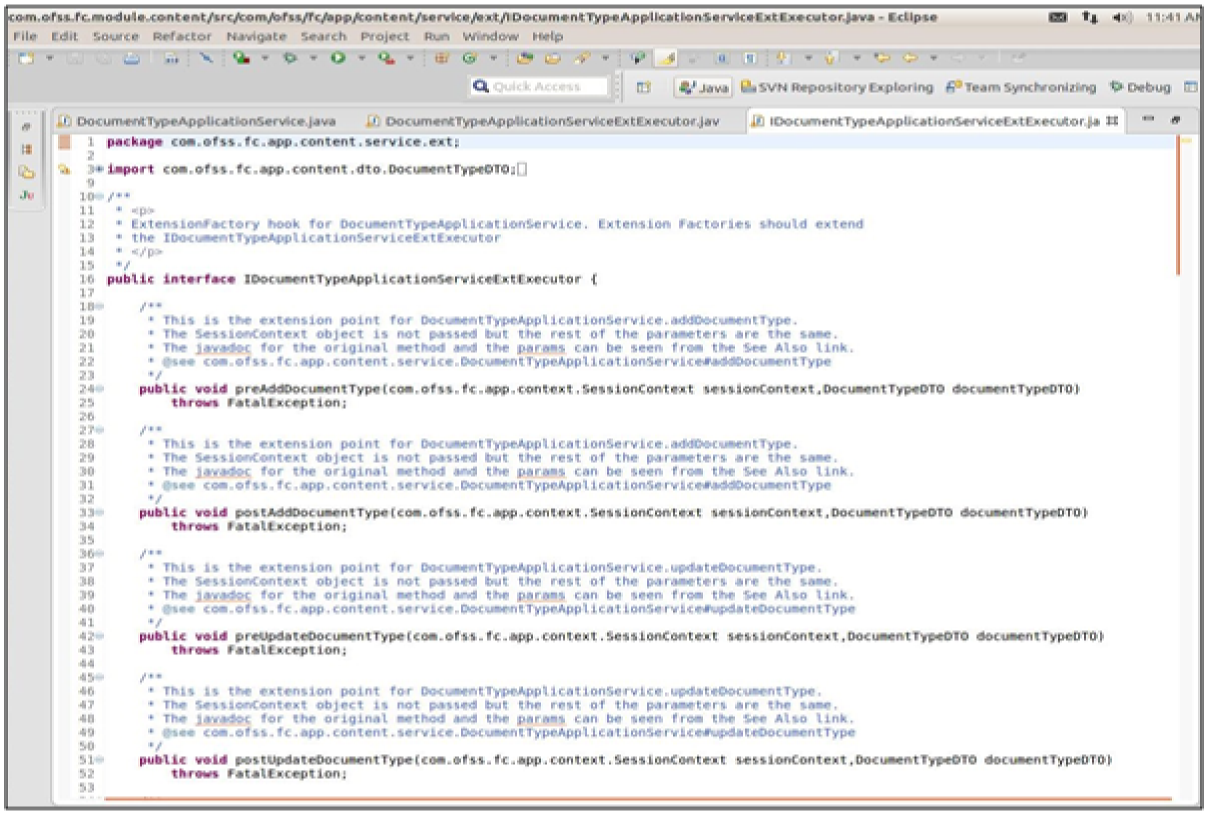
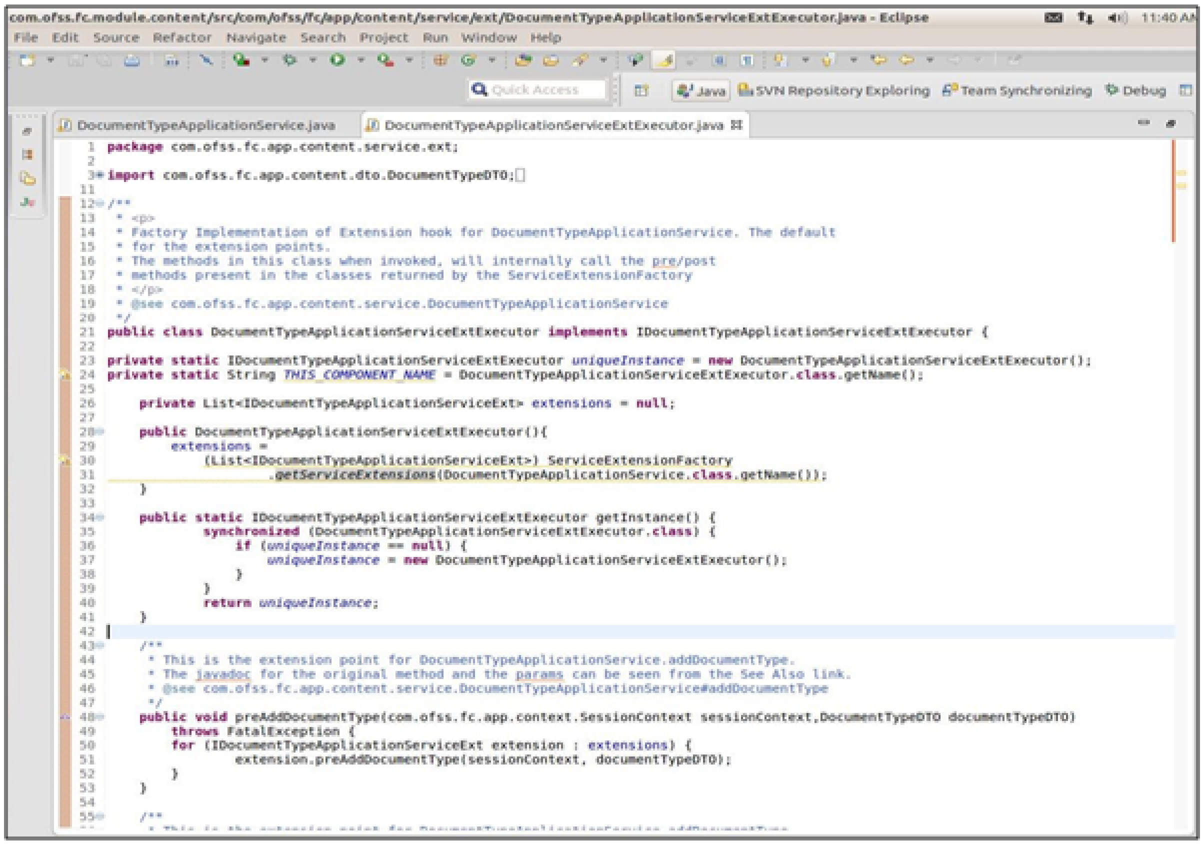

: ApplicationService
: ApplicationServiceExtExecutor
: ServiceExtensionFactory
: IApplicationServiceExt
: DomainService
1 : preServiceMethod(): void
2 : List<ApplicationServiceExt> := getServiceExtensions()
3
4 : preServiceMethod(): void
5
6
7 : businessLogic(): void
8
9 : postServiceMethod(): void
10 : postServiceMethod(): void
11
12

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


com.ofss.fc.module.content/src/com/ofss/fc/app/content/service/ext/IDocumentTypeApplicationServiceExtExecutor.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
Team Synchronizing
Debug
DocumentTypeApplicationService.java
DocumentTypeApplicationServiceExtExecutor.jav
IDocumentTypeApplicationServiceExtExecutor.ja

```
package com.ofss.fc.app.content.service.ext;

import com.ofss.fc.app.content.dto.DocumentTypeDTO;

/**
 * <p>
 * ExtensionFactory hook for DocumentTypeApplicationService. Extension Factories should extend
 * the IDocumentTypeApplicationServiceExtExecutor
 * </p>
 */
public interface IDocumentTypeApplicationServiceExtExecutor {

    /**
     * This is the extension point for DocumentTypeApplicationService.addDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#addDocumentType
     */
    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationService.addDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#addDocumentType
     */
    public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationService.updateDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#updateDocumentType
     */
    public void preUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationService.updateDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#updateDocumentType
     */
    public void postUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException;

```

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


com.ofss.fc.module.content/src/com/ofss/fc/app/content/service/ext/DocumentTypeApplicationServiceExtExecutor.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
Team Synchronizing
Debug
DocumentTypeApplicationService.java
DocumentTypeApplicationServiceExtExecutor.java

```java
package com.ofss.fc.app.content.service.ext;

import com.ofss.fc.app.content.dto.DocumentTypeDTO;

/**
 * <p>
 * Factory Implementation of Extension hook for DocumentTypeApplicationService. The default
 * for the extension points.
 * The methods in this class when invoked, will internally call the pre/post
 * methods present in the classes returned by the ServiceExtensionFactory
 * </p>
 * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService
 */
public class DocumentTypeApplicationServiceExtExecutor implements IDocumentTypeApplicationServiceExtExecutor {

private static IDocumentTypeApplicationServiceExtExecutor uniqueInstance = new DocumentTypeApplicationServiceExtExecutor();
private static String THIS_COMPONENT_NAME = DocumentTypeApplicationServiceExtExecutor.class.getName();

    private List<IDocumentTypeApplicationServiceExt> extensions = null;

    public DocumentTypeApplicationServiceExtExecutor(){
        extensions =
            (List<IDocumentTypeApplicationServiceExt>) ServiceExtensionFactory
                    .getServiceExtensions(DocumentTypeApplicationService.class.getName());
    }

    public static IDocumentTypeApplicationServiceExtExecutor getInstance() {
            synchronized (DocumentTypeApplicationServiceExtExecutor.class) {
                if (uniqueInstance == null) {
                    uniqueInstance = new DocumentTypeApplicationServiceExtExecutor();
                }
            }
            return uniqueInstance;
    }

    /**
     * This is the extension point for DocumentTypeApplicationService.addDocumentType.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.content.service.DocumentTypeApplicationService#addDocumentType
     */
    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
        throws FatalException {
        for (IDocumentTypeApplicationServiceExt extension : extensions) {
                extension.preAddDocumentType(sessionContext, documentTypeDTO);
        }
    }

    /**
```

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
```

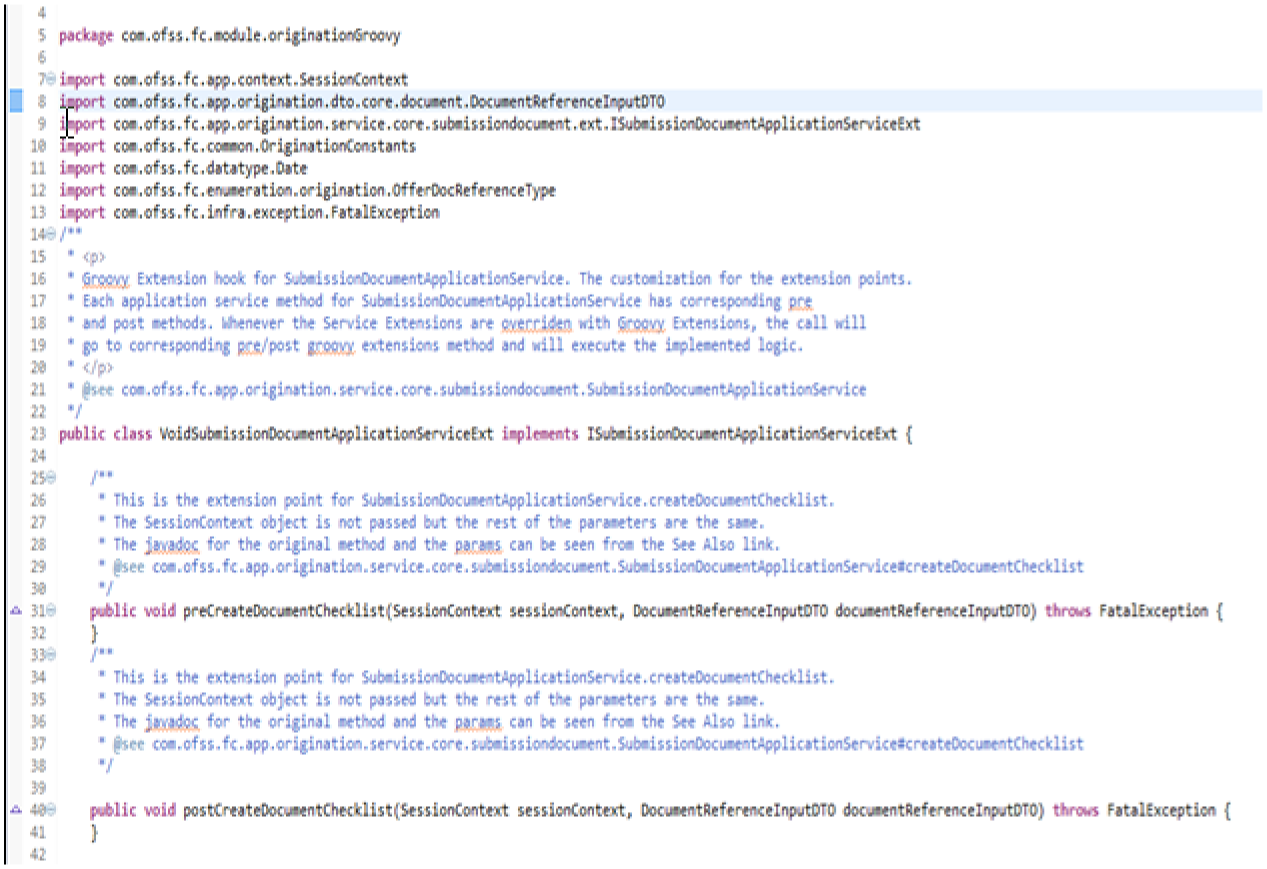
It is possible to configure multiple implementations of pre / post extensions for an executing service in this layer. This is achieved with the help of the extension executor which has the capability to loop through a set of extension implementations which conform to the extension interface which is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

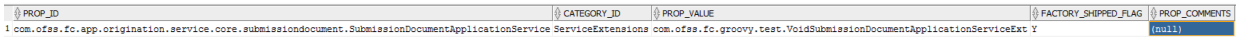
Figure 4–7 Application Service Extension Using Groovy



```

package com.ofss.fc.module.originationGroovy

import com.ofss.fc.app.context.SessionContext
import com.ofss.fc.app.origination.dto.core.document.DocumentReferenceInputDTO
import com.ofss.fc.app.origination.service.core.submissiondocument.ext.ISubmissionDocumentApplicationServiceExt
import com.ofss.fc.common.OriginationConstants
import com.ofss.fc.datatype.Date
import com.ofss.fc.enumeration.origination.OfferDocReferenceType
import com.ofss.fc.infra.exception.FatalException
/**
 * <p>
 * Groovy Extension hook for SubmissionDocumentApplicationService. The customization for the extension points.
 * Each application service method for SubmissionDocumentApplicationService has corresponding pre
 * and post methods. Whenever the Service Extensions are overriden with Groovy Extensions, the call will
 * go to corresponding pre/post groovy extensions method and will execute the implemented logic.
 * </p>
 * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService
 */
public class VoidSubmissionDocumentApplicationServiceExt implements ISubmissionDocumentApplicationServiceExt {

    /**
     * This is the extension point for SubmissionDocumentApplicationService.createDocumentChecklist.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService#createDocumentChecklist
     */
    public void preCreateDocumentChecklist(SessionContext sessionContext, DocumentReferenceInputDTO documentReferenceInputDTO) throws FatalException {
    }
    /**
     * This is the extension point for SubmissionDocumentApplicationService.createDocumentChecklist.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService#createDocumentChecklist
     */

    public void postCreateDocumentChecklist(SessionContext sessionContext, DocumentReferenceInputDTO documentReferenceInputDTO) throws FatalException {
    }

```

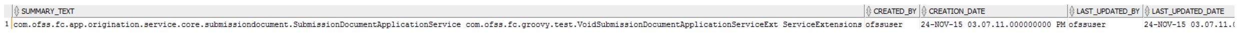
Provide the fully qualified name of the above groovy implementation in flx_fw_config_all_b against the corresponding service extension prop_id and category_id.

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)

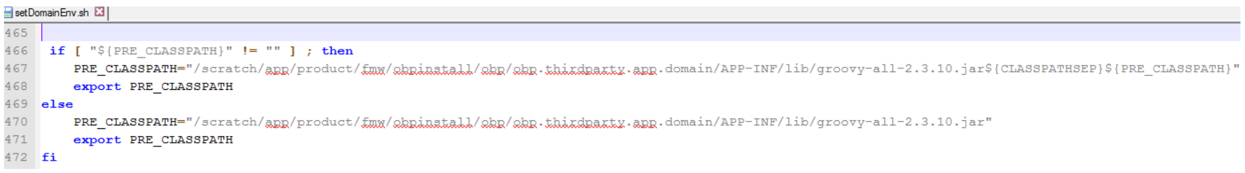
Figure 4–9 SUMMARY_TEXT



	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



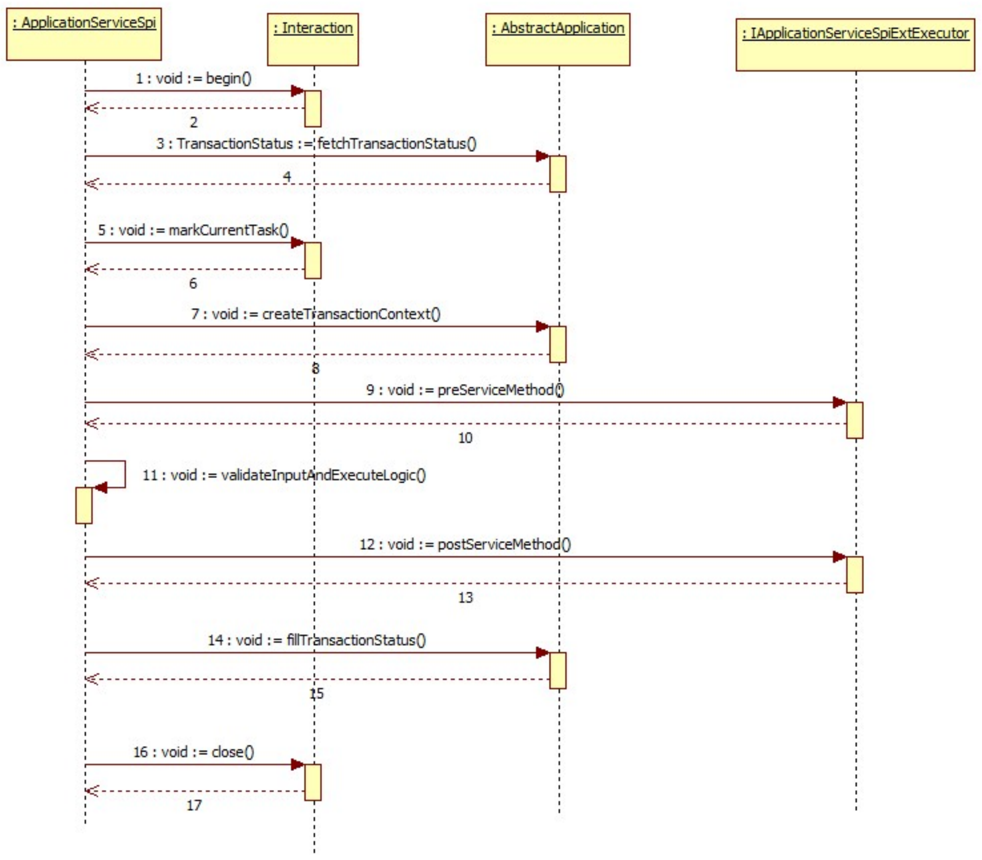
setDomainEnv.sh

```

 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
```

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

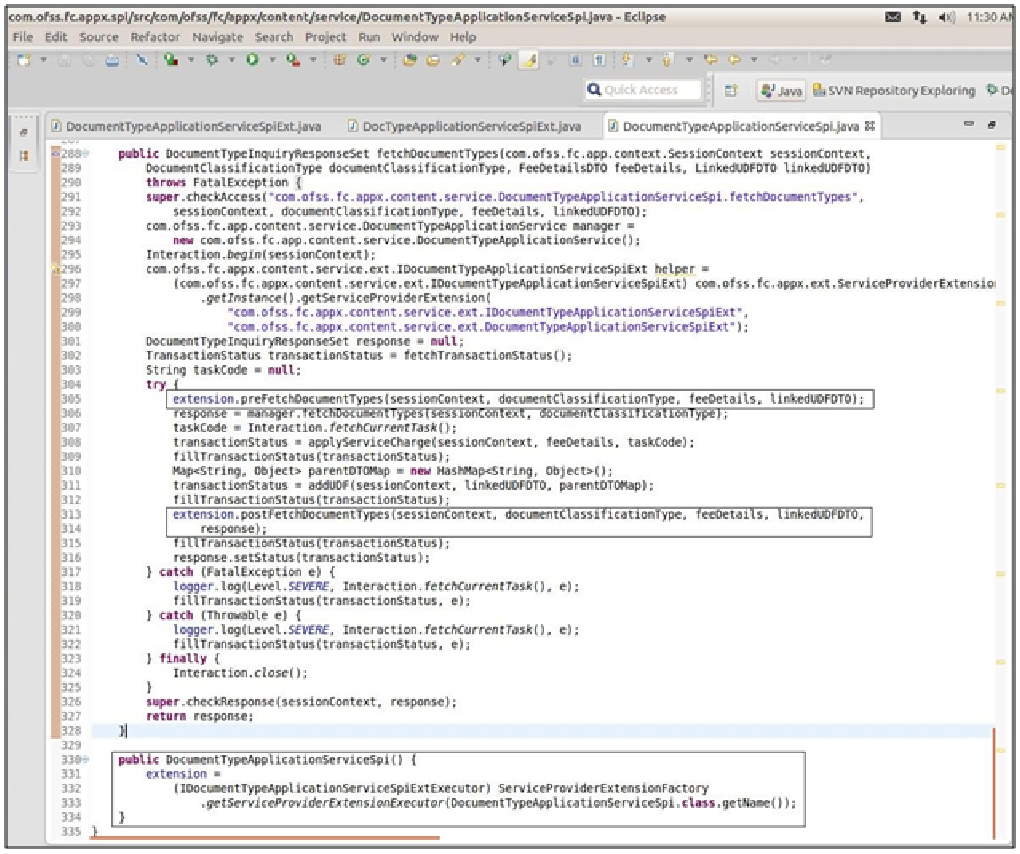

: ApplicationServiceSpi
: Interaction
: AbstractApplication
: IApplicationServiceSpiExtExecutor
1 : void := begin()
2
3 : TransactionStatus := fetchTransactionStatus()
4
5 : void := markCurrentTask()
6
7 : void := createTransactionContext()
8
9 : void := preServiceMethod()
10
11 : void := validateInputAndExecuteLogic()
12 : void := postServiceMethod()
13
14 : void := fillTransactionStatus()
15
16 : void := close()
17

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


com.ofss.fc.appx.spi/src/com/ofss/fc/appx/content/service/DocumentTypeApplicationServiceSpi.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
DocumentTypeApplicationServiceSpiExt.java
DocTypeApplicationServiceSpiExt.java
DocumentTypeApplicationServiceSpi.java

```
    public DocumentTypeInquiryResponseSet fetchDocumentTypes(com.ofss.fc.app.context.SessionContext sessionContext,
        DocumentClassificationType documentClassificationType, FeeDetailsDTO feeDetails, LinkedUDFDTO linkedUDFDTO)
        throws FatalException {
        super.checkAccess("com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi.fetchDocumentTypes",
            sessionContext, documentClassificationType, feeDetails, linkedUDFDTO);
        com.ofss.fc.app.content.service.DocumentTypeApplicationService manager =
            new com.ofss.fc.app.content.service.DocumentTypeApplicationService();
        Interaction.begin(sessionContext);
        com.ofss.fc.appx.content.service.ext.IDocumentTypeApplicationServiceSpiExt helper =
            (com.ofss.fc.appx.content.service.ext.IDocumentTypeApplicationServiceSpiExt) com.ofss.fc.appx.ext.ServiceProviderExtensio
                .getInstance().getServiceProviderExtension(
                    "com.ofss.fc.appx.content.service.ext.IDocumentTypeApplicationServiceSpiExt",
                    "com.ofss.fc.appx.content.service.ext.DocumentTypeApplicationServiceSpiExt");
        DocumentTypeInquiryResponseSet response = null;
        TransactionStatus transactionStatus = fetchTransactionStatus();
        String taskCode = null;
        try {
            extension.preFetchDocumentTypes(sessionContext, documentClassificationType, feeDetails, linkedUDFDTO);
            response = manager.fetchDocumentTypes(sessionContext, documentClassificationType);
            taskCode = Interaction.fetchCurrentTask();
            transactionStatus = applyServiceCharge(sessionContext, feeDetails, taskCode);
            fillTransactionStatus(transactionStatus);
            Map<String, Object> parentDTOMap = new HashMap<String, Object>();
            transactionStatus = addUDF(sessionContext, linkedUDFDTO, parentDTOMap);
            fillTransactionStatus(transactionStatus);
            extension.postFetchDocumentTypes(sessionContext, documentClassificationType, feeDetails, linkedUDFDTO,
                response);
            fillTransactionStatus(transactionStatus);
            response.setStatus(transactionStatus);
        } 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 {
            Interaction.close();
        }
        super.checkResponse(sessionContext, response);
        return response;
    }

    public DocumentTypeApplicationServiceSpi() {
        extension =
            (IDocumentTypeApplicationServiceSpiExtExecutor) ServiceProviderExtensionFactory
                .getServiceProviderExtensionExecutor(DocumentTypeApplicationServiceSpi.class.getName());
    }
}
```

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

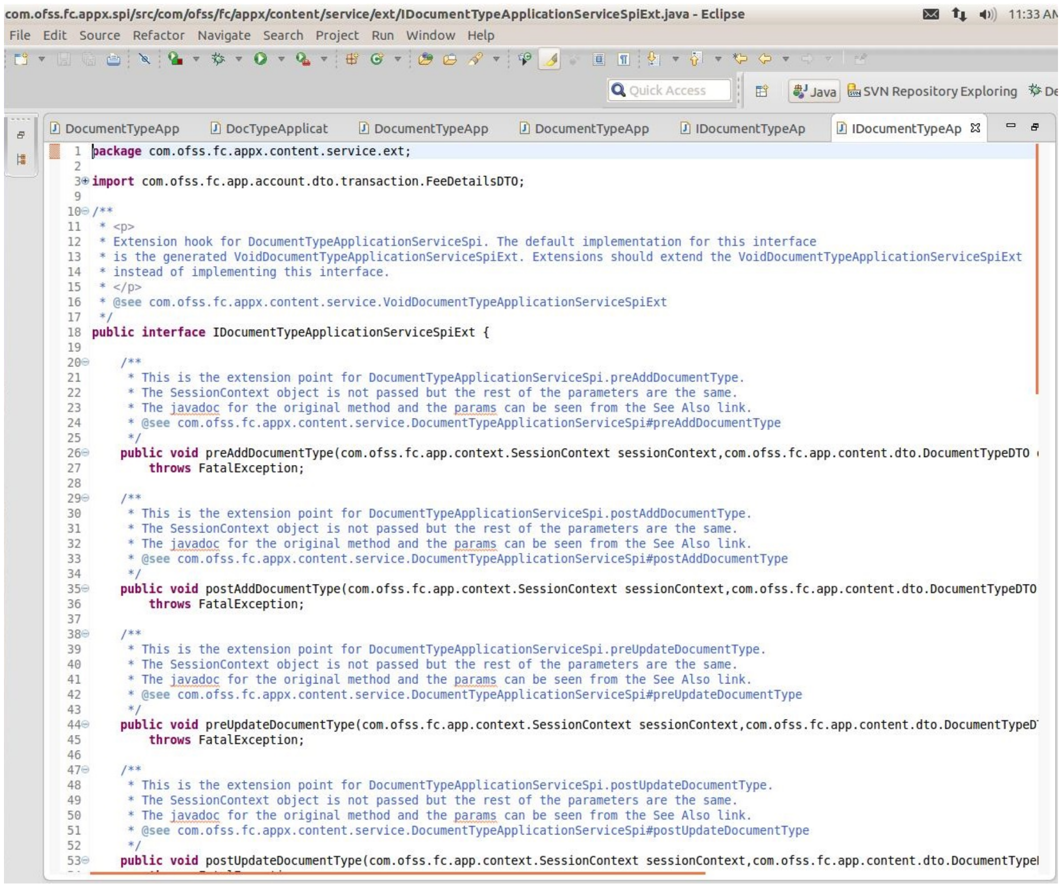
4.2.1 Extended Application Service Extension Interface

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


com.ofss.fc.appx.spi/src/com/ofss/fc/appx/content/service/ext/IDocumentTypeApplicationServiceSpiExt.java - Eclipse
11:33 AM
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
DocumentTypeApp
DocTypeApplicat
DocumentTypeApp
DocumentTypeApp
IDocumentTypeAp
IDocumentTypeAp

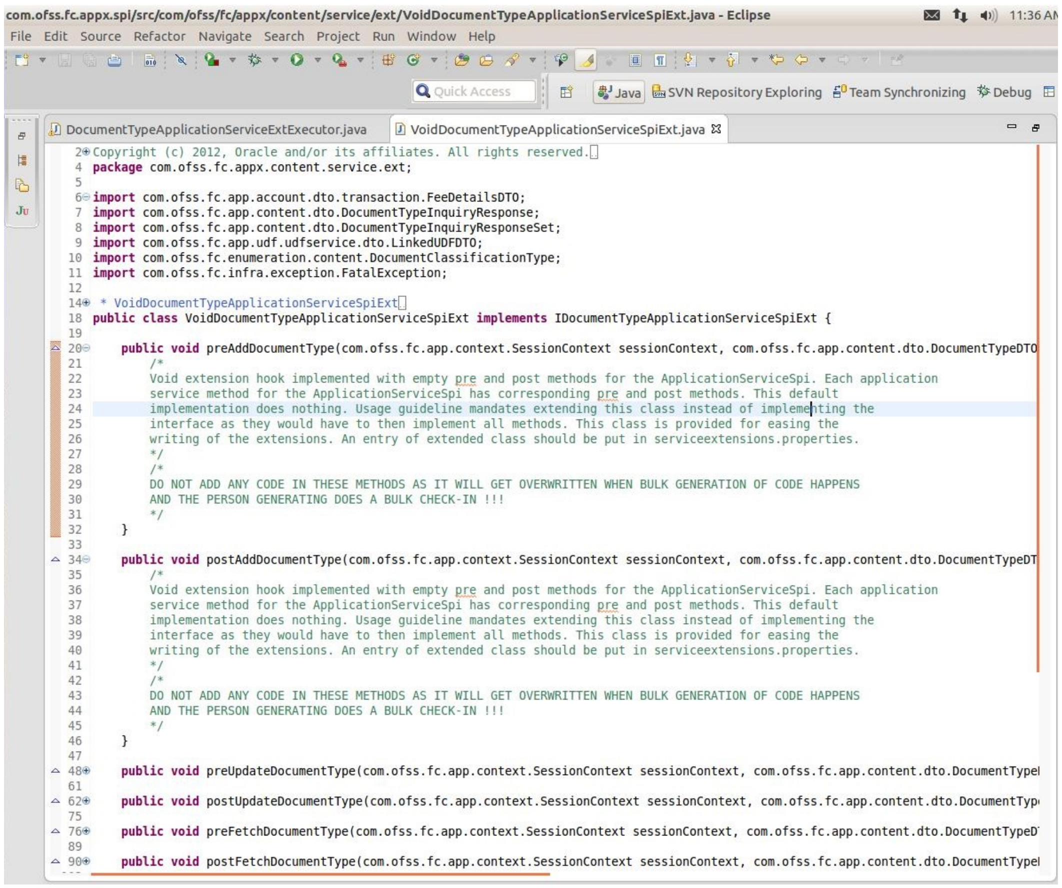
```
package com.ofss.fc.appx.content.service.ext;

import com.ofss.fc.app.account.dto.transaction.FeeDetailsDTO;

/**
 * <p>
 * Extension hook for DocumentTypeApplicationServiceSpi. The default implementation for this interface
 * is the generated VoidDocumentTypeApplicationServiceSpiExt. Extensions should extend the VoidDocumentTypeApplicationServiceSpiExt
 * instead of implementing this interface.
 * </p>
 * @see com.ofss.fc.appx.content.service.VoidDocumentTypeApplicationServiceSpiExt
 */
public interface IDocumentTypeApplicationServiceSpiExt {

    /**
     * This is the extension point for DocumentTypeApplicationServiceSpi.preAddDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#preAddDocumentType
     */
    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeDTO
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationServiceSpi.postAddDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#postAddDocumentType
     */
    public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeDTO
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationServiceSpi.preUpdateDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#preUpdateDocumentType
     */
    public void preUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeD
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationServiceSpi.postUpdateDocumentType.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#postUpdateDocumentType
     */
    public void postUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentType
```

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


com.ofss.fc.appx.spi/src/com/ofss/fc/appx/content/service/ext/VoidDocumentTypeApplicationServiceSpiExt.java - Eclipse
11:36 AM
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
Team Synchronizing
Debug
DocumentTypeApplicationServiceExtExecutor.java
VoidDocumentTypeApplicationServiceSpiExt.java

```
Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.appx.content.service.ext;

import com.ofss.fc.app.account.dto.transaction.FeeDetailsDTO;
import com.ofss.fc.app.content.dto.DocumentTypeInquiryResponse;
import com.ofss.fc.app.content.dto.DocumentTypeInquiryResponseSet;
import com.ofss.fc.app.udf.udfservice.dto.LinkedUDFDTO;
import com.ofss.fc.enumeration.content.DocumentClassificationType;
import com.ofss.fc.infra.exception.FatalException;

 * VoidDocumentTypeApplicationServiceSpiExt
public class VoidDocumentTypeApplicationServiceSpiExt implements IDocumentTypeApplicationServiceSpiExt {

    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentTypeDTO
        /*
        Void extension hook implemented with empty pre and post methods for the ApplicationServiceSpi. Each application
        service method for the ApplicationServiceSpi has corresponding pre and post methods. This default
        implementation does nothing. Usage guideline mandates extending this class instead of implementing the
        interface as they would have to then implement all methods. This class is provided for easing the
        writing of the extensions. An entry of extended class should be put in serviceextensions.properties.
        */
        /*
        DO NOT ADD ANY CODE IN THESE METHODS AS IT WILL GET OVERWRITTEN WHEN BULK GENERATION OF CODE HAPPENS
        AND THE PERSON GENERATING DOES A BULK CHECK-IN !!!
        */
    }

    public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentTypeDT
        /*
        Void extension hook implemented with empty pre and post methods for the ApplicationServiceSpi. Each application
        service method for the ApplicationServiceSpi has corresponding pre and post methods. This default
        implementation does nothing. Usage guideline mandates extending this class instead of implementing the
        interface as they would have to then implement all methods. This class is provided for easing the
        writing of the extensions. An entry of extended class should be put in serviceextensions.properties.
        */
        /*
        DO NOT ADD ANY CODE IN THESE METHODS AS IT WILL GET OVERWRITTEN WHEN BULK GENERATION OF CODE HAPPENS
        AND THE PERSON GENERATING DOES A BULK CHECK-IN !!!
        */
    }

    public void preUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentTypeI

    public void postUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentTyp

    public void preFetchDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentTypeD

    public void postFetchDocumentType(com.ofss.fc.app.context.SessionContext sessionContext, com.ofss.fc.app.content.dto.DocumentTypeI
```

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

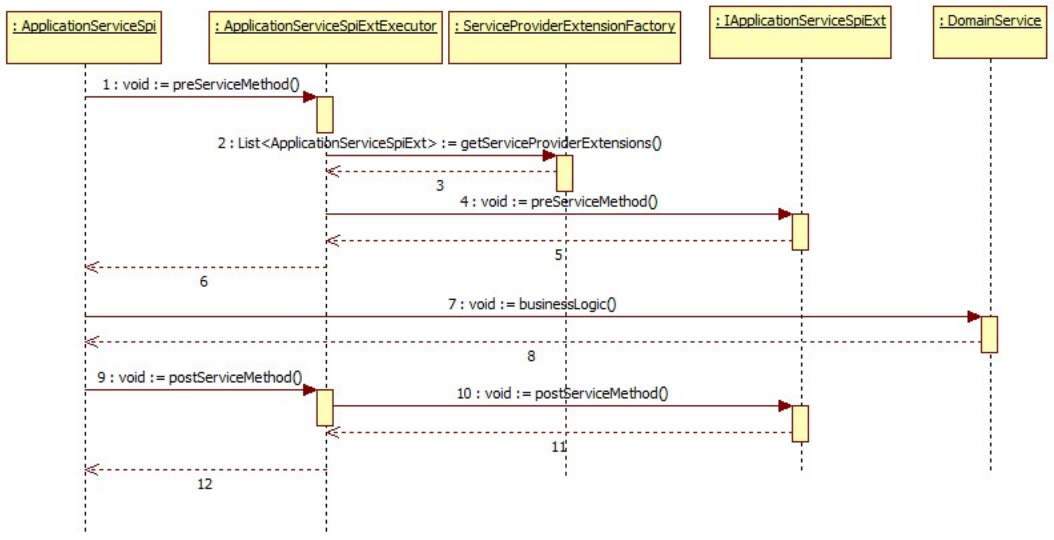
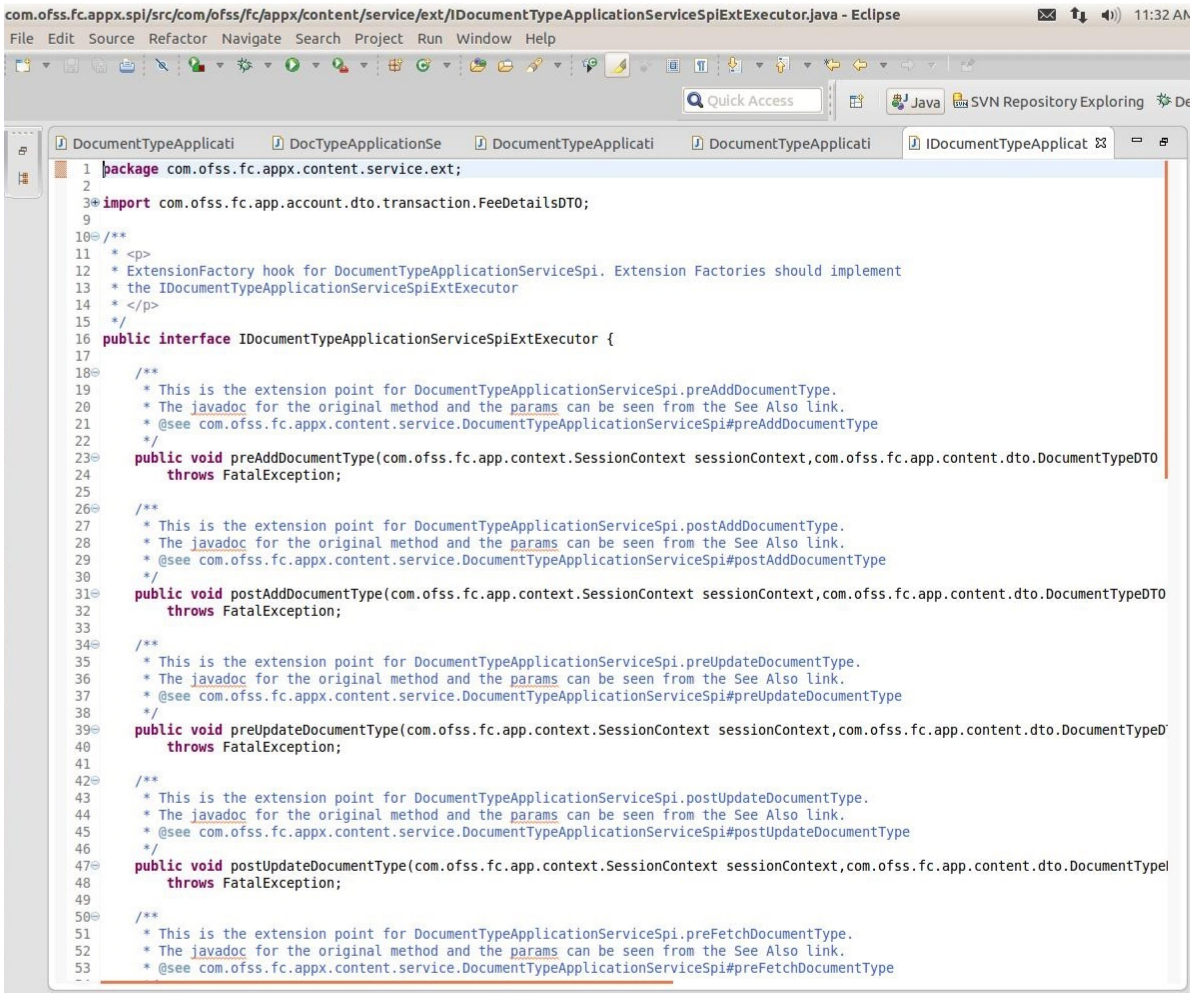
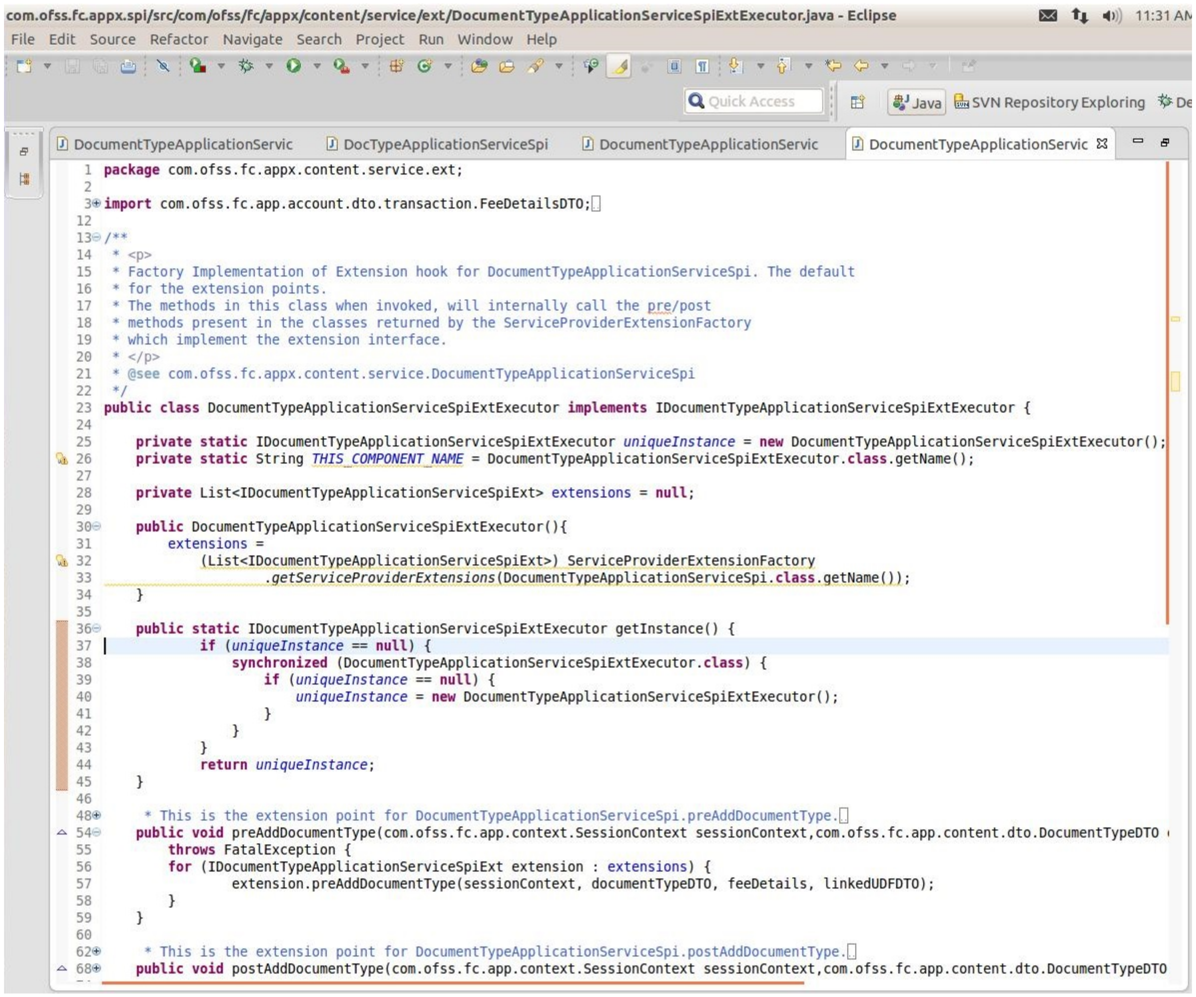

: ApplicationServiceSpi
: ApplicationServiceSpiExtExecutor
: ServiceProviderExtensionFactory
: IApplicationServiceSpiExt
: DomainService
1 : void := preServiceMethod()
2 : List<ApplicationServiceSpiExt> := getServiceProviderExtensions()
3
4 : void := preServiceMethod()
5
6
7 : void := businessLogic()
8
9 : void := postServiceMethod()
10 : void := postServiceMethod()
11
12

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


com.ofss.fc.appx.spi/src/com/ofss/fc/appx/content/service/ext/IDocumentTypeApplicationServiceSpiExtExecutor.java - Eclipse
11:32 AM
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
DocumentTypeApplicati
DocTypeApplicationSe
DocumentTypeApplicati
DocumentTypeApplicati
IDocumentTypeApplicat

```
package com.ofss.fc.appx.content.service.ext;

import com.ofss.fc.app.account.dto.transaction.FeeDetailsDTO;

/**
 * <p>
 * ExtensionFactory hook for DocumentTypeApplicationServiceSpi. Extension Factories should implement
 * the IDocumentTypeApplicationServiceSpiExtExecutor
 * </p>
 */
public interface IDocumentTypeApplicationServiceSpiExtExecutor {

    /**
     * This is the extension point for DocumentTypeApplicationServiceSpi.preAddDocumentType.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#preAddDocumentType
     */
    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeDTO
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationServiceSpi.postAddDocumentType.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#postAddDocumentType
     */
    public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeDTO
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationServiceSpi.preUpdateDocumentType.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#preUpdateDocumentType
     */
    public void preUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeD
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationServiceSpi.postUpdateDocumentType.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#postUpdateDocumentType
     */
    public void postUpdateDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentType
        throws FatalException;

    /**
     * This is the extension point for DocumentTypeApplicationServiceSpi.preFetchDocumentType.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi#preFetchDocumentType
```

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


com.ofss.fc.appx.spi/src/com/ofss/fc/appx/content/service/ext/DocumentTypeApplicationServiceSpiExtExecutor.java - Eclipse
11:31 AM
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
DocumentTypeApplicationServic
DocTypeApplicationServiceSpi
DocumentTypeApplicationServic
DocumentTypeApplicationServic

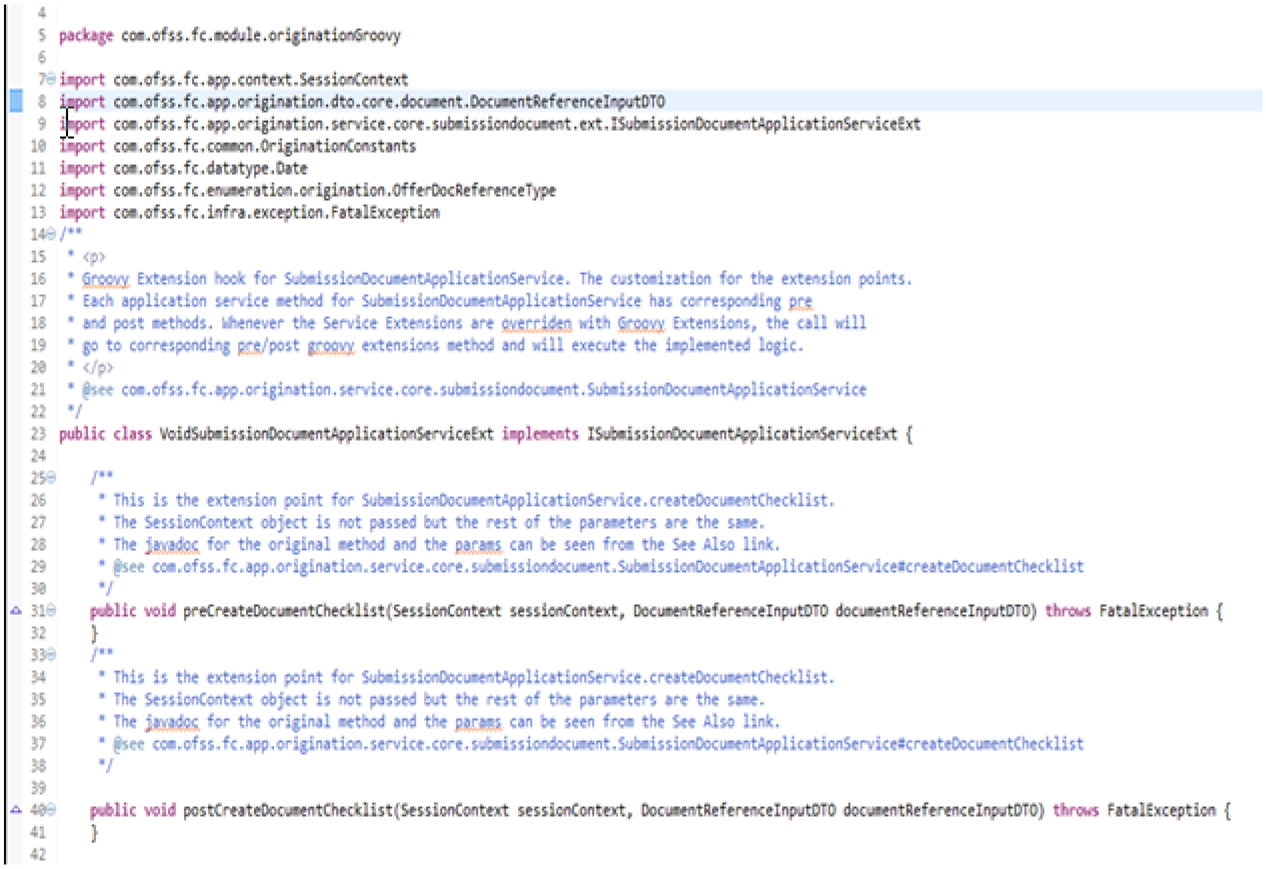
```
package com.ofss.fc.appx.content.service.ext;

import com.ofss.fc.app.account.dto.transaction.FeeDetailsDTO;

/**
 * <p>
 * Factory Implementation of Extension hook for DocumentTypeApplicationServiceSpi. The default
 * for the extension points.
 * The methods in this class when invoked, will internally call the pre/post
 * methods present in the classes returned by the ServiceProviderExtensionFactory
 * which implement the extension interface.
 * </p>
 * @see com.ofss.fc.appx.content.service.DocumentTypeApplicationServiceSpi
 */
public class DocumentTypeApplicationServiceSpiExtExecutor implements IDocumentTypeApplicationServiceSpiExtExecutor {

    private static IDocumentTypeApplicationServiceSpiExtExecutor uniqueInstance = new DocumentTypeApplicationServiceSpiExtExecutor();
    private static String THIS_COMPONENT_NAME = DocumentTypeApplicationServiceSpiExtExecutor.class.getName();

    private List<IDocumentTypeApplicationServiceSpiExt> extensions = null;

    public DocumentTypeApplicationServiceSpiExtExecutor(){
        extensions =
            (List<IDocumentTypeApplicationServiceSpiExt>) ServiceProviderExtensionFactory
                .getServiceProviderExtensions(DocumentTypeApplicationServiceSpi.class.getName());
    }

    public static IDocumentTypeApplicationServiceSpiExtExecutor getInstance() {
        if (uniqueInstance == null) {
            synchronized (DocumentTypeApplicationServiceSpiExtExecutor.class) {
                if (uniqueInstance == null) {
                    uniqueInstance = new DocumentTypeApplicationServiceSpiExtExecutor();
                }
            }
        }
        return uniqueInstance;
    }

     * This is the extension point for DocumentTypeApplicationServiceSpi.preAddDocumentType.
    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeDTO
        throws FatalException {
        for (IDocumentTypeApplicationServiceSpiExt extension : extensions) {
                extension.preAddDocumentType(sessionContext, documentTypeDTO, feeDetails, linkedUDFDTO);
        }
    }

     * This is the extension point for DocumentTypeApplicationServiceSpi.postAddDocumentType.
    public void postAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,com.ofss.fc.app.content.dto.DocumentTypeDTO
```

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.

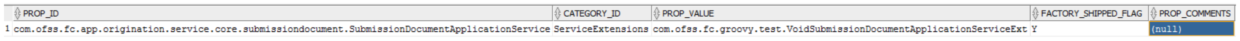
Figure 4–18 Application Service Appx Extension using Groovy



```
package com.ofss.fc.module.originationGroovy

import com.ofss.fc.app.context.SessionContext
import com.ofss.fc.app.origination.dto.core.document.DocumentReferenceInputDTO
import com.ofss.fc.app.origination.service.core.submissiondocument.ext.ISubmissionDocumentApplicationServiceExt
import com.ofss.fc.common.OriginationConstants
import com.ofss.fc.datatype.Date
import com.ofss.fc.enumeration.origination.OfferDocReferenceType
import com.ofss.fc.infra.exception.FatalException
/**
 * <p>
 * Groovy Extension hook for SubmissionDocumentApplicationService. The customization for the extension points.
 * Each application service method for SubmissionDocumentApplicationService has corresponding pre
 * and post methods. Whenever the Service Extensions are overriden with Groovy Extensions, the call will
 * go to corresponding pre/post groovy extensions method and will execute the implemented logic.
 * </p>
 * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService
 */
public class VoidSubmissionDocumentApplicationServiceExt implements ISubmissionDocumentApplicationServiceExt {

    /**
     * This is the extension point for SubmissionDocumentApplicationService.createDocumentChecklist.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService#createDocumentChecklist
     */
    public void preCreateDocumentChecklist(SessionContext sessionContext, DocumentReferenceInputDTO documentReferenceInputDTO) throws FatalException {
    }
    /**
     * This is the extension point for SubmissionDocumentApplicationService.createDocumentChecklist.
     * The SessionContext object is not passed but the rest of the parameters are the same.
     * The javadoc for the original method and the params can be seen from the See Also link.
     * @see com.ofss.fc.app.origination.service.core.submissiondocument.SubmissionDocumentApplicationService#createDocumentChecklist
     */

    public void postCreateDocumentChecklist(SessionContext sessionContext, DocumentReferenceInputDTO documentReferenceInputDTO) throws FatalException {
    }

```

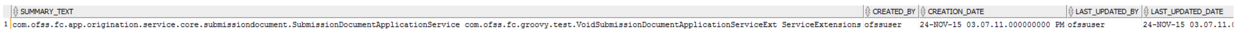
Provide the fully qualified name of the above groovy implementation in flx_fw_config_all_b against the corresponding service extension prop_id and category_id.

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)

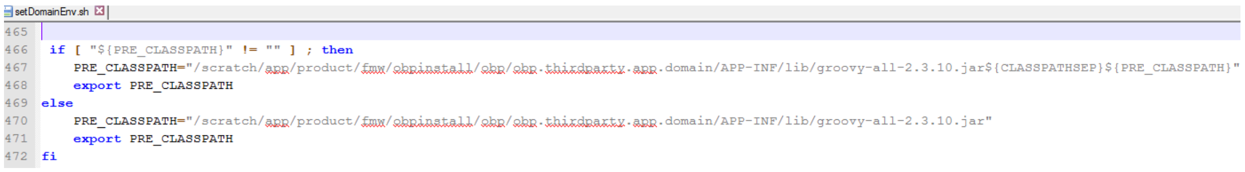
Figure 4–20 SUMMARY_TEXT



	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

```

 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
```

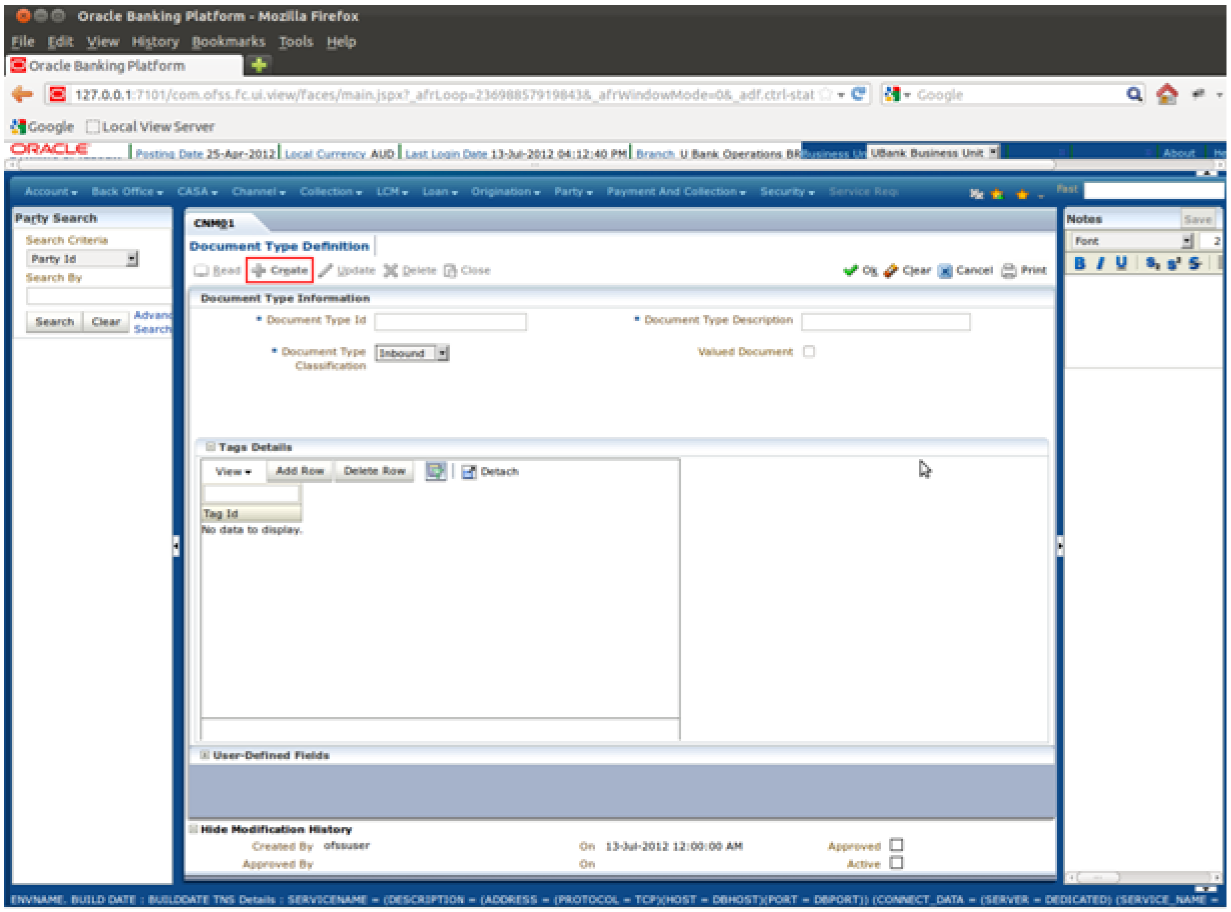
4.3 End-to-End Example of an Extension

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


Oracle Banking Platform - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Account Back Office CASA Channel Collection LCM Loan Origination Party Payment And Collection Security
Party Search
Search Criteria
Party Id
Search By
Search Clear
CNM01
Document Type Definition
Read Create Update Delete Close
Ok Clear Cancel Print
Document Type Information
* Document Type Id
* Document Type Description
* Document Type Classification
Valued Document
Tags Details
View Add Row Delete Row Detach
Tag Id
No data to display.
User-Defined Fields
Hide Modification History
Notes
Save

Posting Date 25-Apr-2012 | Local Currency AUD | Last Login Date 13-Jul-2012 04:12:40 PM | Branch U Bank Operations BR | Business Unit UBank Business Unit

Document Type Classification: Inbound

Created By ofssuser On 13-Jul-2012 12:00:00 AM Approved

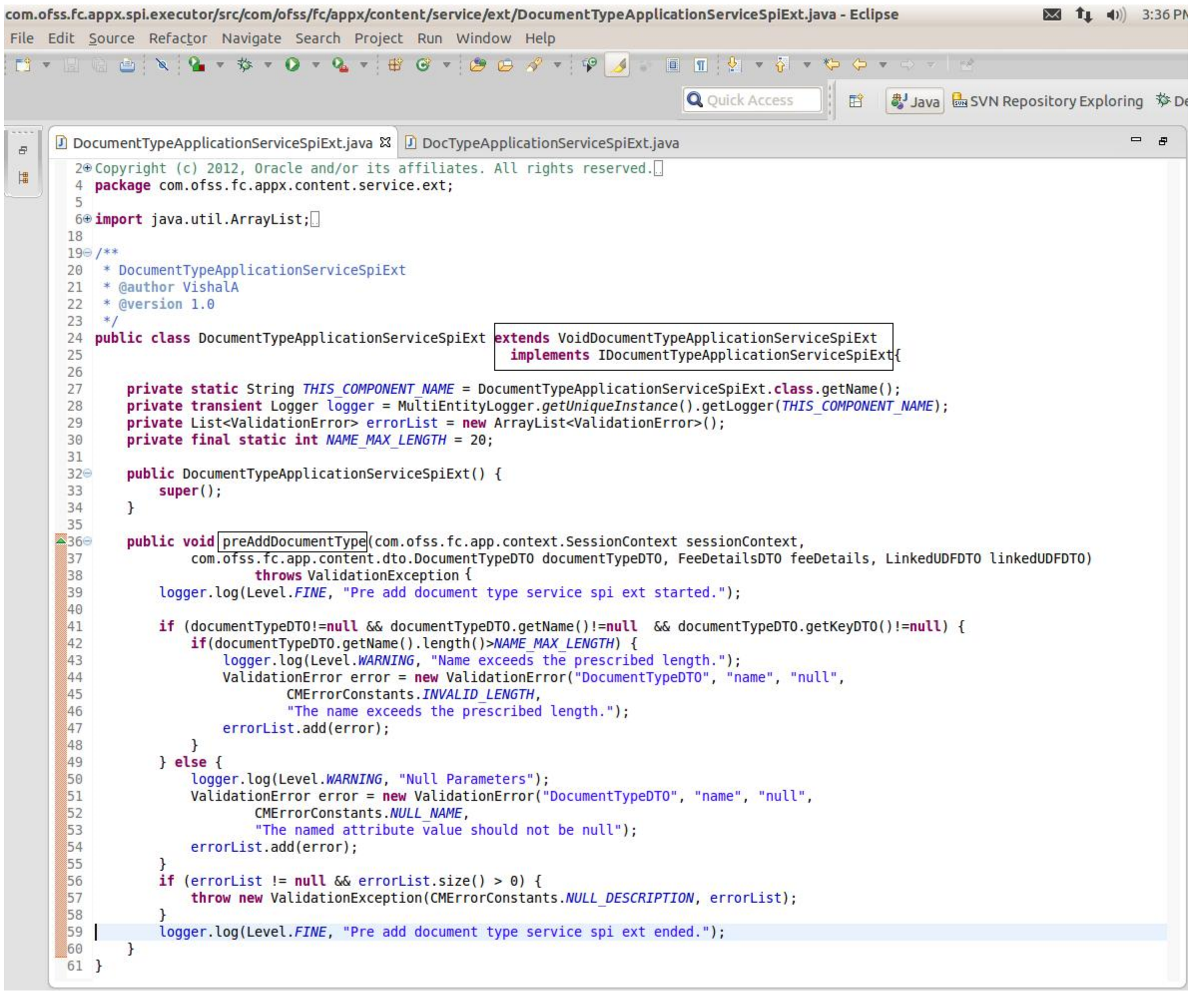
Approved By On Active

No data to display.

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.

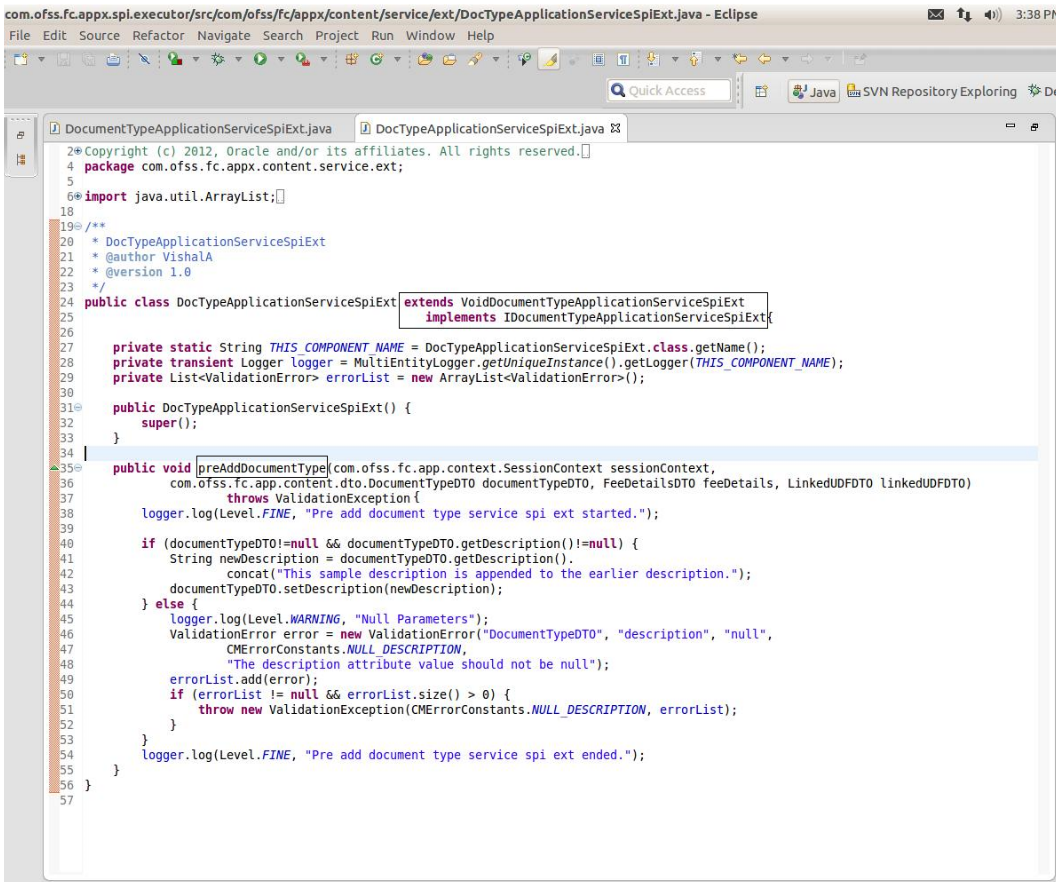
In this example, we have added multiple extensions to this service of the appx layer through the extension executor, where the update of the description is done in one of the extension and check the length of name in another in the pre extension method.

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


com.ofss.fc.appx.spi.executor/src/com/ofss/fc/appx/content/service/ext/DocumentTypeApplicationServiceSpiExt.java - Eclipse
3:36 PM
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
DocumentTypeApplicationServiceSpiExt.java
DocTypeApplicationServiceSpiExt.java

```
Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.appx.content.service.ext;

import java.util.ArrayList;

/**
 * DocumentTypeApplicationServiceSpiExt
 * @author VishalA
 * @version 1.0
 */
public class DocumentTypeApplicationServiceSpiExt extends VoidDocumentTypeApplicationServiceSpiExt
    implements IDocumentTypeApplicationServiceSpiExt{

    private static String THIS_COMPONENT_NAME = DocumentTypeApplicationServiceSpiExt.class.getName();
    private transient Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(THIS_COMPONENT_NAME);
    private List<ValidationError> errorList = new ArrayList<ValidationError>();
    private final static int NAME_MAX_LENGTH = 20;

    public DocumentTypeApplicationServiceSpiExt() {
        super();
    }

    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,
            com.ofss.fc.app.content.dto.DocumentTypeDTO documentTypeDTO, FeeDetailsDTO feeDetails, LinkedUDFDTO linkedUDFDTO)
                    throws ValidationException {
        logger.log(Level.FINE, "Pre add document type service spi ext started.");

        if (documentTypeDTO!=null && documentTypeDTO.getName()!=null  && documentTypeDTO.getKeyDTO()!=null) {
            if(documentTypeDTO.getName().length()>NAME_MAX_LENGTH) {
                logger.log(Level.WARNING, "Name exceeds the prescribed length.");
                ValidationError error = new ValidationError("DocumentTypeDTO", "name", "null",
                        CMErrorConstants.INVALID_LENGTH,
                        "The name exceeds the prescribed length.");
                errorList.add(error);
            }
        } else {
            logger.log(Level.WARNING, "Null Parameters");
            ValidationError error = new ValidationError("DocumentTypeDTO", "name", "null",
                    CMErrorConstants.NULL_NAME,
                    "The named attribute value should not be null");
            errorList.add(error);
        }
        if (errorList != null && errorList.size() > 0) {
            throw new ValidationException(CMErrorConstants.NULL_DESCRIPTION, errorList);
        }
        logger.log(Level.FINE, "Pre add document type service spi ext ended.");
    }
}
```

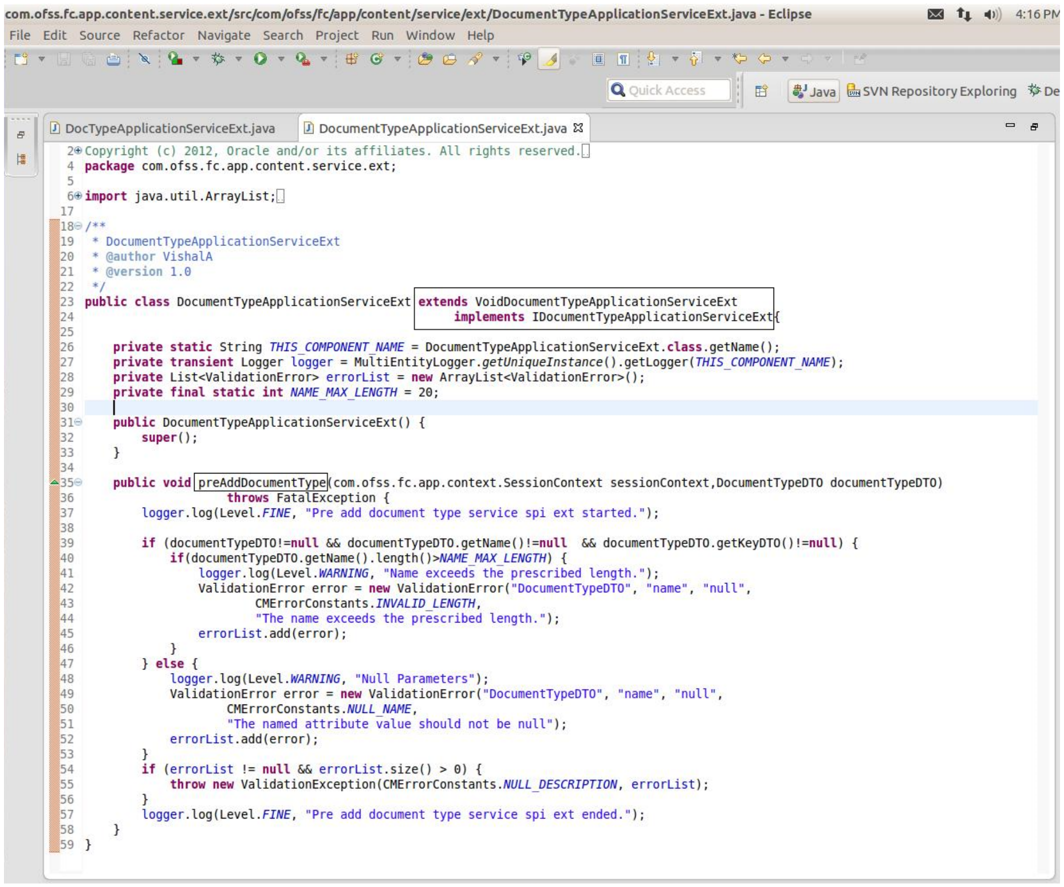
Figure 4–24 Doc Type Application Service Spi Ext - Appx Layer


com.ofss.fc.appx.spi.executor/src/com/ofss/fc/appx/content/service/ext/DocTypeApplicationServiceSpiExt.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
DocumentTypeApplicationServiceSpiExt.java
DocTypeApplicationServiceSpiExt.java

```
Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.appx.content.service.ext;

import java.util.ArrayList;

/**
 * DocTypeApplicationServiceSpiExt
 * @author VishalA
 * @version 1.0
 */
public class DocTypeApplicationServiceSpiExt extends VoidDocumentTypeApplicationServiceSpiExt
        implements IDocumentTypeApplicationServiceSpiExt{

    private static String THIS_COMPONENT_NAME = DocTypeApplicationServiceSpiExt.class.getName();
    private transient Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(THIS_COMPONENT_NAME);
    private List<ValidationError> errorList = new ArrayList<ValidationError>();

    public DocTypeApplicationServiceSpiExt() {
        super();
    }

    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,
            com.ofss.fc.app.content.dto.DocumentTypeDTO documentTypeDTO, FeeDetailsDTO feeDetails, LinkedUDFDTO linkedUDFDTO)
                    throws ValidationException{
        logger.log(Level.FINE, "Pre add document type service spi ext started.");

        if (documentTypeDTO!=null && documentTypeDTO.getDescription()!=null) {
            String newDescription = documentTypeDTO.getDescription().
                    concat("This sample description is appended to the earlier description.");
            documentTypeDTO.setDescription(newDescription);
        } else {
            logger.log(Level.WARNING, "Null Parameters");
            ValidationError error = new ValidationError("DocumentTypeDTO", "description", "null",
                    CMErrorConstants.NULL_DESCRIPTION,
                    "The description attribute value should not be null");
            errorList.add(error);
            if (errorList != null && errorList.size() > 0) {
                throw new ValidationException(CMErrorConstants.NULL_DESCRIPTION, errorList);
            }
        }
        logger.log(Level.FINE, "Pre add document type service spi ext ended.");
    }
}
```

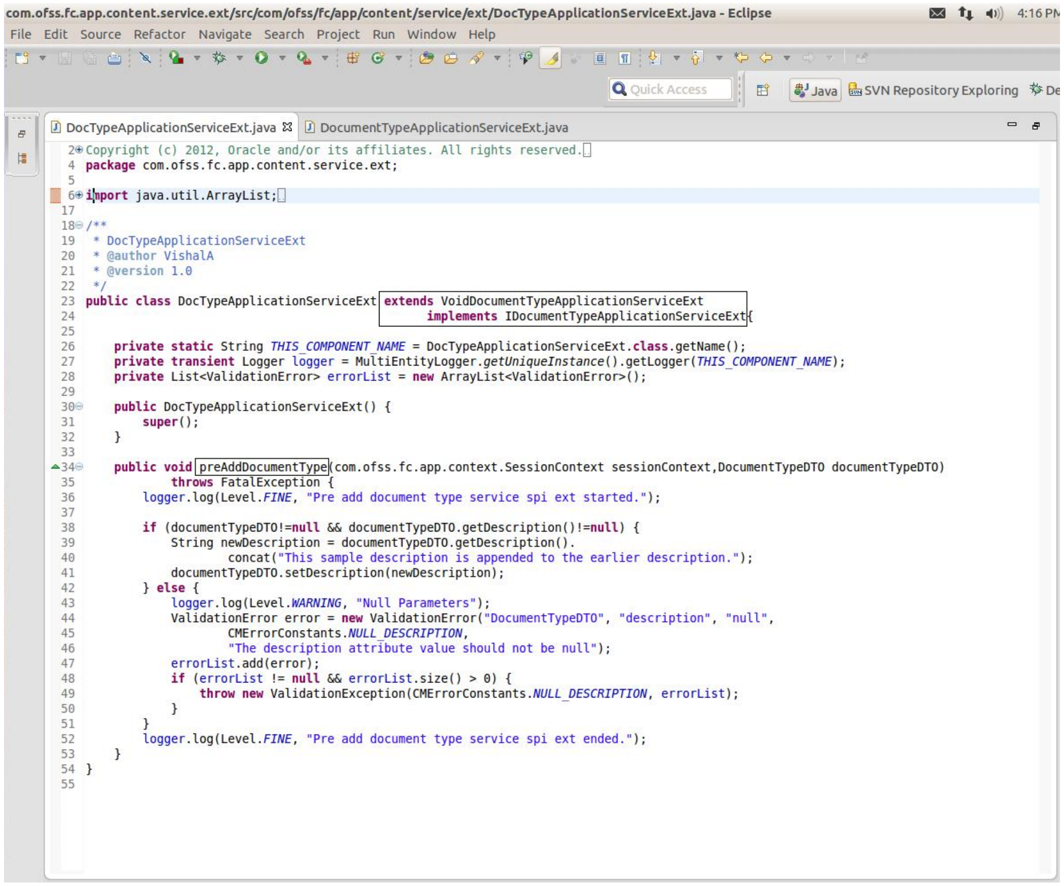
In this example, we have added multiple extensions to the service of the app layer through the extension executor. We have implemented a not null and size check on the document tags in pre hook of the app layer to validate that document tags are sent as input in the application service.

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


com.ofss.fc.app.content.service.ext/src/com/ofss/fc/app/content/service/ext/DocumentTypeApplicationServiceExt.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
DocTypeApplicationServiceExt.java
DocumentTypeApplicationServiceExt.java

```java
Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.app.content.service.ext;

import java.util.ArrayList;

/**
 * DocumentTypeApplicationServiceExt
 * @author VishalA
 * @version 1.0
 */
public class DocumentTypeApplicationServiceExt extends VoidDocumentTypeApplicationServiceExt
            implements IDocumentTypeApplicationServiceExt{

    private static String THIS_COMPONENT_NAME = DocumentTypeApplicationServiceExt.class.getName();
    private transient Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(THIS_COMPONENT_NAME);
    private List<ValidationError> errorList = new ArrayList<ValidationError>();
    private final static int NAME_MAX_LENGTH = 20;

    public DocumentTypeApplicationServiceExt() {
        super();
    }

    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
                    throws FatalException {
        logger.log(Level.FINE, "Pre add document type service spi ext started.");

        if (documentTypeDTO!=null && documentTypeDTO.getName()!=null  && documentTypeDTO.getKeyDTO()!=null) {
            if(documentTypeDTO.getName().length()>NAME_MAX_LENGTH) {
                logger.log(Level.WARNING, "Name exceeds the prescribed length.");
                ValidationError error = new ValidationError("DocumentTypeDTO", "name", "null",
                        CMErrorConstants.INVALID_LENGTH,
                        "The name exceeds the prescribed length.");
                errorList.add(error);
            }
        } else {
            logger.log(Level.WARNING, "Null Parameters");
            ValidationError error = new ValidationError("DocumentTypeDTO", "name", "null",
                    CMErrorConstants.NULL_NAME,
                    "The named attribute value should not be null");
            errorList.add(error);
        }
        if (errorList != null && errorList.size() > 0) {
            throw new ValidationException(CMErrorConstants.NULL_DESCRIPTION, errorList);
        }
        logger.log(Level.FINE, "Pre add document type service spi ext ended.");
    }
}
```

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


com.ofss.fc.app.content.service.ext/src/com/ofss/fc/app/content/service/ext/DocTypeApplicationServiceExt.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
SVN Repository Exploring
DocTypeApplicationServiceExt.java
DocumentTypeApplicationServiceExt.java

```java
Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.app.content.service.ext;

import java.util.ArrayList;

/**
 * DocTypeApplicationServiceExt
 * @author VishalA
 * @version 1.0
 */
public class DocTypeApplicationServiceExt extends VoidDocumentTypeApplicationServiceExt
        implements IDocumentTypeApplicationServiceExt{

    private static String THIS_COMPONENT_NAME = DocTypeApplicationServiceExt.class.getName();
    private transient Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(THIS_COMPONENT_NAME);
    private List<ValidationError> errorList = new ArrayList<ValidationError>();

    public DocTypeApplicationServiceExt() {
        super();
    }

    public void preAddDocumentType(com.ofss.fc.app.context.SessionContext sessionContext,DocumentTypeDTO documentTypeDTO)
            throws FatalException {
        logger.log(Level.FINE, "Pre add document type service spi ext started.");

        if (documentTypeDTO!=null && documentTypeDTO.getDescription()!=null) {
            String newDescription = documentTypeDTO.getDescription().
                    concat("This sample description is appended to the earlier description.");
            documentTypeDTO.setDescription(newDescription);
        } else {
            logger.log(Level.WARNING, "Null Parameters");
            ValidationError error = new ValidationError("DocumentTypeDTO", "description", "null",
                    CMErrorConstants.NULL_DESCRIPTION,
                    "The description attribute value should not be null");
            errorList.add(error);
            if (errorList != null && errorList.size() > 0) {
                throw new ValidationException(CMErrorConstants.NULL_DESCRIPTION, errorList);
            }
        }
        logger.log(Level.FINE, "Pre add document type service spi ext ended.");
    }
}
```

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:

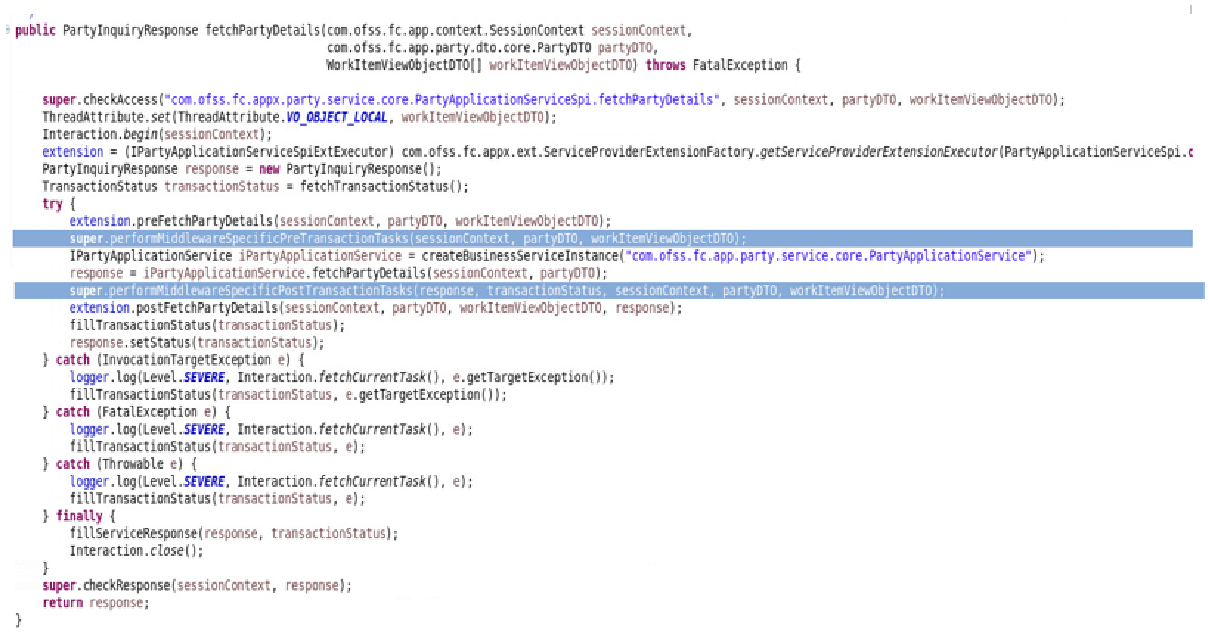
1. OBP signatures and Spi content will be same across all sites irrespective of OBP-middleware or Product processor implementation.
2. No appreciable change is required when the bank migrates from OBP Middleware to a full-fledged OBP product processor implementation.
3. OBP Middleware signatures are 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



```java
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;
}
```

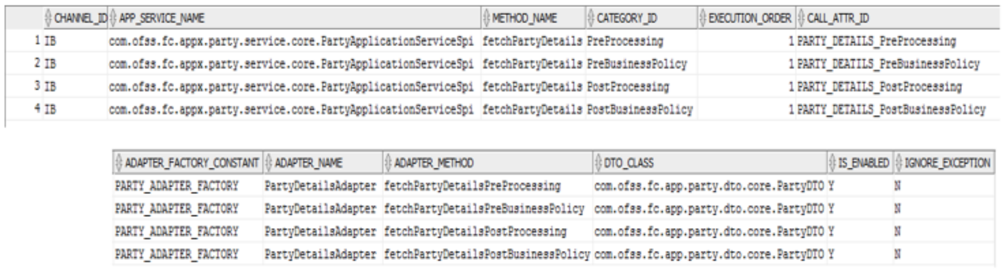
4.4.2 Sample Configuration

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



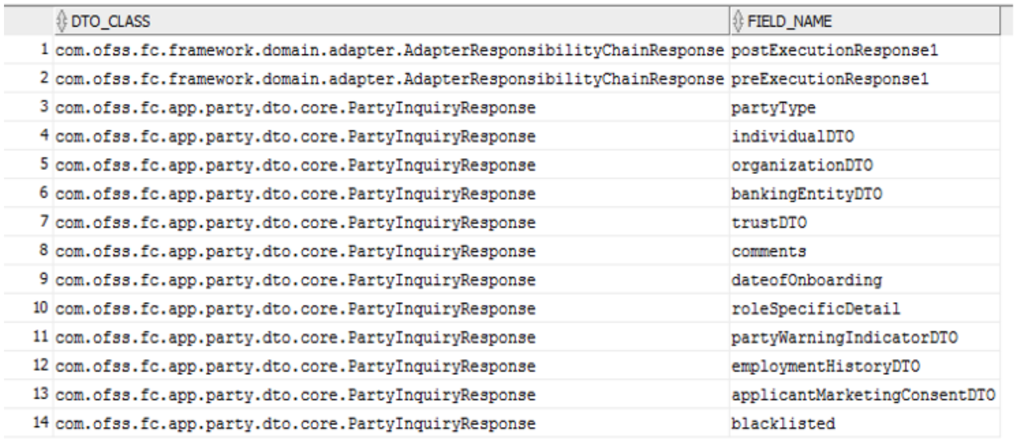
	CHANNEL_ID	APP_SERVICE_NAME	METHOD_NAME	CATEGORY_ID	EXECUTION_ORDER	CALL_ATTR_ID
1	IB	com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi	fetchPartyDetails	PreProcessing	1	PARTY_DETAILS_PreProcessing
2	IB	com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi	fetchPartyDetails	PreBusinessPolicy	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

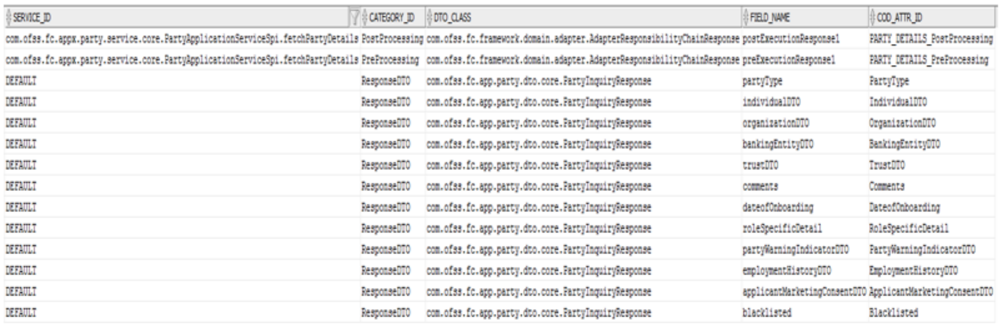


	DTO_CLASS	FIELD_NAME
1	com.ofss.fc.framework.domain.adapter.AdapterResponsibilityChainResponse	postExecutionResponse1
2	com.ofss.fc.framework.domain.adapter.AdapterResponsibilityChainResponse	preExecutionResponse1
3	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	partyType
4	com.ofss.fc.app.party.dto.core.PartyInquiryResponse	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

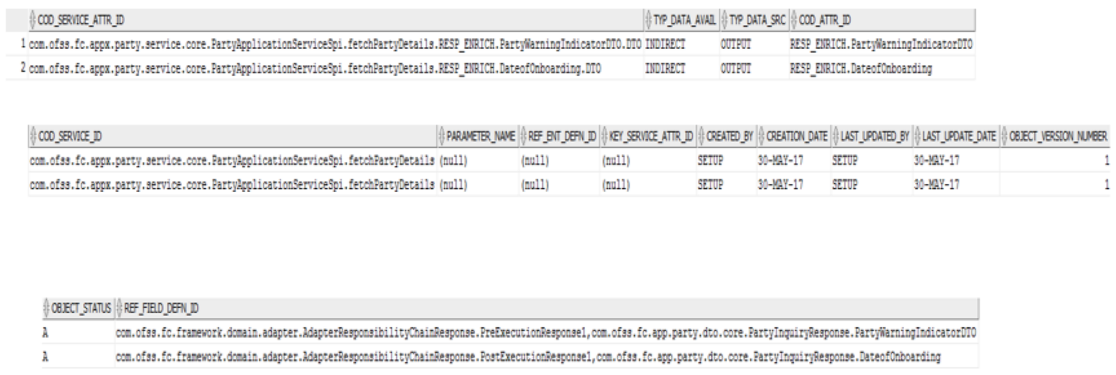


SERVICE_ID	CATEGORY_ID	DTO_CLASS	FIELD_NAME	COD_ATTR_ID
com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails	PostProcessing	com.ofss.fc.framework.domain.adapter.AdapterResponsibilityChainResponse	postExecutionResponse1	PARTY_DETAILS_PostProcessing
com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails	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	organizationDTO	OrganizationDTO
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

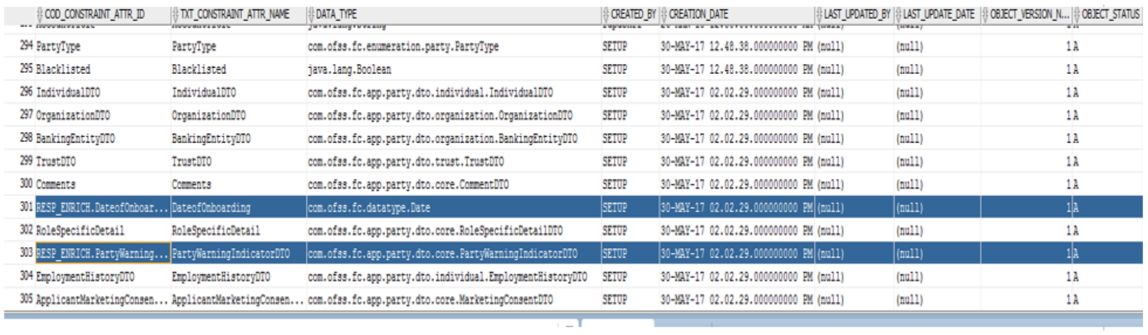
COD_SERVICE_ID	PARAMETER_NAME	REF_ENT_DEFN_ID	KEY_SERVICE_ATTR_ID	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE	OBJECT_VERSION_NUMBER
com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails	(null)	(null)	(null)	SETUP	30-MAY-17	SETUP	30-MAY-17	1
com.ofss.fc.appx.party.service.core.PartyApplicationServiceSpi.fetchPartyDetails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

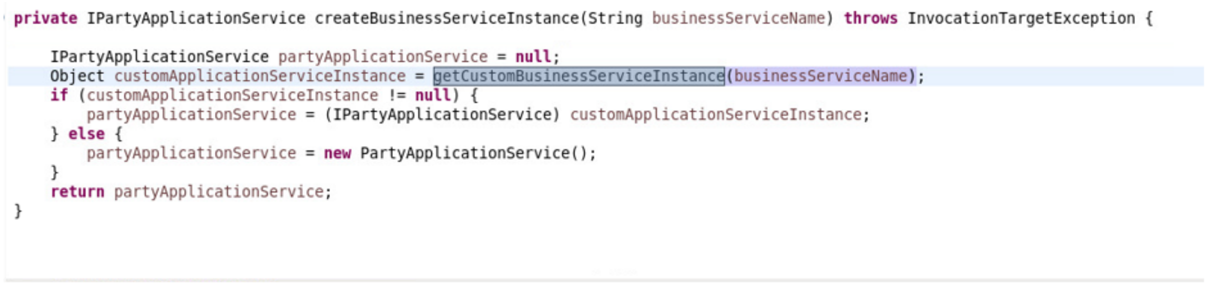


	COD_CONSTRAINT_ATTR_ID	TXT_CONSTRAINT_ATTR_NAME	DATA_TYPE	CREATED_BY	CREATION_DATE	LAST_UPDATED_BY	LAST_UPDATE_DATE	OBJECT_VERSION_N...	OBJECT_STATUS
294	PartyType	PartyType	com.ofss.fc.enumeration.party.PartyType	SETUP	30-MAY-17 12.48.38.000000000 PM	(null)	(null)	1	A
295	Blacklisted	Blacklisted	java.lang.Boolean	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.

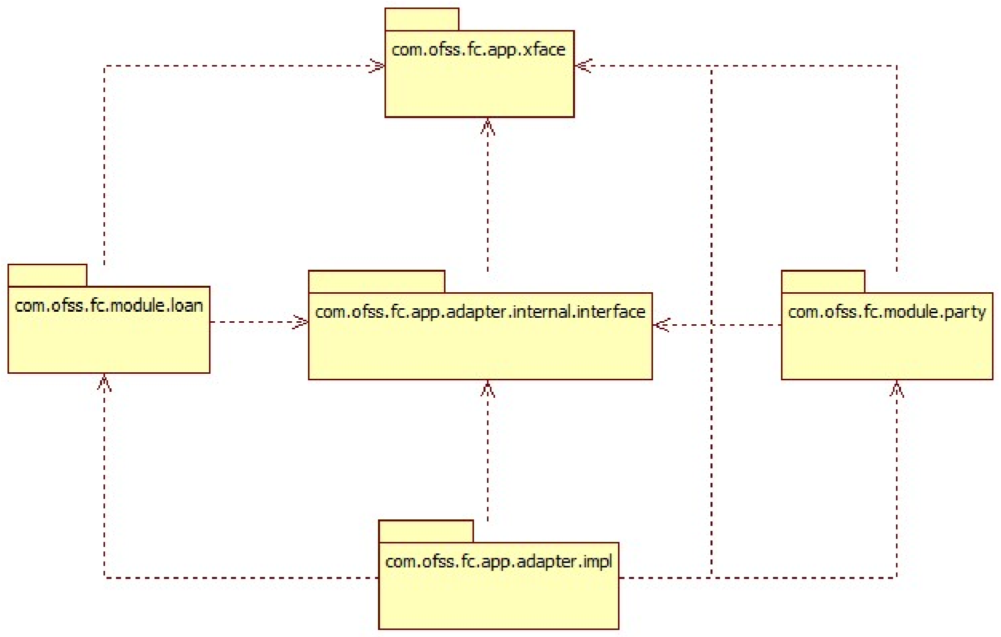
Table 6–1 Components of Adapter Implementation

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


com.ofss.fc.app.xface
com.ofss.fc.module.loan
com.ofss.fc.app.adapter.internal.interface
com.ofss.fc.module.party
com.ofss.fc.app.adapter.impl

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.

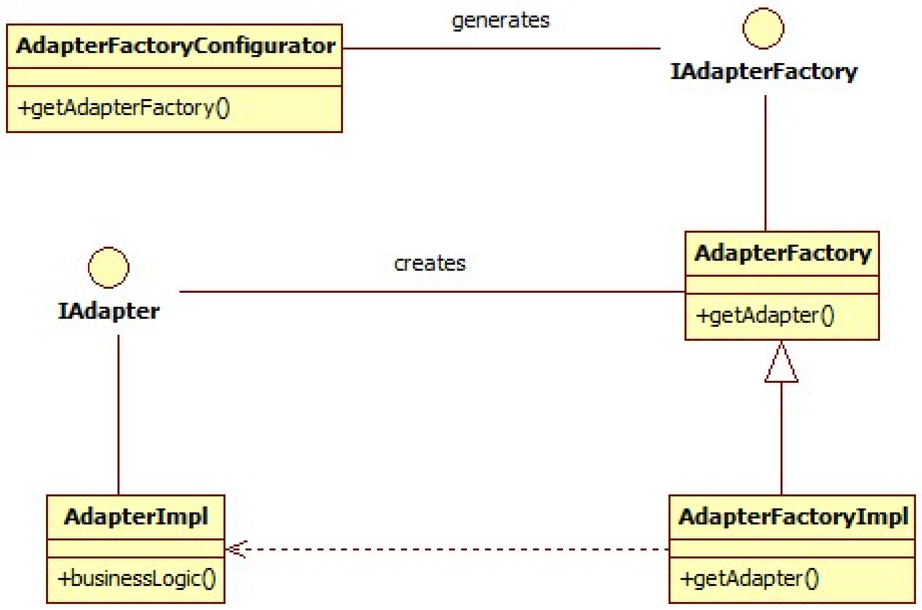
6.1.2 Adapter Mechanism Class Diagram

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.

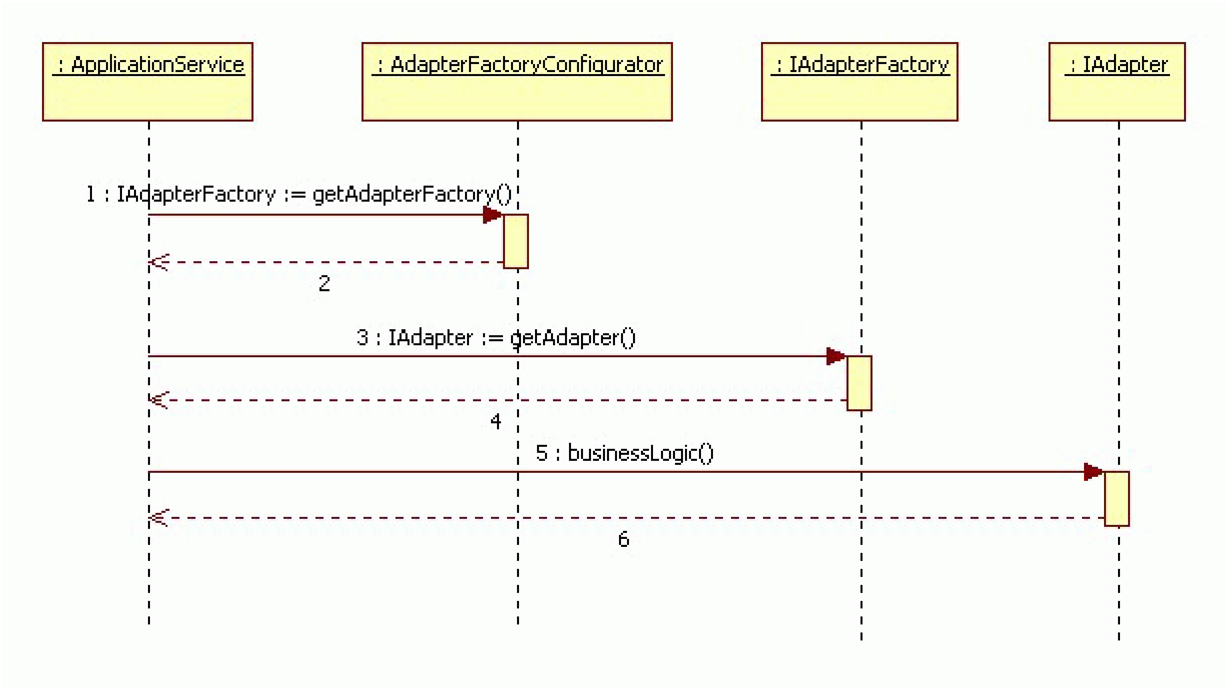
Figure 6–2 Adapter Mechanism Class Diagram



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


: ApplicationService
: AdapterFactoryConfigurator
: IAdapterFactory
: IAdapter
1 : IAdapterFactory := getAdapterFactory()
2
3 : IAdapter := getAdapter()
4
5 : businessLogic()
6

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.

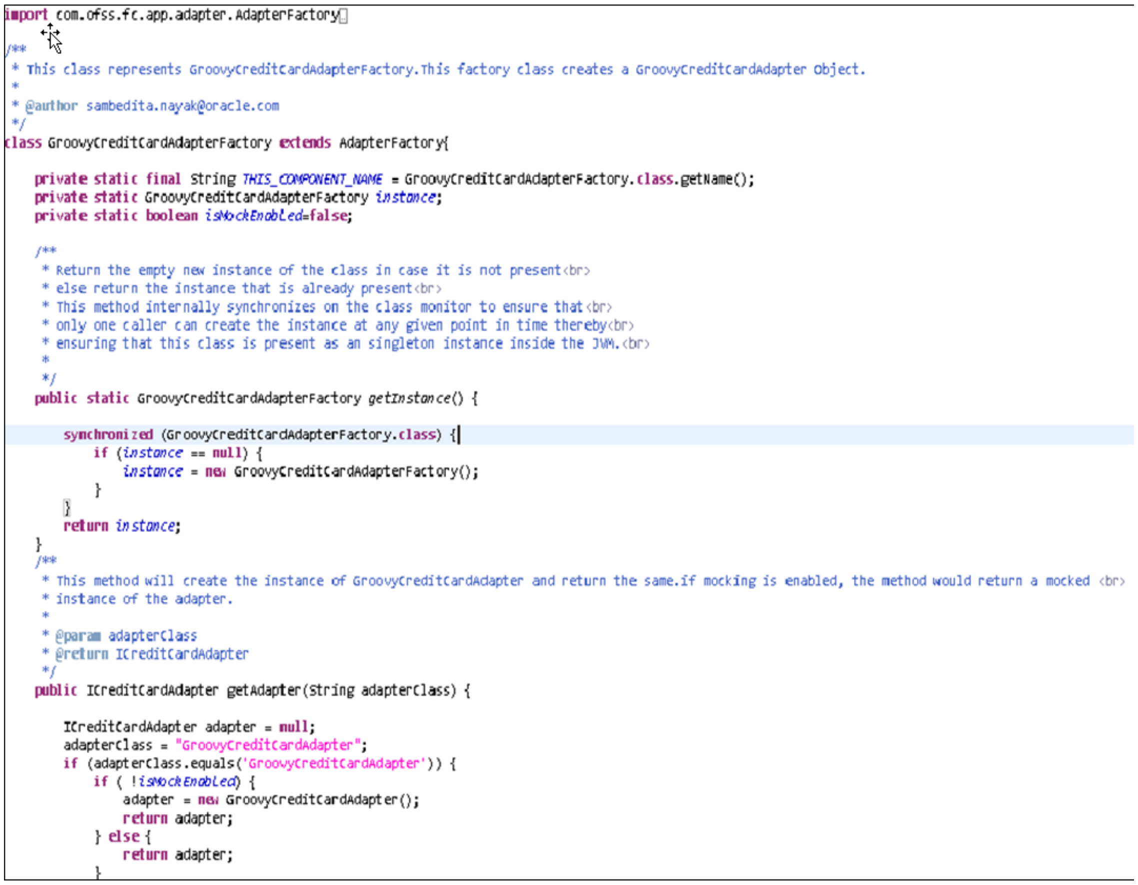
Existing product adapter will be overridden by the new custom made adapter for Groovy. This new Groovy adapter would contain groovy implementation methods which might call groovy files internally to perform desired functionality.

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.

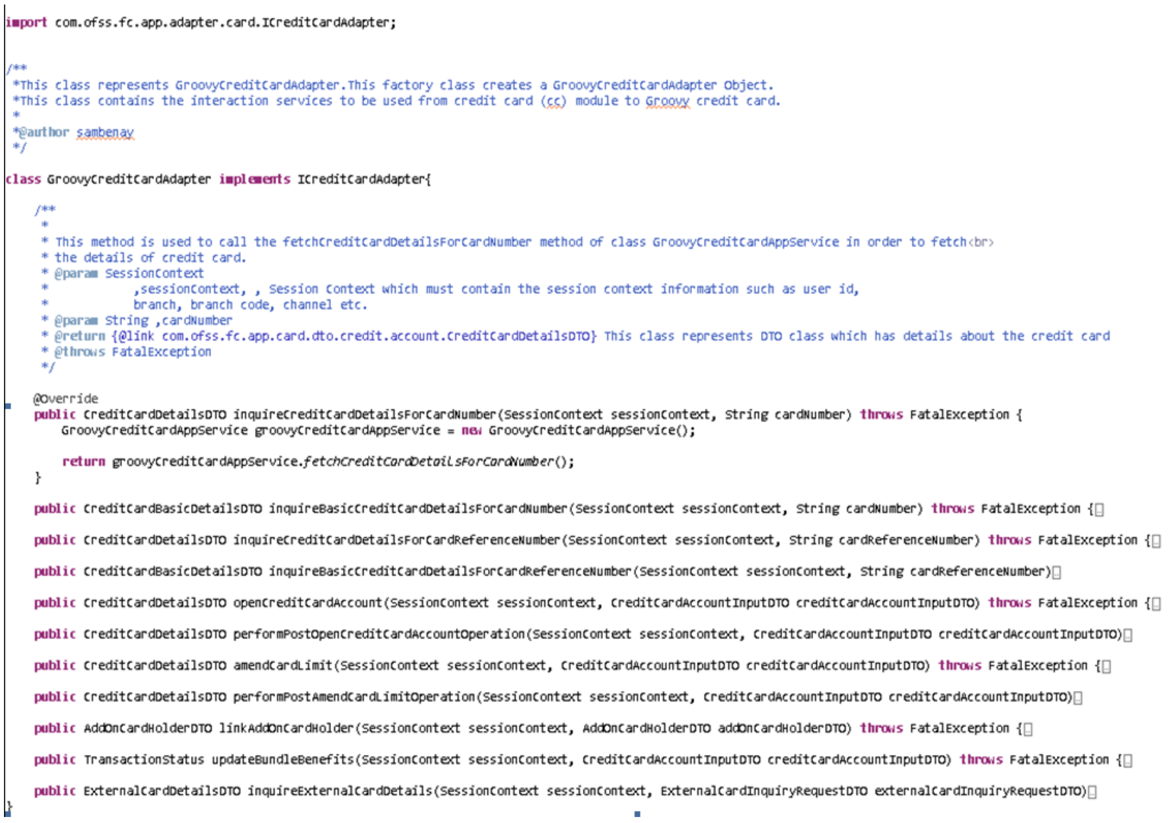
Figure 6–4 Adapter Implementation Using Groovy



```
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

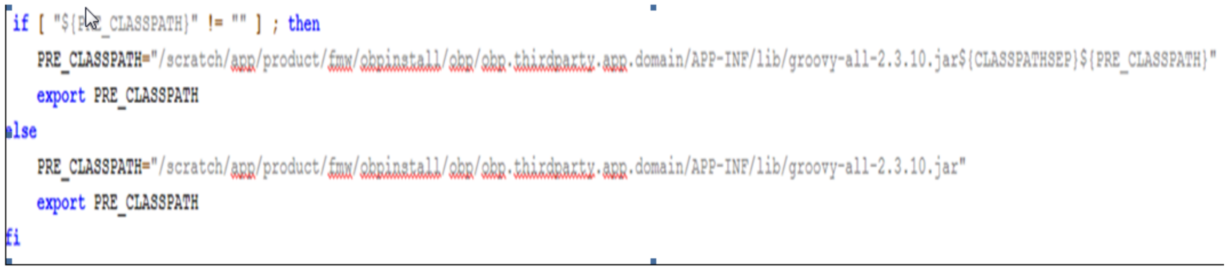
```
<Preference name="GroovyCreditCardAdapterFactory"
    PreferencesProvider="com.ofss.fc.infra.config.impl.DBBasedPropertyProvider"
    parent="jdbcpreference"
    propertyFileName="select prop_id, prop_value from flx_fw_config_all_b where category_id='GroovyCreditCardAdapterFactory'"
    syncTimeInterval="600000"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

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.

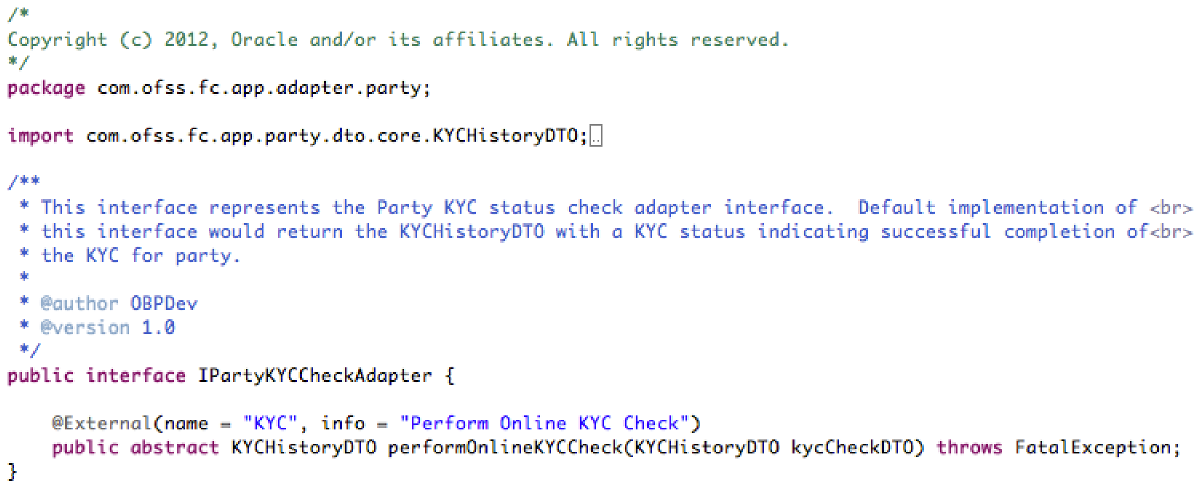
6.3.2 Custom Adapter Example 2 – PartyKYCCheckAdapter

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;
}
```

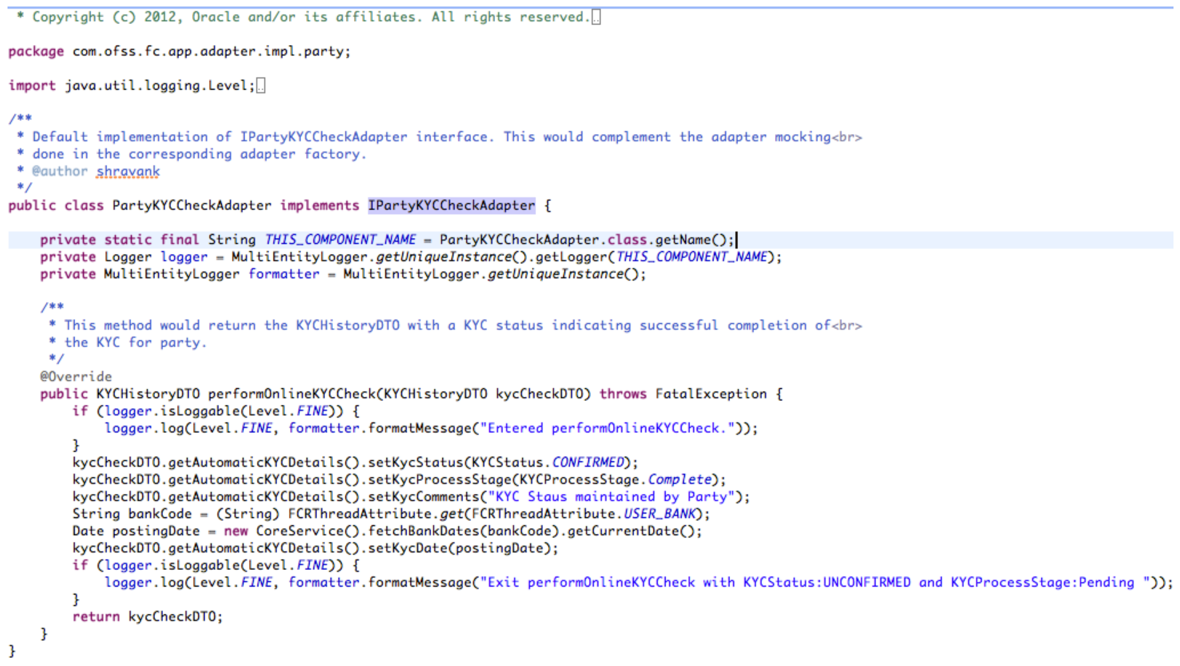
This adapter is integrated in product and the default implementation of the KYC check returns a successful KYC check as shown below. This is depicted in the code snippets below.

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;
    }
}
```

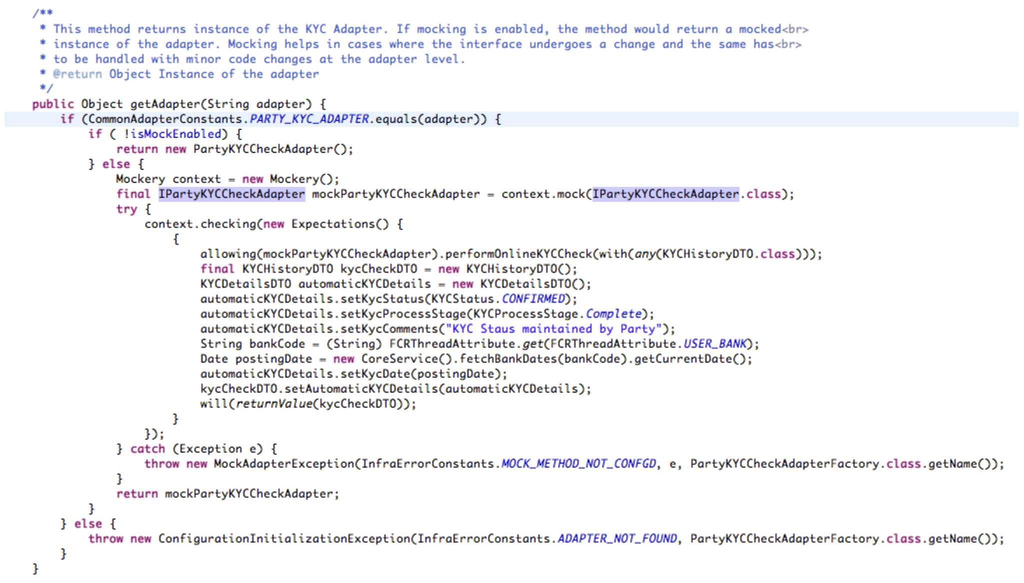
```
… PartyKYCCheckAdapter performOnlineKYCCheck Method …
public KYCHistoryDTO performOnlineKYCCheck(KYCHistoryDTO
kycCheckDTO) throws FatalException {
kycCheckDTO.getAutomaticKYCDetails().setKycStatus
(KYCStatus.CONFIRMED);
kycCheckDTO.getAutomaticKYCDetails().setKycProcessStage
(KYCProcessStage.Complete);
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.

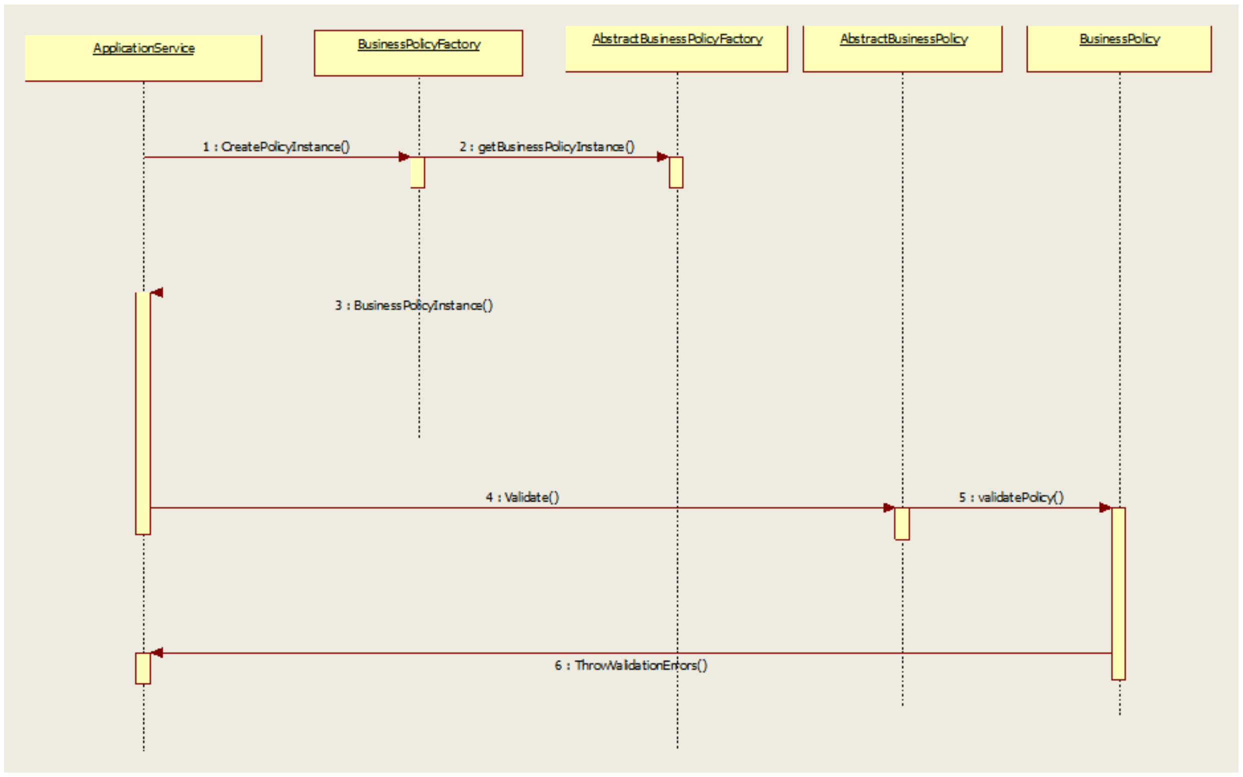
```
Original entry
PARTY_KYC_ADAPTER_
FACTORY=com.ofss.fc.app.adapter.impl.party.PartyKYCCheckAdapterFac
tory
Changed entry
PARTY_KYC_ADAPTER_FACTORY=
com.ofss.fc.app.adapter.impl.party.CustomPartyKYCCheckAdapterFacto
ry
```

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


ApplicationService
BusinessPolicyFactory
AbstractBusinessPolicyFactory
AbstractBusinessPolicy
BusinessPolicy
1 : CreatePolicyInstance()
2 : getBusinessPolicyInstance()
3 : BusinessPolicyInstance()
4 : Validate()
5 : validatePolicy()
6 : ThrowValidationErrors()

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.

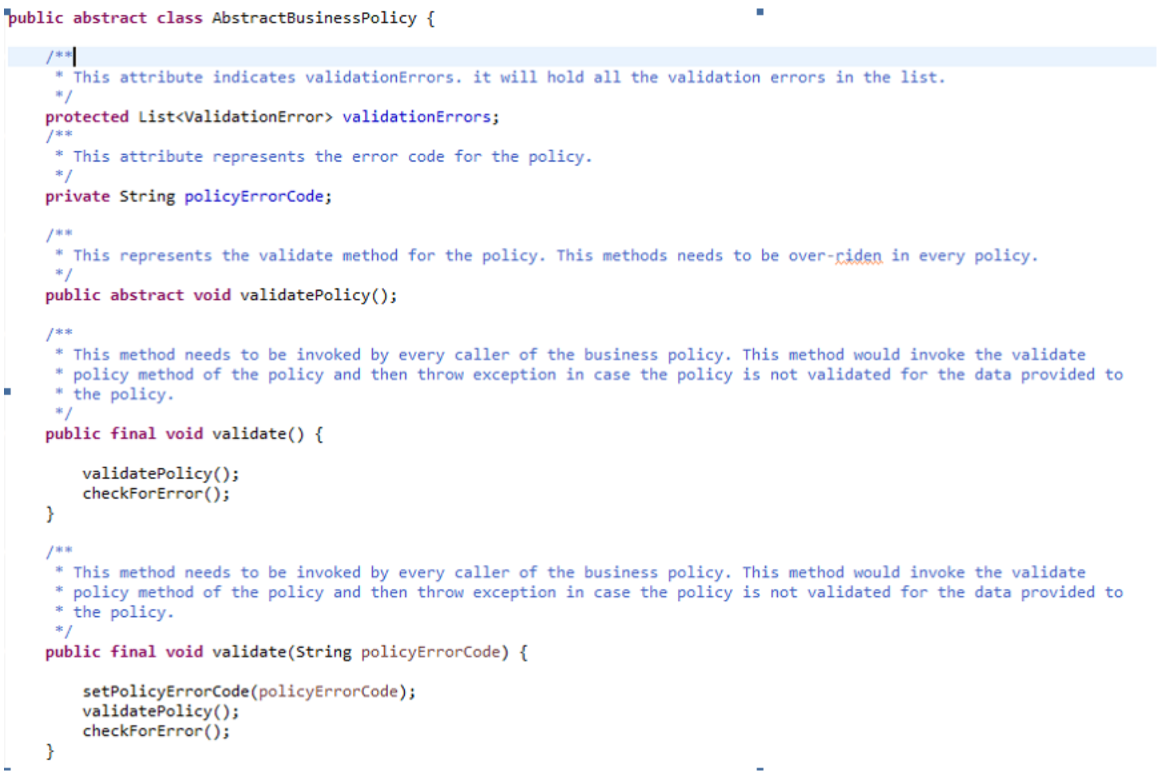
7.1 Base Implementation of Business Policy

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

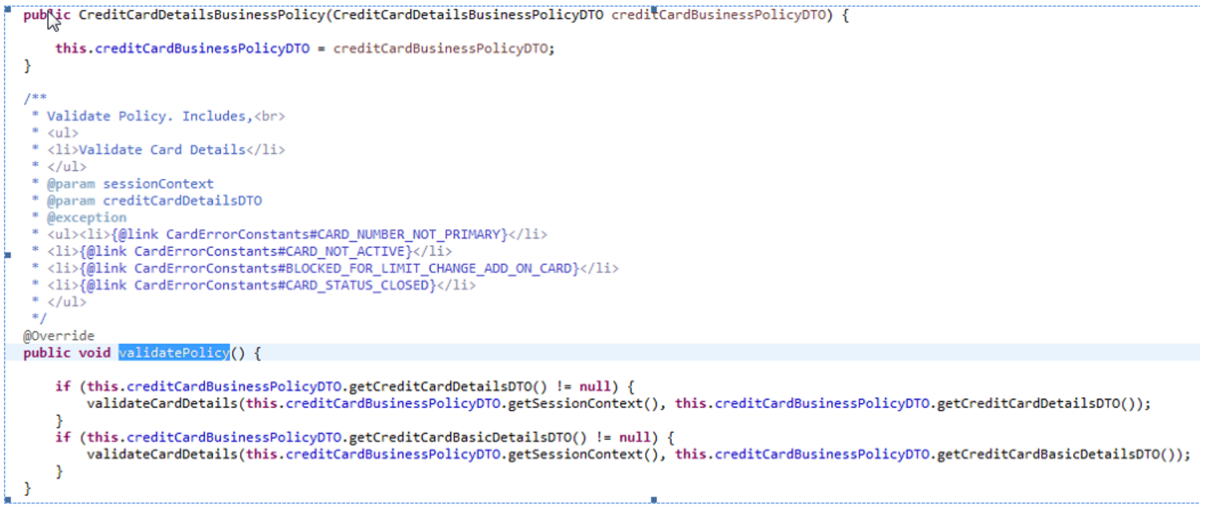
Figure 7–2 validate() method in AbstractBusinessPolicy.java



```
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

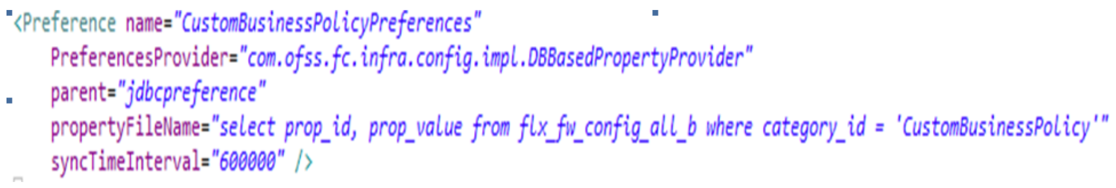
Custom implementation of business policy is achieved by defining a preference for customBusinessPolicy in preferences.xml which represents a query to the FLX_FW_CONFIG_ALL_B table in the database. To override a base business policy, class name of the custom business policy with the policy code is inserted 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);
```

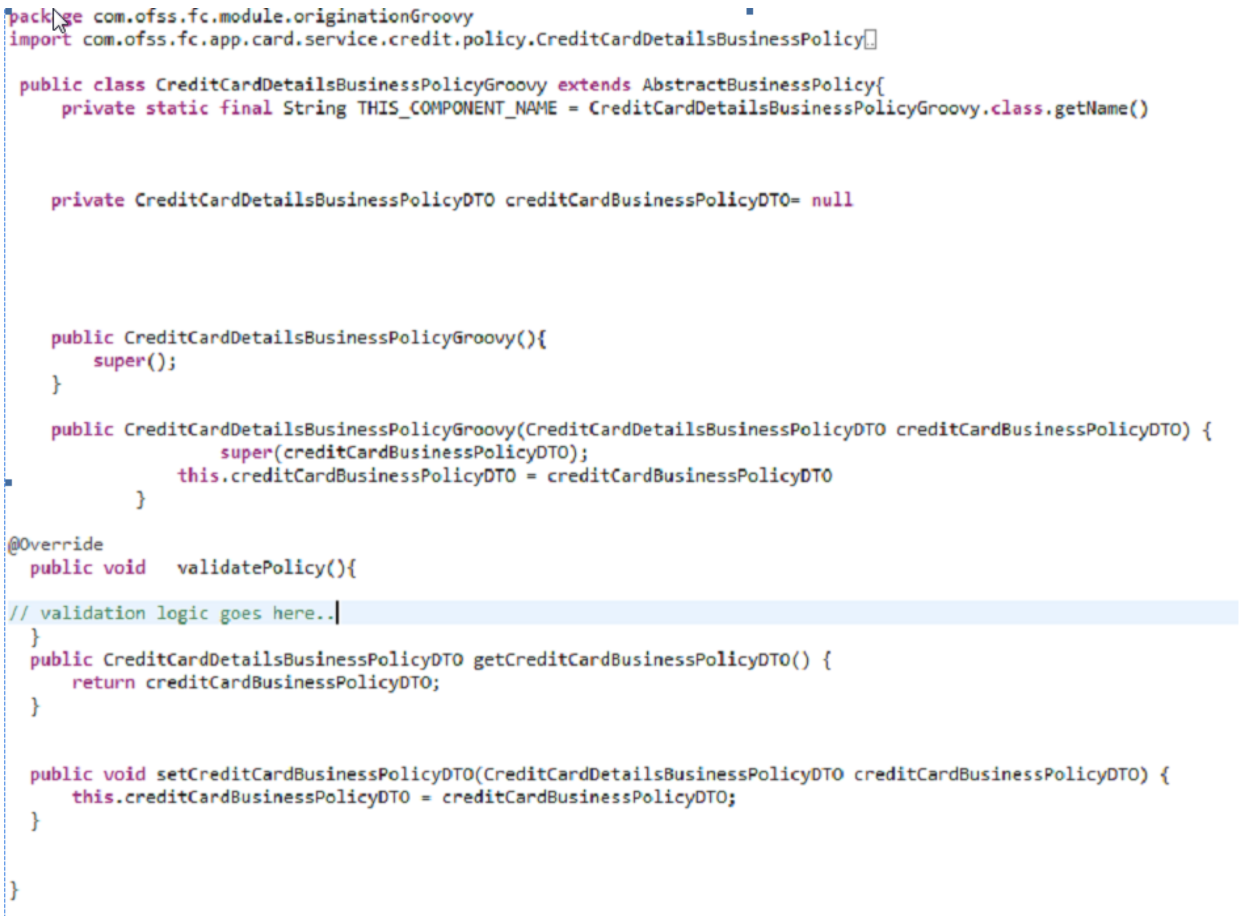
7.4 Extensions Using Groovy

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 OBP Extensibility Support Using Eclipse Plugin

OBP Eclipse Plugin has been updated to support OBP Extensibility features like run-time inclusion of Application Service SPI Extensions and Business Policy Extensions in the form of uploadable Groovy files.

8.1 Configure Eclipse Preferences for OBP Service Plugin

Following are the steps to configure eclipse preferences for OBP service plugin:

1. Click on Windows>Preferences.

Figure 8–1 Java Eclipse - Select Preferences

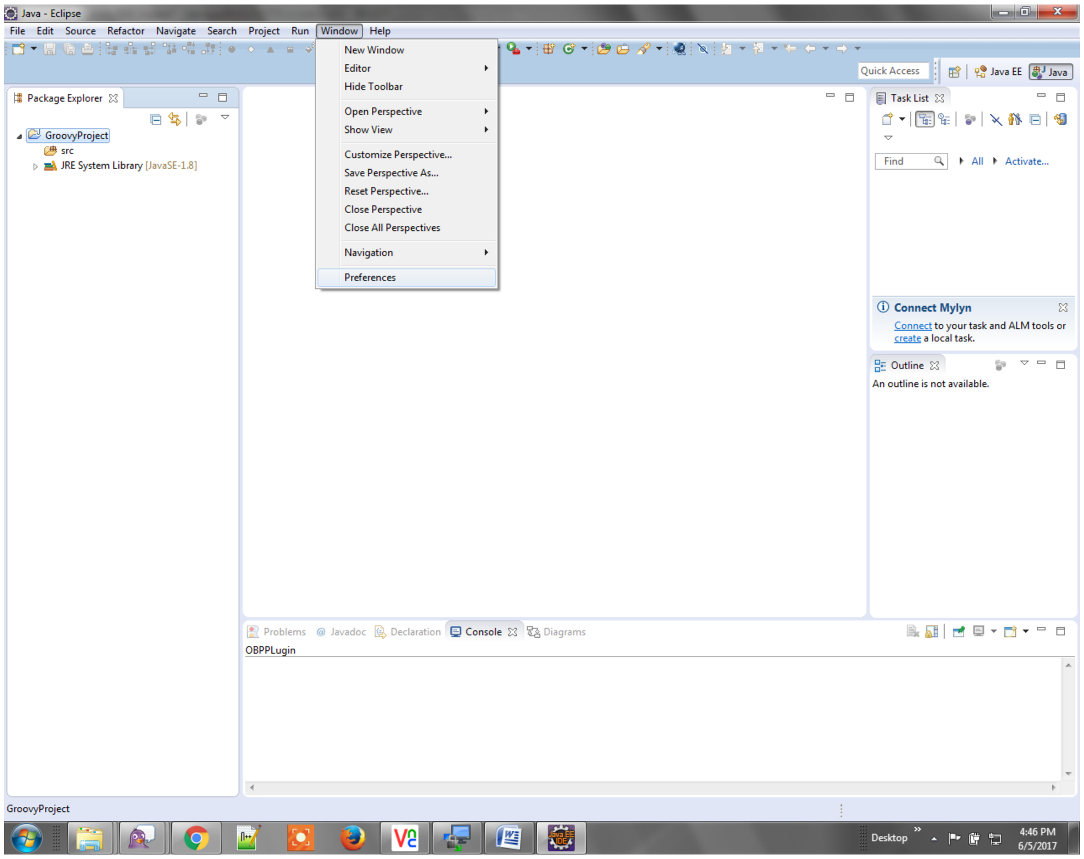
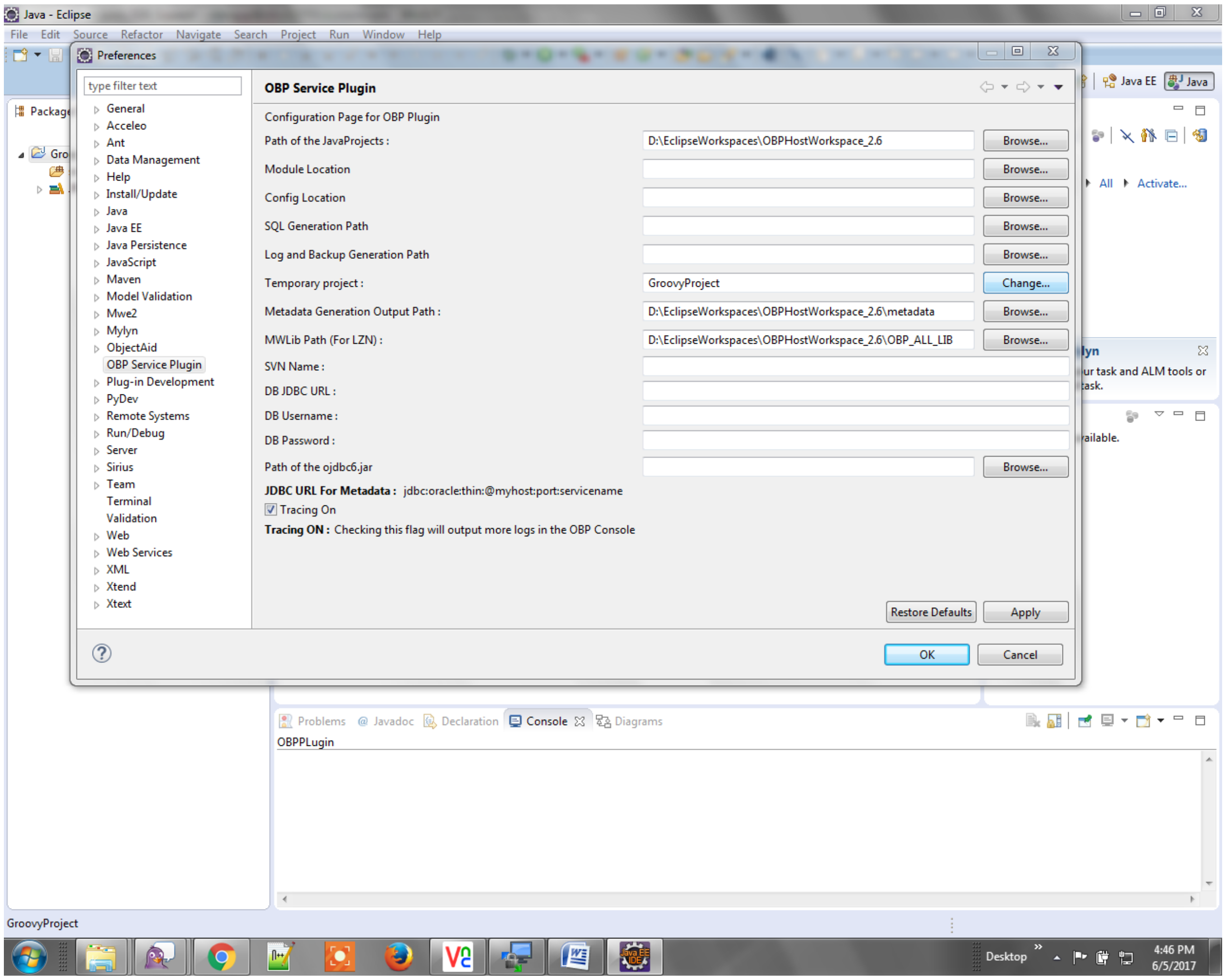


Figure 8–2 Preferences Dialog Box - OBP Service Plugin



OBP Service Plugin

Configuration Page for OBP Plugin

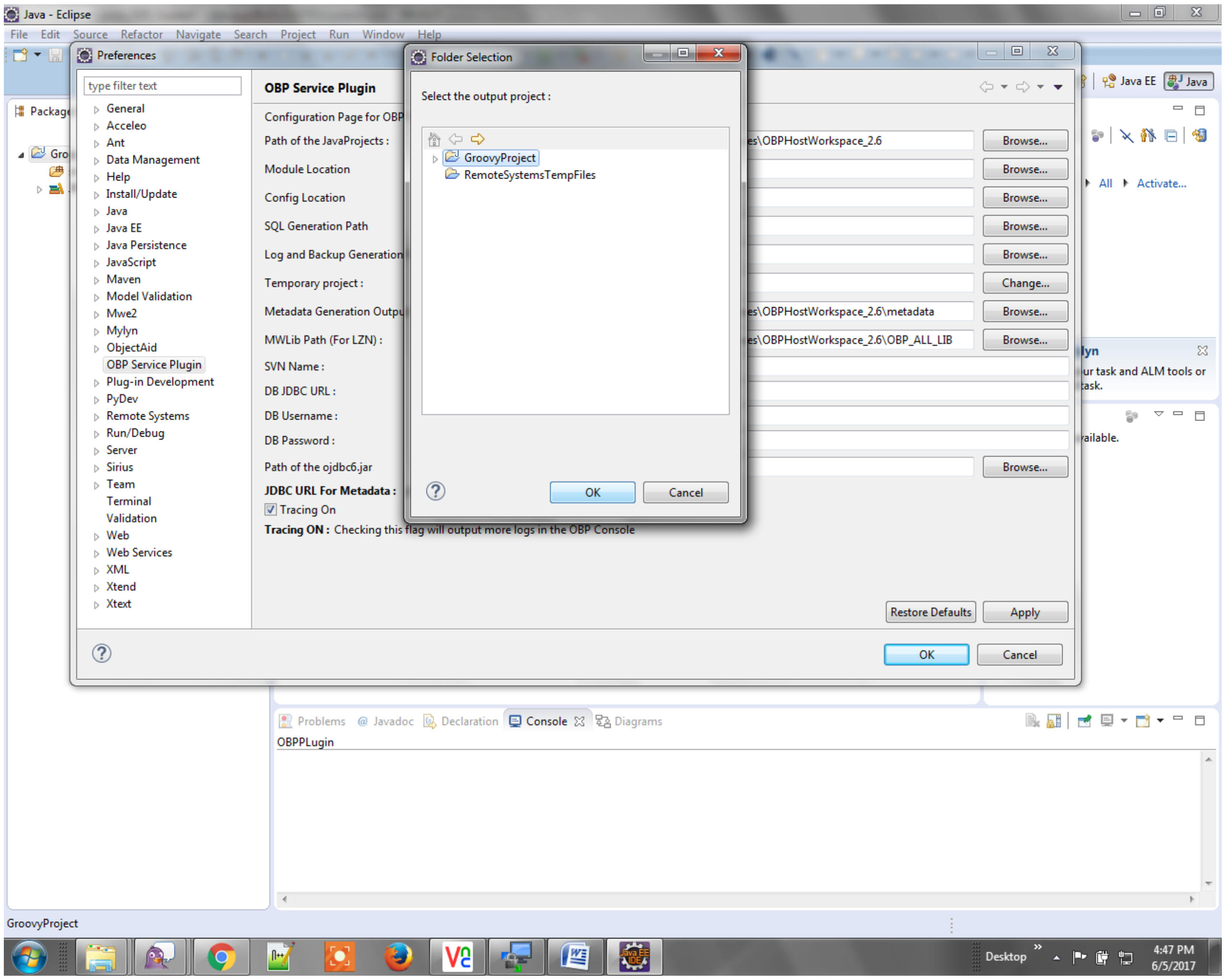
Path of the JavaProjects :	D:\EclipseWorkspaces\OBPHostWorkspace_2.6
Module Location	
Config Location	
SQL Generation Path	
Log and Backup Generation Path	
Temporary project :	GroovyProject
Metadata Generation Output Path :	D:\EclipseWorkspaces\OBPHostWorkspace_2.6\metadata
MWLib Path (For LZN) :	D:\EclipseWorkspaces\OBPHostWorkspace_2.6\OBP_ALL_LIB
SVN Name :	
DB JDBC URL :	
DB Username :	
DB Password :	
Path of the ojdbc6.jar	

JDBC URL For Metadata : jdbc:oracle:thin:@myhost:port:servicename

[✓] Tracing On

Tracing ON : Checking this flag will output more logs in the OBP Console

Figure 8–3 Folder Selection



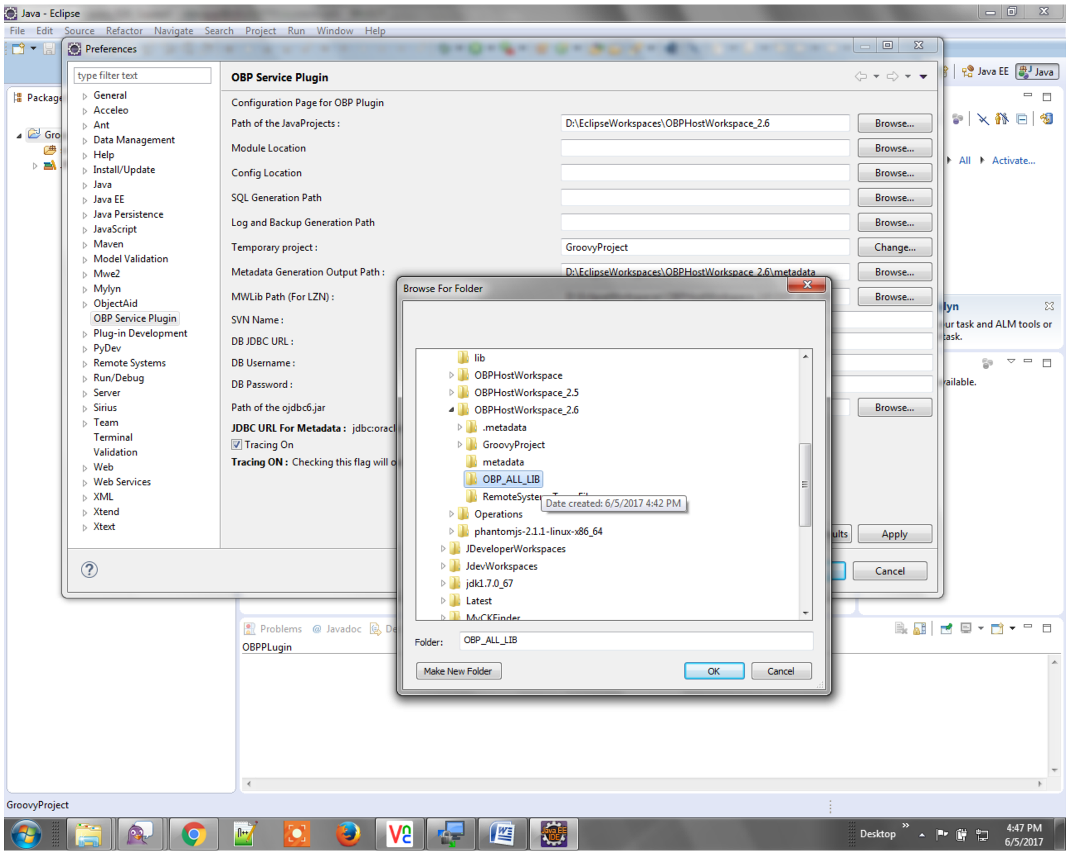
Configuration Page for OBP
Path of the JavaProjects : ...\OBPHostWorkspace_2.6
Module Location
Config Location
SQL Generation Path
Log and Backup Generation
Temporary project :
Metadata Generation Outpu ...\OBPHostWorkspace_2.6\metadata
MWLib Path (For LZN) : ...\OBPHostWorkspace_2.6\OBP_ALL_LIB
SVN Name :
DB JDBC URL :
DB Username :
DB Password :
Path of the ojdbc6.jar
JDBC URL For Metadata :
☑ Tracing On
Tracing ON : Checking this flag will output more logs in the OBP Console

Select the output project :

OBPPLugin

2. The parameter Temporary Project has to be configured to point to the base project where the Groovy Extension Files have to be saved.

Figure 8–4 Browse for Folder



OBP Service Plugin

Configuration Page for OBP Plugin

Path of the JavaProjects : D:\EclipseWorkspaces\OBPHostWorkspace_2.6

Module Location

Config Location

SQL Generation Path

Log and Backup Generation Path

Temporary project : GroovyProject

Metadata Generation Output Path : D:\EclipseWorkspaces\OBPHostWorkspace_2.6\metadata

MWLib Path (For LZN) :

SVN Name :

DB JDBC URL :

DB Username :

DB Password :

Path of the ojdbc6.jar

JDBC URL For Metadata : jdbc:oracl

☑ Tracing On

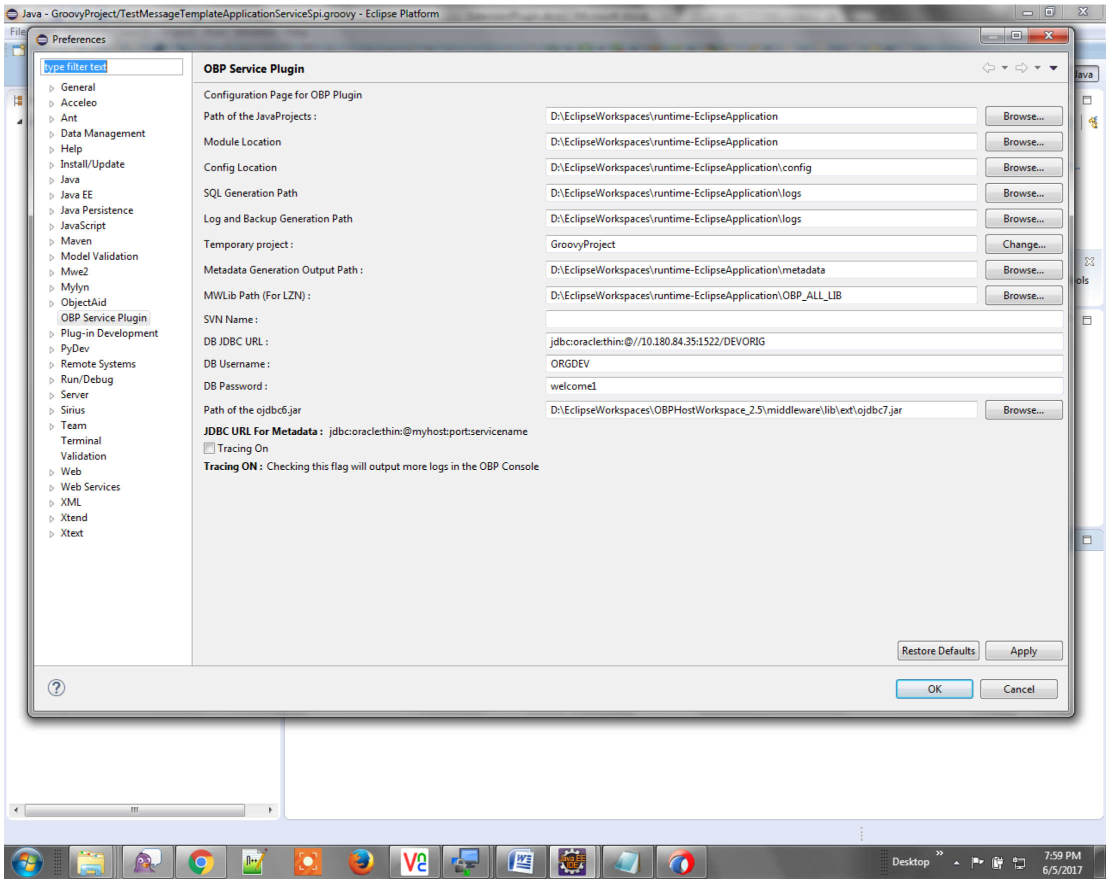
Tracing ON : Checking this flag will o

Browse For Folder

Folder: OBP_ALL_LIB

3. The parameter “MWLib Path” has to be configured to point to the directory where all the OBP Host jar files have been kept.

Figure 8–5 Configuring MWLib Path Parameter



OBP Service Plugin

Configuration Page for OBP Plugin

Path of the JavaProjects :	D:\EclipseWorkspaces\runtime-EclipseApplication
Module Location	D:\EclipseWorkspaces\runtime-EclipseApplication
Config Location	D:\EclipseWorkspaces\runtime-EclipseApplication\config
SQL Generation Path	D:\EclipseWorkspaces\runtime-EclipseApplication\logs
Log and Backup Generation Path	D:\EclipseWorkspaces\runtime-EclipseApplication\logs
Temporary project :	GroovyProject
Metadata Generation Output Path :	D:\EclipseWorkspaces\runtime-EclipseApplication\metadata
MWLib Path (For LZN) :	D:\EclipseWorkspaces\runtime-EclipseApplication\OBP_ALL_LIB
SVN Name :	
DB JDBC URL :	jdbc:oracle:thin:@//10.180.84.35:1522/DEVORIG
DB Username :	ORGDEV
DB Password :	welcome1
Path of the ojdbc6.jar	D:\EclipseWorkspaces\OBPHostWorkspace_2.5\middleware\lib\ext\ojdbc7.jar

JDBC URL For Metadata : jdbc:oracle:thin:@myhost:port:servicename

☐ Tracing On

Tracing ON : Checking this flag will output more logs in the OBP Console

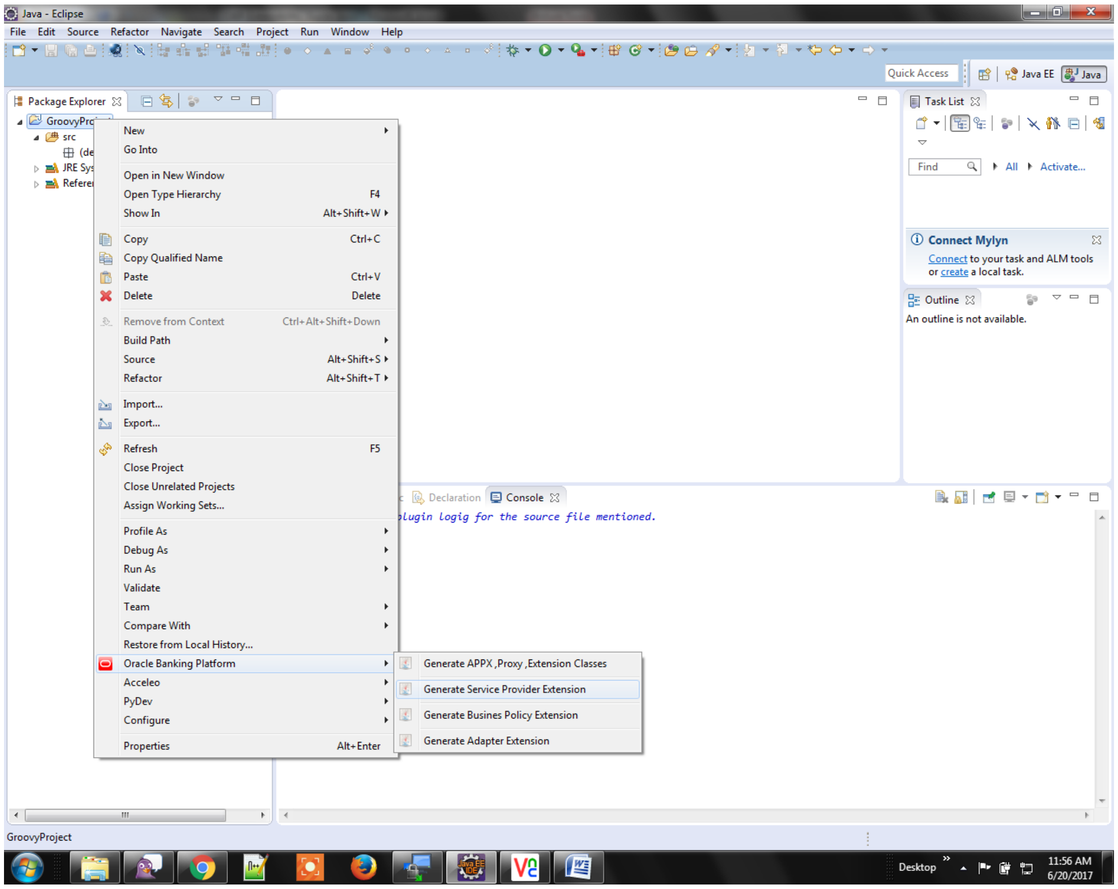
4. The rest of the OBP Service Plugin parameters can be configured as usual.

8.2 Support for Application Service Provider Extension

8.2.1 Generate Application Service Provider Extension

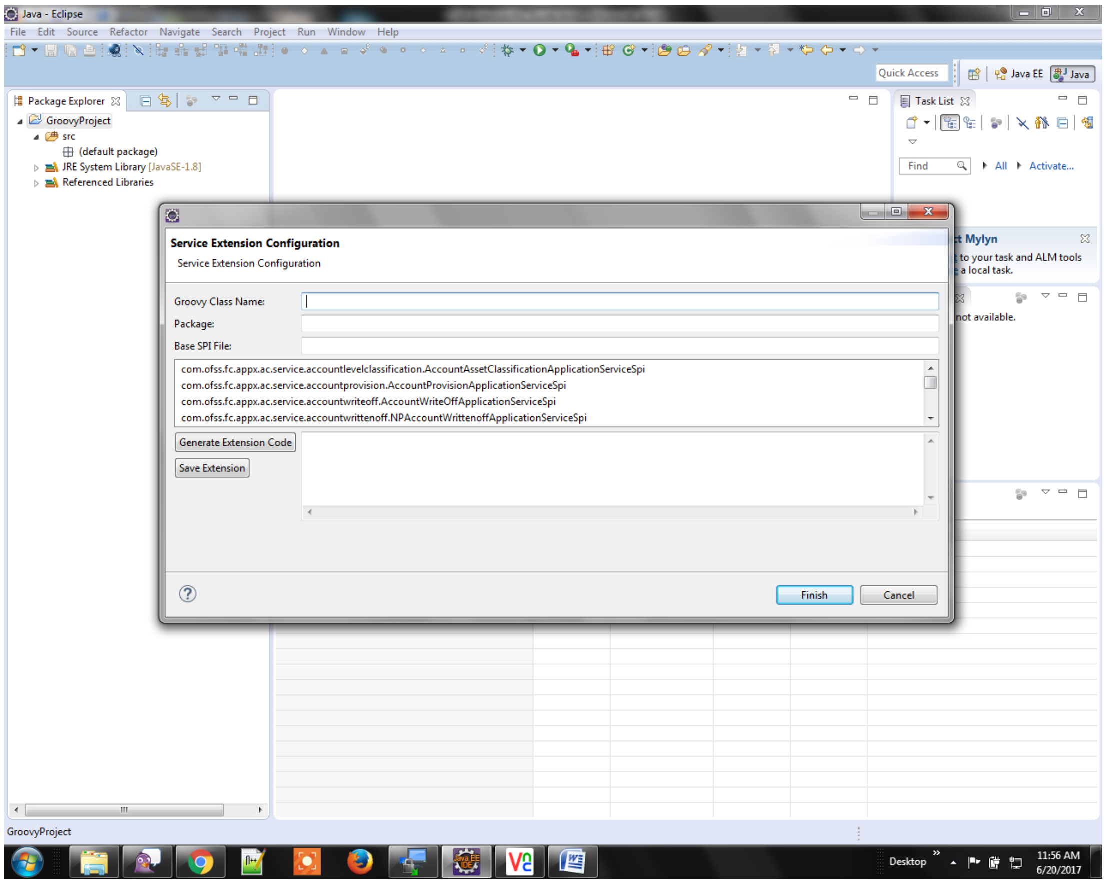
1. The parameter Temporary Project has been configured to point to the base project where the Groovy Extension Files have to be saved.
2. Right click on this project and select Oracle Banking Platform > Generate Service Provider Extension.

Figure 8–6 Java Eclipse - Select Generate Service Provider Extension



The below wizard appears with a list of Base SPI Files.

Figure 8–7 Service Extension Configuration



Service Extension Configuration

Service Extension Configuration

Groovy Class Name:

Package:

Base SPI File:

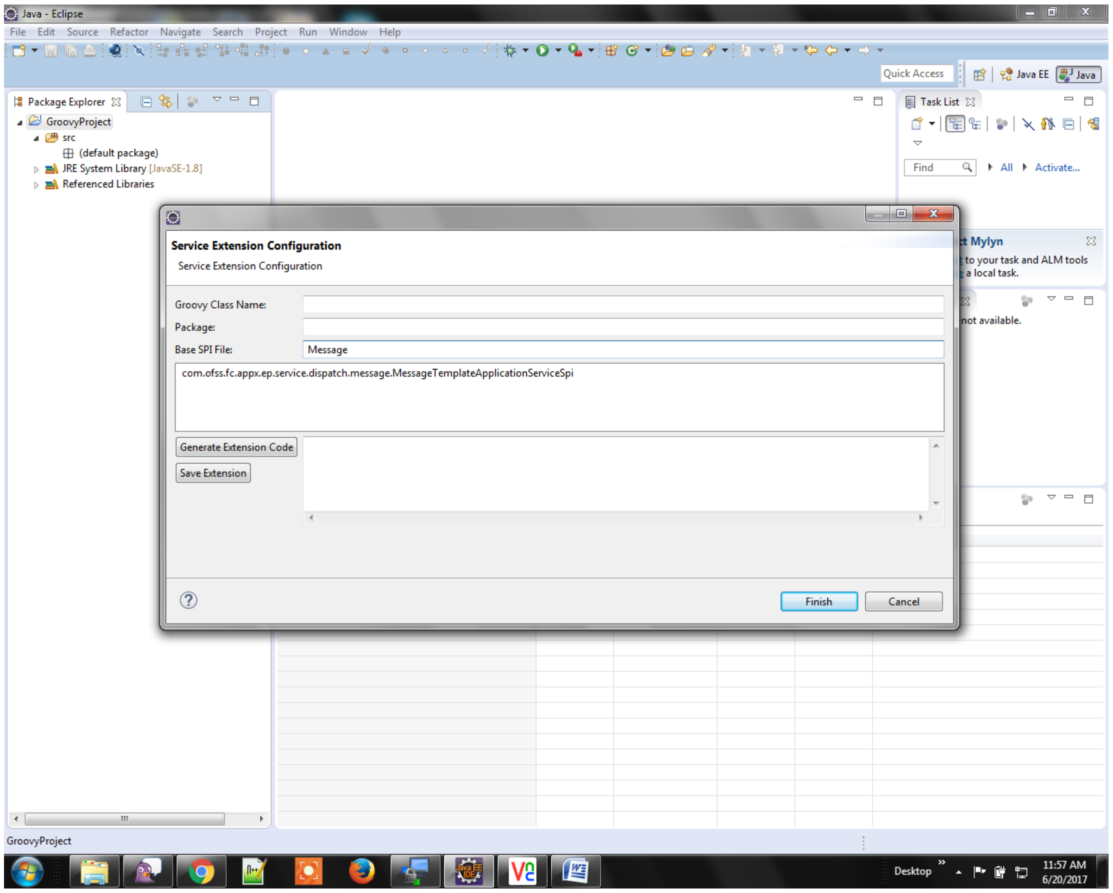
```
com.ofss.fc.appx.ac.service.accountlevelclassification.AccountAssetClassificationApplicationServiceSpi
com.ofss.fc.appx.ac.service.accountprovision.AccountProvisionApplicationServiceSpi
com.ofss.fc.appx.ac.service.accountwriteoff.AccountWriteOffApplicationServiceSpi
com.ofss.fc.appx.ac.service.accountwrittenoff.NPAccountWrittenoffApplicationServiceSpi
```

Generate Extension Code

Save Extension

3. Enter a search keyword to filter the list for the required Base SPI file.

Figure 8–8 Enter Search Keyword to Filter Base SPI File



Service Extension Configuration

Service Extension Configuration

Groovy Class Name:

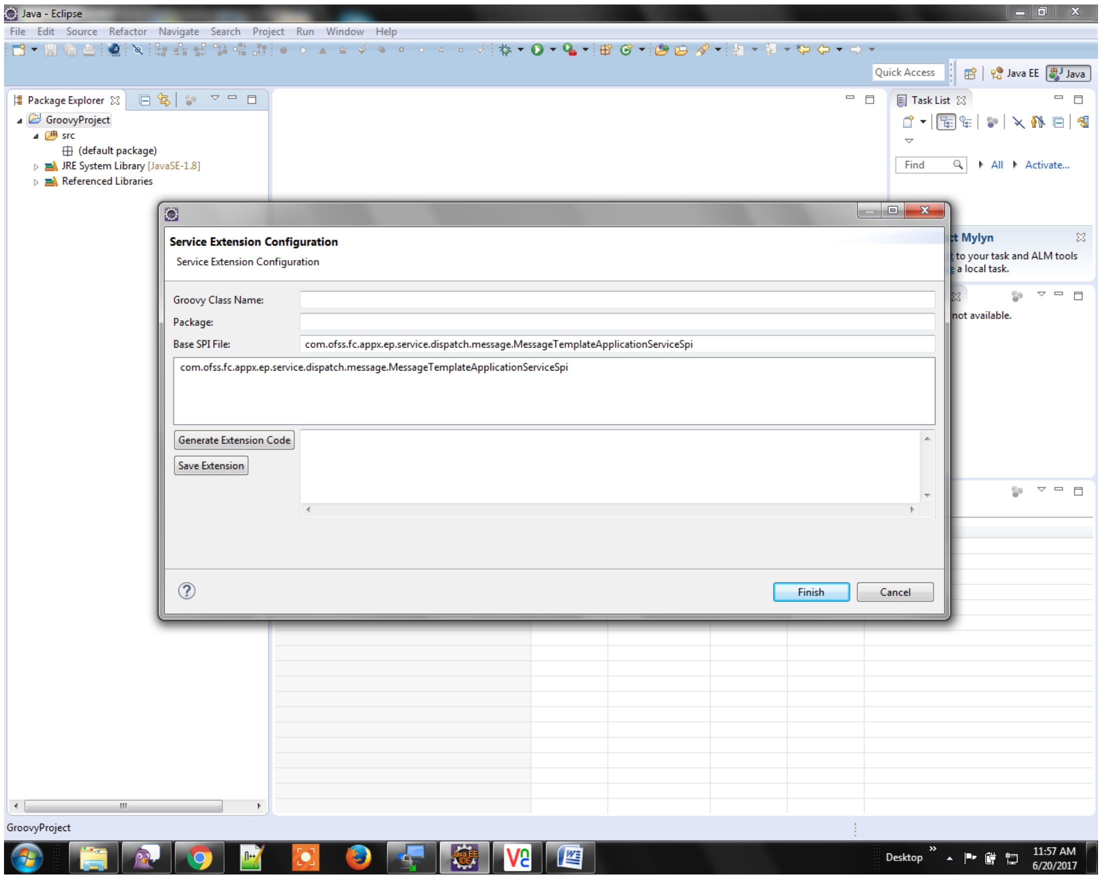
Package:

Base SPI File: Message

com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi

4. Select the filtered Base SPI file from the list such that it appears as the input for Base SPI file.

Figure 8–9 Select Base SPI File

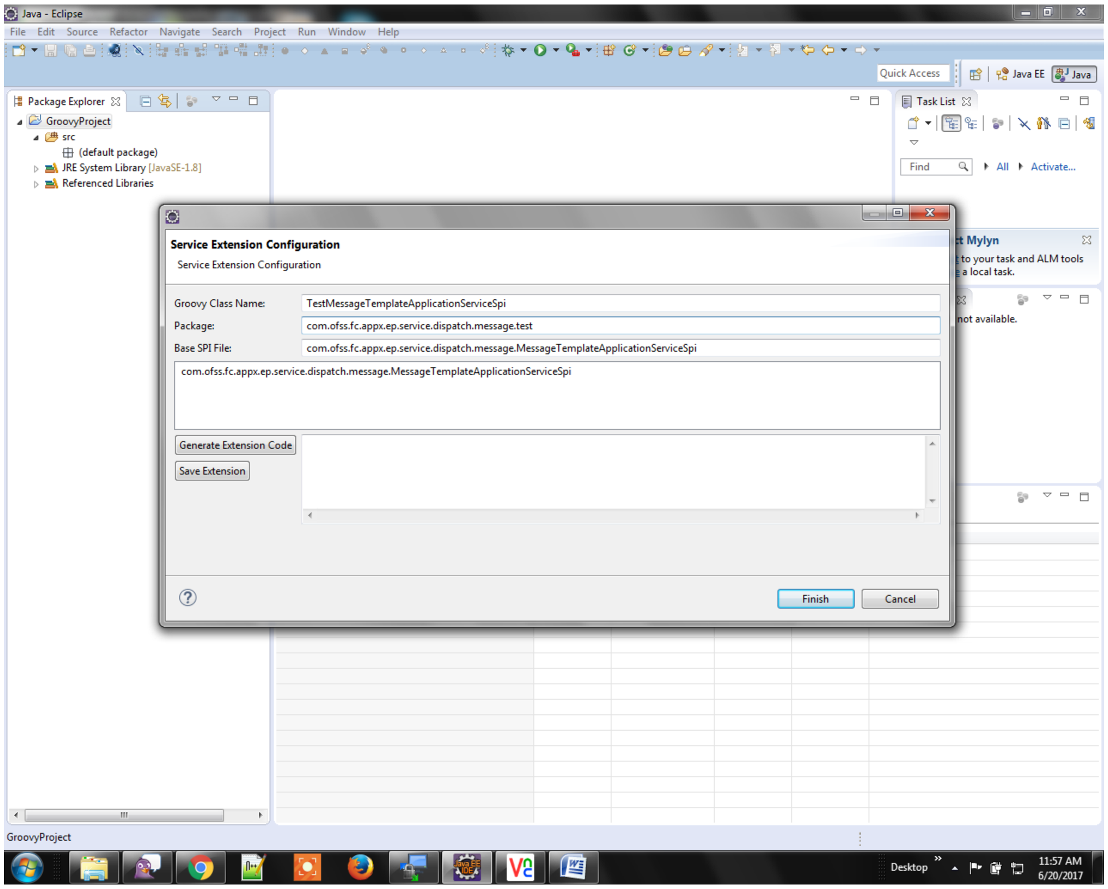

Service Extension Configuration
Service Extension Configuration
Generate Extension Code
Save Extension
Finish
Cancel

Groovy Class Name:
Package:
Base SPI File: com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi

com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi

5. Appropriately set values for the extension class name and package.

Figure 8–10 Set Extension Class Name and Package



Service Extension Configuration

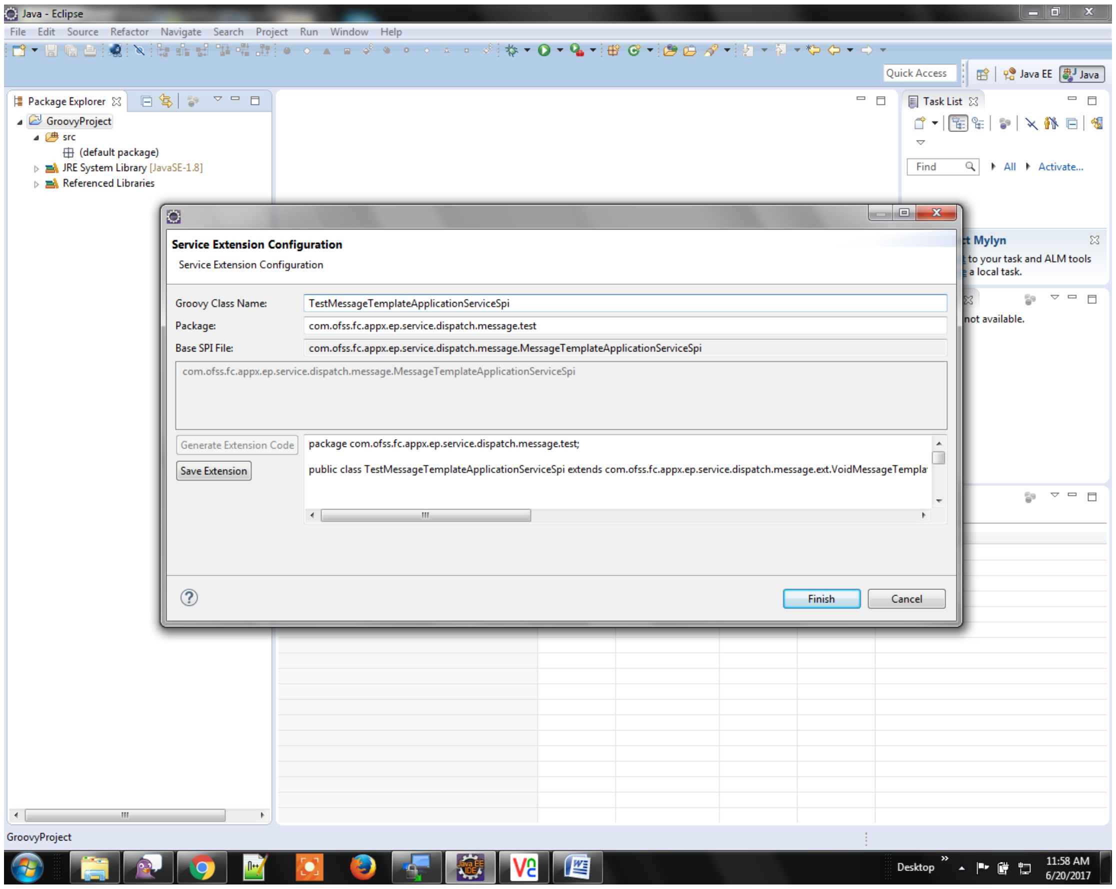
Service Extension Configuration

Groovy Class Name:	TestMessageTemplateApplicationServiceSpi
Package:	com.ofss.fc.appx.ep.service.dispatch.message.test
Base SPI File:	com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi

com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi

6. Click on Generate Extension Code.

Figure 8–11 Click Generate Extension Code



Service Extension Configuration

Service Extension Configuration

Groovy Class Name: TestMessageTemplateApplicationServiceSpi

Package: com.ofss.fc.appx.ep.service.dispatch.message.test

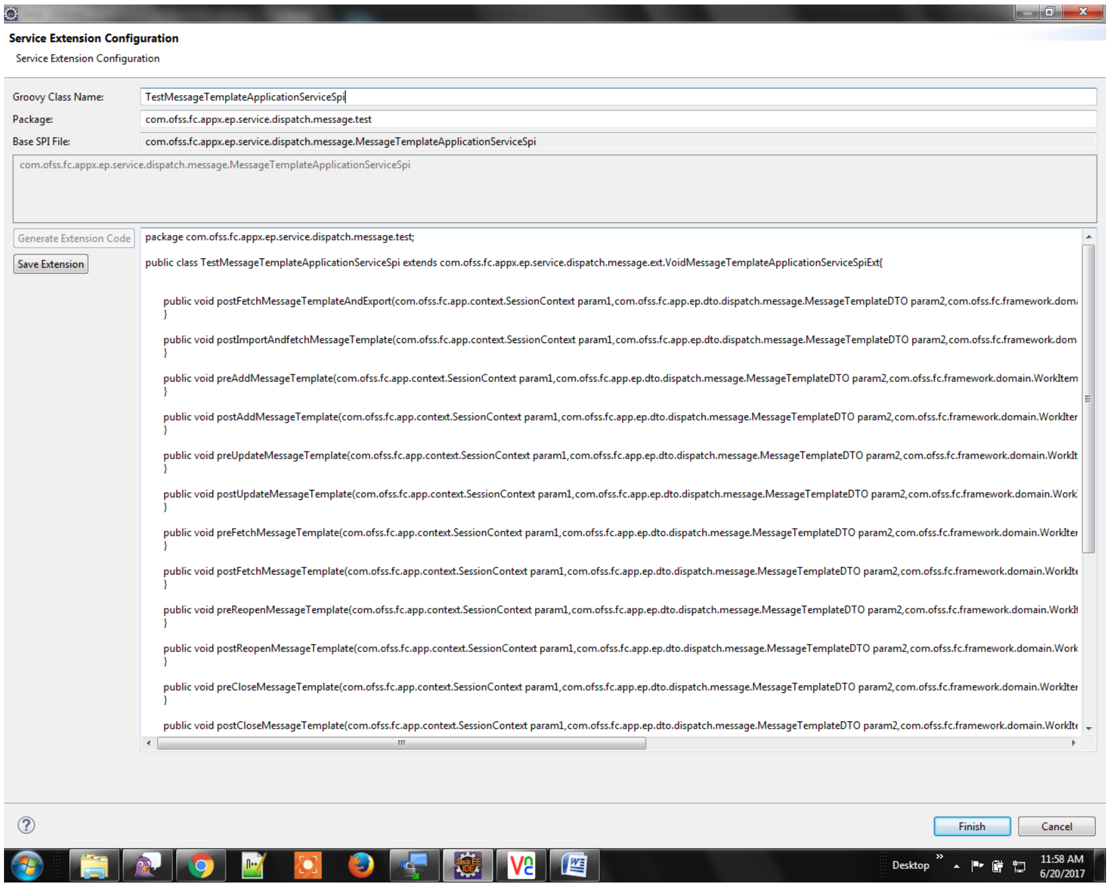
Base SPI File: com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi

com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi

```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dispatch.message.ext.VoidMessageTempla
```

7. The code gets generated with the extension hooks.

Figure 8–12 Extension Code Generated with Extension Hooks



Groovy Class Name: TestMessageTemplateApplicationServiceSpi

Package: com.ofss.fc.appx.ep.service.dispatch.message.test

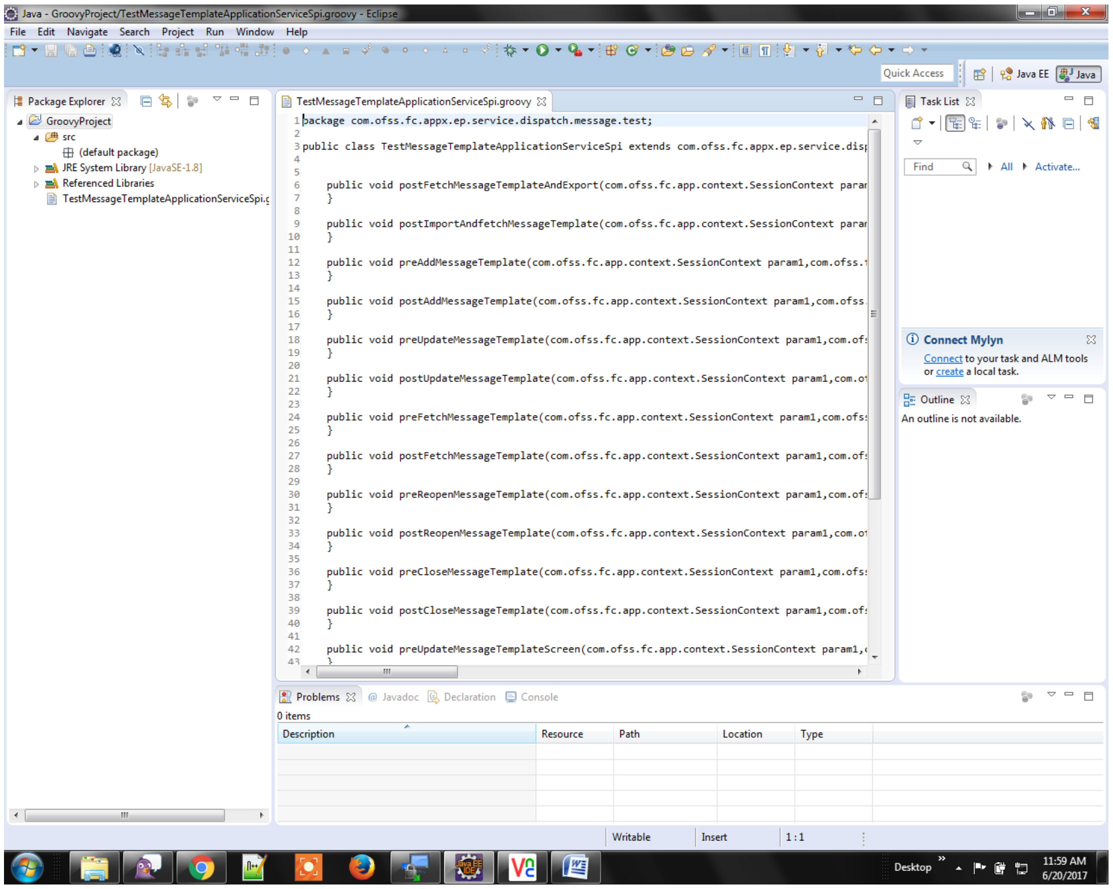
Base SPI File: com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi

com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi

```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dispatch.message.ext.VoidMessageTemplateApplicationServiceSpiExt{


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.dom
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.dom
    }

    public void preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.domain.WorkItem
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.domain.WorkIter
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.domain.WorkIt
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.domain.Work
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.domain.WorkIter
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.domain.WorkIte
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.domain.WorkIt
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.domain.Work
    }

    public void preCloseMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.domain.WorkIter
    }

    public void postCloseMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateDTO param2,com.ofss.fc.framework.domain.WorkIte
```

8. Click on Save Extension and Finish.

Figure 8–13 Save Extension and Finish


Java - GroovyProject/TestMessageTemplateApplicationServiceSpi.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
(default package)
JRE System Library [JavaSE-1.8]
Referenced Libraries
TestMessageTemplateApplicationServiceSpi.g
TestMessageTemplateApplicationServiceSpi.groovy
Task List
Find
All
Activate...
Connect Mylyn
Outline
Problems
Javadoc
Declaration
Console
Writable
Insert
1 : 1
Desktop
11:59 AM
6/20/2017

```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.disp


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext parar
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext parar
    }

    public void preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void preCloseMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postCloseMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void preUpdateMessageTemplateScreen(com.ofss.fc.app.context.SessionContext param1,c
    }
```

Connect to your task and ALM tools or create a local task.

An outline is not available.

0 items

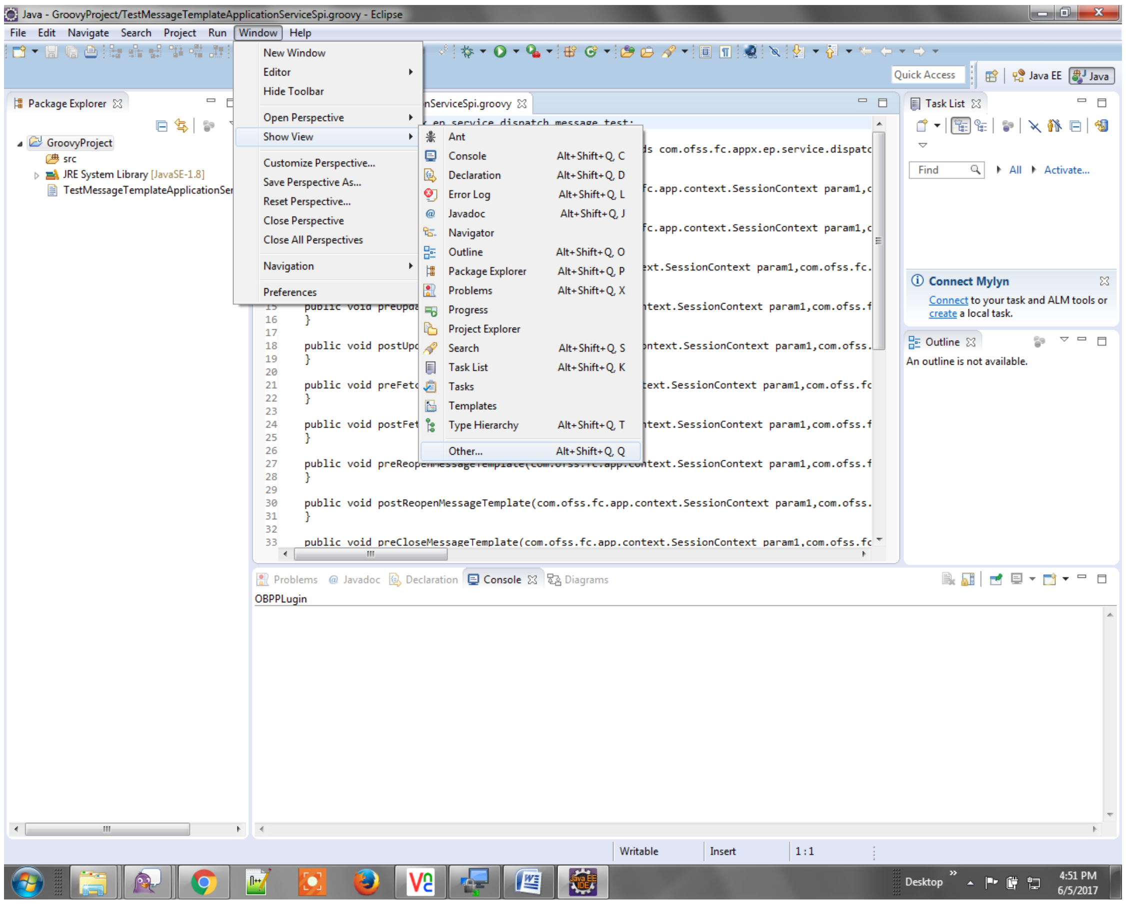
Description	Resource	Path	Location	Type

9. The extension gets saved in the project created to contain the Extension classes generated through the plugin. You can add all the code required in the pre and post hooks for extensibility of the base Application Service.

8.2.2 Configure OBP Extensibility Server Explorer - View

1. In eclipse click on Window > Show View > Other.

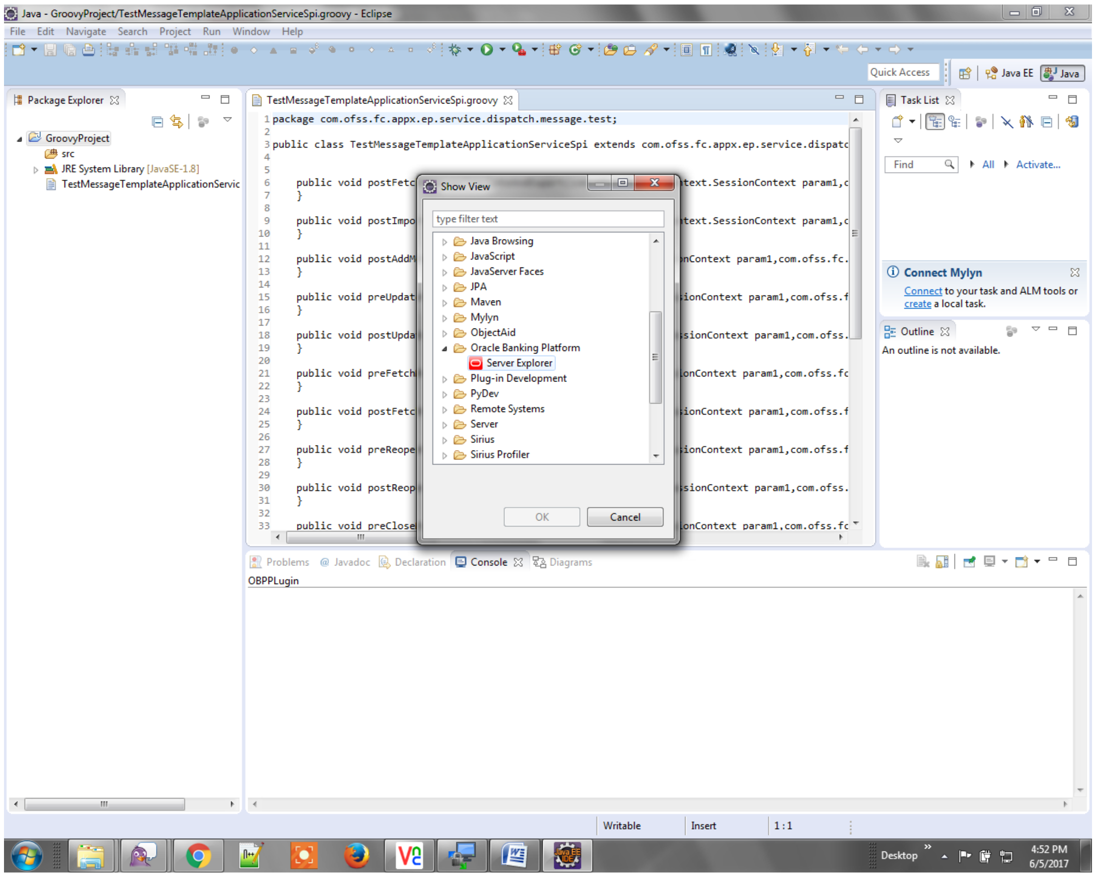
Figure 8–14 Java Eclipse



```
    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preCloseMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc
```

2. Click on Oracle Banking Platform > Server Explorer.

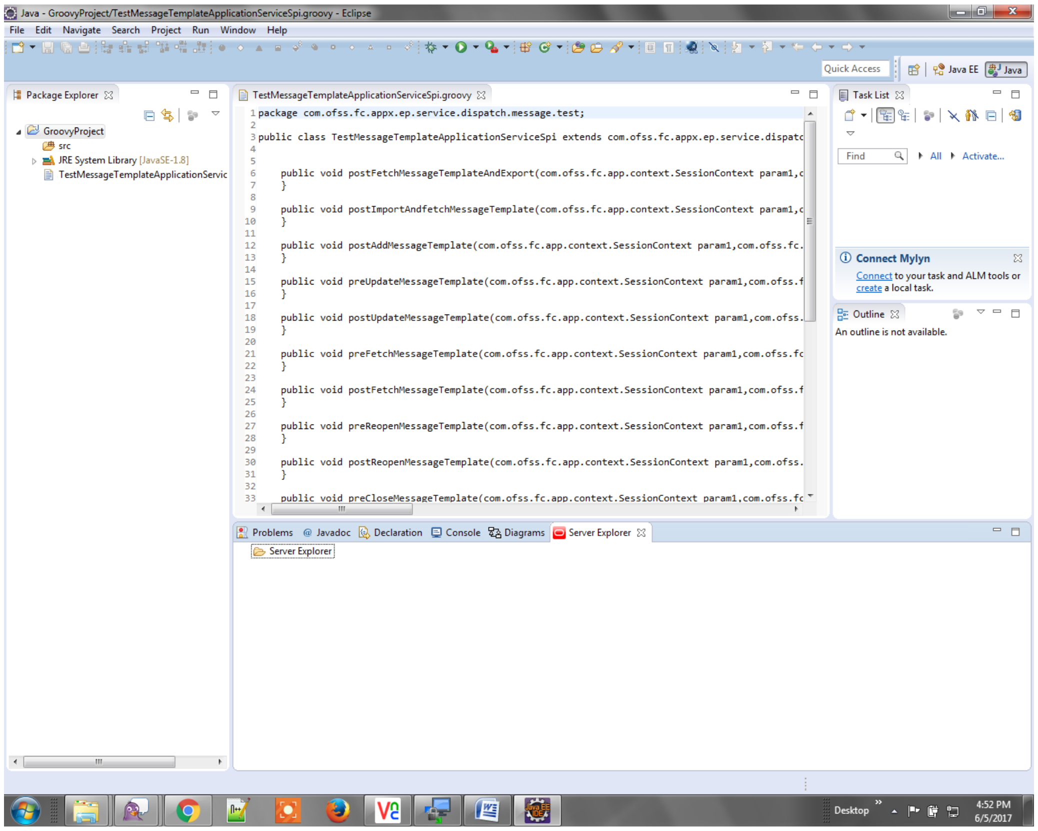
Figure 8–15 Click Server Explorer



```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dispatc


    public void postFetc                        text.SessionContext param1,c
    }

    public void postImpo                        text.SessionContext param1,c
    }

    public void postAddM                        nContext param1,com.ofss.fc.
    }

    public void preUpdat                        ionContext param1,com.ofss.f
    }

    public void postUpda                        sionContext param1,com.ofss.
    }

    public void preFetch                        onContext param1,com.ofss.fc
    }

    public void postFetc                        ionContext param1,com.ofss.f
    }

    public void preReope                        ionContext param1,com.ofss.f
    }

    public void postReop                        sionContext param1,com.ofss.
    }

    public void preClose                        onContext param1,com.ofss.fc
```

3. The ServerExplorer view tab opens up.

Figure 8–16 Server Explorer View tab

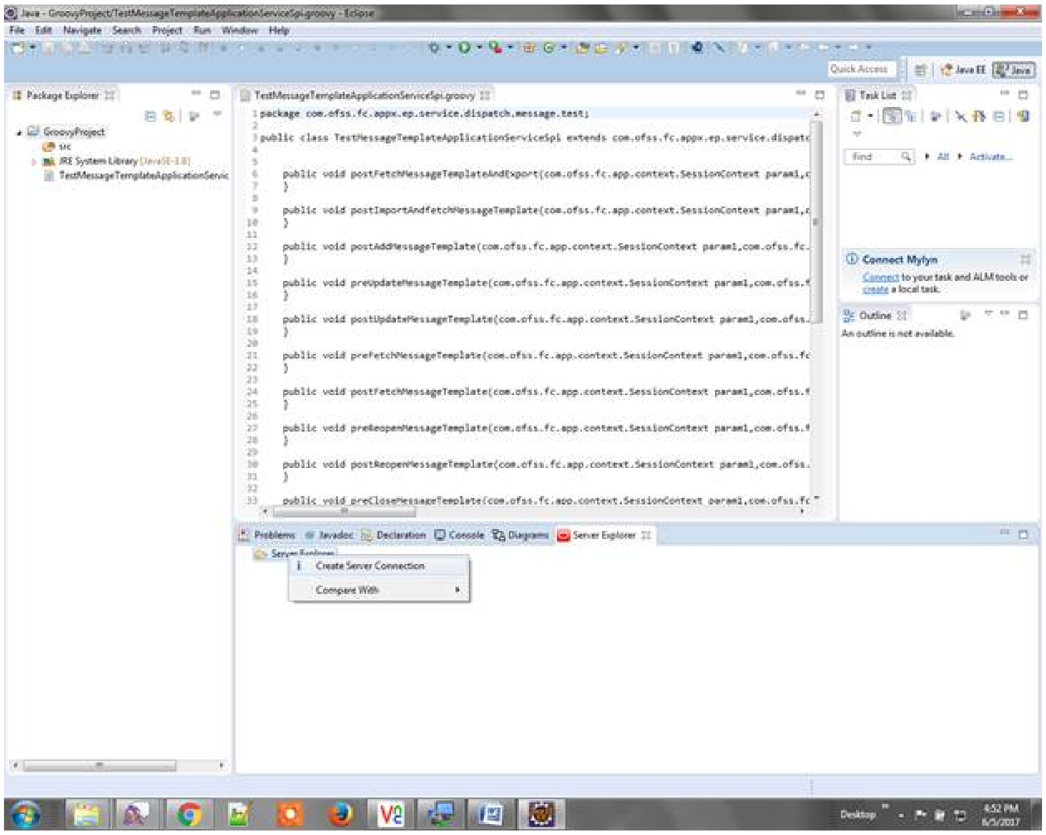


```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dispatc


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext param1,c
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,c
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preCloseMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc
```

Java - GroovyProject/TestMessageTemplateApplicationServiceSpi.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
TestMessageTemplateApplicationServic
TestMessageTemplateApplicationServiceSpi.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Diagrams
Server Explorer
Server Explorer
Desktop
4:52 PM
6/5/2017

4. Right click on Server Explorer and click Create Server Connection.

Figure 8–17 Create Server Connection

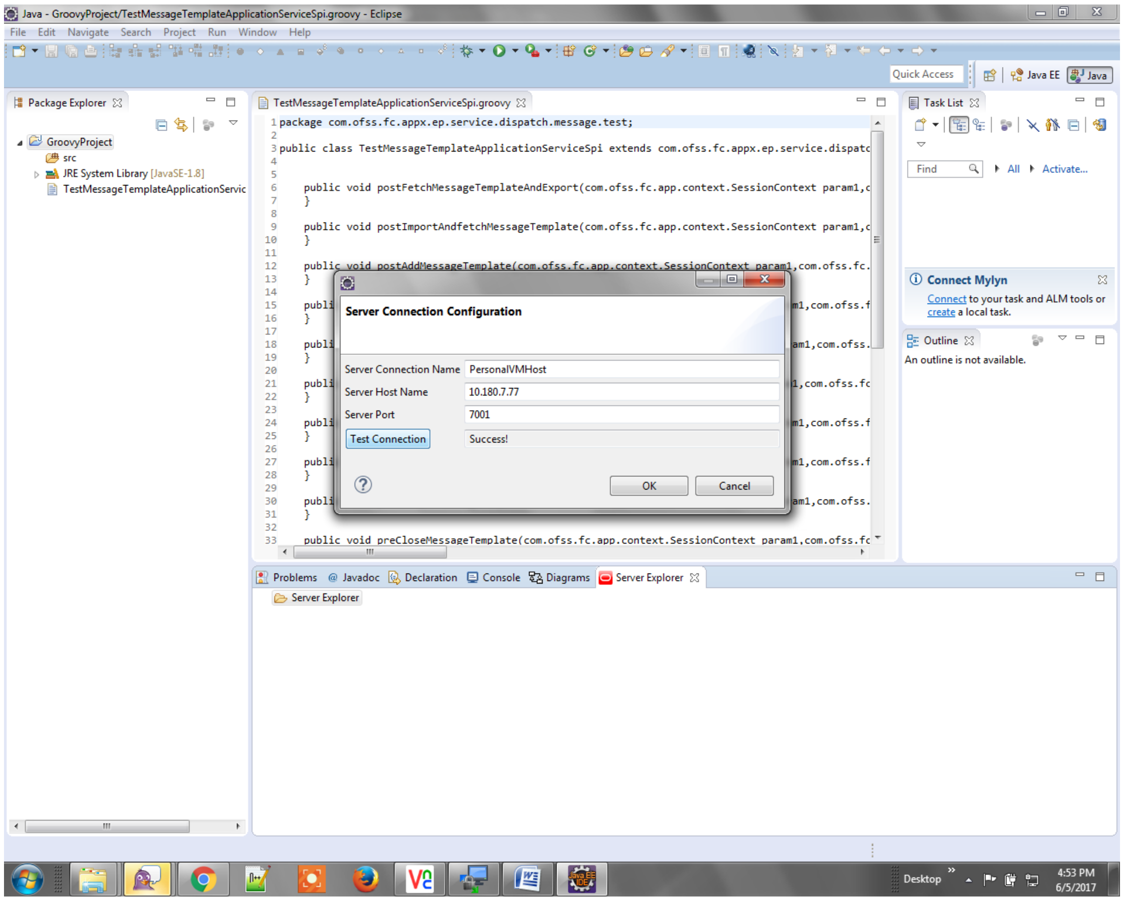


```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dispatc


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext param1,c
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,c
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void prefetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preCloseMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc
```

Java - GroovyProject/TestMessageTemplateApplicationServiceSpi.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
TestMessageTemplateApplicationServic
TestMessageTemplateApplicationServiceSpi.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Diagrams
Server Explorer
Server Explorer
Create Server Connection
Compare With
Desktop
4:52 PM
6/5/2017

5. Provide values for Connection, name, ip, and port and test connection and click ok.

Figure 8–18 Provide Details for Server Configuration



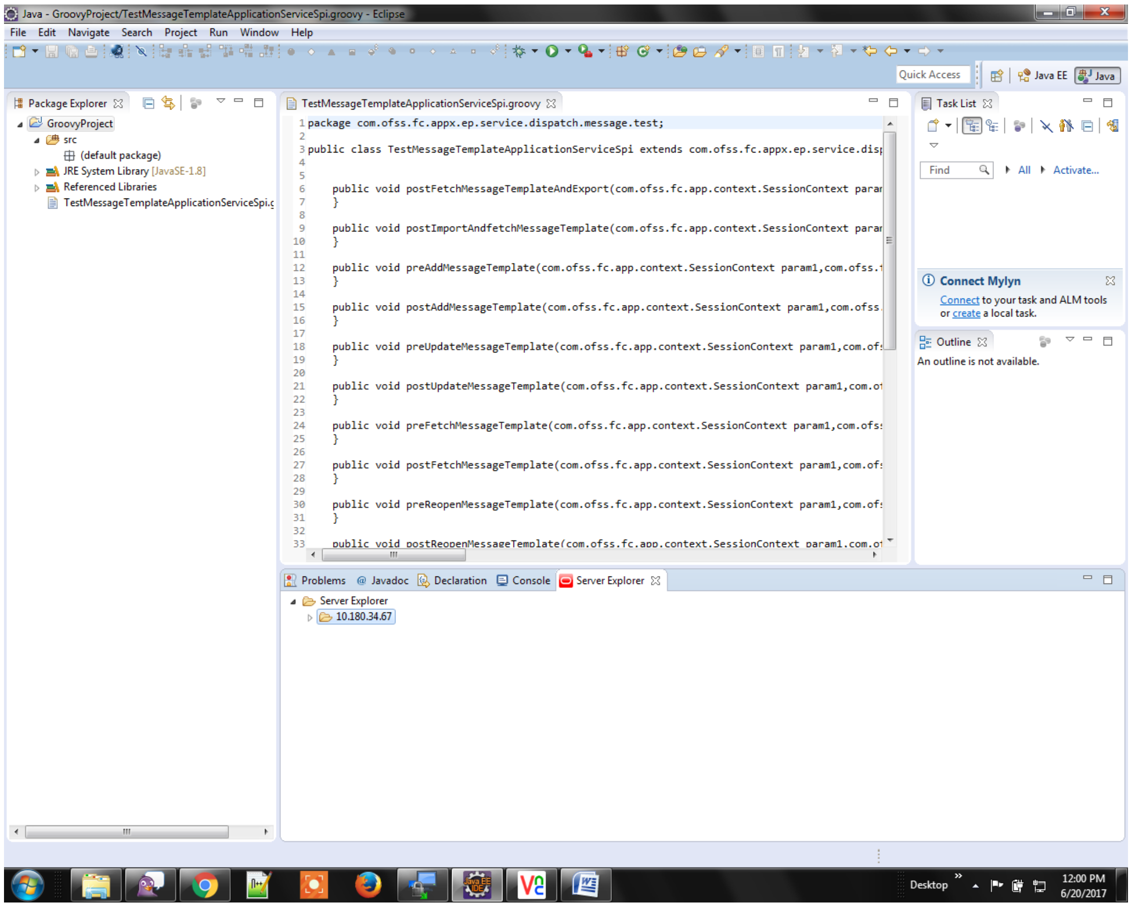
```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dispatc


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext param1,c
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,c
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.
    }
```

Server Connection Configuration

Server Connection Name	PersonalVMHost
Server Host Name	10.180.7.77
Server Port	7001
Test Connection	Success!

6. The configured server appears as a child of Server Explorer. The configured server also lists all Application Service SPI Extensions and all Business Policy Extensions and all Adapter Extensions already deployed in server.

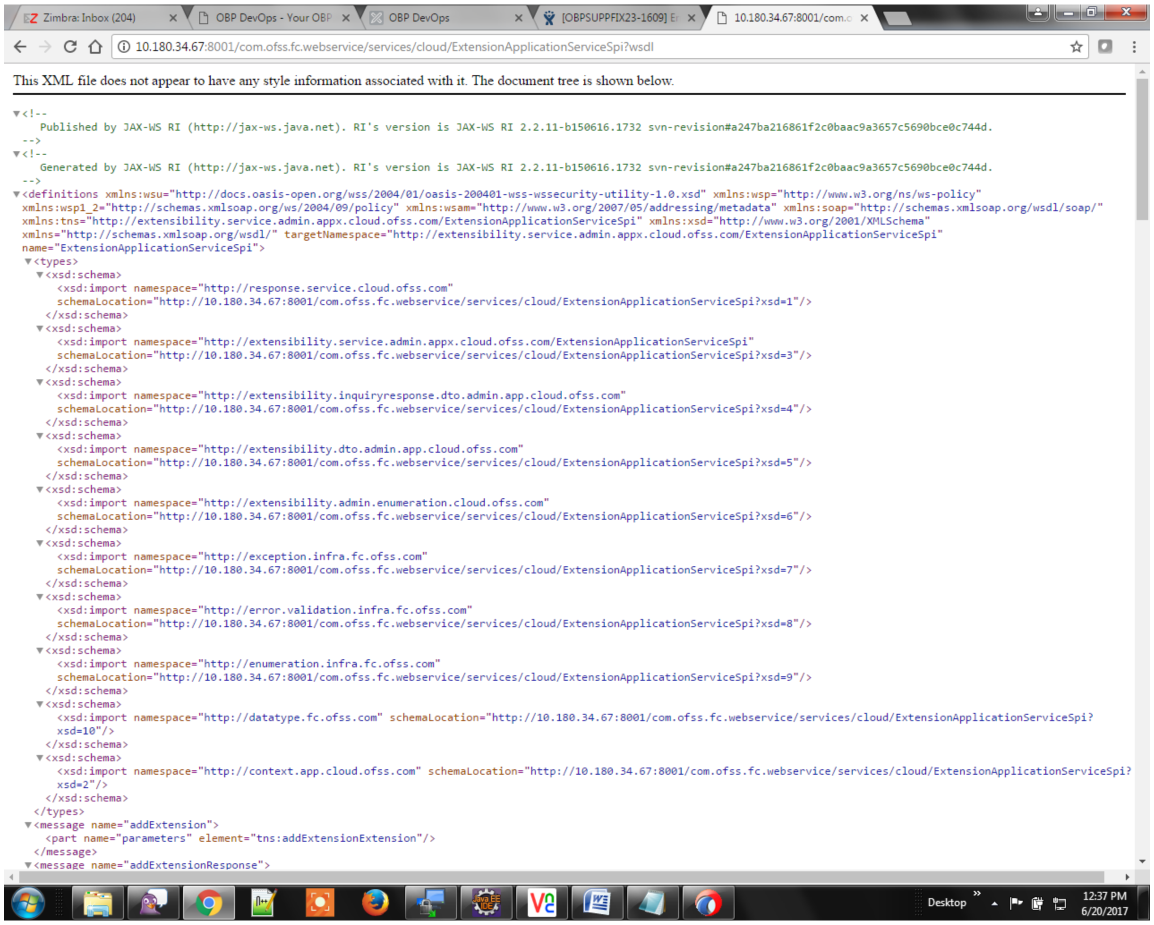
Figure 8–19 Server Configured


Java - GroovyProject/TestMessageTemplateApplicationServiceSpi.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
(default package)
JRE System Library [JavaSE-1.8]
Referenced Libraries
TestMessageTemplateApplicationServiceSpi.g
TestMessageTemplateApplicationServiceSpi.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Server Explorer
10.180.34.67
Desktop
12:00 PM
6/20/2017

```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.disp


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext paran
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext paran
    }

    public void preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
```

8.2.3 Exposed Webservice for Application Service SPI Extensions

Figure 8–20 ExtensionApplicationServiceSpi



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<!--
   Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.11-b150616.1732 svn-revision#a247ba216861f2c0baac9a3657c5690bce0c744d.
 -->
▼<!--
   Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.11-b150616.1732 svn-revision#a247ba216861f2c0baac9a3657c5690bce0c744d.
 -->
▼<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
 xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tns="http://extensibility.service.admin.appx.cloud.ofss.com/ExtensionApplicationServiceSpi" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://extensibility.service.admin.appx.cloud.ofss.com/ExtensionApplicationServiceSpi"
 name="ExtensionApplicationServiceSpi">
  ▼<types>
    ▼<xsd:schema>
       <xsd:import namespace="http://response.service.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/ExtensionApplicationServiceSpi?xsd=1"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.service.admin.appx.cloud.ofss.com/ExtensionApplicationServiceSpi"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/ExtensionApplicationServiceSpi?xsd=3"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.inquiryresponse.dto.admin.app.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/ExtensionApplicationServiceSpi?xsd=4"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.dto.admin.app.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/ExtensionApplicationServiceSpi?xsd=5"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.admin.enumeration.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/ExtensionApplicationServiceSpi?xsd=6"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://exception.infra.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/ExtensionApplicationServiceSpi?xsd=7"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://error.validation.infra.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/ExtensionApplicationServiceSpi?xsd=8"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://enumeration.infra.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/ExtensionApplicationServiceSpi?xsd=9"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://datatype.fc.ofss.com" schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/ExtensionApplicationServiceSpi?
       xsd=10"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://context.app.cloud.ofss.com" schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/ExtensionApplicationServiceSpi?
       xsd=2"/>
     </xsd:schema>
   </types>
  ▼<message name="addExtension">
     <part name="parameters" element="tns:addExtensionExtension"/>
   </message>
  ▼<message name="addExtensionResponse">
```

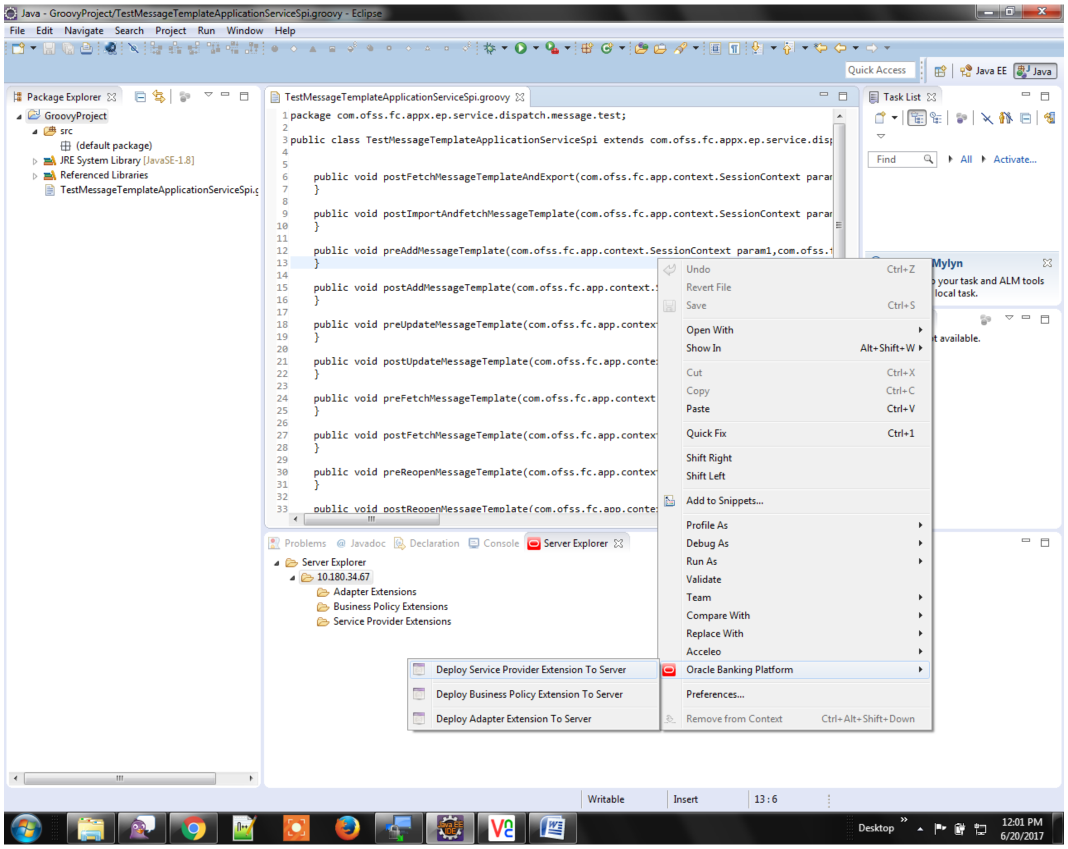
ExtensionApplicationServiceSpi is the web service that exposes the web services to add service provider extensions, fetch service provider extensions and delete service provider extensions. These services are used by the eclipse plugin to deploy extensions, fetch extensions and undeploy extensions at runtime.

8.2.4 Deploy Application Service SPI to Server

Perform the following steps to deploy the application service SPI to server:

1. Right Click on the Extension Class in the Package Explorer or the Code Editor and click on Oracle Banking Platform > Deploy Service Provider Extension To Server.

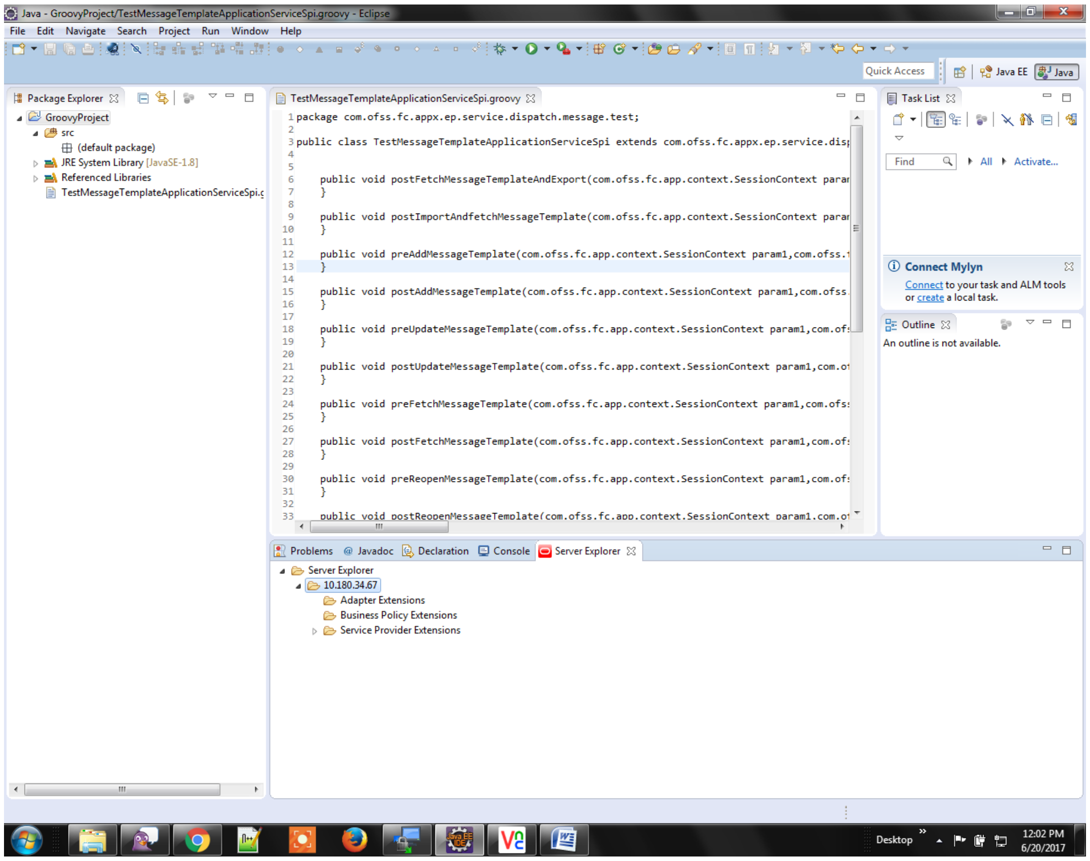
Figure 8–21 Java Eclipse



```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dis


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext para
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext para
    }

    public void preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.contex
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.conte
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.contex
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.contex
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.conte
```

2. It is compulsory to have selected the server under Server Explorer in which the deployment has to occur.

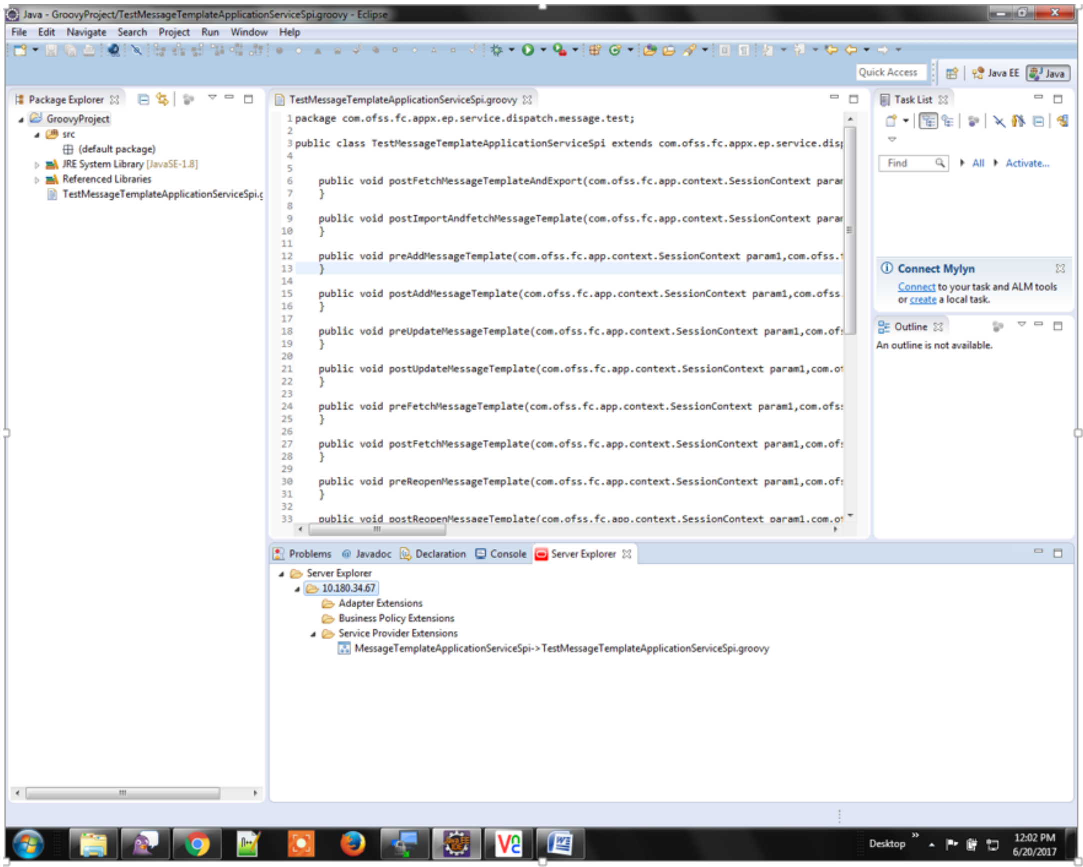
Figure 8–22 Select Server Explorer to Deploy Extension


Java - GroovyProject/TestMessageTemplateApplicationServiceSpi.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
(default package)
JRE System Library [JavaSE-1.8]
Referenced Libraries
TestMessageTemplateApplicationServiceSpi.g
TestMessageTemplateApplicationServiceSpi.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Server Explorer
10.180.34.67
Adapter Extensions
Business Policy Extensions
Service Provider Extensions
Desktop
12:02 PM
6/20/2017

```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.disp


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext parar
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext parar
    }

    public void preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
```

3. The Extension gets deployed in the server.

Figure 8–23 Extension Deployed



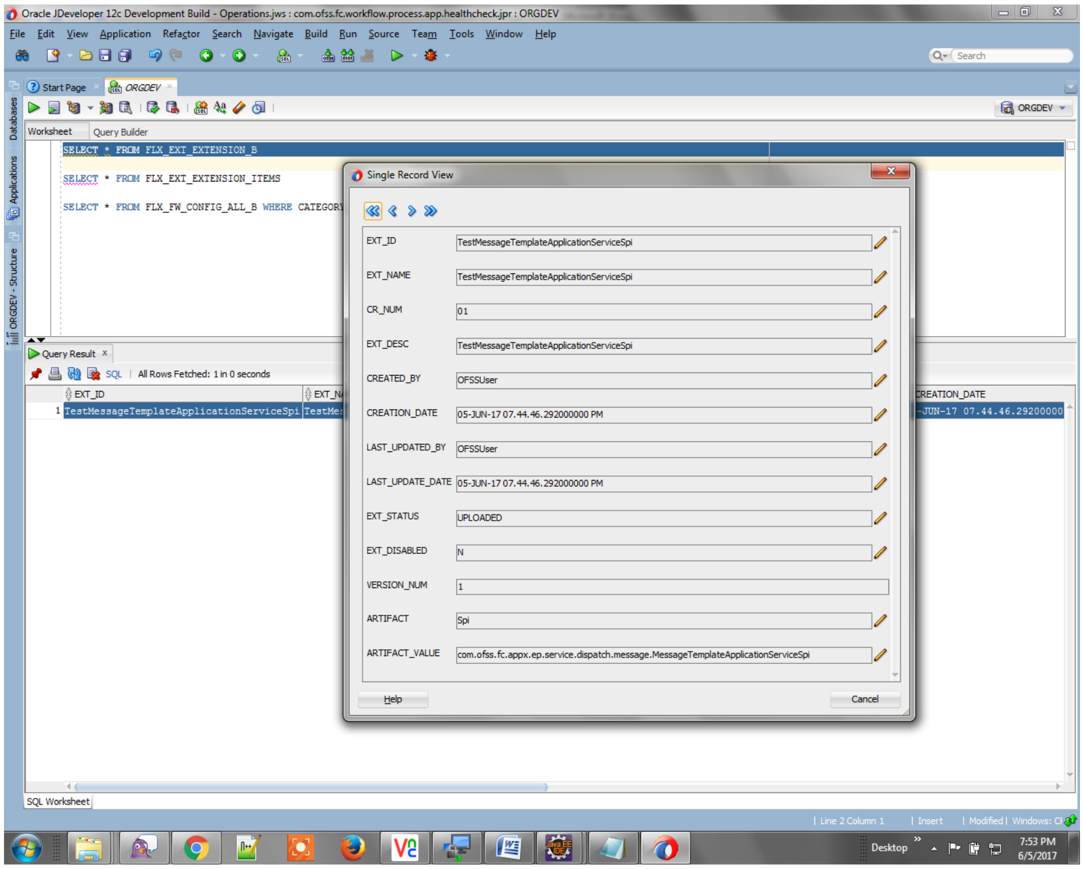
```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dis


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext para
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext para
    }

    public void preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.o
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.o
```

```
Server Explorer
  10.180.34.67
    Adapter Extensions
    Business Policy Extensions
    Service Provider Extensions
      MessageTemplateApplicationServiceSpi->TestMessageTemplateApplicationServiceSpi.groovy
```

4. The Extension gets deployed in the server and appears under the appropriate child of Server configured under Server Explorer.

8.2.5 Database Inserts: Application Service SPI Extension Deployment

Figure 8–24 Application Service SPI Extension Deployment - Single Record View

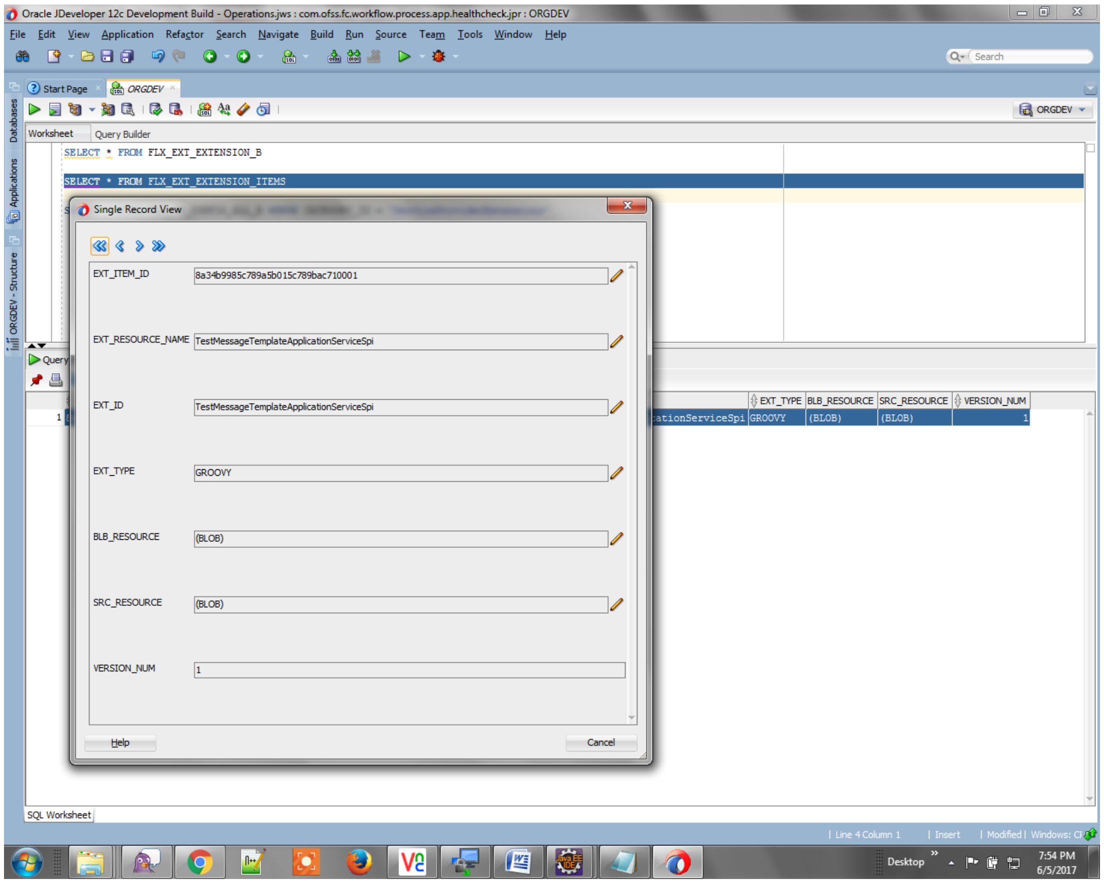

Oracle JDeveloper 12c Development Build - Operations.jws : com.ofss.fc.workflow.process.app.healthcheck.jpr : ORGDEV
File Edit View Application Refactor Search Navigate Build Run Source Team Tools Window Help
Start Page
ORGDEV
Worksheet
Query Builder
Query Result
SQL Worksheet
Single Record View
Help
Cancel

```
SELECT * FROM FLX_EXT_EXTENSION_B

SELECT * FROM FLX_EXT_EXTENSION_ITEMS

SELECT * FROM FLX_FW_CONFIG_ALL_B WHERE CATEGORY
```

All Rows Fetched: 1 in 0 seconds

EXT_ID	TestMessageTemplateApplicationServiceSpi
EXT_NAME	TestMessageTemplateApplicationServiceSpi
CR_NUM	01
EXT_DESC	TestMessageTemplateApplicationServiceSpi
CREATED_BY	OFSSUser
CREATION_DATE	05-JUN-17 07.44.46.292000000 PM
LAST_UPDATED_BY	OFSSUser
LAST_UPDATE_DATE	05-JUN-17 07.44.46.292000000 PM
EXT_STATUS	UPLOADED
EXT_DISABLED	N
VERSION_NUM	1
ARTIFACT	Spi
ARTIFACT_VALUE	com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi

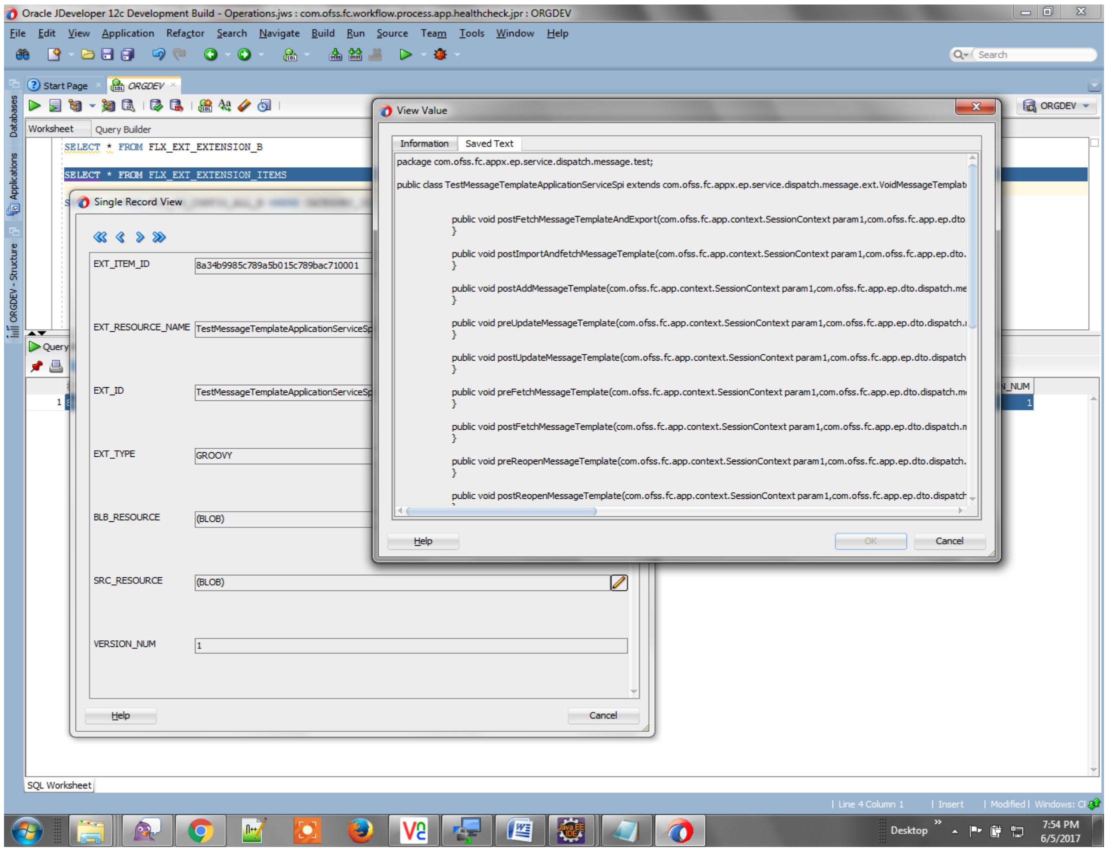
Figure 8–25 Application Service SPI Extension Deployment - Single Record View


Oracle JDeveloper 12c Development Build - Operations.jws : com.ofss.fc.workflow.process.app.healthcheck.jpr : ORGDEV
File Edit View Application Refactor Search Navigate Build Run Source Team Tools Window Help
Start Page
ORGDEV
Worksheet
Query Builder
Single Record View
Help
Cancel
SQL Worksheet

```
SELECT * FROM FLX_EXT_EXTENSION_B

SELECT * FROM FLX_EXT_EXTENSION_ITEMS
```

EXT_ITEM_ID	8a34b9985c789a5b015c789bac710001
EXT_RESOURCE_NAME	TestMessageTemplateApplicationServiceSpi
EXT_ID	TestMessageTemplateApplicationServiceSpi
EXT_TYPE	GROOVY
BLB_RESOURCE	(BLOB)
SRC_RESOURCE	(BLOB)
VERSION_NUM	1

Figure 8–26 Application Service SPI Extension Deployment - View Value

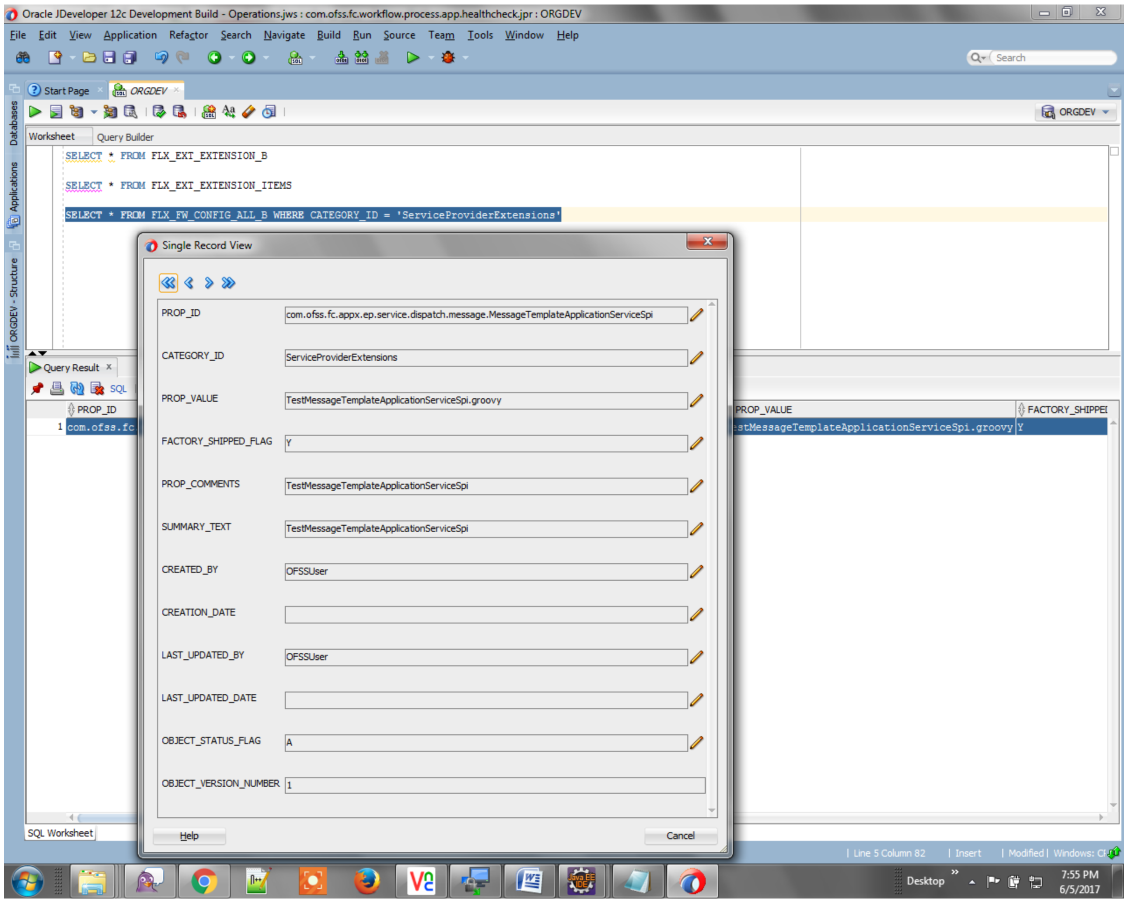

Oracle JDeveloper 12c Development Build - Operations.jws : com.ofss.fc.workflow.process.app.healthcheck.jpr : ORGDEV
File Edit View Application Refactor Search Navigate Build Run Source Team Tools Window Help
Start Page
ORGDEV
Worksheet
Query Builder
Single Record View
View Value
Information
Saved Text
Help
OK
Cancel
SQL Worksheet
Line 4 Column 1
Insert
Modified
Desktop
7:54 PM
6/5/2017

```
SELECT * FROM FLX_EXT_EXTENSION_B

SELECT * FROM FLX_EXT_EXTENSION_ITEMS
```

EXT_ITEM_ID	8a34b9985c789a5b015c789bac710001
EXT_RESOURCE_NAME	TestMessageTemplateApplicationServiceSp
EXT_ID	TestMessageTemplateApplicationServiceSp
EXT_TYPE	GROOVY
BLB_RESOURCE	(BLOB)
SRC_RESOURCE	(BLOB)
VERSION_NUM	1

```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dispatch.message.ext.VoidMessageTemplat


            public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.
            }

            public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.
            }

            public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.me
            }

            public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.
            }

            public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch
            }

            public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.m
            }

            public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.n
            }

            public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch.
            }

            public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.fc.app.ep.dto.dispatch
```

Figure 8–27 Application Service SPI Extension Deployment - Single Record View



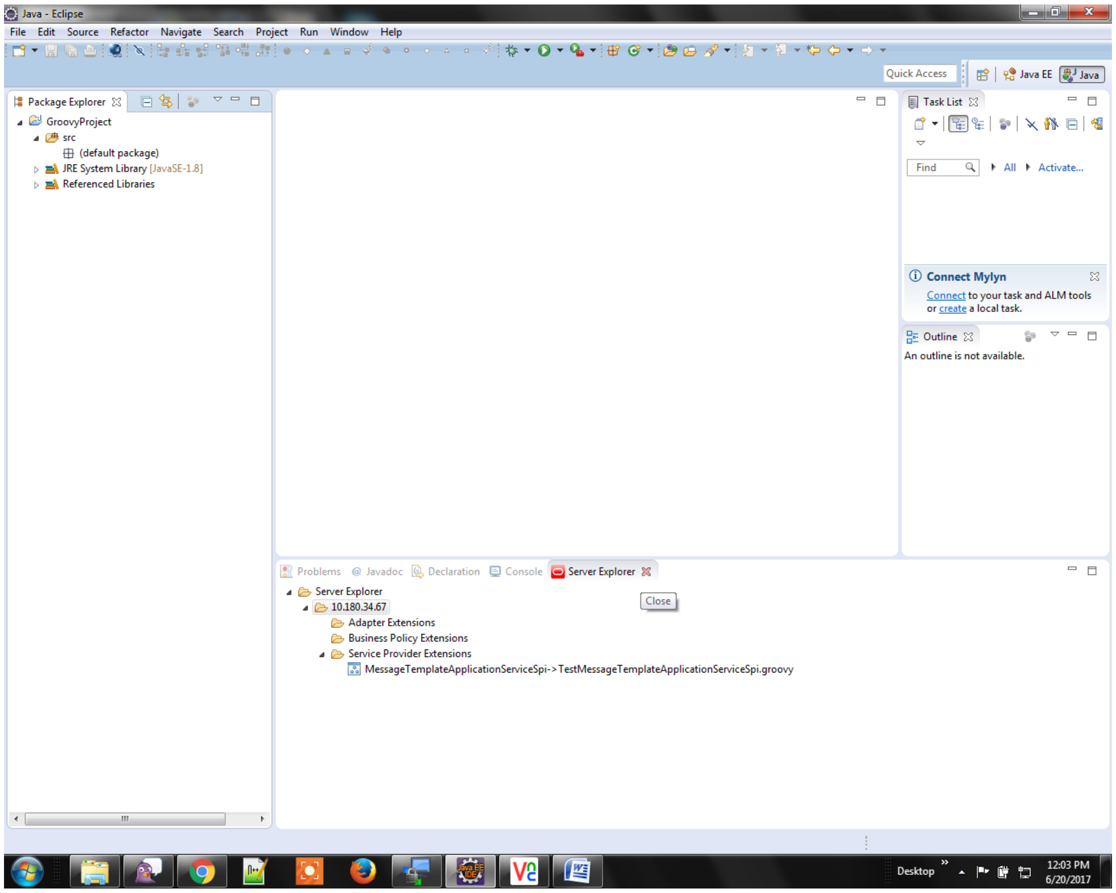
```
SELECT * FROM FLX_EXT_EXTENSION_B

SELECT * FROM FLX_EXT_EXTENSION_ITEMS

SELECT * FROM FLX_FW_CONFIG_ALL_B WHERE CATEGORY_ID = 'ServiceProviderExtensions'
```

Single Record View

PROP_ID	com.ofss.fc.appx.ep.service.dispatch.message.MessageTemplateApplicationServiceSpi
CATEGORY_ID	ServiceProviderExtensions
PROP_VALUE	TestMessageTemplateApplicationServiceSpi.groovy
FACTORY_SHIPPED_FLAG	Y
PROP_COMMENTS	TestMessageTemplateApplicationServiceSpi
SUMMARY_TEXT	TestMessageTemplateApplicationServiceSpi
CREATED_BY	OFSSUser
CREATION_DATE	
LAST_UPDATED_BY	OFSSUser
LAST_UPDATED_DATE	
OBJECT_STATUS_FLAG	A
OBJECT_VERSION_NUMBER	1

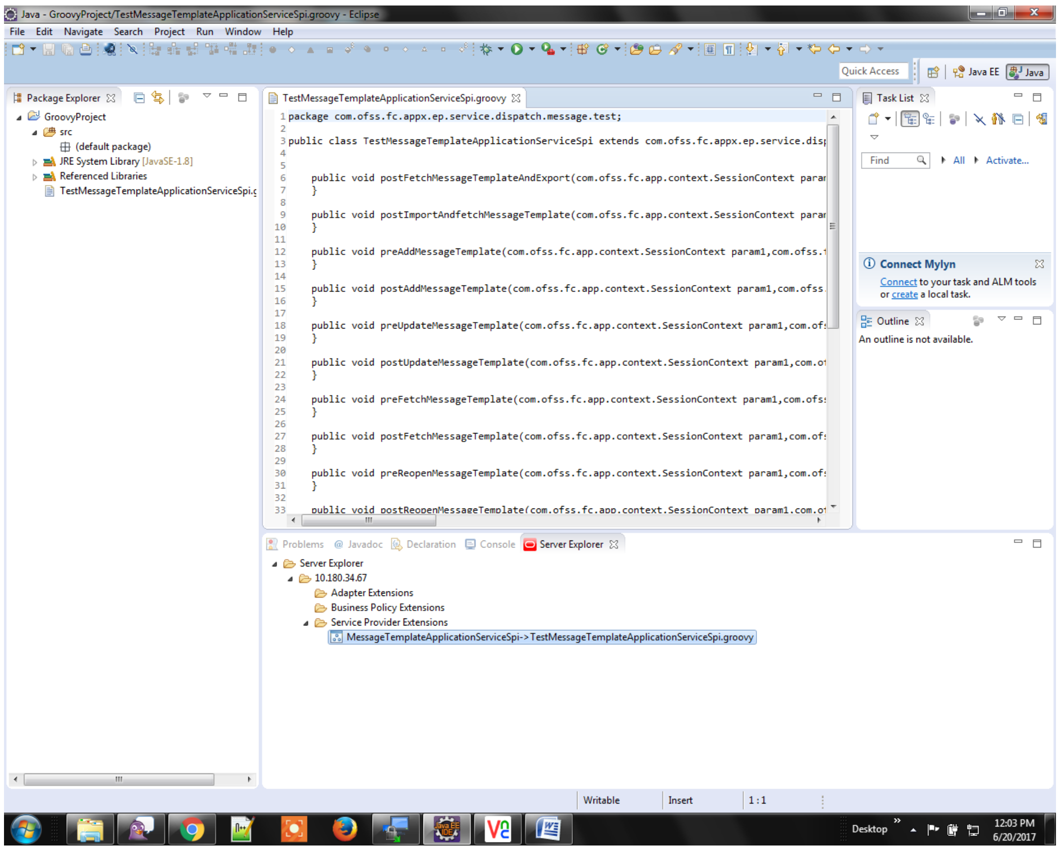
8.2.6 Fetching Deployed Application Service SPI Extension

Figure 8–28 Java Eclipse - Fetching Deployed Application Service SPI Extension


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
(default package)
JRE System Library [JavaSE-1.8]
Referenced Libraries
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Close
Server Explorer
10.180.34.67
Adapter Extensions
Business Policy Extensions
Service Provider Extensions
MessageTemplateApplicationServiceSpi->TestMessageTemplateApplicationServiceSpi.groovy
Desktop
12:03 PM
6/20/2017

1. For the purpose of fetching a deployed Extension from the Server, you can click on the appropriate Extension under the appropriate child under Server Explorer.

Figure 8–29 Click on Extension under Server Explorer



```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dis


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext parar
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext parar
    }

    public void preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.1
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
```

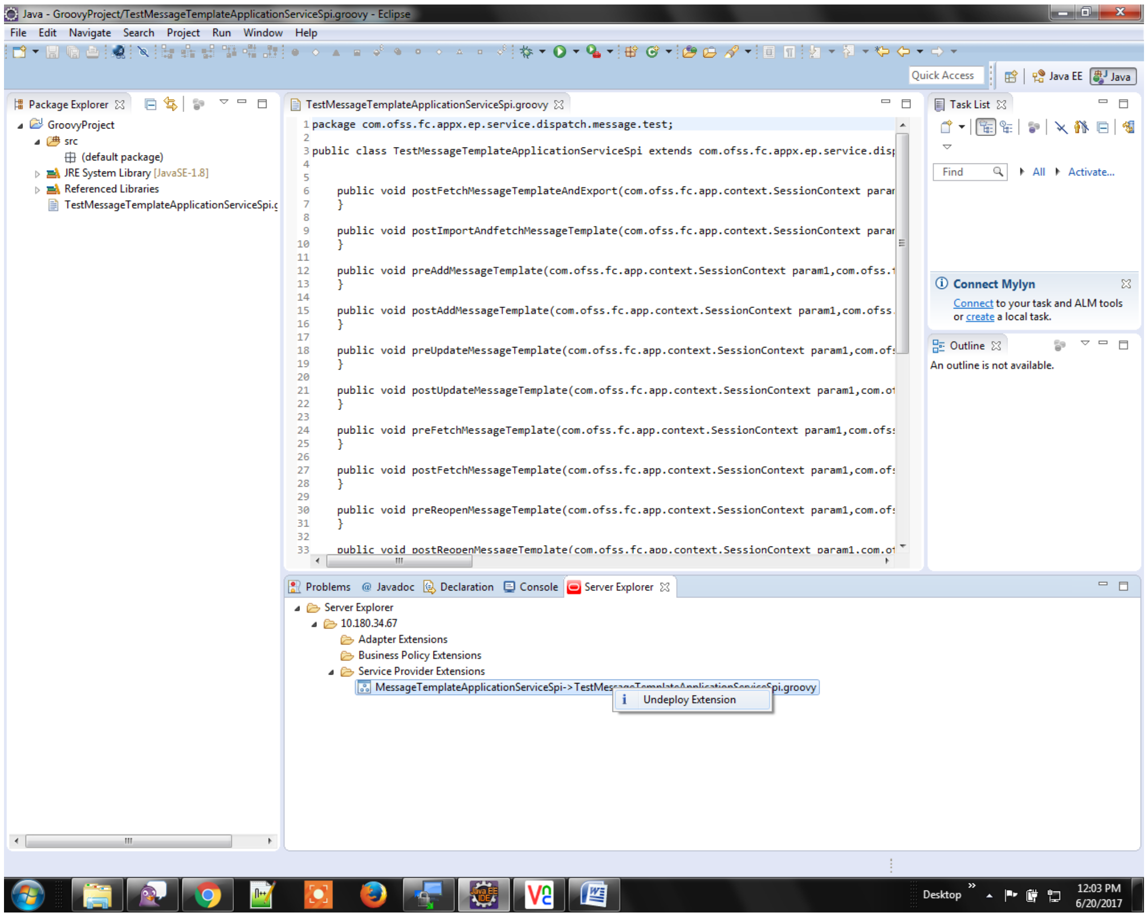
```
Server Explorer
  10.180.34.67
    Adapter Extensions
    Business Policy Extensions
    Service Provider Extensions
      MessageTemplateApplicationServiceSpi->TestMessageTemplateApplicationServiceSpi.groovy
```

2. The extension gets saved in the project created to contain the Extension classes generated through the plugin. Also the Extension code opens up in the Code Editor.

8.2.7 Undeploying Application Service SPI Extension

Perform the following steps for undeploying application service SPI extension:

Figure 8–30 Java Eclipse - Undeploying Application Service SPI Extension



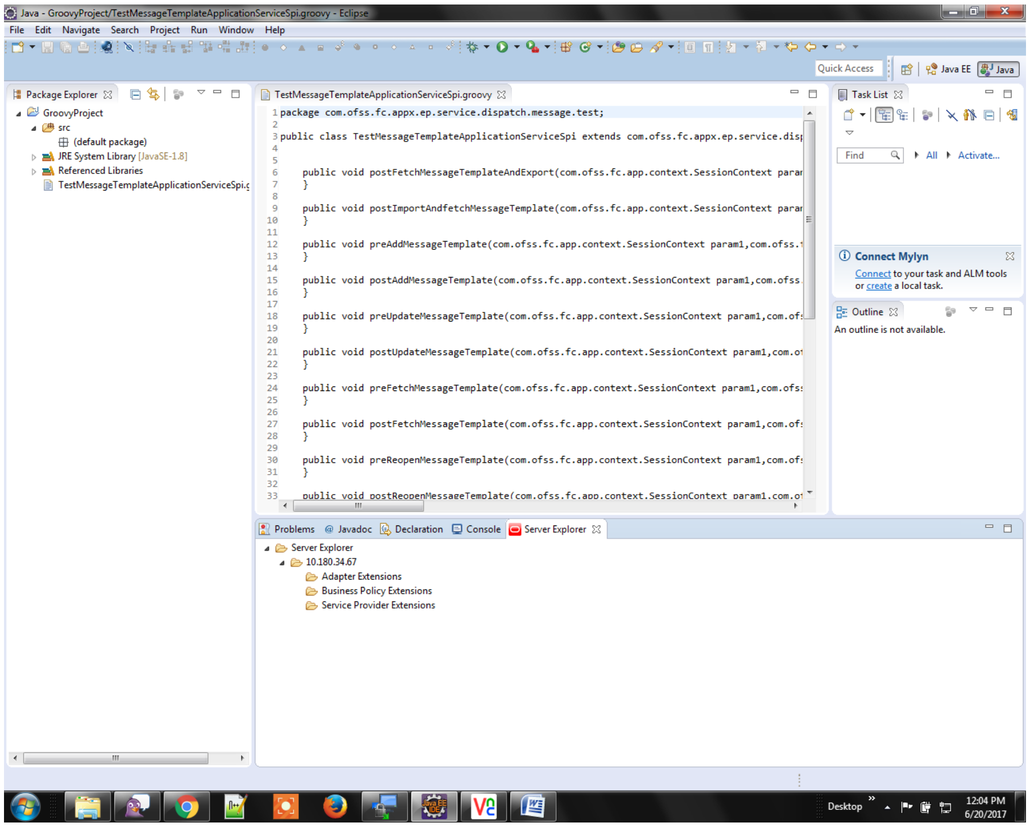
```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.disp


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext parar
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext parar
    }

    public void preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
```

```
Server Explorer
  10.180.34.67
    Adapter Extensions
    Business Policy Extensions
    Service Provider Extensions
      MessageTemplateApplicationServiceSpi->TestMessageTemplateApplicationServiceSpi.groovy
```

Java - GroovyProject/TestMessageTemplateApplicationServiceSpi.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Package Explorer
GroovyProject
src
(default package)
JRE System Library [JavaSE-1.8]
Referenced Libraries
TestMessageTemplateApplicationServiceSpi.g
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Undeploy Extension
Quick Access
Java EE
Java
Desktop
12:03 PM
6/20/2017

1. For the purpose of undeploying the Extension from the Server, you right click on the specific Extension under the appropriate child of the Server configured under Server Explorer.

Figure 8–31 Click on Extension under Server Explorer



```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.disp


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext param
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext param
    }

    public void preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void preReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postReopenMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
```

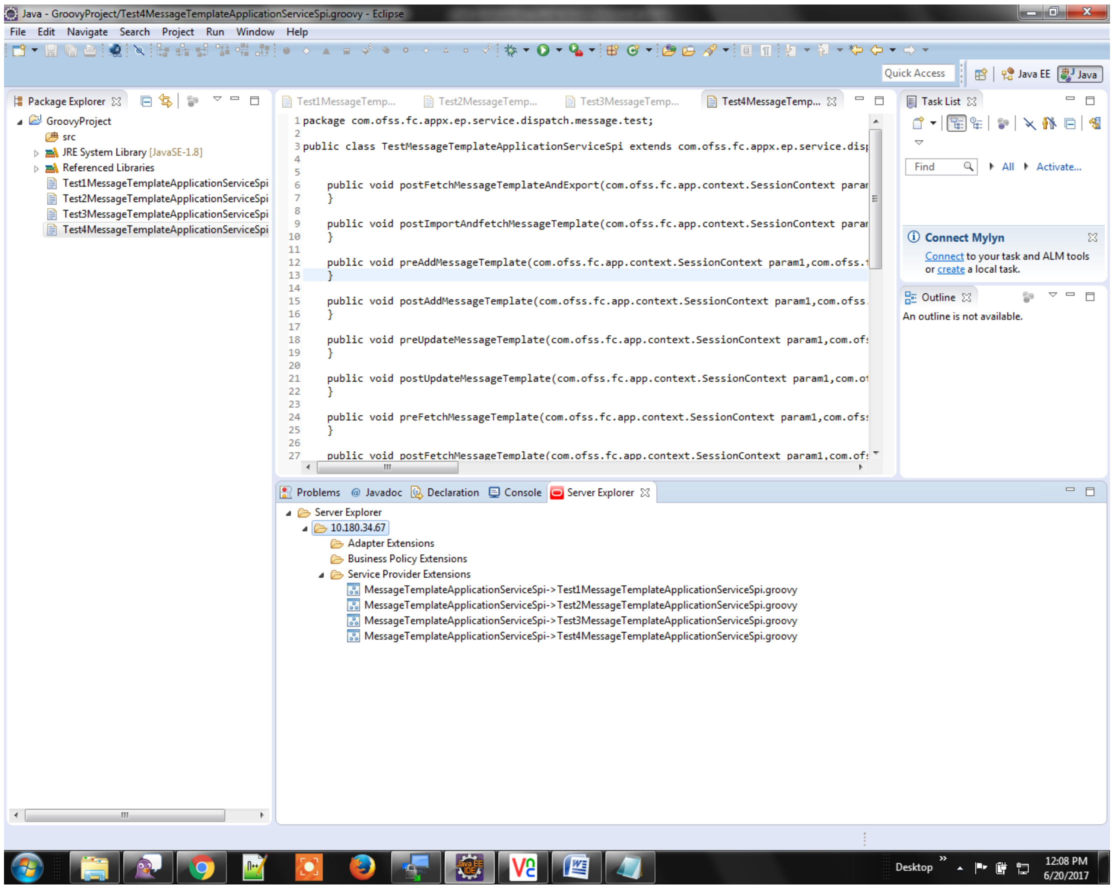
- Server Explorer
  - 10.180.34.67
    - Adapter Extensions
    - Business Policy Extensions
    - Service Provider Extensions

2. The specific Extension under the appropriate child of the Server configured under Server Explorer disappears and gets undeployed from the Server.

8.2.8 Case of Multiple Application Service SPI Extensions

1. You can choose to add multiple Groovy extensions for the same Application Service SPI. These will appear in the order in which they were added under Server Explorer>Server>Service Provider Extensions>.

Figure 8–32 Adding multiple Groovy extensions for the same Application Service SPI



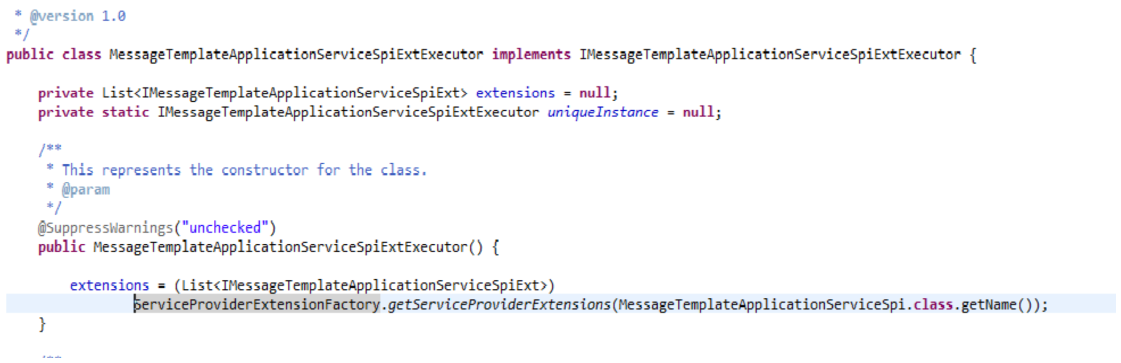
```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.disp


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext parar
    }

    public void postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext parar
    }

    public void preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.f
    }

    public void postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
    }

    public void preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of
    }

    public void preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
    }

    public void postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs
```

```
Server Explorer
  10.180.34.67
    Adapter Extensions
    Business Policy Extensions
    Service Provider Extensions
      MessageTemplateApplicationServiceSpi->Test1MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test2MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test3MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test4MessageTemplateApplicationServiceSpi.groovy
```

8.2.9 Inclusion of Groovy Extension in Actual Code Flow

1. The Application Service SPI Ext Executor derives a list of Application Service SPI Extensions through ServiceProviderExtensionFactory.getServiceProviderExtensions(<ApplicationServiceSpiName>).

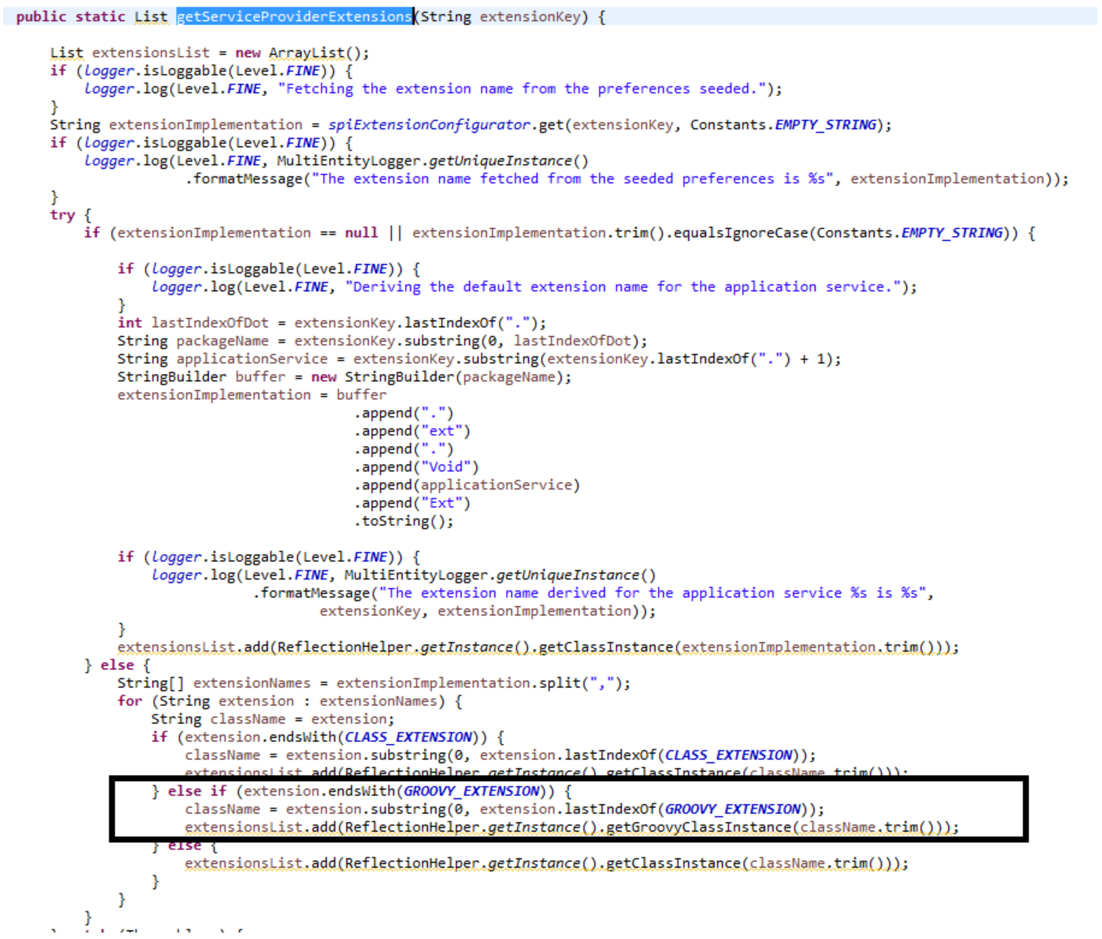
Figure 8–33 ServiceProviderExtensionFactory.getServiceProviderExtensions



```
 * @version 1.0
 */
public class MessageTemplateApplicationServiceSpiExtExecutor implements IMessageTemplateApplicationServiceSpiExtExecutor {

    private List<IMessageTemplateApplicationServiceSpiExt> extensions = null;
    private static IMessageTemplateApplicationServiceSpiExtExecutor uniqueInstance = null;

    /**
     * This represents the constructor for the class.
     * @param
     */
    @SuppressWarnings("unchecked")
    public MessageTemplateApplicationServiceSpiExtExecutor() {

        extensions = (List<IMessageTemplateApplicationServiceSpiExt>)
                ServiceProviderExtensionFactory.getServiceProviderExtensions(MessageTemplateApplicationServiceSpi.class.getName());
    }
```

Subsequently the Groovy Extensions are compiled and included in the Code Flow.

Figure 8–34 Groovy Extensions compiled and included in Code Flow



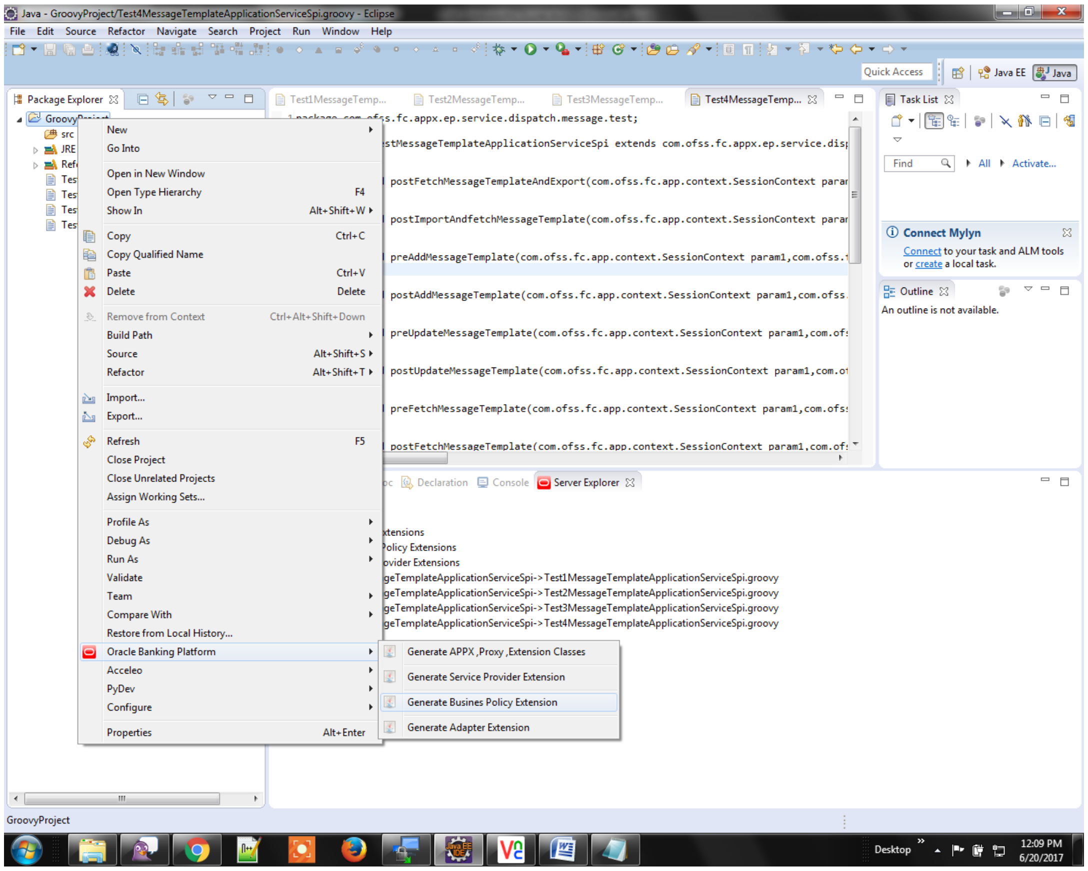
```
public static List getServiceProviderExtensions(String extensionKey) {

    List extensionsList = new ArrayList();
    if (logger.isLoggable(Level.FINE)) {
        logger.log(Level.FINE, "Fetching the extension name from the preferences seeded.");
    }
    String extensionImplementation = spiExtensionConfigurator.get(extensionKey, Constants.EMPTY_STRING);
    if (logger.isLoggable(Level.FINE)) {
        logger.log(Level.FINE, MultiEntityLogger.getUniqueInstance()
                .formatMessage("The extension name fetched from the seeded preferences is %s", extensionImplementation));
    }
    try {
        if (extensionImplementation == null || extensionImplementation.trim().equalsIgnoreCase(Constants.EMPTY_STRING)) {

            if (logger.isLoggable(Level.FINE)) {
                logger.log(Level.FINE, "Deriving the default extension name for the application service.");
            }
            int lastIndexOfDot = extensionKey.lastIndexOf(".");
            String packageName = extensionKey.substring(0, lastIndexOfDot);
            String applicationService = extensionKey.substring(extensionKey.lastIndexOf(".") + 1);
            StringBuilder buffer = new StringBuilder(packageName);
            extensionImplementation = buffer
                                    .append(".")
                                    .append("ext")
                                    .append(".")
                                    .append("Void")
                                    .append(applicationService)
                                    .append("Ext")
                                    .toString();

            if (logger.isLoggable(Level.FINE)) {
                logger.log(Level.FINE, MultiEntityLogger.getUniqueInstance()
                        .formatMessage("The extension name derived for the application service %s is %s",
                                extensionKey, extensionImplementation));
            }
            extensionsList.add(ReflectionHelper.getInstance().getClassInstance(extensionImplementation.trim()));
        } else {
            String[] extensionNames = extensionImplementation.split(",");
            for (String extension : extensionNames) {
                String className = extension;
                if (extension.endsWith(CLASS_EXTENSION)) {
                    className = extension.substring(0, extension.lastIndexOf(CLASS_EXTENSION));
                    extensionsList.add(ReflectionHelper.getInstance().getClassInstance(className.trim()));
                } else if (extension.endsWith(GROOVY_EXTENSION)) {
                    className = extension.substring(0, extension.lastIndexOf(GROOVY_EXTENSION));
                    extensionsList.add(ReflectionHelper.getInstance().getGroovyClassInstance(className.trim()));
                } else {
                    extensionsList.add(ReflectionHelper.getInstance().getClassInstance(className.trim()));
                }
            }
        }
```

8.3 Support for Business Policy Extension

8.3.1 Generate Business Policy Extension

1. The parameter Temporary Project has been configured to point to the base project where the Groovy Extension Files have to be saved.
2. Right click on this project and select Oracle Banking Platform > Generate Business Policy Extension.

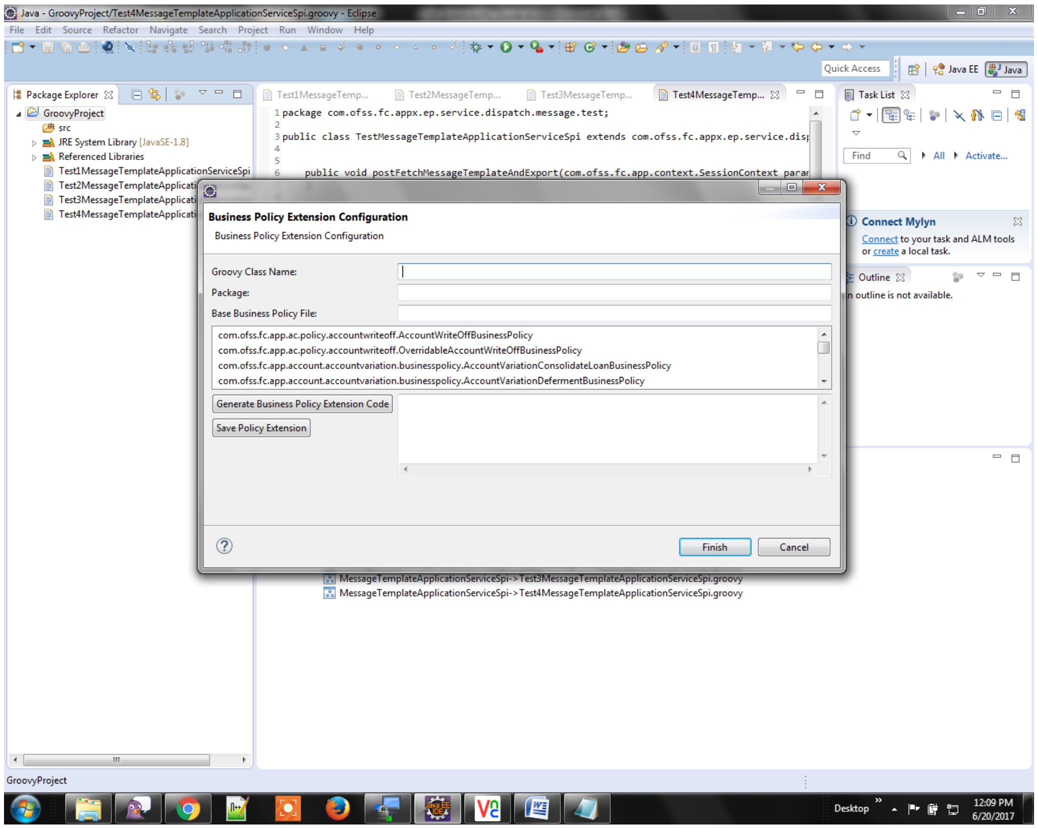
Figure 8–35 Generate Business Policy Extension



```
package com.ofss.fc.appx.ep.service.dispatch.message.test;
stMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.disp
postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext parar
postImportAndfetchMessageTemplate(com.ofss.fc.app.context.SessionContext parar
preAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss.
postAddMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofss
preUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of:
postUpdateMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.o
preFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.ofs:
postFetchMessageTemplate(com.ofss.fc.app.context.SessionContext param1,com.of:
```

```
xtensions
olicy Extensions
ovider Extensions
geTemplateApplicationServiceSpi->Test1MessageTemplateApplicationServiceSpi.groovy
geTemplateApplicationServiceSpi->Test2MessageTemplateApplicationServiceSpi.groovy
geTemplateApplicationServiceSpi->Test3MessageTemplateApplicationServiceSpi.groovy
geTemplateApplicationServiceSpi->Test4MessageTemplateApplicationServiceSpi.groovy
```

Figure 8–36 Business Policy Extension Configuration


Java - GroovyProject/Test4MessageTemplateApplicationServiceSpi.groovy - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicati
Test3MessageTemplateApplicati
Test4MessageTemplateApplicati
Test1MessageTemp...
Test2MessageTemp...
Test3MessageTemp...
Test4MessageTemp...
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
n outline is not available.
Generate Business Policy Extension Code
Save Policy Extension
Finish
Cancel
GroovyProject
Desktop
12:09 PM
6/20/2017

```
package com.ofss.fc.appx.ep.service.dispatch.message.test;

public class TestMessageTemplateApplicationServiceSpi extends com.ofss.fc.appx.ep.service.dis


    public void postFetchMessageTemplateAndExport(com.ofss.fc.app.context.SessionContext parar
```

Business Policy Extension Configuration

Business Policy Extension Configuration

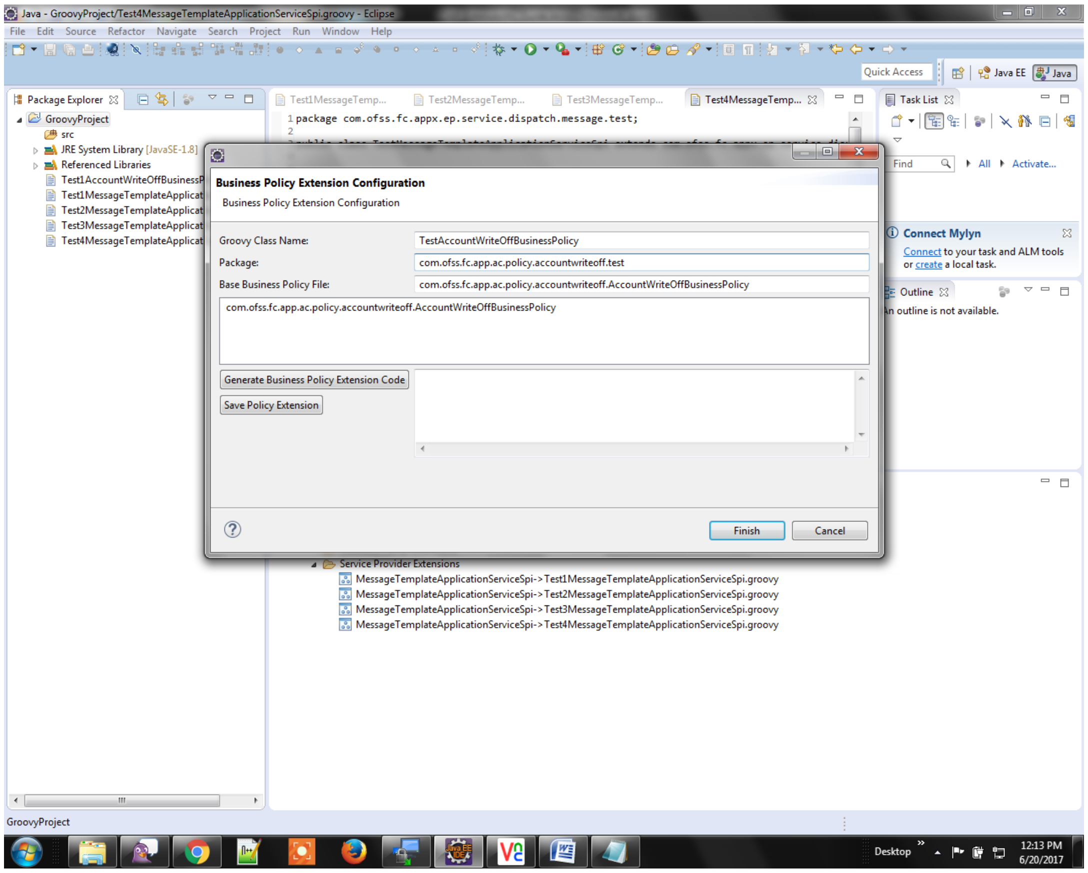
Groovy Class Name:	
Package:	
Base Business Policy File:	

com.ofss.fc.app.ac.policy.accountwriteoff.AccountWriteOffBusinessPolicy
com.ofss.fc.app.ac.policy.accountwriteoff.OverridableAccountWriteOffBusinessPolicy
com.ofss.fc.app.account.accountvariation.businesspolicy.AccountVariationConsolidateLoanBusinessPolicy
com.ofss.fc.app.account.accountvariation.businesspolicy.AccountVariationDefermentBusinessPolicy

MessageTemplateApplicationServiceSpi->Test3MessageTemplateApplicationServiceSpi.groovy
MessageTemplateApplicationServiceSpi->Test4MessageTemplateApplicationServiceSpi.groovy

3. The above wizard appears with a list of Business Policy Files.

Figure 8–37 Select Base Business Policy file


Java - GroovyProject/Test4MessageTemplateApplicationServiceSpi.groovy - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessP
Test1MessageTemplateApplicat
Test2MessageTemplateApplicat
Test3MessageTemplateApplicat
Test4MessageTemplateApplicat
Test1MessageTemp...
Test2MessageTemp...
Test3MessageTemp...
Test4MessageTemp...
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Generate Business Policy Extension Code
Save Policy Extension
Finish
Cancel
GroovyProject
Desktop
12:13 PM
6/20/2017

```
package com.ofss.fc.appx.ep.service.dispatch.message.test;
```

Business Policy Extension Configuration

Business Policy Extension Configuration

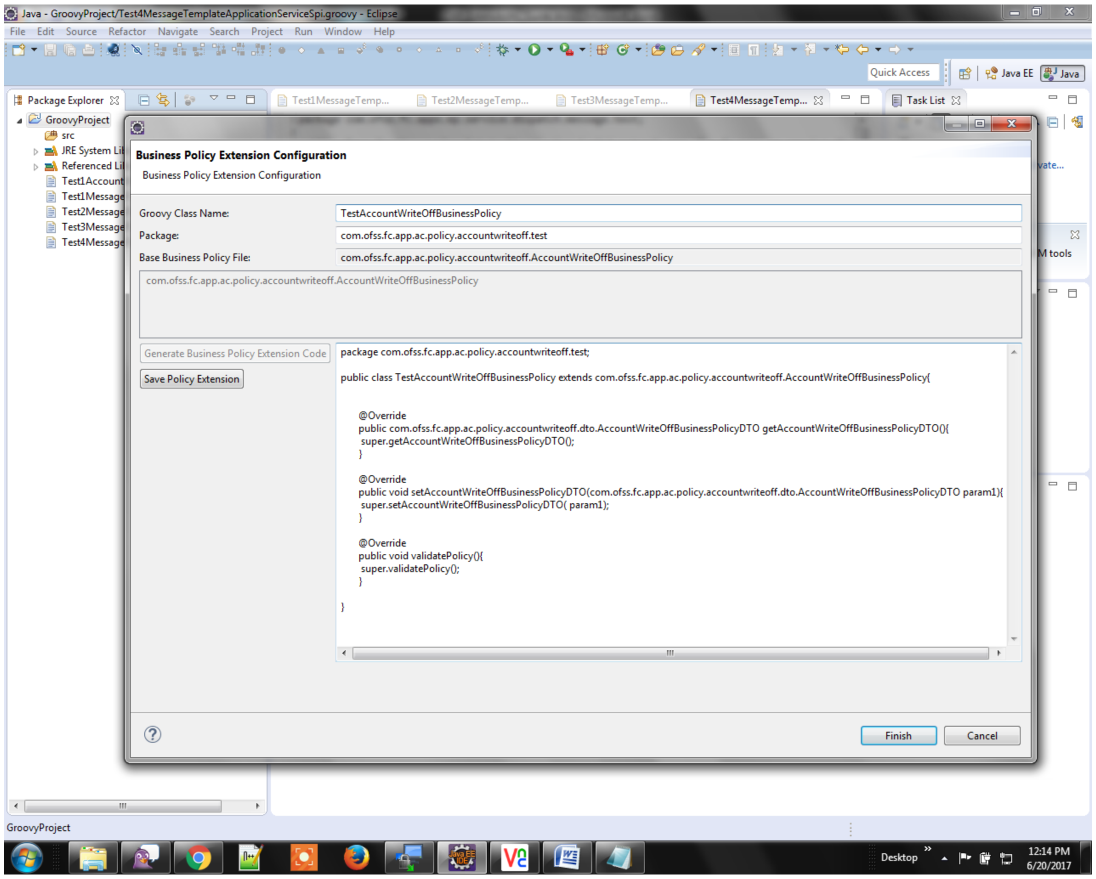
Groovy Class Name:	TestAccountWriteOffBusinessPolicy
Package:	com.ofss.fc.app.ac.policy.accountwriteoff.test
Base Business Policy File:	com.ofss.fc.app.ac.policy.accountwriteoff.AccountWriteOffBusinessPolicy

com.ofss.fc.app.ac.policy.accountwriteoff.AccountWriteOffBusinessPolicy

Service Provider Extensions
- MessageTemplateApplicationServiceSpi-> Test1MessageTemplateApplicationServiceSpi.groovy
- MessageTemplateApplicationServiceSpi-> Test2MessageTemplateApplicationServiceSpi.groovy
- MessageTemplateApplicationServiceSpi-> Test3MessageTemplateApplicationServiceSpi.groovy
- MessageTemplateApplicationServiceSpi-> Test4MessageTemplateApplicationServiceSpi.groovy

4. Select the filtered Base Business Policy file from the list such that it appears as the input for Base Business Policy file.
5. Appropriately set values for the Extension Class Name and package.

Figure 8–38 Enter Extension Class Name and Package



Business Policy Extension Configuration

Business Policy Extension Configuration

Groovy Class Name: TestAccountWriteOffBusinessPolicy

Package: com.ofss.fc.app.ac.policy.accountwriteoff.test

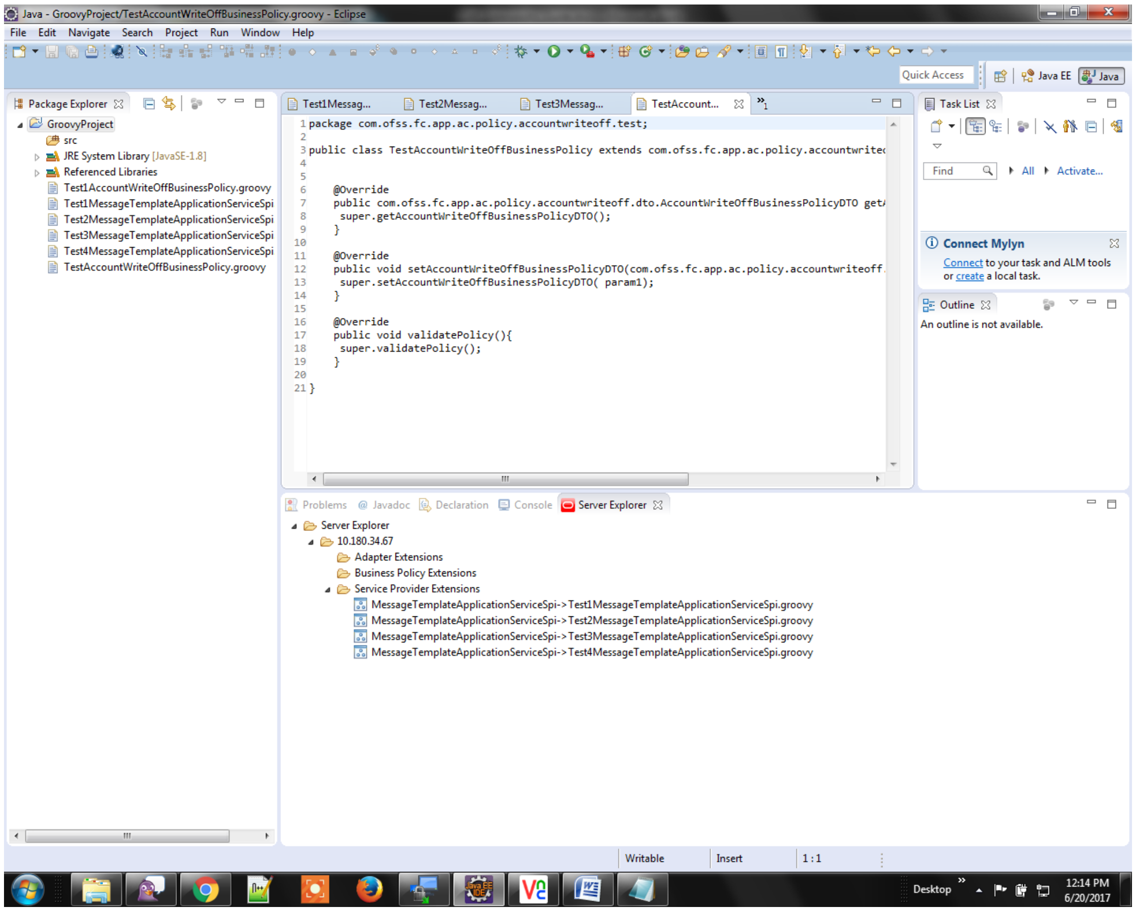
Base Business Policy File: com.ofss.fc.app.ac.policy.accountwriteoff.AccountWriteOffBusinessPolicy

com.ofss.fc.app.ac.policy.accountwriteoff.AccountWriteOffBusinessPolicy

```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;

public class TestAccountWriteOffBusinessPolicy extends com.ofss.fc.app.ac.policy.accountwriteoff.AccountWriteOffBusinessPolicy{


    @Override
    public com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO getAccountWriteOffBusinessPolicyDTO(){
     super.getAccountWriteOffBusinessPolicyDTO();
    }

    @Override
    public void setAccountWriteOffBusinessPolicyDTO(com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO param1){
     super.setAccountWriteOffBusinessPolicyDTO( param1);
    }

    @Override
    public void validatePolicy(){
     super.validatePolicy();
    }

}
```

6. Click on Generate Busness Policy Extension Code.
7. The code gets generated with the base business policy methods.
8. Click on Save Policy Extension and Finish.

Figure 8–39 Click Save Policy Extension and Finish



```
GroovyProject
    src
    JRE System Library [JavaSE-1.8]
    Referenced Libraries
    Test1AccountWriteOffBusinessPolicy.groovy
    Test1MessageTemplateApplicationServiceSpi
    Test2MessageTemplateApplicationServiceSpi
    Test3MessageTemplateApplicationServiceSpi
    Test4MessageTemplateApplicationServiceSpi
    TestAccountWriteOffBusinessPolicy.groovy
```

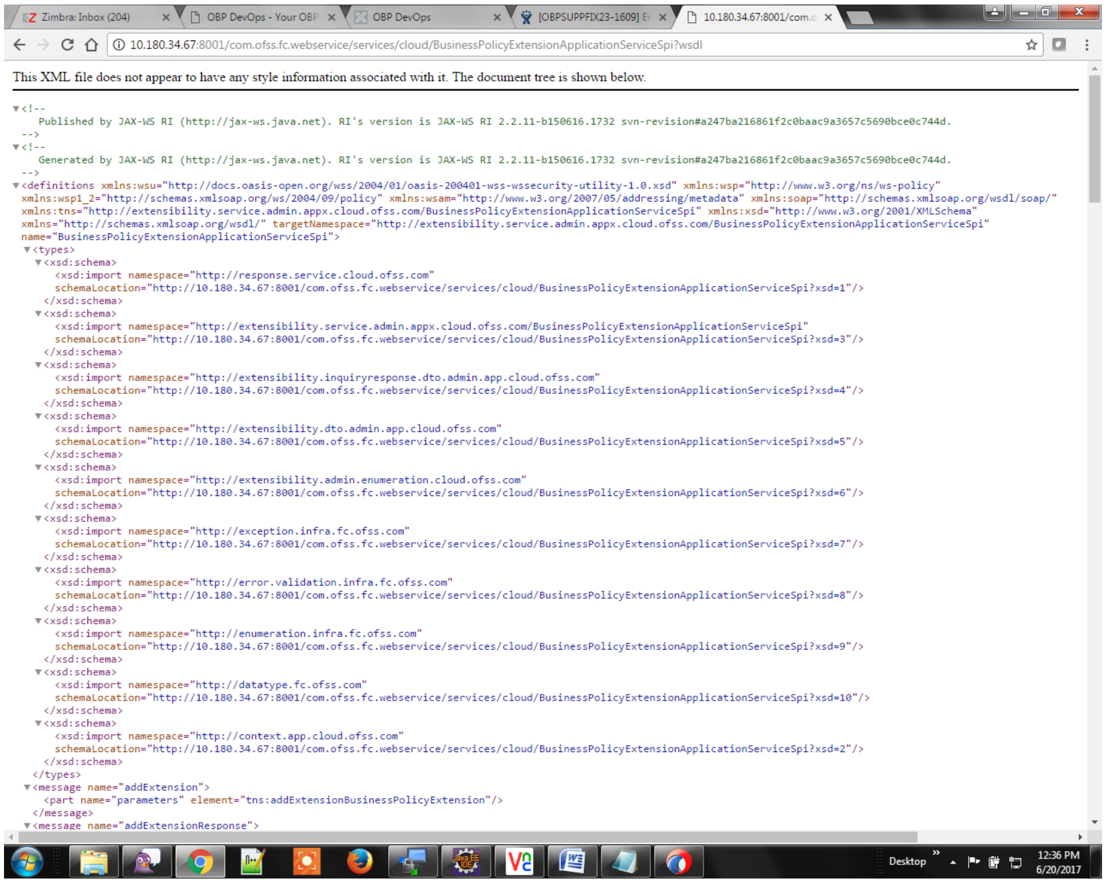
```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;

public class TestAccountWriteOffBusinessPolicy extends com.ofss.fc.app.ac.policy.accountwrite


    @Override
    public com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO getA
     super.getAccountWriteOffBusinessPolicyDTO();
    }

    @Override
    public void setAccountWriteOffBusinessPolicyDTO(com.ofss.fc.app.ac.policy.accountwriteoff.
     super.setAccountWriteOffBusinessPolicyDTO( param1);
    }

    @Override
    public void validatePolicy(){
     super.validatePolicy();
    }

}
```

```
Server Explorer
    10.180.34.67
        Adapter Extensions
        Business Policy Extensions
        Service Provider Extensions
            MessageTemplateApplicationServiceSpi->Test1MessageTemplateApplicationServiceSpi.groovy
            MessageTemplateApplicationServiceSpi->Test2MessageTemplateApplicationServiceSpi.groovy
            MessageTemplateApplicationServiceSpi->Test3MessageTemplateApplicationServiceSpi.groovy
            MessageTemplateApplicationServiceSpi->Test4MessageTemplateApplicationServiceSpi.groovy
```

9. The extension gets saved in the project created to contain the Extension classes generated through the plugin. You can add all the code required in the base methods for extensibility of the base Business Policy.

8.3.2 Exposed Webservice for Business Policy Extensions

Figure 8–40 Business Policy Extension Application ServiceSpi



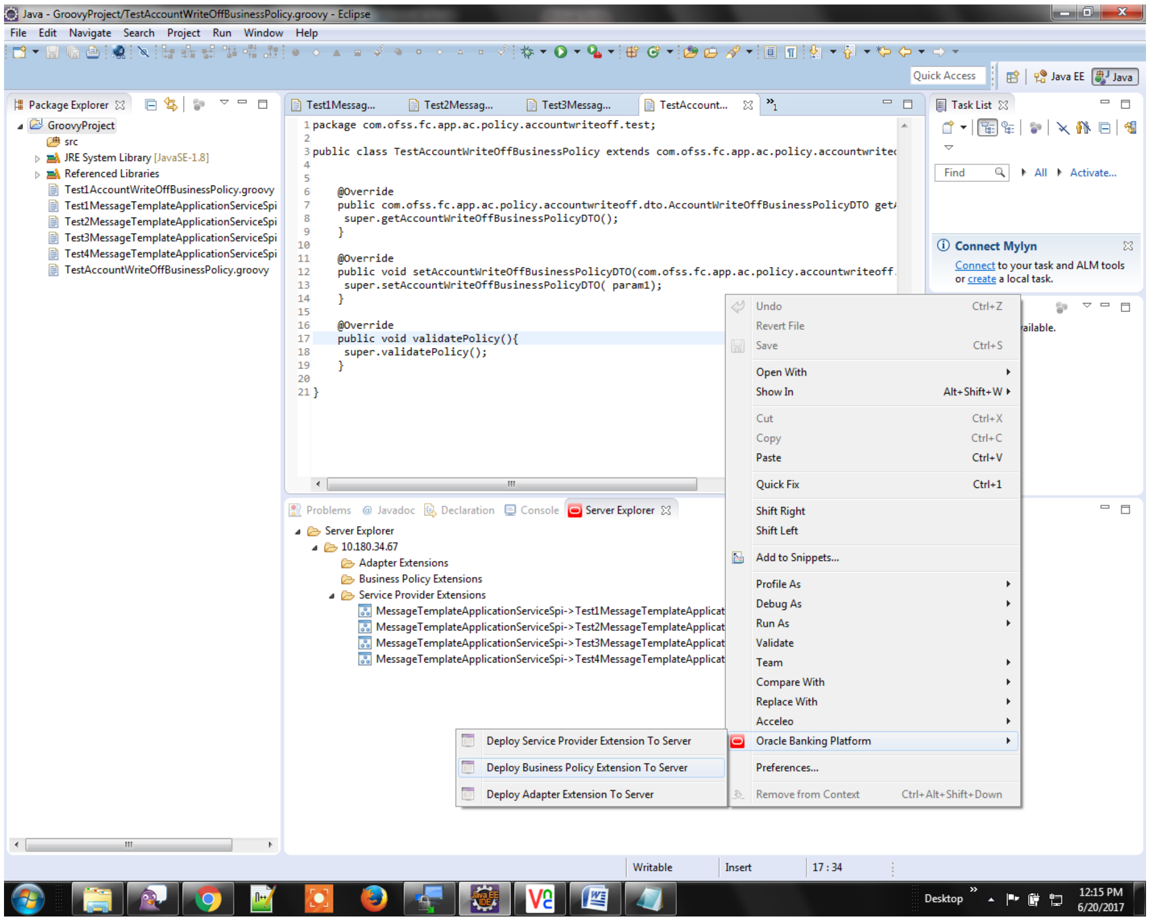
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<!--
    Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.11-b150616.1732 svn-revision#a247ba216861f2c0baac9a3657c5690bce0c744d.
 -->
▼<!--
    Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.11-b150616.1732 svn-revision#a247ba216861f2c0baac9a3657c5690bce0c744d.
 -->
▼<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
 xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tns="http://extensibility.service.admin.appx.cloud.ofss.com/BusinessPolicyExtensionApplicationServiceSpi" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://extensibility.service.admin.appx.cloud.ofss.com/BusinessPolicyExtensionApplicationServiceSpi"
 name="BusinessPolicyExtensionApplicationServiceSpi">
  ▼<types>
    ▼<xsd:schema>
       <xsd:import namespace="http://response.service.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/BusinessPolicyExtensionApplicationServiceSpi?xsd=1"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.service.admin.appx.cloud.ofss.com/BusinessPolicyExtensionApplicationServiceSpi"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/BusinessPolicyExtensionApplicationServiceSpi?xsd=3"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.inquiryresponse.dto.admin.app.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/BusinessPolicyExtensionApplicationServiceSpi?xsd=4"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.dto.admin.app.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/BusinessPolicyExtensionApplicationServiceSpi?xsd=5"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.admin.enumeration.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/BusinessPolicyExtensionApplicationServiceSpi?xsd=6"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://exception.infra.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/BusinessPolicyExtensionApplicationServiceSpi?xsd=7"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://error.validation.infra.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/BusinessPolicyExtensionApplicationServiceSpi?xsd=8"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://enumeration.infra.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/BusinessPolicyExtensionApplicationServiceSpi?xsd=9"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://datatype.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/BusinessPolicyExtensionApplicationServiceSpi?xsd=10"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://context.app.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/BusinessPolicyExtensionApplicationServiceSpi?xsd=2"/>
     </xsd:schema>
   </types>
  ▼<message name="addExtension">
     <part name="parameters" element="tns:addExtensionBusinessPolicyExtension"/>
   </message>
  ▼<message name="addExtensionResponse">
```

Business Policy Extension Application ServiceSpi is the web service that exposes the web services to add business policy extensions, fetch business policy extensions and delete business policy extensions.These services are used by the eclipse plugin to deploy extensions, fetch extensions and undeploy extensions at runtime.

8.3.3 Deploy Business Policy Extension to Server

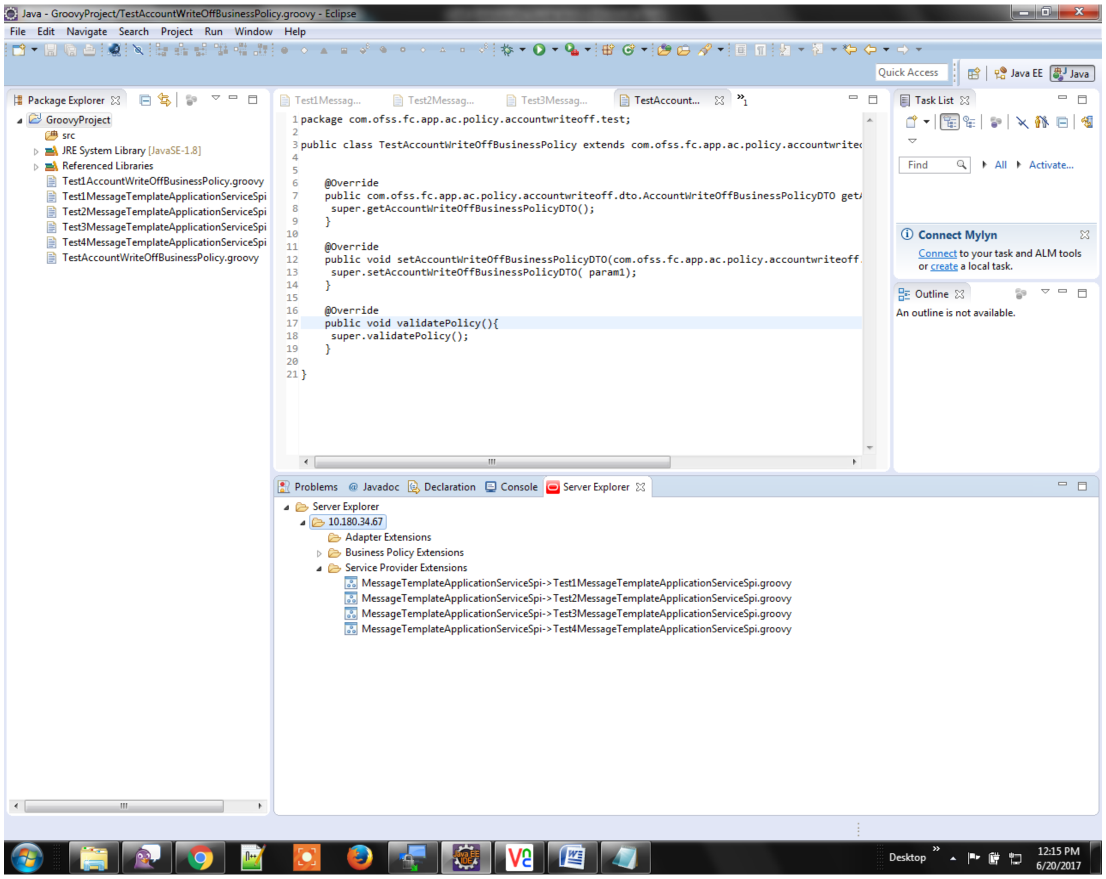
Figure 8–41 Click Deploy Business Policy Extension To Server


Java - GroovyProject/TestAccountWriteOffBusinessPolicy.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
Test1Messag...
Test2Messag...
Test3Messag...
TestAccount...
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Problems
Javadoc
Declaration
Console
Server Explorer
Server Explorer
10.180.34.67
Adapter Extensions
Business Policy Extensions
Service Provider Extensions
MessageTemplateApplicationServiceSpi->Test1MessageTemplateApplicat
MessageTemplateApplicationServiceSpi->Test2MessageTemplateApplicat
MessageTemplateApplicationServiceSpi->Test3MessageTemplateApplicat
MessageTemplateApplicationServiceSpi->Test4MessageTemplateApplicat
Undo Ctrl+Z
Revert File
Save Ctrl+S
Open With
Show In Alt+Shift+W
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Quick Fix Ctrl+1
Shift Right
Shift Left
Add to Snippets...
Profile As
Debug As
Run As
Validate
Team
Compare With
Replace With
Acceleo
Oracle Banking Platform
Preferences...
Remove from Context Ctrl+Alt+Shift+Down
Deploy Service Provider Extension To Server
Deploy Business Policy Extension To Server
Deploy Adapter Extension To Server
Writable
Insert
17 : 34
Desktop
12:15 PM
6/20/2017

```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;

public class TestAccountWriteOffBusinessPolicy extends com.ofss.fc.app.ac.policy.accountwritec


    @Override
    public com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO getA
     super.getAccountWriteOffBusinessPolicyDTO();
    }

    @Override
    public void setAccountWriteOffBusinessPolicyDTO(com.ofss.fc.app.ac.policy.accountwriteoff.
     super.setAccountWriteOffBusinessPolicyDTO( param1);
    }

    @Override
    public void validatePolicy(){
     super.validatePolicy();
    }

}
```

1. Right Click on the Extension Class in the Package Explorer or the Code Editor and click on Oracle Banking Platform > Deploy Business Policy Extension To Server.
2. It is compulsory to have selected the Server under Server Explorer in which the deployment has to occur.

Figure 8–42 Select Server



Java - GroovyProject/TestAccountWriteOffBusinessPolicy.groovy - Eclipse

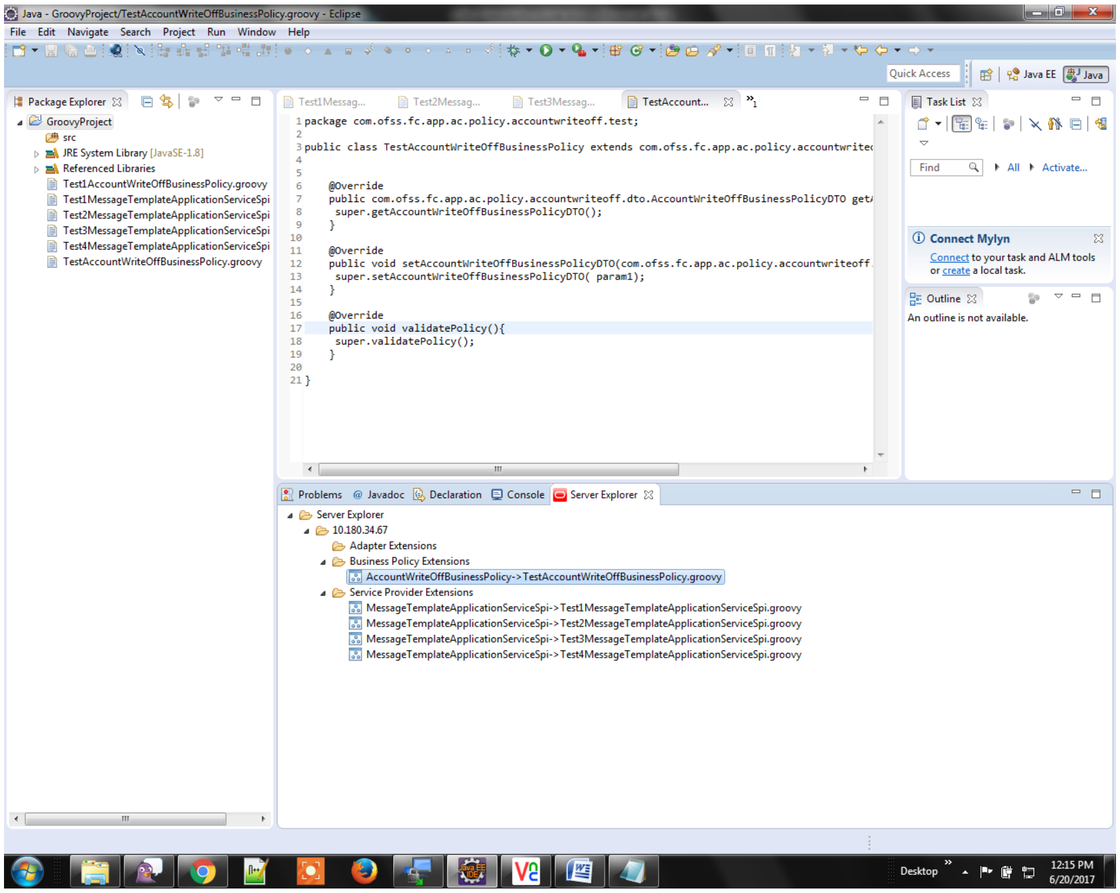
```
GroovyProject
    src
    JRE System Library [JavaSE-1.8]
    Referenced Libraries
    Test1AccountWriteOffBusinessPolicy.groovy
    Test1MessageTemplateApplicationServiceSpi
    Test2MessageTemplateApplicationServiceSpi
    Test3MessageTemplateApplicationServiceSpi
    Test4MessageTemplateApplicationServiceSpi
    TestAccountWriteOffBusinessPolicy.groovy
```

```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;

public class TestAccountWriteOffBusinessPolicy extends com.ofss.fc.app.ac.policy.accountwritec


    @Override
    public com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO getA
     super.getAccountWriteOffBusinessPolicyDTO();
    }

    @Override
    public void setAccountWriteOffBusinessPolicyDTO(com.ofss.fc.app.ac.policy.accountwriteoff.
     super.setAccountWriteOffBusinessPolicyDTO( param1);
    }

    @Override
    public void validatePolicy(){
     super.validatePolicy();
    }

}
```

```
Server Explorer
    10.180.34.67
        Adapter Extensions
        Business Policy Extensions
        Service Provider Extensions
            MessageTemplateApplicationServiceSpi->Test1MessageTemplateApplicationServiceSpi.groovy
            MessageTemplateApplicationServiceSpi->Test2MessageTemplateApplicationServiceSpi.groovy
            MessageTemplateApplicationServiceSpi->Test3MessageTemplateApplicationServiceSpi.groovy
            MessageTemplateApplicationServiceSpi->Test4MessageTemplateApplicationServiceSpi.groovy
```

3. The extension gets deployed in the server.

Figure 8–43 Extension Deployed on Server



Java - GroovyProject/TestAccountWriteOffBusinessPolicy.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
Test1Messag...
Test2Messag...
Test3Messag...
TestAccount...
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Desktop
12:15 PM
6/20/2017

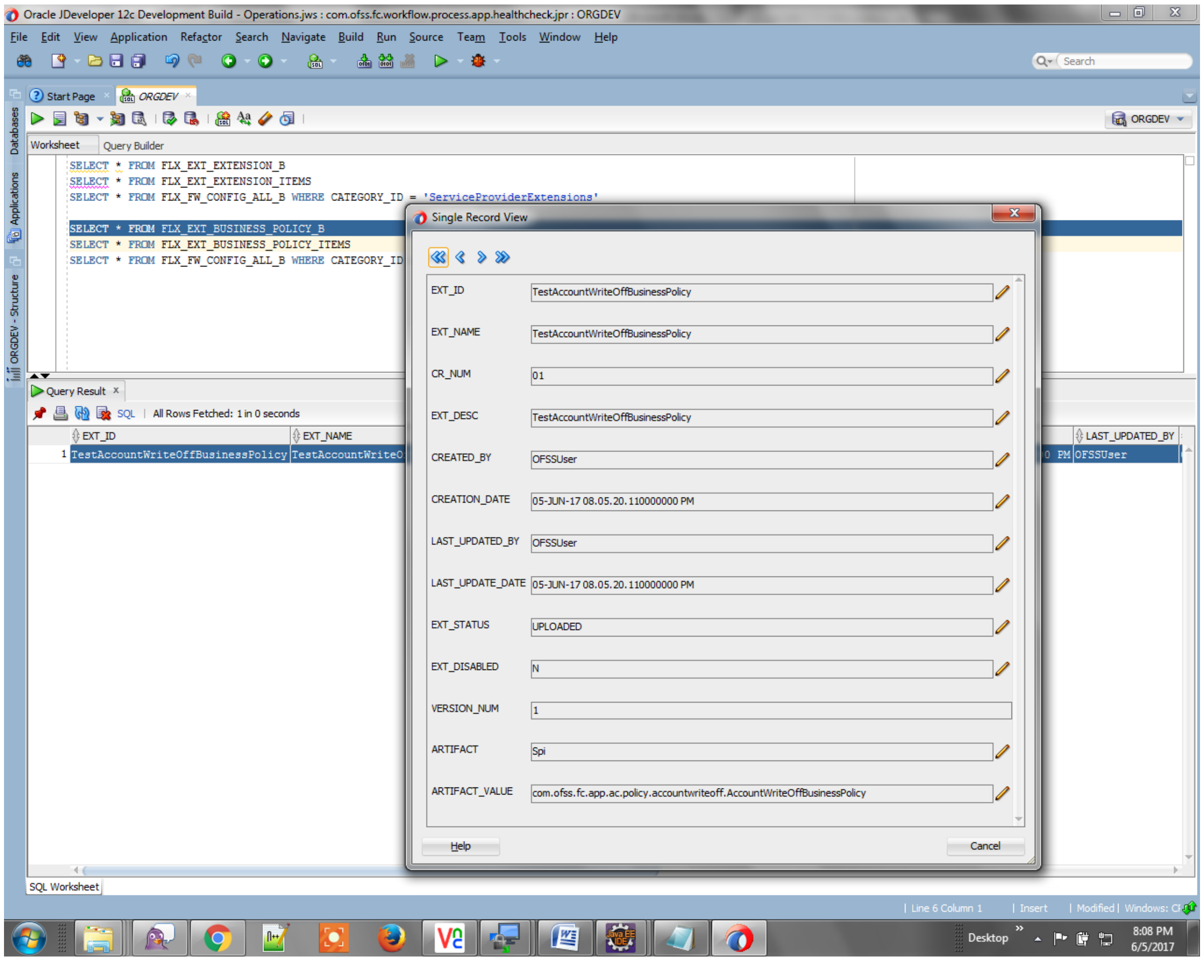
```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;

public class TestAccountWriteOffBusinessPolicy extends com.ofss.fc.app.ac.policy.accountwritec


    @Override
    public com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO get/
     super.getAccountWriteOffBusinessPolicyDTO();
    }

    @Override
    public void setAccountWriteOffBusinessPolicyDTO(com.ofss.fc.app.ac.policy.accountwriteoff.
     super.setAccountWriteOffBusinessPolicyDTO( param1);
    }

    @Override
    public void validatePolicy(){
     super.validatePolicy();
    }

}
```

```
Server Explorer
  10.180.34.67
    Adapter Extensions
    Business Policy Extensions
      AccountWriteOffBusinessPolicy->TestAccountWriteOffBusinessPolicy.groovy
    Service Provider Extensions
      MessageTemplateApplicationServiceSpi->Test1MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test2MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test3MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test4MessageTemplateApplicationServiceSpi.groovy
```

4. The Extension gets deployed in the server and appears under the appropriate child of Server configured under Server Explorer.

8.3.4 Database Inserts: Business Policy Extension Deployment

Figure 8–44 Business Policy Extension Deployment - Single Record View



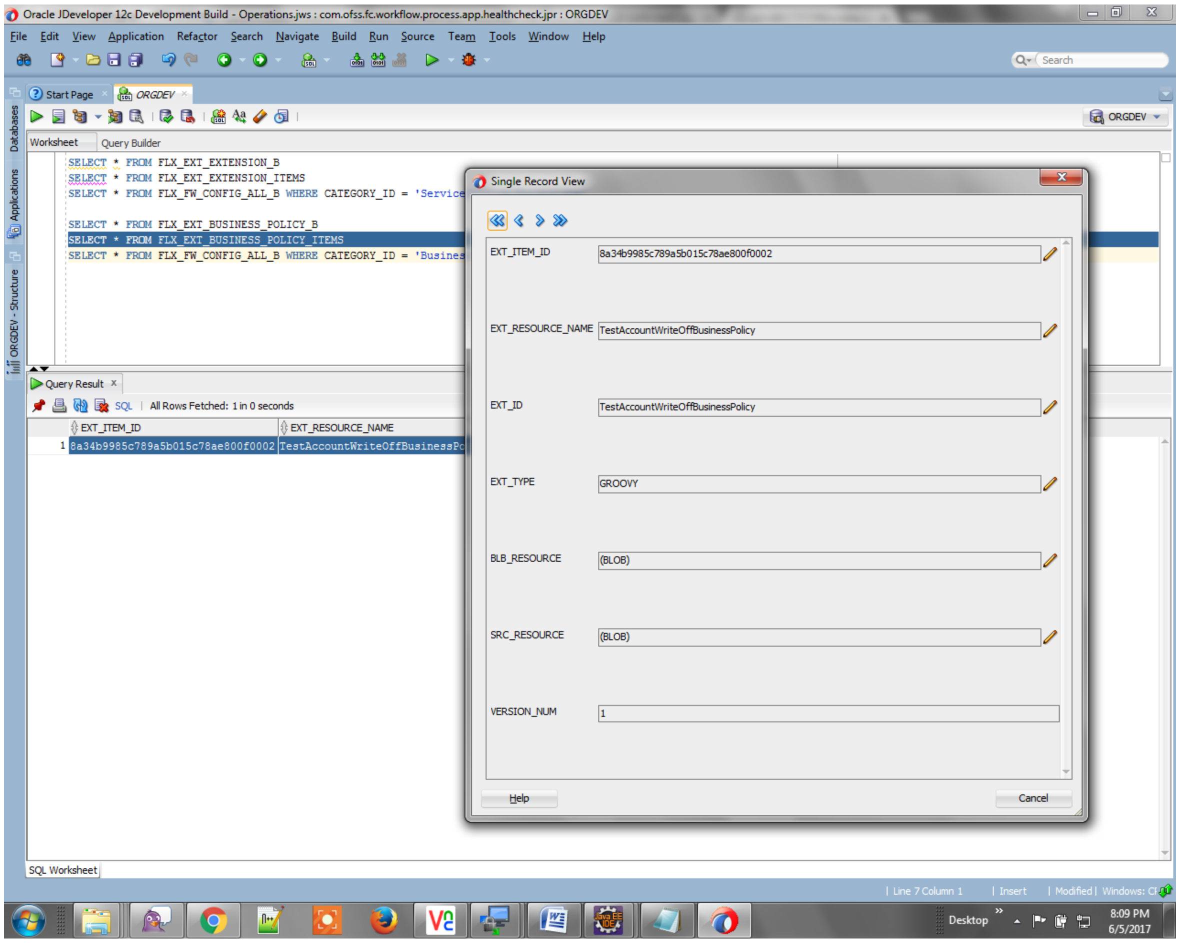
```
SELECT * FROM FLX_EXT_EXTENSION_B
SELECT * FROM FLX_EXT_EXTENSION_ITEMS
SELECT * FROM FLX_FW_CONFIG_ALL_B WHERE CATEGORY_ID = 'ServiceProviderExtensions'

SELECT * FROM FLX_EXT_BUSINESS_POLICY_B
SELECT * FROM FLX_EXT_BUSINESS_POLICY_ITEMS
SELECT * FROM FLX_FW_CONFIG_ALL_B WHERE CATEGORY_ID
```

All Rows Fetched: 1 in 0 seconds

Single Record View

EXT_ID	TestAccountWriteOffBusinessPolicy
EXT_NAME	TestAccountWriteOffBusinessPolicy
CR_NUM	01
EXT_DESC	TestAccountWriteOffBusinessPolicy
CREATED_BY	OFSSUser
CREATION_DATE	05-JUN-17 08.05.20.110000000 PM
LAST_UPDATED_BY	OFSSUser
LAST_UPDATE_DATE	05-JUN-17 08.05.20.110000000 PM
EXT_STATUS	UPLOADED
EXT_DISABLED	N
VERSION_NUM	1
ARTIFACT	Spi
ARTIFACT_VALUE	com.ofss.fc.app.ac.policy.accountwriteoff.AccountWriteOffBusinessPolicy

Figure 8–45 Business Policy Extension Deployment - Single Record View

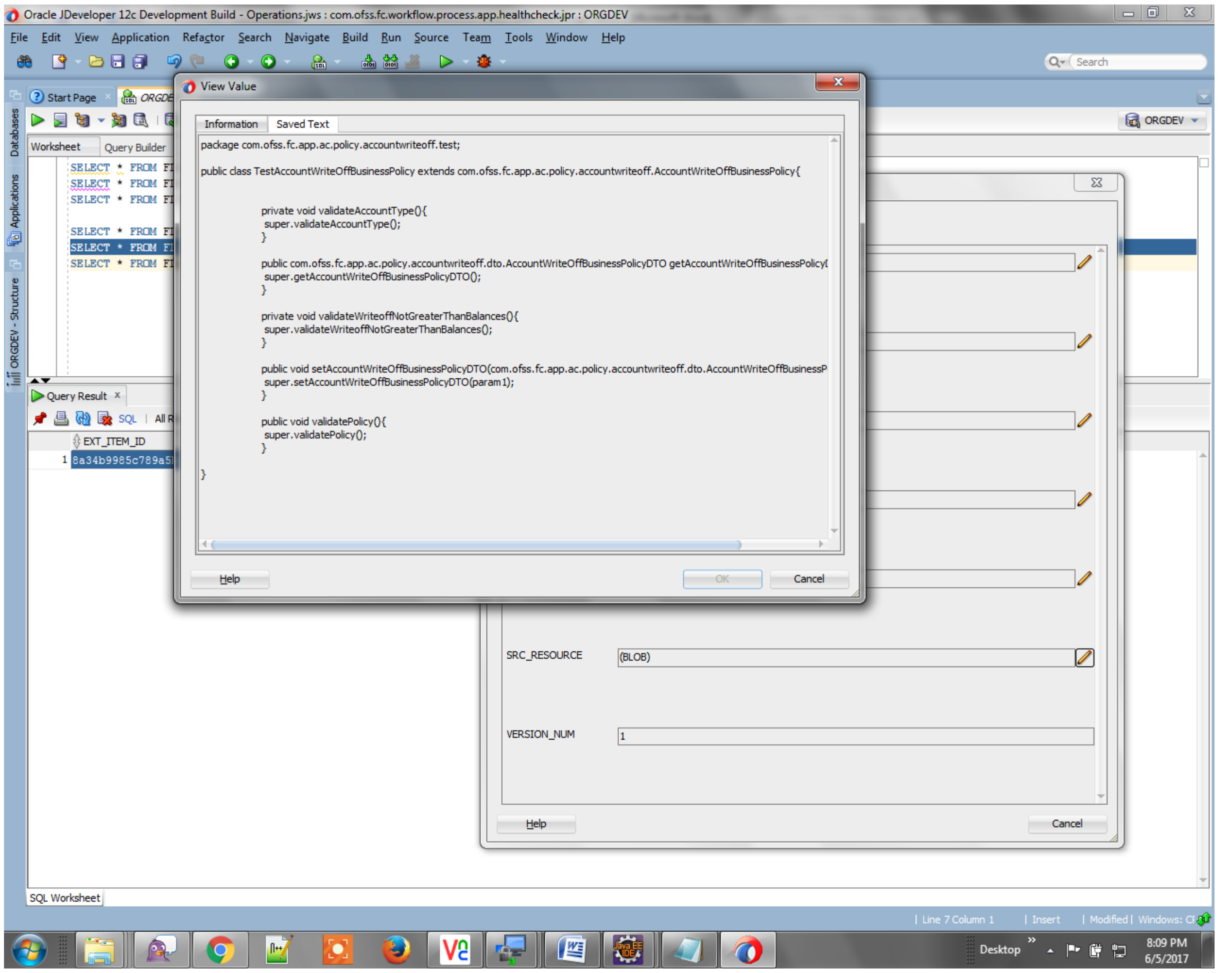

Oracle JDeveloper 12c Development Build - Operations.jws : com.ofss.fc.workflow.process.app.healthcheck.jpr : ORGDEV
File Edit View Application Refactor Search Navigate Build Run Source Team Tools Window Help
Start Page
ORGDEV
Worksheet
Query Builder
Query Result
SQL
All Rows Fetched: 1 in 0 seconds
Single Record View
Help
Cancel
SQL Worksheet
Line 7 Column 1
Insert
Modified
Desktop
8:09 PM
6/5/2017

```
SELECT * FROM FLX_EXT_EXTENSION_B
SELECT * FROM FLX_EXT_EXTENSION_ITEMS
SELECT * FROM FLX_FW_CONFIG_ALL_B WHERE CATEGORY_ID = 'Service

SELECT * FROM FLX_EXT_BUSINESS_POLICY_B
SELECT * FROM FLX_EXT_BUSINESS_POLICY_ITEMS
SELECT * FROM FLX_FW_CONFIG_ALL_B WHERE CATEGORY_ID = 'Busines
```

	EXT_ITEM_ID	EXT_RESOURCE_NAME
1	8a34b9985c789a5b015c78ae800f0002	TestAccountWriteOffBusinessPc

EXT_ITEM_ID	8a34b9985c789a5b015c78ae800f0002
EXT_RESOURCE_NAME	TestAccountWriteOffBusinessPolicy
EXT_ID	TestAccountWriteOffBusinessPolicy
EXT_TYPE	GROOVY
BLB_RESOURCE	(BLOB)
SRC_RESOURCE	(BLOB)
VERSION_NUM	1

Figure 8–46 Business Policy Extension Deployment - View Value

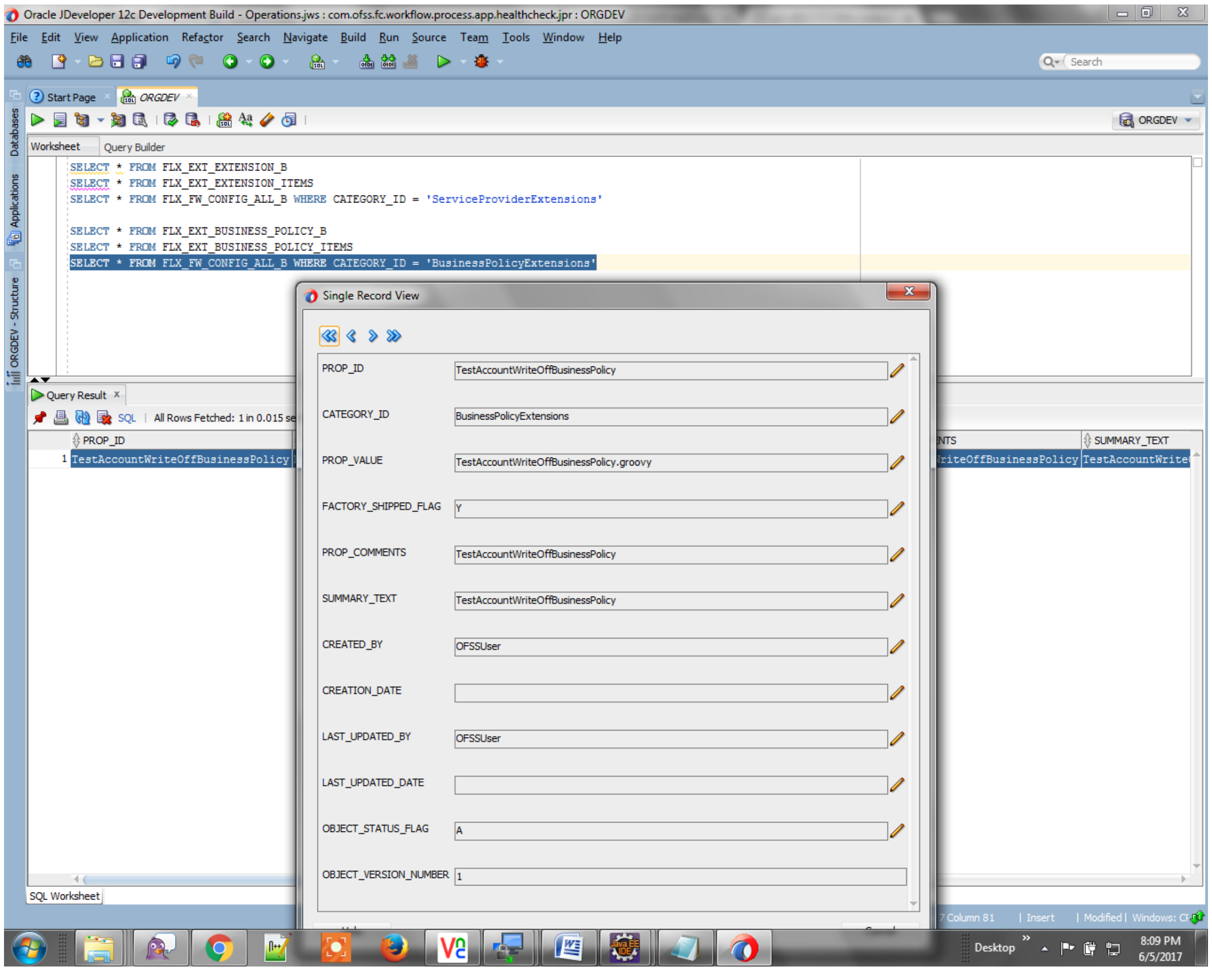


```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;

public class TestAccountWriteOffBusinessPolicy extends com.ofss.fc.app.ac.policy.accountwriteoff.AccountWriteOffBusinessPolicy{


        private void validateAccountType(){
         super.validateAccountType();
        }

        public com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO getAccountWriteOffBusinessPolicy
         super.getAccountWriteOffBusinessPolicyDTO();
        }

        private void validateWriteoffNotGreaterThanBalances(){
         super.validateWriteoffNotGreaterThanBalances();
        }

        public void setAccountWriteOffBusinessPolicyDTO(com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessP
         super.setAccountWriteOffBusinessPolicyDTO(param1);
        }

        public void validatePolicy(){
         super.validatePolicy();
        }

}
```

SRC_RESOURCE (BLOB)

VERSION_NUM 1

Figure 8–47 Business Policy Extension Deployment - Single Record View



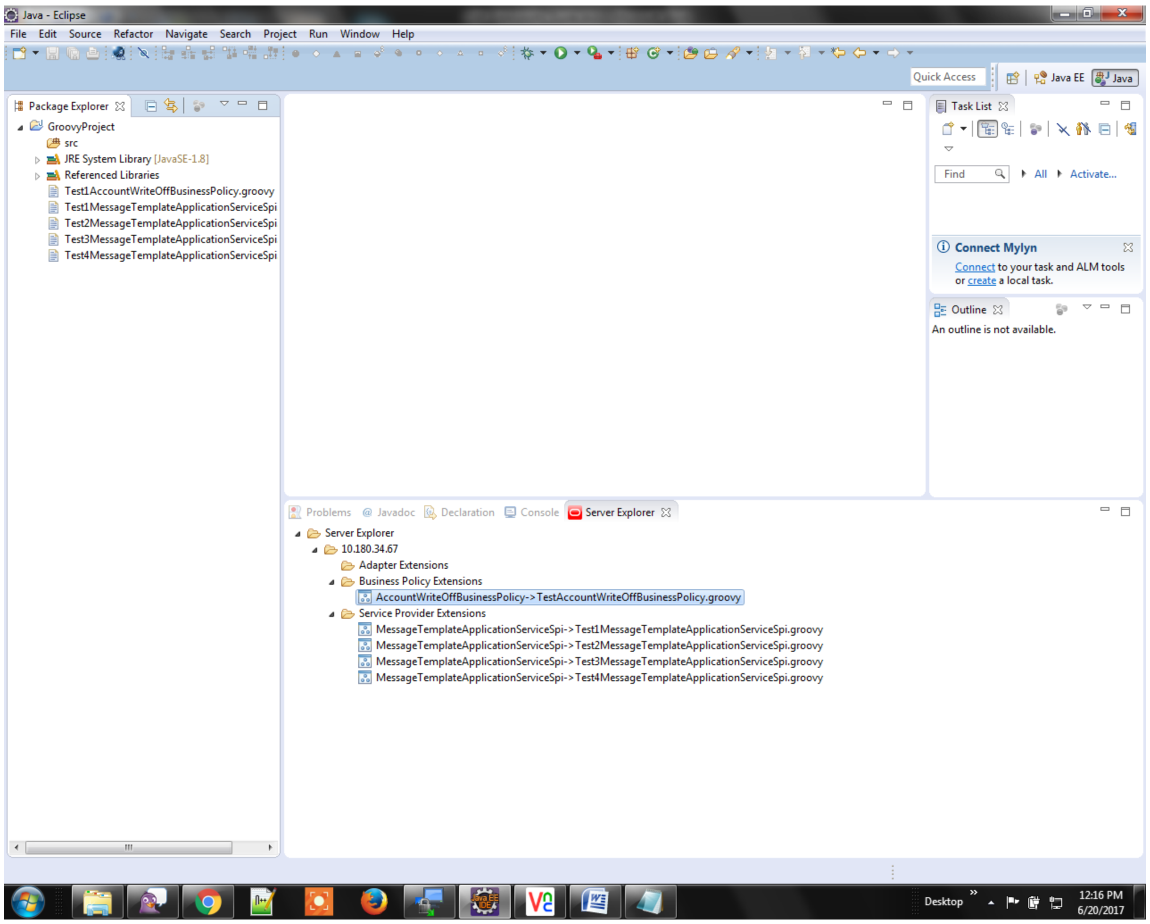
```
SELECT * FROM FLX_EXT_EXTENSION_B
SELECT * FROM FLX_EXT_EXTENSION_ITEMS
SELECT * FROM FLX_FW_CONFIG_ALL_B WHERE CATEGORY_ID = 'ServiceProviderExtensions'

SELECT * FROM FLX_EXT_BUSINESS_POLICY_B
SELECT * FROM FLX_EXT_BUSINESS_POLICY_ITEMS
SELECT * FROM FLX_FW_CONFIG_ALL_B WHERE CATEGORY_ID = 'BusinessPolicyExtensions'
```

Single Record View

PROP_ID	TestAccountWriteOffBusinessPolicy
CATEGORY_ID	BusinessPolicyExtensions
PROP_VALUE	TestAccountWriteOffBusinessPolicy.groovy
FACTORY_SHIPPED_FLAG	Y
PROP_COMMENTS	TestAccountWriteOffBusinessPolicy
SUMMARY_TEXT	TestAccountWriteOffBusinessPolicy
CREATED_BY	OFSSUser
CREATION_DATE	
LAST_UPDATED_BY	OFSSUser
LAST_UPDATED_DATE	
OBJECT_STATUS_FLAG	A
OBJECT_VERSION_NUMBER	1

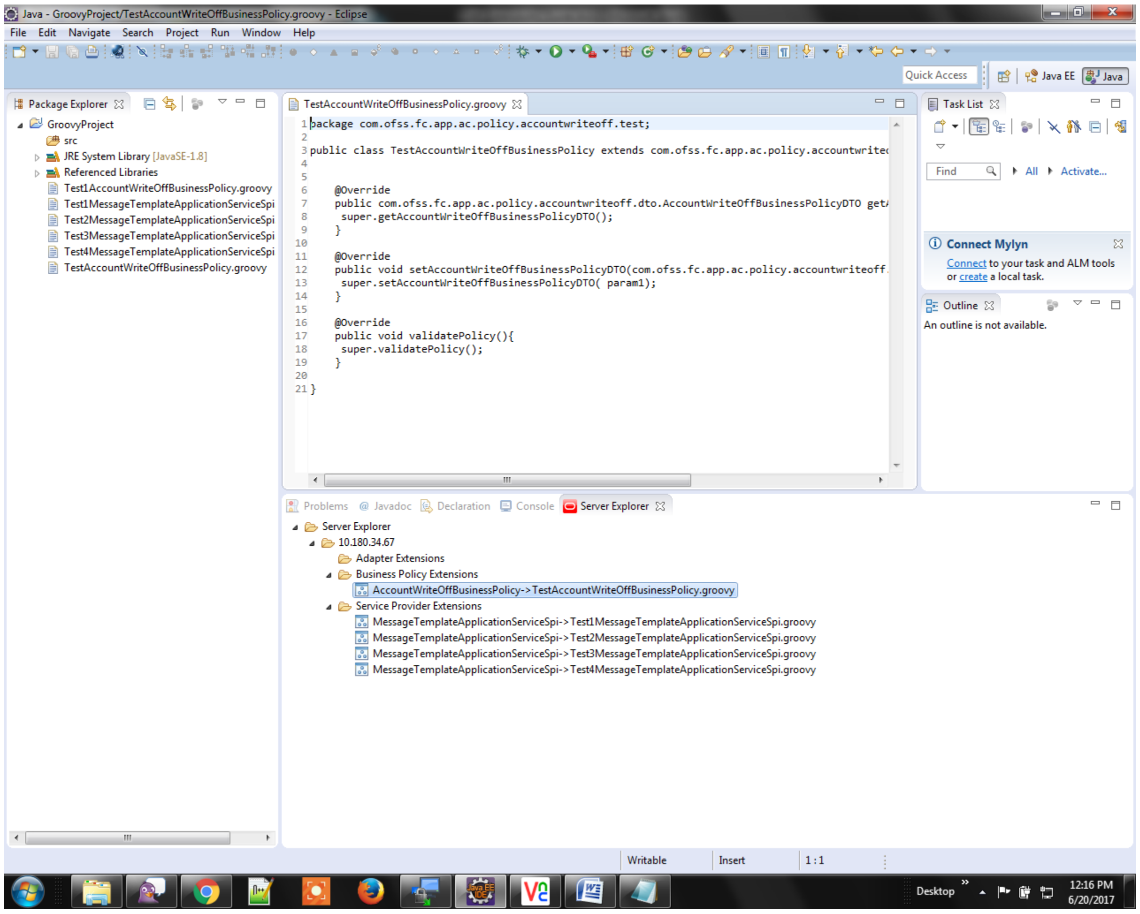
8.3.5 Fetching Deployed Business Policy Extension

Figure 8–48 Fetching Deployed Business Policy Extension


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Server Explorer
10.180.34.67
Adapter Extensions
Business Policy Extensions
AccountWriteOffBusinessPolicy->TestAccountWriteOffBusinessPolicy.groovy
Service Provider Extensions
MessageTemplateApplicationServiceSpi->Test1MessageTemplateApplicationServiceSpi.groovy
MessageTemplateApplicationServiceSpi->Test2MessageTemplateApplicationServiceSpi.groovy
MessageTemplateApplicationServiceSpi->Test3MessageTemplateApplicationServiceSpi.groovy
MessageTemplateApplicationServiceSpi->Test4MessageTemplateApplicationServiceSpi.groovy
Desktop
12:16 PM
6/20/2017

1. For the purpose of fetching a deployed Extension from the Server, you can click on the appropriate Extension under the appropriate child under Server Explorer.

Figure 8–49 Click Extension under Server Explorer



```
GroovyProject
    src
    JRE System Library [JavaSE-1.8]
    Referenced Libraries
    Test1AccountWriteOffBusinessPolicy.groovy
    Test1MessageTemplateApplicationServiceSpi
    Test2MessageTemplateApplicationServiceSpi
    Test3MessageTemplateApplicationServiceSpi
    Test4MessageTemplateApplicationServiceSpi
    TestAccountWriteOffBusinessPolicy.groovy
```

TestAccountWriteOffBusinessPolicy.groovy

```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;

public class TestAccountWriteOffBusinessPolicy extends com.ofss.fc.app.ac.policy.accountwritec


    @Override
    public com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO get/
     super.getAccountWriteOffBusinessPolicyDTO();
    }

    @Override
    public void setAccountWriteOffBusinessPolicyDTO(com.ofss.fc.app.ac.policy.accountwriteoff.
     super.setAccountWriteOffBusinessPolicyDTO( param1);
    }

    @Override
    public void validatePolicy(){
     super.validatePolicy();
    }

}
```

Connect Mylyn

Connect to your task and ALM tools or create a local task.

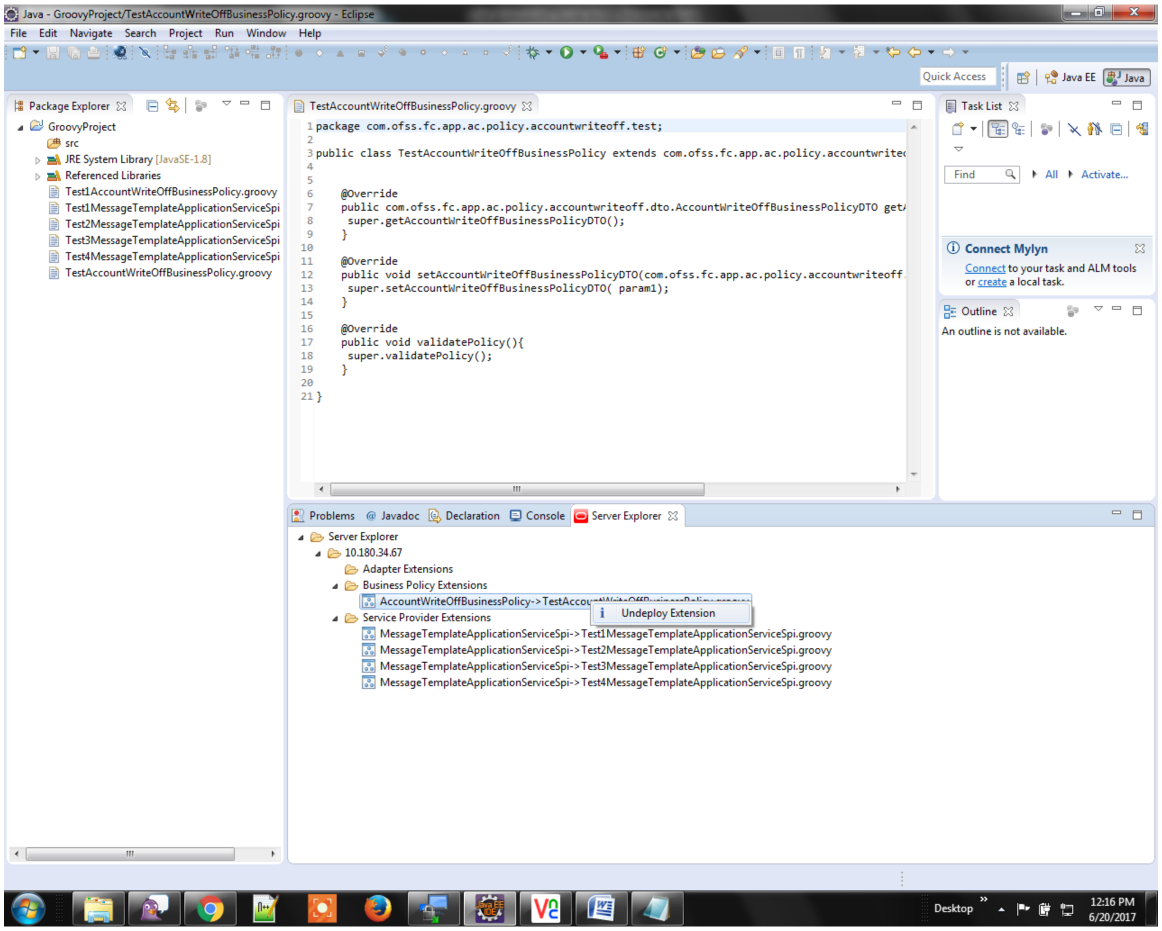
An outline is not available.

```
Server Explorer
    10.180.34.67
        Adapter Extensions
        Business Policy Extensions
            AccountWriteOffBusinessPolicy->TestAccountWriteOffBusinessPolicy.groovy
        Service Provider Extensions
            MessageTemplateApplicationServiceSpi->Test1MessageTemplateApplicationServiceSpi.groovy
            MessageTemplateApplicationServiceSpi->Test2MessageTemplateApplicationServiceSpi.groovy
            MessageTemplateApplicationServiceSpi->Test3MessageTemplateApplicationServiceSpi.groovy
            MessageTemplateApplicationServiceSpi->Test4MessageTemplateApplicationServiceSpi.groovy
```

2. The extension gets saved in the project created to contain the Extension classes generated through the plugin. Also the Extension code opens up in the Code Editor.

8.3.6 Undeploying Business Policy Extension from Server

Figure 8–50 Undeploying the Extension from Server

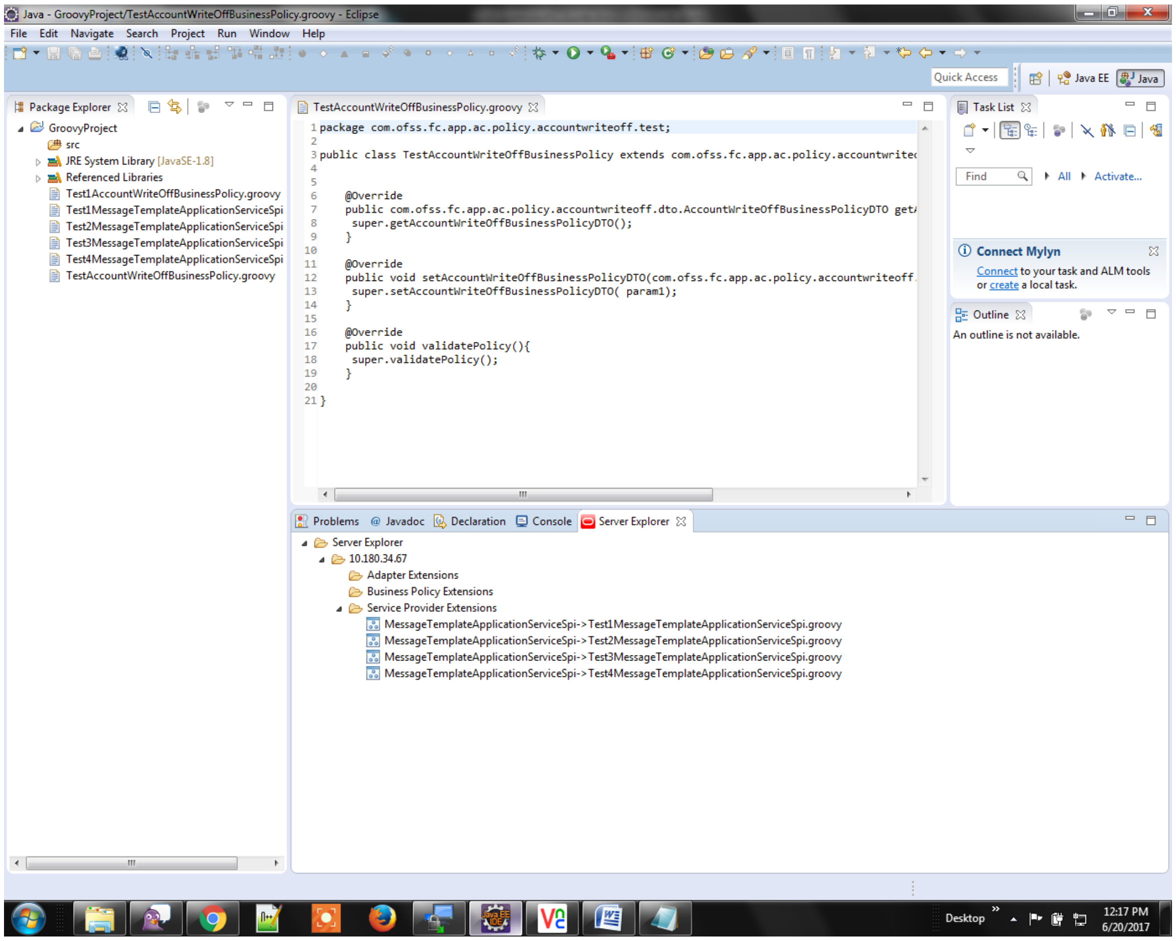

Java - GroovyProject/TestAccountWriteOffBusinessPolicy.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
TestAccountWriteOffBusinessPolicy.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Undeploy Extension
Desktop
12:16 PM
6/20/2017

```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;

public class TestAccountWriteOffBusinessPolicy extends com.ofss.fc.app.ac.policy.accountwritec


    @Override
    public com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO get/
     super.getAccountWriteOffBusinessPolicyDTO();
    }

    @Override
    public void setAccountWriteOffBusinessPolicyDTO(com.ofss.fc.app.ac.policy.accountwriteoff.
     super.setAccountWriteOffBusinessPolicyDTO( param1);
    }

    @Override
    public void validatePolicy(){
     super.validatePolicy();
    }

}
```

```
Server Explorer
  10.180.34.67
    Adapter Extensions
    Business Policy Extensions
      AccountWriteOffBusinessPolicy->TestAccou
    Service Provider Extensions
      MessageTemplateApplicationServiceSpi->Test1MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test2MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test3MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test4MessageTemplateApplicationServiceSpi.groovy
```

1. For the purpose of undeploying the Extension from the Server, you right click on the specific Extension under the appropriate child of the Server configured under Server Explorer.

Figure 8–51 Undeploying the Extension from Server



```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;

public class TestAccountWriteOffBusinessPolicy extends com.ofss.fc.app.ac.policy.accountwritec


    @Override
    public com.ofss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO get/
     super.getAccountWriteOffBusinessPolicyDTO();
    }

    @Override
    public void setAccountWriteOffBusinessPolicyDTO(com.ofss.fc.app.ac.policy.accountwriteoff.
     super.setAccountWriteOffBusinessPolicyDTO( param1);
    }

    @Override
    public void validatePolicy(){
     super.validatePolicy();
    }

}
```

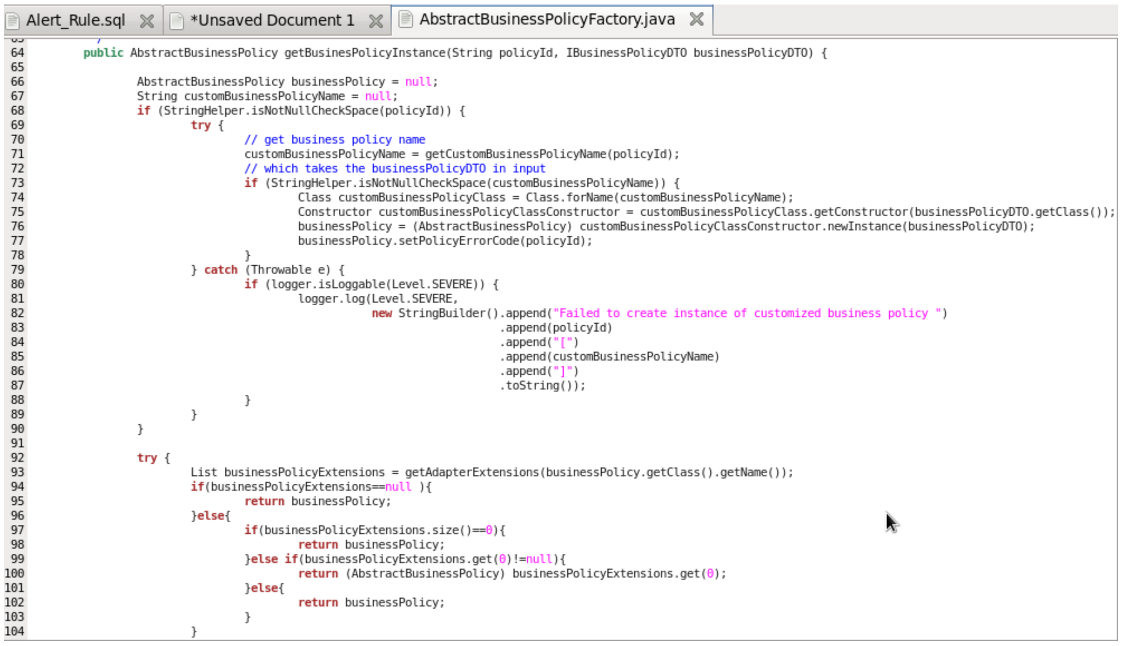
```
Server Explorer
  10.180.34.67
    Adapter Extensions
    Business Policy Extensions
    Service Provider Extensions
      MessageTemplateApplicationServiceSpi->Test1MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test2MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test3MessageTemplateApplicationServiceSpi.groovy
      MessageTemplateApplicationServiceSpi->Test4MessageTemplateApplicationServiceSpi.groovy
```

2. The specific Extension under the appropriate child of the Server configured under "Server Explorer" disappears and gets undeployed from the Server.

8.3.7 Inclusion of Groovy Extension in Actual Code Flow

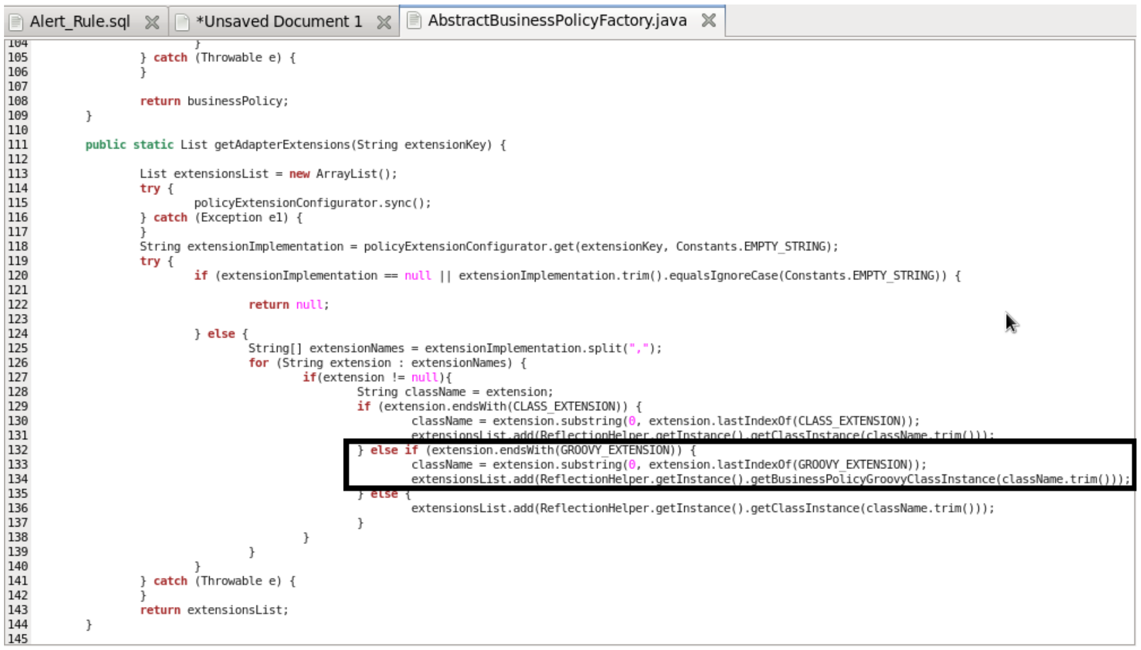
1. Whenever an AbstractBusinessPolicyFactory child class invokes getBusinessPolicyInstance() method the parent class corresponding method gets invoked that deciphers the appropriate Business Policy Instance.

Figure 8–52 AbstractBusinessPolicyFactory.java



```java
    public AbstractBusinessPolicy getBusinesPolicyInstance(String policyId, IBusinessPolicyDTO businessPolicyDTO) {

        AbstractBusinessPolicy businessPolicy = null;
        String customBusinessPolicyName = null;
        if (StringHelper.isNotNullCheckSpace(policyId)) {
            try {
                // get business policy name
                customBusinessPolicyName = getCustomBusinessPolicyName(policyId);
                // which takes the businessPolicyDTO in input
                if (StringHelper.isNotNullCheckSpace(customBusinessPolicyName)) {
                    Class customBusinessPolicyClass = Class.forName(customBusinessPolicyName);
                    Constructor customBusinessPolicyClassConstructor = customBusinessPolicyClass.getConstructor(businessPolicyDTO.getClass());
                    businessPolicy = (AbstractBusinessPolicy) customBusinessPolicyClassConstructor.newInstance(businessPolicyDTO);
                    businessPolicy.setPolicyErrorCode(policyId);
                }
            } catch (Throwable e) {
                if (logger.isLoggable(Level.SEVERE)) {
                    logger.log(Level.SEVERE,
                            new StringBuilder().append("Failed to create instance of customized business policy ")
                                                .append(policyId)
                                                .append("[")
                                                .append(customBusinessPolicyName)
                                                .append("]")
                                                .toString());
                }
            }
        }

        try {
            List businessPolicyExtensions = getAdapterExtensions(businessPolicy.getClass().getName());
            if(businessPolicyExtensions==null ){
                return businessPolicy;
            }else{
                if(businessPolicyExtensions.size()==0){
                    return businessPolicy;
                }else if(businessPolicyExtensions.get(0)!=null){
                    return (AbstractBusinessPolicy) businessPolicyExtensions.get(0);
                }else{
                    return businessPolicy;
                }
            }
```

Figure 8–53 AbstractBusinessPolicyFactory.java



```java
            }
        } catch (Throwable e) {
        }

        return businessPolicy;
    }

    public static List getAdapterExtensions(String extensionKey) {

        List extensionsList = new ArrayList();
        try {
            policyExtensionConfigurator.sync();
        } catch (Exception e1) {
        }
        String extensionImplementation = policyExtensionConfigurator.get(extensionKey, Constants.EMPTY_STRING);
        try {
            if (extensionImplementation == null || extensionImplementation.trim().equalsIgnoreCase(Constants.EMPTY_STRING)) {

                return null;

            } else {
                String[] extensionNames = extensionImplementation.split(",");
                for (String extension : extensionNames) {
                    if(extension != null){
                        String className = extension;
                        if (extension.endsWith(CLASS_EXTENSION)) {
                            className = extension.substring(0, extension.lastIndexOf(CLASS_EXTENSION));
                            extensionsList.add(ReflectionHelper.getInstance().getClassInstance(className.trim()));
                        } else if (extension.endsWith(GROOVY_EXTENSION)) {
                            className = extension.substring(0, extension.lastIndexOf(GROOVY_EXTENSION));
                            extensionsList.add(ReflectionHelper.getInstance().getBusinessPolicyGroovyClassInstance(className.trim()));
                        } else {
                            extensionsList.add(ReflectionHelper.getInstance().getClassInstance(className.trim()));
                        }
                    }
                }
            }
        } catch (Throwable e) {
        }
        return extensionsList;
    }

```

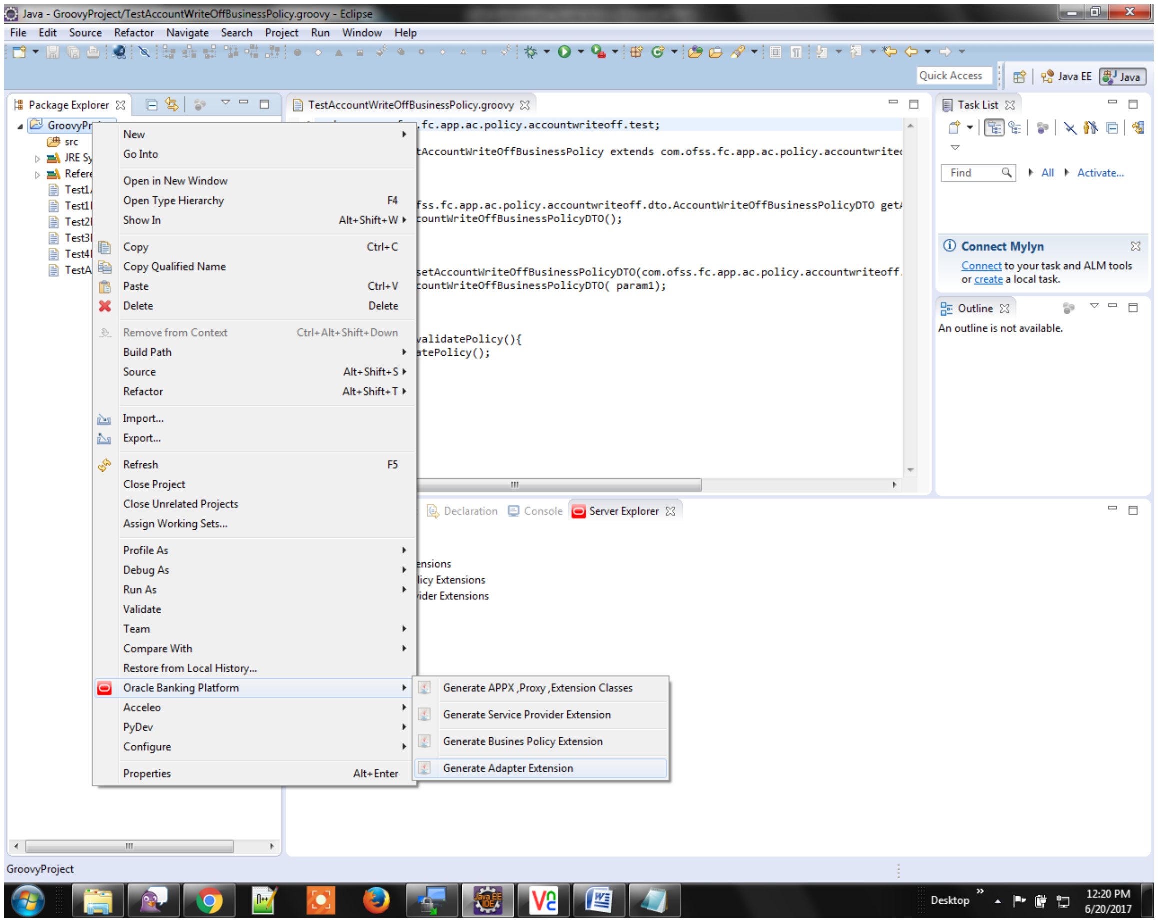
2. The name of this Business Poicy Class is then passed to “getAdapterExtensions(String extensionKey)” which then gets the extension name from FLX_FW_CONFIG_ALL_B based on the CATEGORY_ID=”BusinessPolicyExtensions” and PROP_ID=<Business Policy Class Name>. This Extension is then used to get the Groovy Source code which is then parsed and compiled and returned as the appropriate Business Policy Class.

8.4 Support for Adapter Extension

8.4.1 Generate Adapter Extension

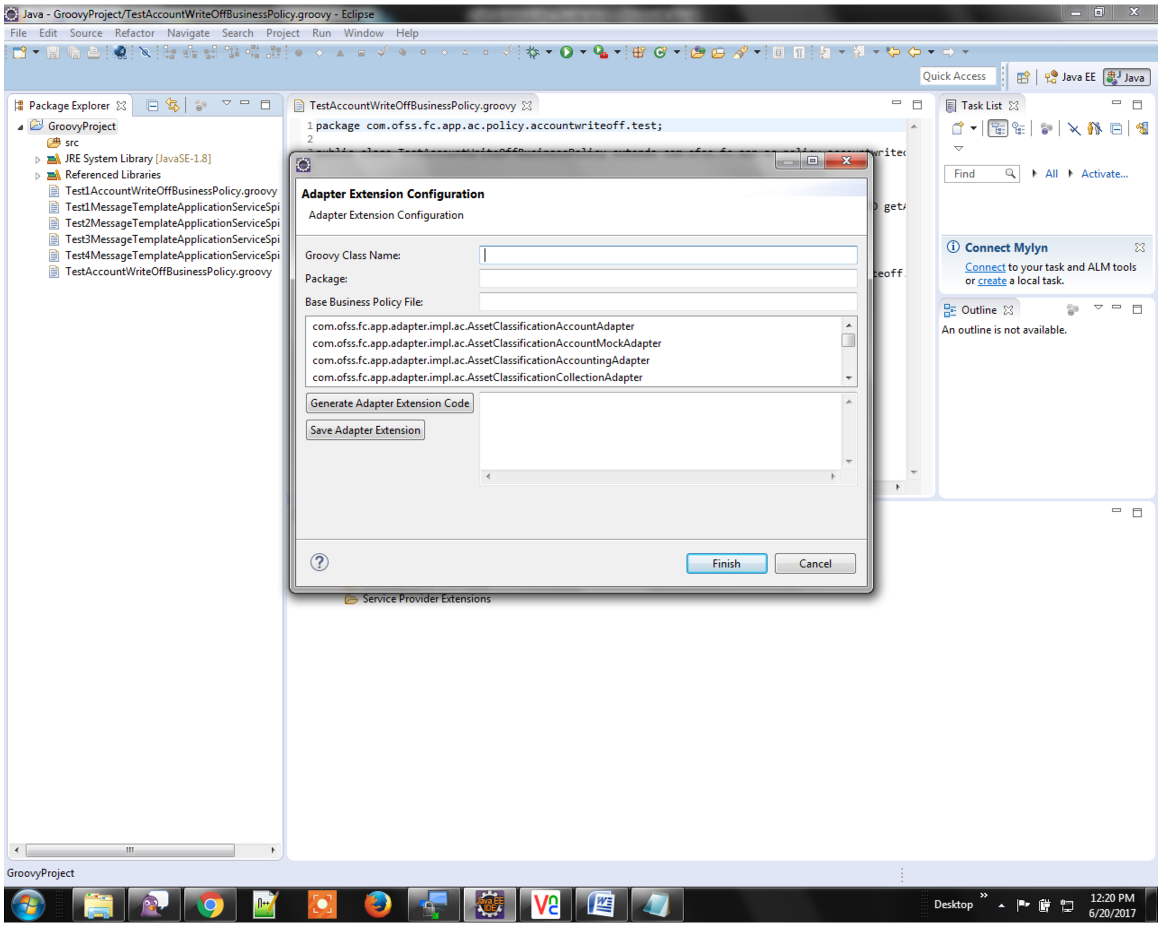
1. The parameter Temporary Project has been configured to point to the base project where the Groovy Extension Files have to be saved.
2. Right click on this project and select Oracle Banking Platform > Generate Adapter Extension.

Figure 8–54 Generate Adapter Extension



```
.fc.app.ac.policy.accountwriteoff.test;

tAccountWriteOffBusinessPolicy extends com.ofss.fc.app.ac.policy.accountwritec

fss.fc.app.ac.policy.accountwriteoff.dto.AccountWriteOffBusinessPolicyDTO get/
countWriteOffBusinessPolicyDTO();

setAccountWriteOffBusinessPolicyDTO(com.ofss.fc.app.ac.policy.accountwriteoff.
countWriteOffBusinessPolicyDTO( param1);

validatePolicy(){
atePolicy();
```

Figure 8–55 Adapter Extension Configuration


Java - GroovyProject/TestAccountWriteOffBusinessPolicy.groovy - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
TestAccountWriteOffBusinessPolicy.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Service Provider Extensions
GroovyProject
Desktop
12:20 PM
6/20/2017

```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;
```

Adapter Extension Configuration

Adapter Extension Configuration

Groovy Class Name:

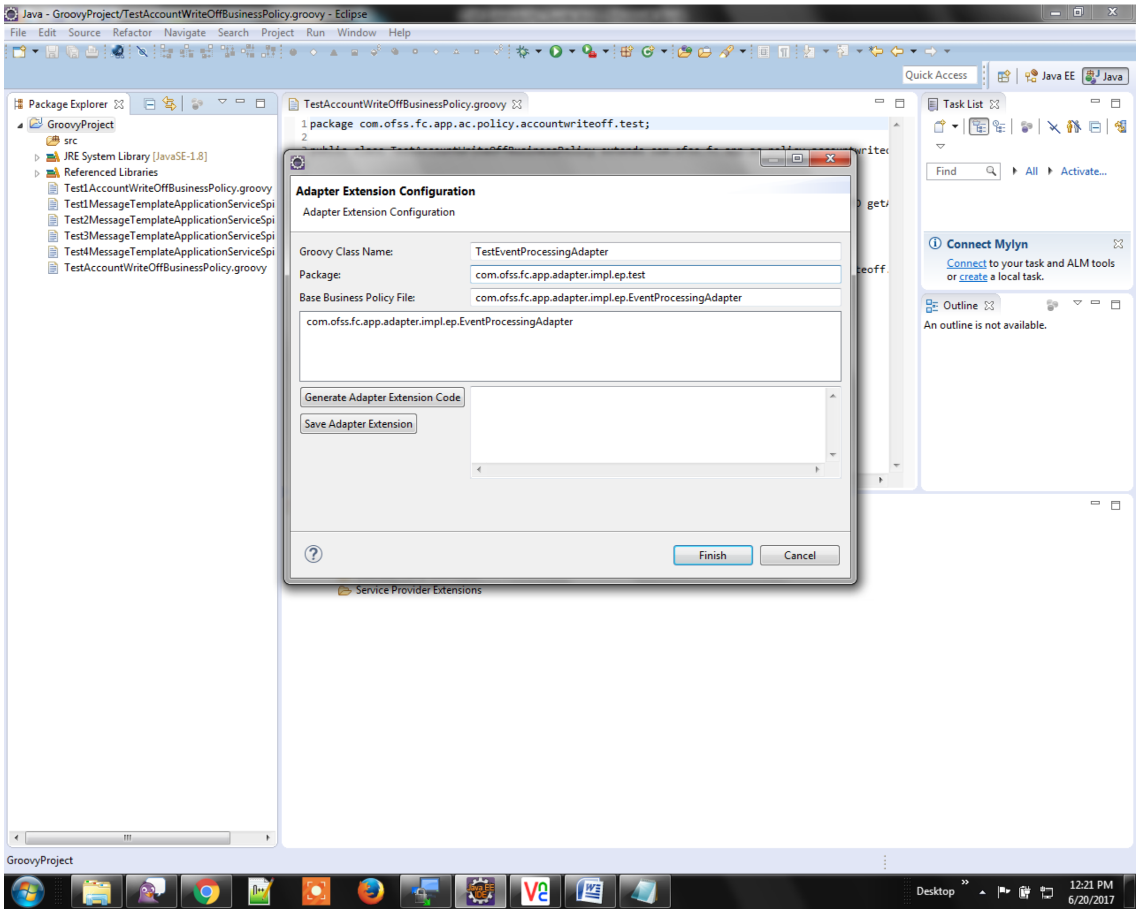
Package:

Base Business Policy File:

```
com.ofss.fc.app.adapter.impl.ac.AssetClassificationAccountAdapter
com.ofss.fc.app.adapter.impl.ac.AssetClassificationAccountMockAdapter
com.ofss.fc.app.adapter.impl.ac.AssetClassificationAccountingAdapter
com.ofss.fc.app.adapter.impl.ac.AssetClassificationCollectionAdapter
```

3. The above wizard appears with a list of Adapter Files.

Figure 8–56 Adapter Extension Configuration


Java - GroovyProject/TestAccountWriteOffBusinessPolicy.groovy - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
TestAccountWriteOffBusinessPolicy.groovy
Task List
Connect Mylyn
Outline
An outline is not available.
Service Provider Extensions
Generate Adapter Extension Code
Save Adapter Extension
Finish
Cancel

```
package com.ofss.fc.app.ac.policy.accountwriteoff.test;
```

Adapter Extension Configuration

Adapter Extension Configuration

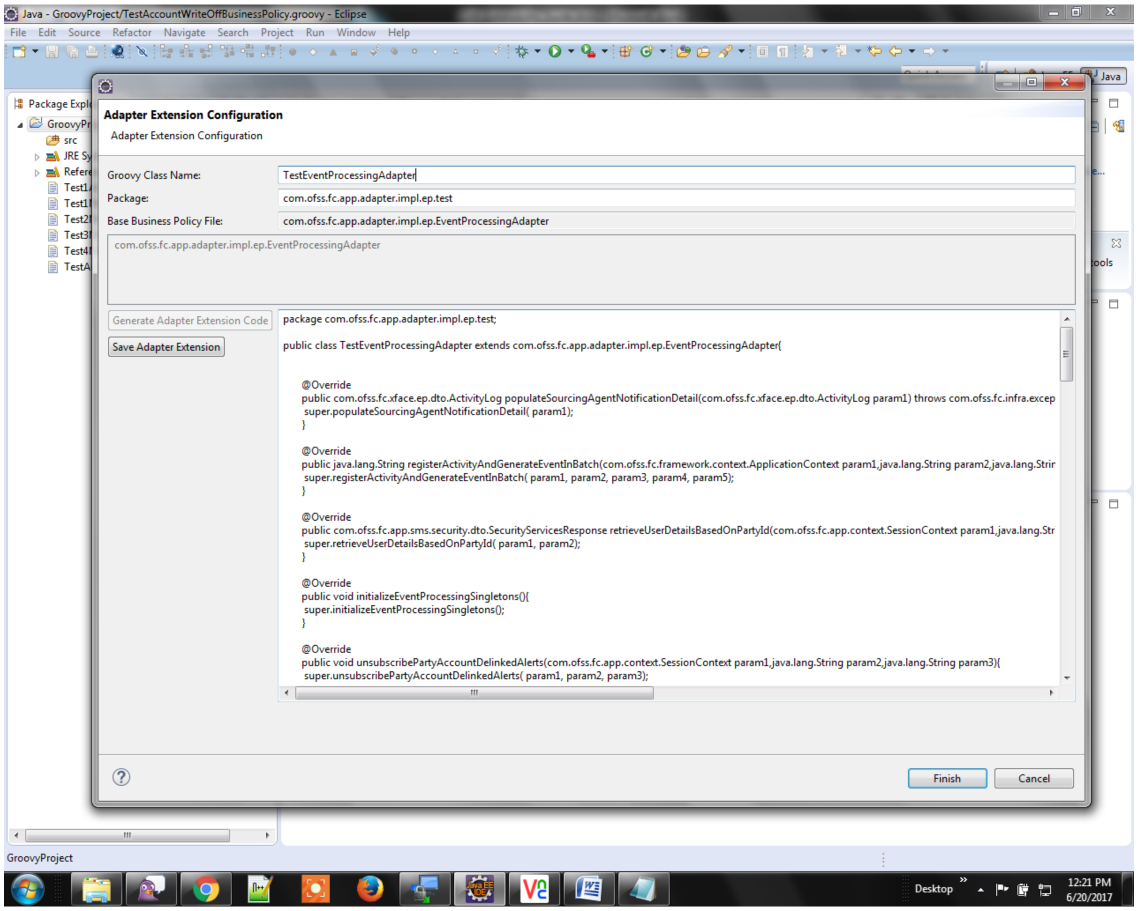
Groovy Class Name:	TestEventProcessingAdapter
Package:	com.ofss.fc.app.adapter.impl.ep.test
Base Business Policy File:	com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapter

com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapter

4. Select the filtered Base Business Policy file from the list such that it appears as the input for Base Adapter file.

5. Appropriately set values for the Extension Class Name and package.

Figure 8–57 Enter Extension Class Name and Package


Java - GroovyProject/TestAccountWriteOffBusinessPolicy.groovy - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Adapter Extension Configuration
Adapter Extension Configuration
Generate Adapter Extension Code
Save Adapter Extension
Finish
Cancel

Groovy Class Name: TestEventProcessingAdapter

Package: com.ofss.fc.app.adapter.impl.ep.test

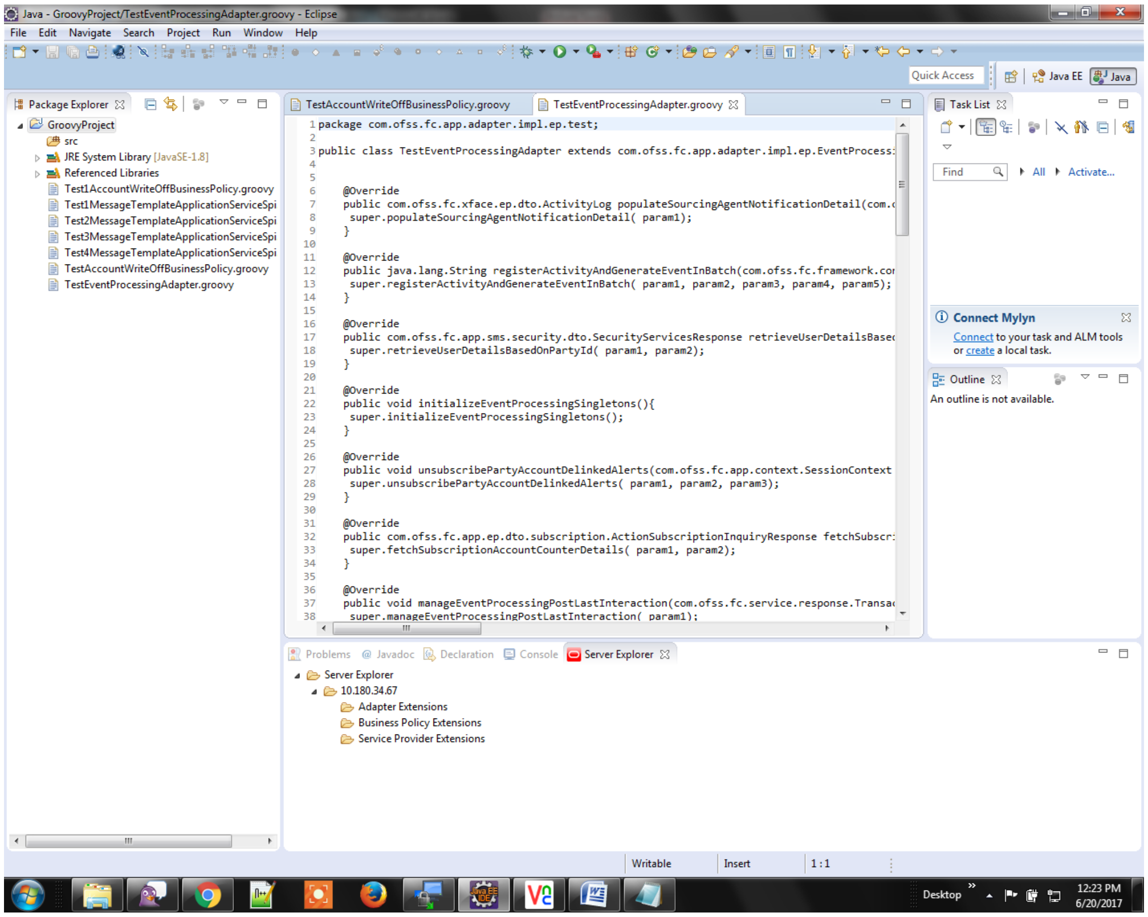
Base Business Policy File: com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapter

com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapter

```
package com.ofss.fc.app.adapter.impl.ep.test;

public class TestEventProcessingAdapter extends com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapter{


    @Override
    public com.ofss.fc.xface.ep.dto.ActivityLog populateSourcingAgentNotificationDetail(com.ofss.fc.xface.ep.dto.ActivityLog param1) throws com.ofss.fc.infra.excep
     super.populateSourcingAgentNotificationDetail( param1);
    }

    @Override
    public java.lang.String registerActivityAndGenerateEventInBatch(com.ofss.fc.framework.context.ApplicationContext param1,java.lang.String param2,java.lang.Strir
     super.registerActivityAndGenerateEventInBatch( param1, param2, param3, param4, param5);
    }

    @Override
    public com.ofss.fc.app.sms.security.dto.SecurityServicesResponse retrieveUserDetailsBasedOnPartyId(com.ofss.fc.app.context.SessionContext param1,java.lang.Str
     super.retrieveUserDetailsBasedOnPartyId( param1, param2);
    }

    @Override
    public void initializeEventProcessingSingletons(){
     super.initializeEventProcessingSingletons();
    }

    @Override
    public void unsubscribePartyAccountDelinkedAlerts(com.ofss.fc.app.context.SessionContext param1,java.lang.String param2,java.lang.String param3){
     super.unsubscribePartyAccountDelinkedAlerts( param1, param2, param3);
```

6. Click on Generate Adapter Extension Code. The code gets generated with the base adapter methods.
7. Click on Save Adapter Extension and Finish.

Figure 8–58 Save Adapter Extension and Finish

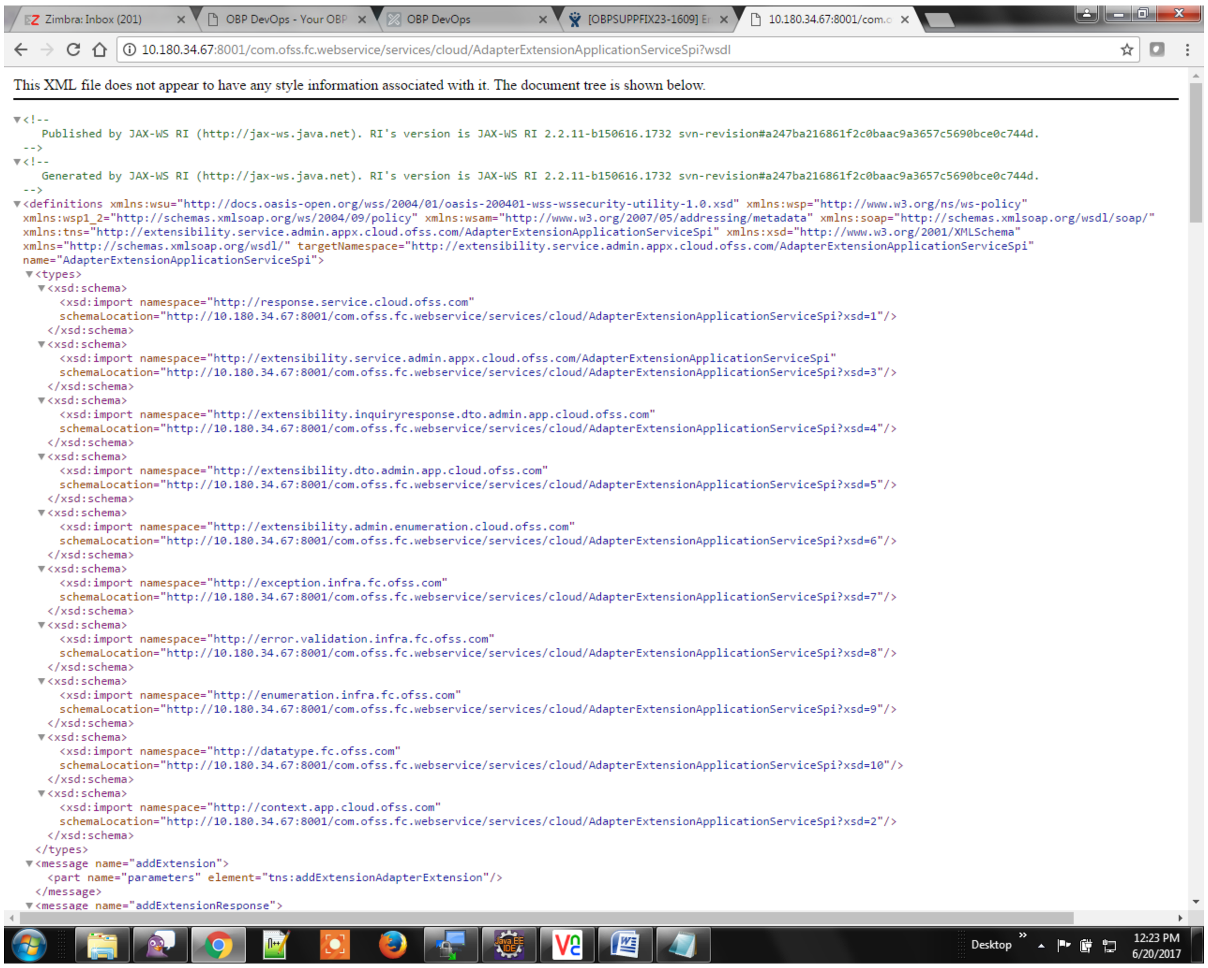


```
package com.ofss.fc.app.adapter.impl.ep.test;

public class TestEventProcessingAdapter extends com.ofss.fc.app.adapter.impl.ep.EventProcess


    @Override
    public com.ofss.fc.xface.ep.dto.ActivityLog populateSourcingAgentNotificationDetail(com.c
     super.populateSourcingAgentNotificationDetail( param1);
    }

    @Override
    public java.lang.String registerActivityAndGenerateEventInBatch(com.ofss.fc.framework.cor
     super.registerActivityAndGenerateEventInBatch( param1, param2, param3, param4, param5);
    }

    @Override
    public com.ofss.fc.app.sms.security.dto.SecurityServicesResponse retrieveUserDetailsBasec
     super.retrieveUserDetailsBasedOnPartyId( param1, param2);
    }

    @Override
    public void initializeEventProcessingSingletons(){
     super.initializeEventProcessingSingletons();
    }

    @Override
    public void unsubscribePartyAccountDelinkedAlerts(com.ofss.fc.app.context.SessionContext
     super.unsubscribePartyAccountDelinkedAlerts( param1, param2, param3);
    }

    @Override
    public com.ofss.fc.app.ep.dto.subscription.ActionSubscriptionInquiryResponse fetchSubscr
     super.fetchSubscriptionAccountCounterDetails( param1, param2);
    }

    @Override
    public void manageEventProcessingPostLastInteraction(com.ofss.fc.service.response.Transac
     super.manageEventProcessingPostLastInteraction( param1);
```

8. The extension gets saved in the project created to contain the Extension classes generated through the plugin. You can add all the code required in the base methods for extensibility of the base Business Policy.

8.4.2 Exposed Webservice for Adapter Extensions

Figure 8–59 Adapter Extension Application Service Spi



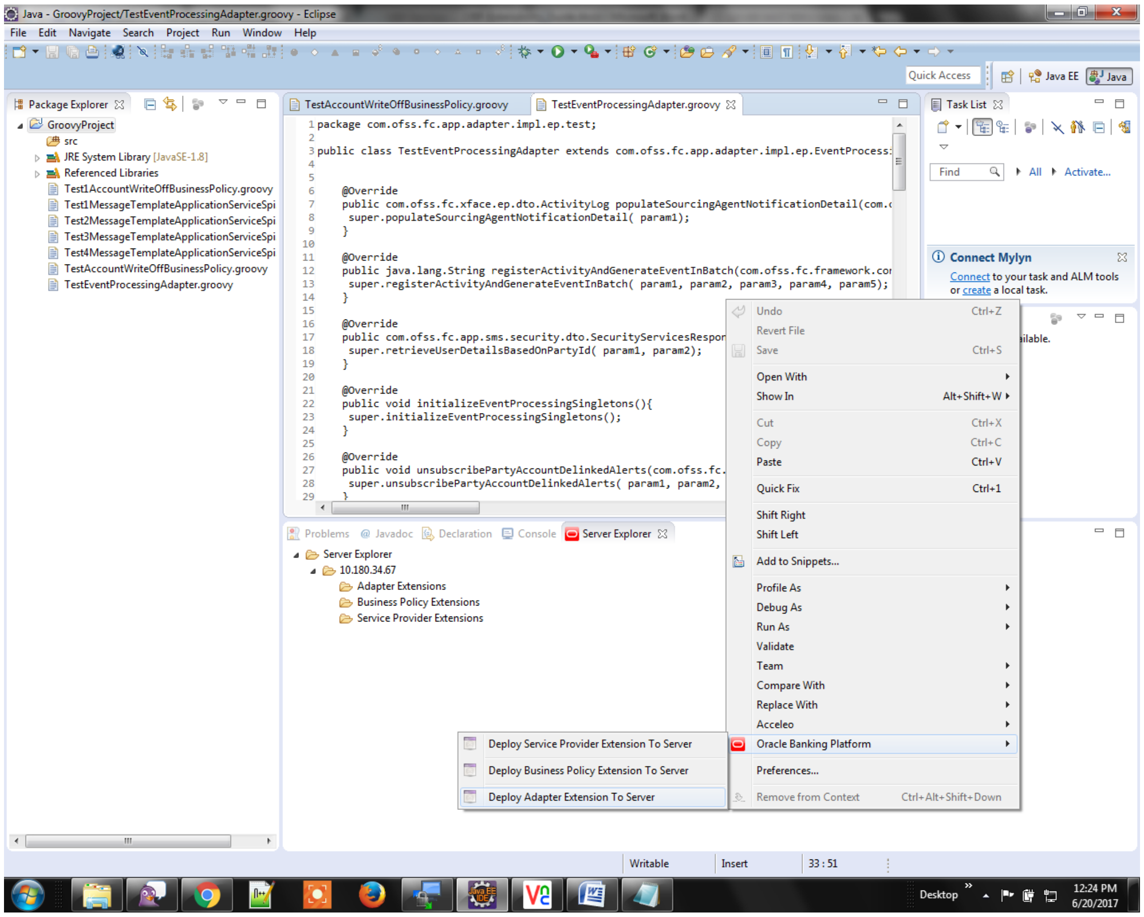
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<!--
    Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.11-b150616.1732 svn-revision#a247ba216861f2c0baac9a3657c5690bce0c744d.
 -->
▼<!--
    Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.11-b150616.1732 svn-revision#a247ba216861f2c0baac9a3657c5690bce0c744d.
 -->
▼<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
 xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tns="http://extensibility.service.admin.appx.cloud.ofss.com/AdapterExtensionApplicationServiceSpi" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://extensibility.service.admin.appx.cloud.ofss.com/AdapterExtensionApplicationServiceSpi"
 name="AdapterExtensionApplicationServiceSpi">
  ▼<types>
    ▼<xsd:schema>
       <xsd:import namespace="http://response.service.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/AdapterExtensionApplicationServiceSpi?xsd=1"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.service.admin.appx.cloud.ofss.com/AdapterExtensionApplicationServiceSpi"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/AdapterExtensionApplicationServiceSpi?xsd=3"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.inquiryresponse.dto.admin.app.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/AdapterExtensionApplicationServiceSpi?xsd=4"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.dto.admin.app.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/AdapterExtensionApplicationServiceSpi?xsd=5"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://extensibility.admin.enumeration.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/AdapterExtensionApplicationServiceSpi?xsd=6"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://exception.infra.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/AdapterExtensionApplicationServiceSpi?xsd=7"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://error.validation.infra.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/AdapterExtensionApplicationServiceSpi?xsd=8"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://enumeration.infra.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/AdapterExtensionApplicationServiceSpi?xsd=9"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://datatype.fc.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/AdapterExtensionApplicationServiceSpi?xsd=10"/>
     </xsd:schema>
    ▼<xsd:schema>
       <xsd:import namespace="http://context.app.cloud.ofss.com"
       schemaLocation="http://10.180.34.67:8001/com.ofss.fc.webservice/services/cloud/AdapterExtensionApplicationServiceSpi?xsd=2"/>
     </xsd:schema>
   </types>
  ▼<message name="addExtension">
     <part name="parameters" element="tns:addExtensionAdapterExtension"/>
   </message>
  ▼<message name="addExtensionResponse">
```

Adapter Extension Application Service Spi is the web service that exposes the web services to add adapter extensions, fetch adapter extensions and delete adapter extensions. These services are used by the eclipse plugin to deploy extensions, fetch extensions and undeploy extensions at runtime.

8.4.3 Deploy Adapter Extension to Server

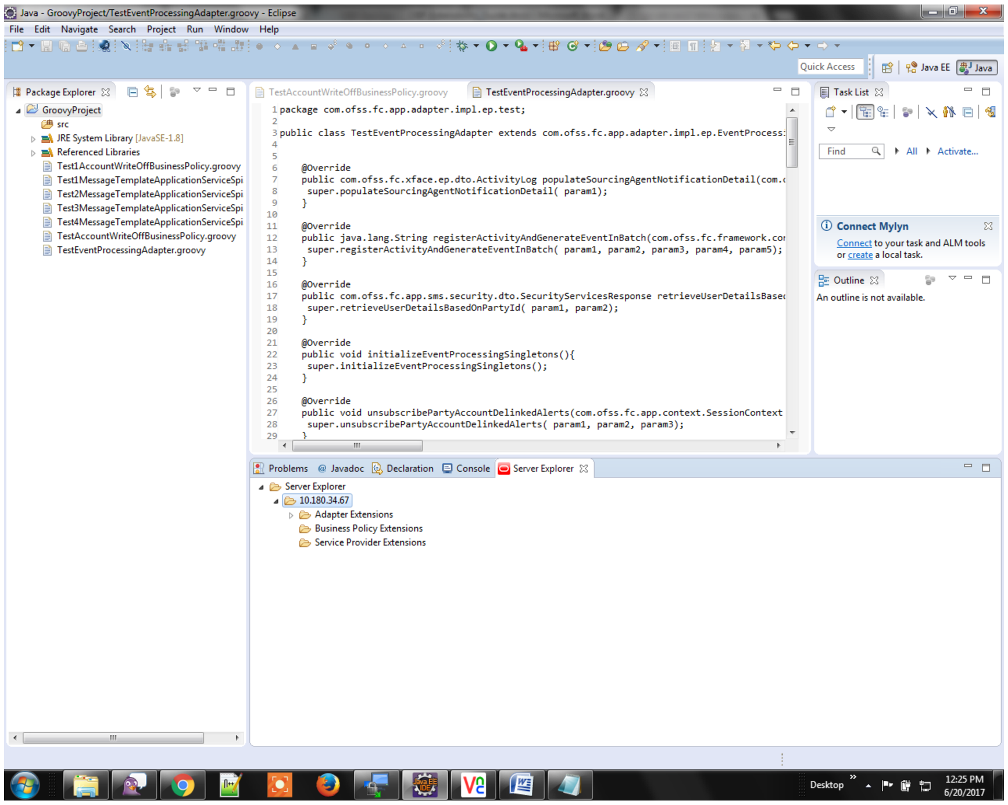
Figure 8–60 Deploy Business Policy Extension To Server


Java - GroovyProject/TestEventProcessingAdapter.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
TestEventProcessingAdapter.groovy
Problems Javadoc Declaration Console Server Explorer
Server Explorer
10.180.34.67
Adapter Extensions
Business Policy Extensions
Service Provider Extensions
Undo Ctrl+Z
Revert File
Save Ctrl+S
Open With
Show In Alt+Shift+W
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Quick Fix Ctrl+1
Shift Right
Shift Left
Add to Snippets...
Profile As
Debug As
Run As
Validate
Team
Compare With
Replace With
Acceleo
Oracle Banking Platform
Preferences...
Remove from Context Ctrl+Alt+Shift+Down
Deploy Service Provider Extension To Server
Deploy Business Policy Extension To Server
Deploy Adapter Extension To Server
Writable Insert 33 : 51

```
package com.ofss.fc.app.adapter.impl.ep.test;

public class TestEventProcessingAdapter extends com.ofss.fc.app.adapter.impl.ep.EventProcess:


    @Override
    public com.ofss.fc.xface.ep.dto.ActivityLog populateSourcingAgentNotificationDetail(com.c
     super.populateSourcingAgentNotificationDetail( param1);
    }

    @Override
    public java.lang.String registerActivityAndGenerateEventInBatch(com.ofss.fc.framework.cor
     super.registerActivityAndGenerateEventInBatch( param1, param2, param3, param4, param5);
    }

    @Override
    public com.ofss.fc.app.sms.security.dto.SecurityServicesRespor
     super.retrieveUserDetailsBasedOnPartyId( param1, param2);
    }

    @Override
    public void initializeEventProcessingSingletons(){
     super.initializeEventProcessingSingletons();
    }

    @Override
    public void unsubscribePartyAccountDelinkedAlerts(com.ofss.fc.
     super.unsubscribePartyAccountDelinkedAlerts( param1, param2,
    }
```

1. Right Click on the Extension Class in the Package Explorer or the Code Editor and click on Oracle Banking Platform > Deploy Business Policy Extension To Server.
2. It is compulsory to have selected the Server under Server Explorer in which the deployment has to occur.

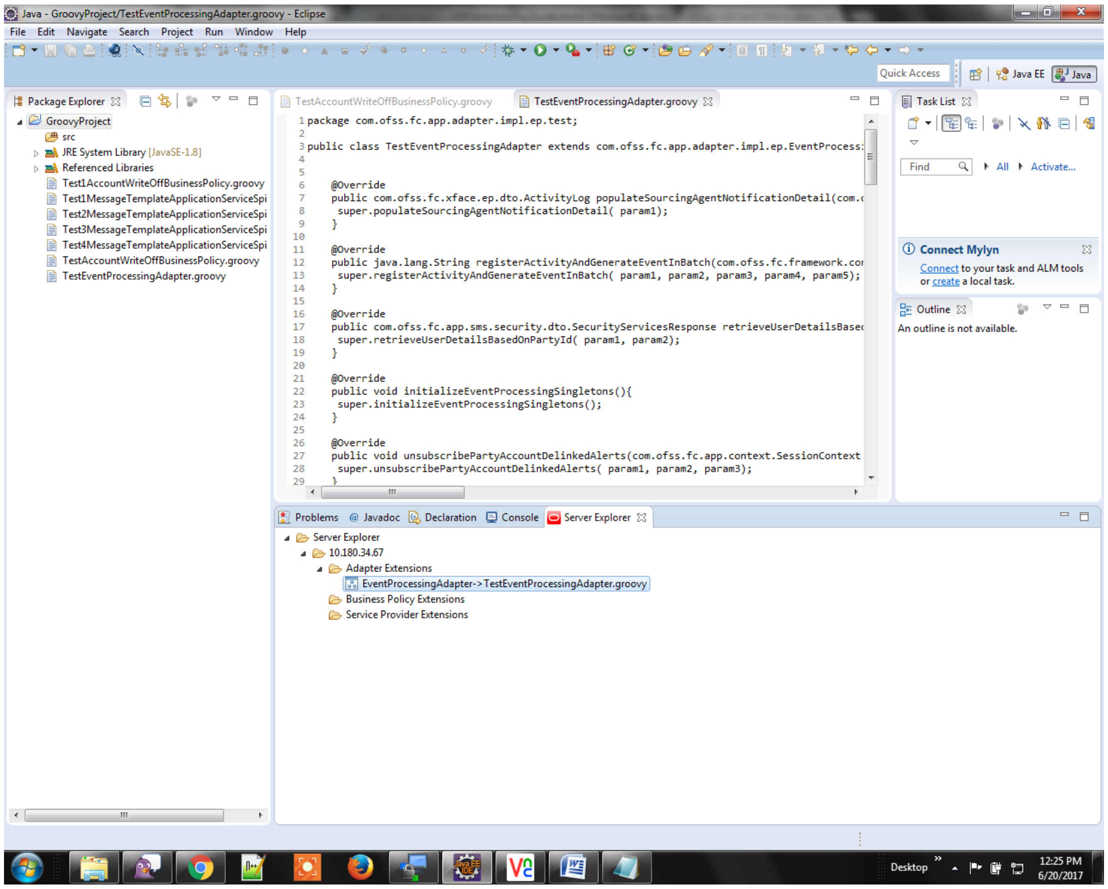
Figure 8–61 Select Server


Java - GroovyProject/TestEventProcessingAdapter.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
TestEventProcessingAdapter.groovy
TestAccountWriteOffBusinessPolicy.groovy
TestEventProcessingAdapter.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Server Explorer
10.180.34.67
Adapter Extensions
Business Policy Extensions
Service Provider Extensions
Desktop
12:25 PM
6/20/2017

```
package com.ofss.fc.app.adapter.impl.ep.test;

public class TestEventProcessingAdapter extends com.ofss.fc.app.adapter.impl.ep.EventProcess:


    @Override
    public com.ofss.fc.xface.ep.dto.ActivityLog populateSourcingAgentNotificationDetail(com.c
     super.populateSourcingAgentNotificationDetail( param1);
    }

    @Override
    public java.lang.String registerActivityAndGenerateEventInBatch(com.ofss.fc.framework.cor
     super.registerActivityAndGenerateEventInBatch( param1, param2, param3, param4, param5);
    }

    @Override
    public com.ofss.fc.app.sms.security.dto.SecurityServicesResponse retrieveUserDetailsBased
     super.retrieveUserDetailsBasedOnPartyId( param1, param2);
    }

    @Override
    public void initializeEventProcessingSingletons(){
     super.initializeEventProcessingSingletons();
    }

    @Override
    public void unsubscribePartyAccountDelinkedAlerts(com.ofss.fc.app.context.SessionContext
     super.unsubscribePartyAccountDelinkedAlerts( param1, param2, param3);
    }
```

3. The Extension gets deployed in the server.

Figure 8–62 Extension Deployed


Java - GroovyProject/TestEventProcessingAdapter.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
TestAccountWriteOffBusinessPolicy.groovy
TestEventProcessingAdapter.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Desktop
12:25 PM
6/20/2017

```
GroovyProject
    src
    JRE System Library [JavaSE-1.8]
    Referenced Libraries
    Test1AccountWriteOffBusinessPolicy.groovy
    Test1MessageTemplateApplicationServiceSpi
    Test2MessageTemplateApplicationServiceSpi
    Test3MessageTemplateApplicationServiceSpi
    Test4MessageTemplateApplicationServiceSpi
    TestAccountWriteOffBusinessPolicy.groovy
    TestEventProcessingAdapter.groovy
```

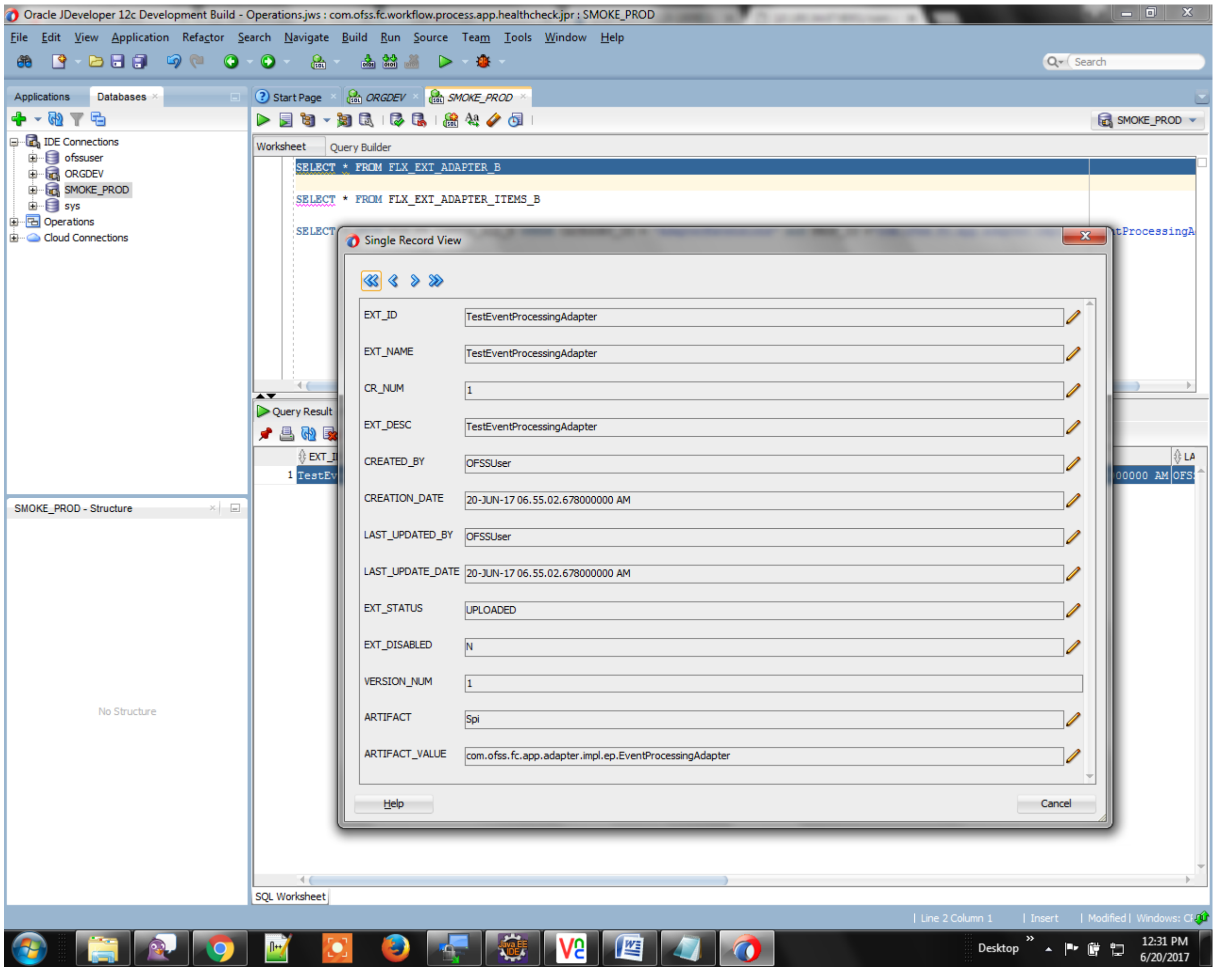
```
package com.ofss.fc.app.adapter.impl.ep.test;

public class TestEventProcessingAdapter extends com.ofss.fc.app.adapter.impl.ep.EventProcess


    @Override
    public com.ofss.fc.xface.ep.dto.ActivityLog populateSourcingAgentNotificationDetail(com.c
     super.populateSourcingAgentNotificationDetail( param1);
    }

    @Override
    public java.lang.String registerActivityAndGenerateEventInBatch(com.ofss.fc.framework.cor
     super.registerActivityAndGenerateEventInBatch( param1, param2, param3, param4, param5);
    }

    @Override
    public com.ofss.fc.app.sms.security.dto.SecurityServicesResponse retrieveUserDetailsBasec
     super.retrieveUserDetailsBasedOnPartyId( param1, param2);
    }

    @Override
    public void initializeEventProcessingSingletons(){
     super.initializeEventProcessingSingletons();
    }

    @Override
    public void unsubscribePartyAccountDelinkedAlerts(com.ofss.fc.app.context.SessionContext
     super.unsubscribePartyAccountDelinkedAlerts( param1, param2, param3);
    }
```

```
Server Explorer
    10.180.34.67
        Adapter Extensions
            EventProcessingAdapter->TestEventProcessingAdapter.groovy
        Business Policy Extensions
        Service Provider Extensions
```

4. The Extension gets deployed in the server and appears under the appropriate child of Server configured under Server Explorer.

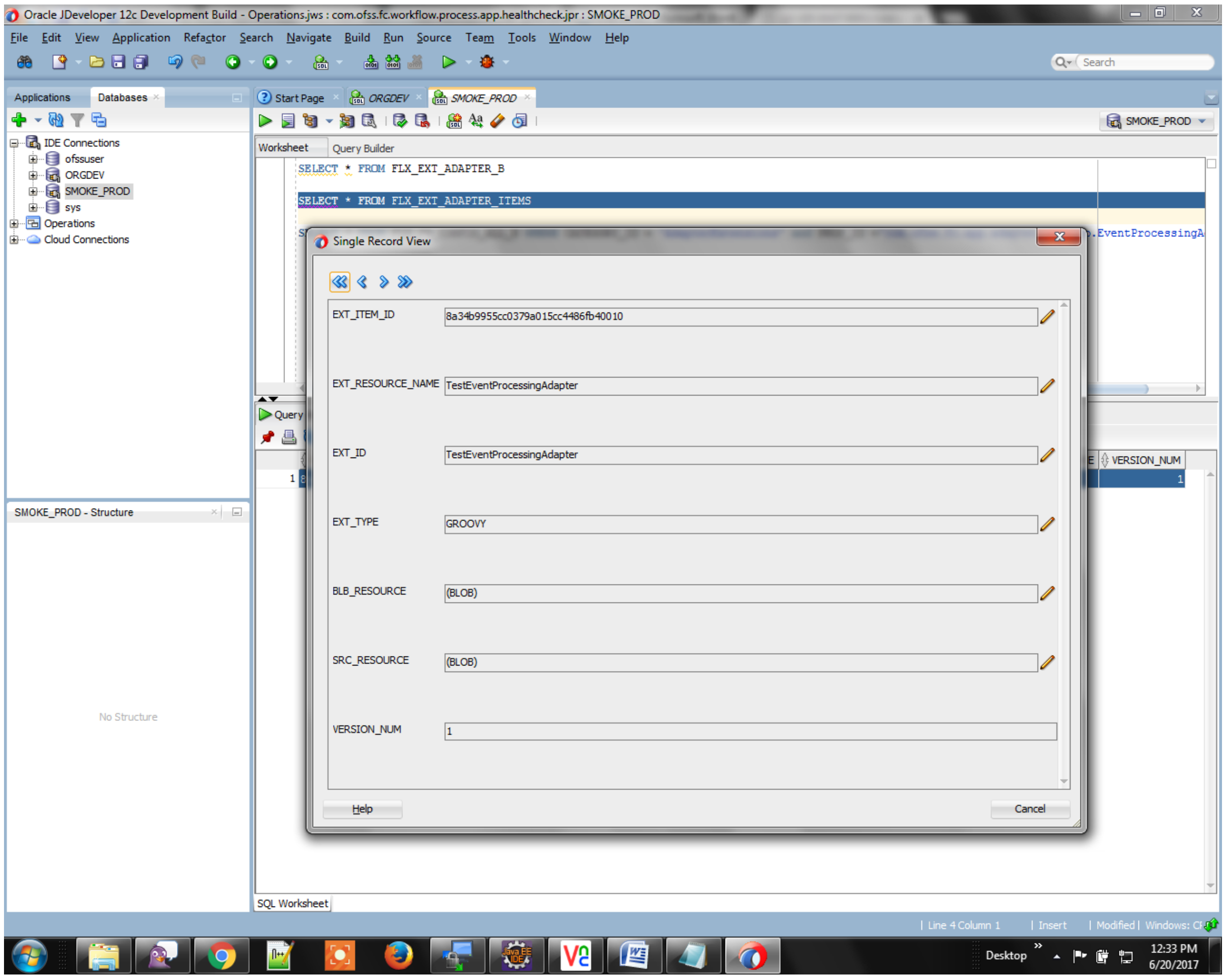
8.4.4 Database Inserts: Adapter Extension Deployment

Figure 8–63 Adapter Extension Deployment - Single Record View


```
SELECT * FROM FLX_EXT_ADAPTER_B

SELECT * FROM FLX_EXT_ADAPTER_ITEMS_B
```

EXT_ID	TestEventProcessingAdapter
EXT_NAME	TestEventProcessingAdapter
CR_NUM	1
EXT_DESC	TestEventProcessingAdapter
CREATED_BY	OFSSUser
CREATION_DATE	20-JUN-17 06.55.02.678000000 AM
LAST_UPDATED_BY	OFSSUser
LAST_UPDATE_DATE	20-JUN-17 06.55.02.678000000 AM
EXT_STATUS	UPLOADED
EXT_DISABLED	N
VERSION_NUM	1
ARTIFACT	Spi
ARTIFACT_VALUE	com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapter

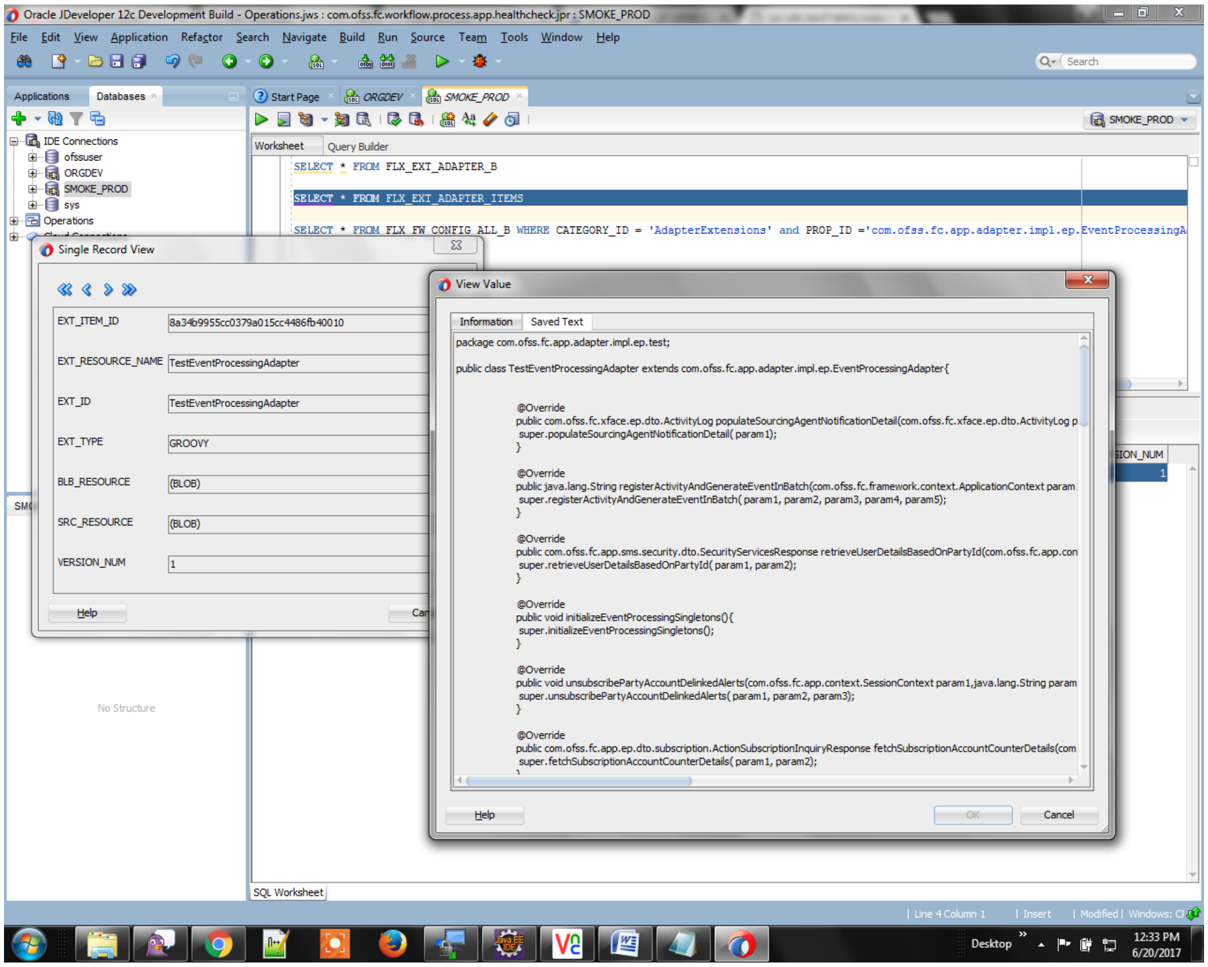
Figure 8–64 Adapter Extension Deployment - Single Record View



```
SELECT * FROM FLX_EXT_ADAPTER_B

SELECT * FROM FLX_EXT_ADAPTER_ITEMS
```

EXT_ITEM_ID	8a34b9955cc0379a015cc4486fb40010
EXT_RESOURCE_NAME	TestEventProcessingAdapter
EXT_ID	TestEventProcessingAdapter
EXT_TYPE	GROOVY
BLB_RESOURCE	(BLOB)
SRC_RESOURCE	(BLOB)
VERSION_NUM	1

Figure 8–65 Adapter Extension Deployment - View Value



```
SELECT * FROM FLX_EXT_ADAPTER_B

SELECT * FROM FLX_EXT_ADAPTER_ITEMS

SELECT * FROM FLX_FW_CONFIG_ALL_B WHERE CATEGORY_ID = 'AdapterExtensions' and PROP_ID ='com.ofss.fc.app.adapter.impl.ep.EventProcessingA
```

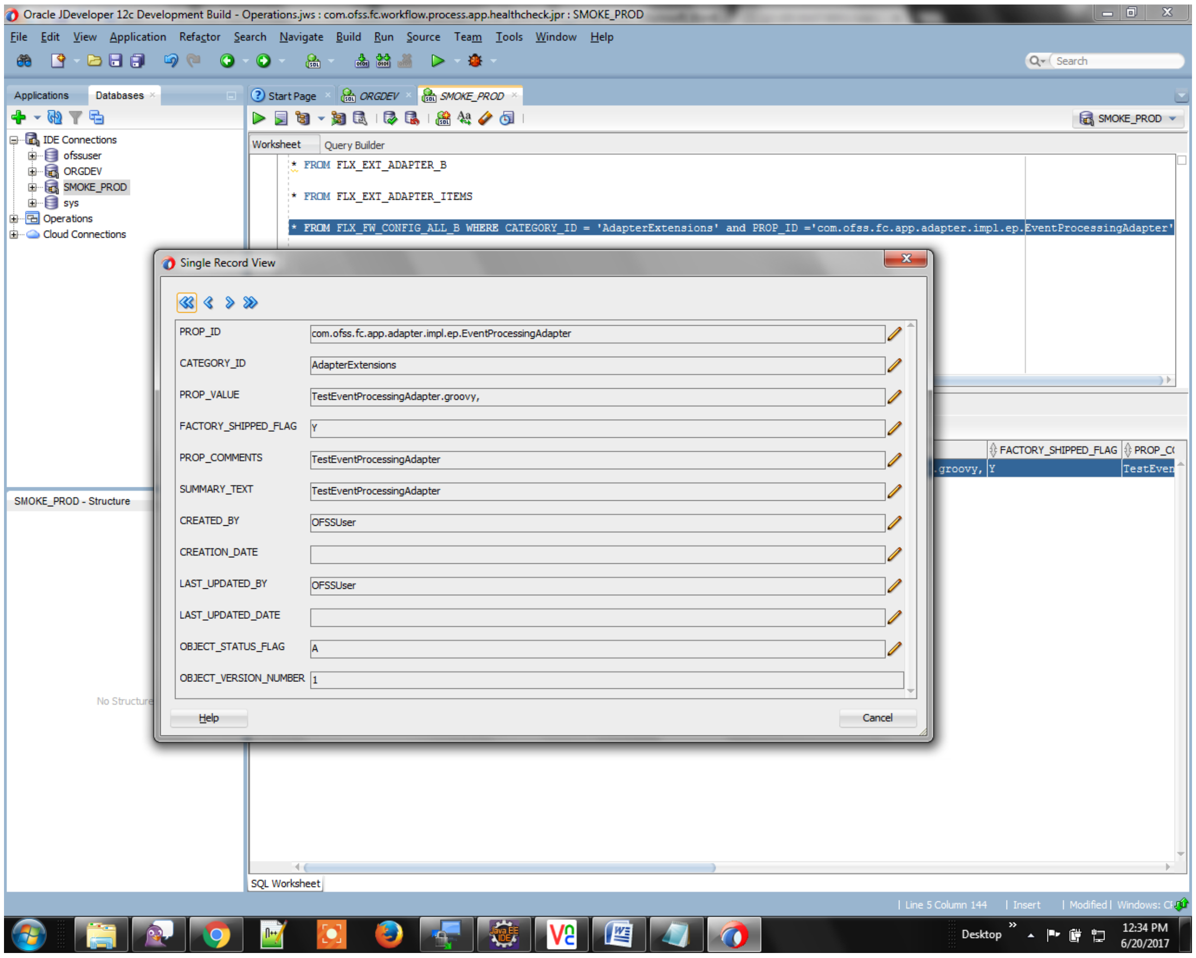
Single Record View

EXT_ITEM_ID	8a34b9955cc0379a015cc4486fb40010
EXT_RESOURCE_NAME	TestEventProcessingAdapter
EXT_ID	TestEventProcessingAdapter
EXT_TYPE	GROOVY
BLB_RESOURCE	(BLOB)
SRC_RESOURCE	(BLOB)
VERSION_NUM	1

View Value — Saved Text

```
package com.ofss.fc.app.adapter.impl.ep.test;

public class TestEventProcessingAdapter extends com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapter{


            @Override
            public com.ofss.fc.xface.ep.dto.ActivityLog populateSourcingAgentNotificationDetail(com.ofss.fc.xface.ep.dto.ActivityLog p
             super.populateSourcingAgentNotificationDetail( param1);
            }

            @Override
            public java.lang.String registerActivityAndGenerateEventInBatch(com.ofss.fc.framework.context.ApplicationContext param
             super.registerActivityAndGenerateEventInBatch( param1, param2, param3, param4, param5);
            }

            @Override
            public com.ofss.fc.app.sms.security.dto.SecurityServicesResponse retrieveUserDetailsBasedOnPartyId(com.ofss.fc.app.con
             super.retrieveUserDetailsBasedOnPartyId( param1, param2);
            }

            @Override
            public void initializeEventProcessingSingletons(){
             super.initializeEventProcessingSingletons();
            }

            @Override
            public void unsubscribePartyAccountDelinkedAlerts(com.ofss.fc.app.context.SessionContext param1,java.lang.String param
             super.unsubscribePartyAccountDelinkedAlerts( param1, param2, param3);
            }

            @Override
            public com.ofss.fc.app.ep.dto.subscription.ActionSubscriptionInquiryResponse fetchSubscriptionAccountCounterDetails(com
             super.fetchSubscriptionAccountCounterDetails( param1, param2);
```

Figure 8–66 Adapter Extension Deployment - Single Record View



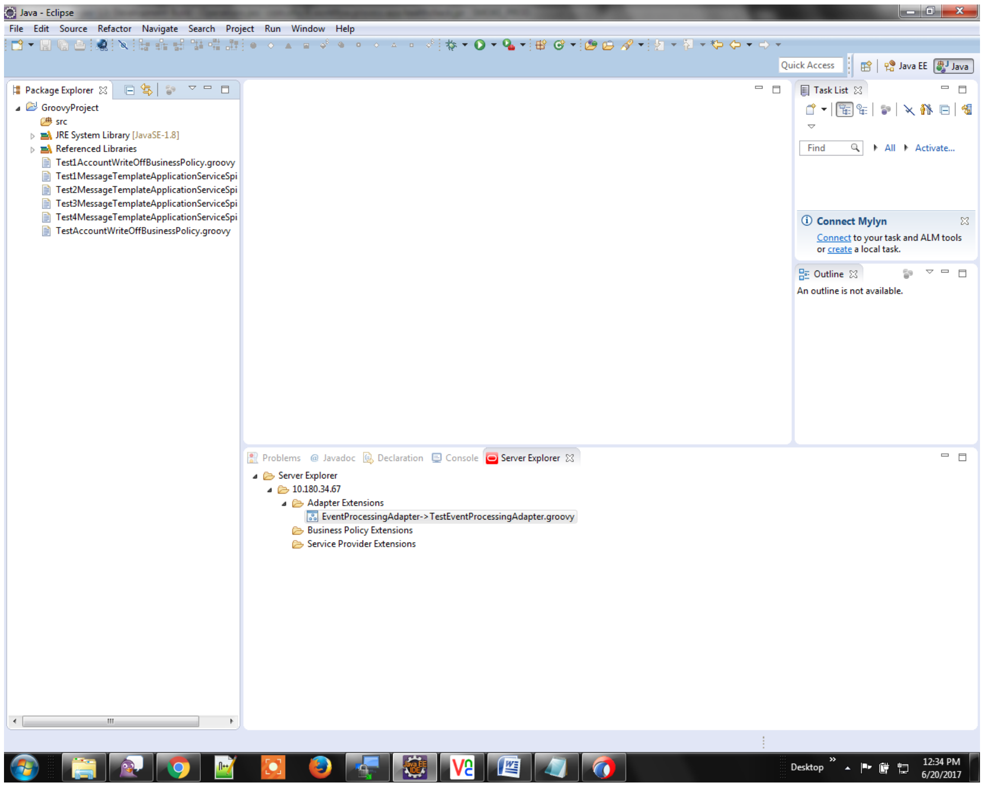
```
* FROM FLX_EXT_ADAPTER_B

* FROM FLX_EXT_ADAPTER_ITEMS

* FROM FLX_FW_CONFIG_ALL_B WHERE CATEGORY_ID = 'AdapterExtensions' and PROP_ID ='com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapter'
```

Single Record View

PROP_ID	com.ofss.fc.app.adapter.impl.ep.EventProcessingAdapter
CATEGORY_ID	AdapterExtensions
PROP_VALUE	TestEventProcessingAdapter.groovy,
FACTORY_SHIPPED_FLAG	Y
PROP_COMMENTS	TestEventProcessingAdapter
SUMMARY_TEXT	TestEventProcessingAdapter
CREATED_BY	OFSSUser
CREATION_DATE	
LAST_UPDATED_BY	OFSSUser
LAST_UPDATED_DATE	
OBJECT_STATUS_FLAG	A
OBJECT_VERSION_NUMBER	1

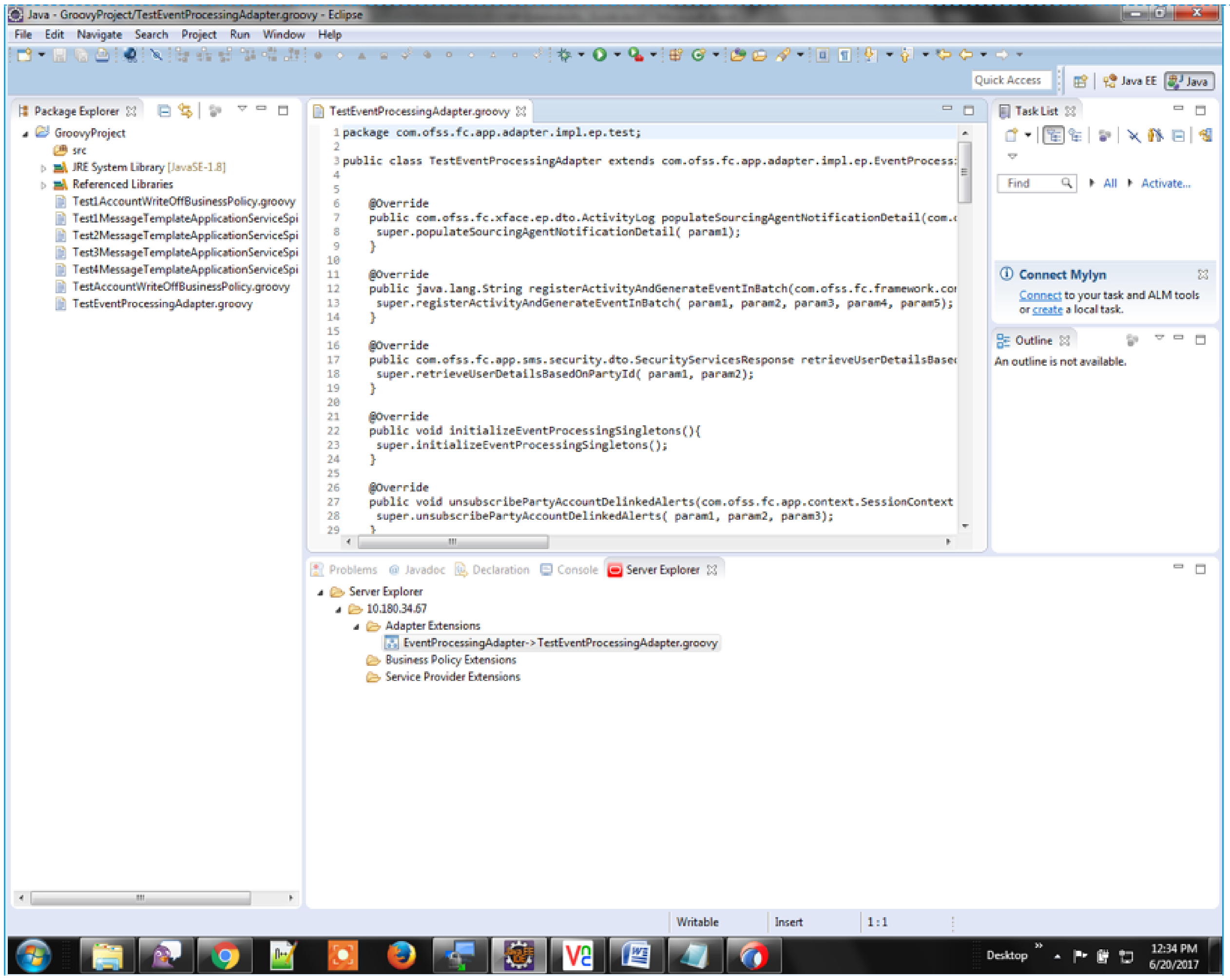
8.4.5 Fetching Deployed Adapter Extension

Figure 8–67 Fetching Deployed Adapter Extension


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Server Explorer
10.180.34.67
Adapter Extensions
EventProcessingAdapter->TestEventProcessingAdapter.groovy
Business Policy Extensions
Service Provider Extensions
Desktop
12:34 PM
6/20/2017

1. For the purpose of fetching a deployed Extension from the Server, you can click on the appropriate Extension under the appropriate child under Server Explorer.

Figure 8–68 Click Extension from Server



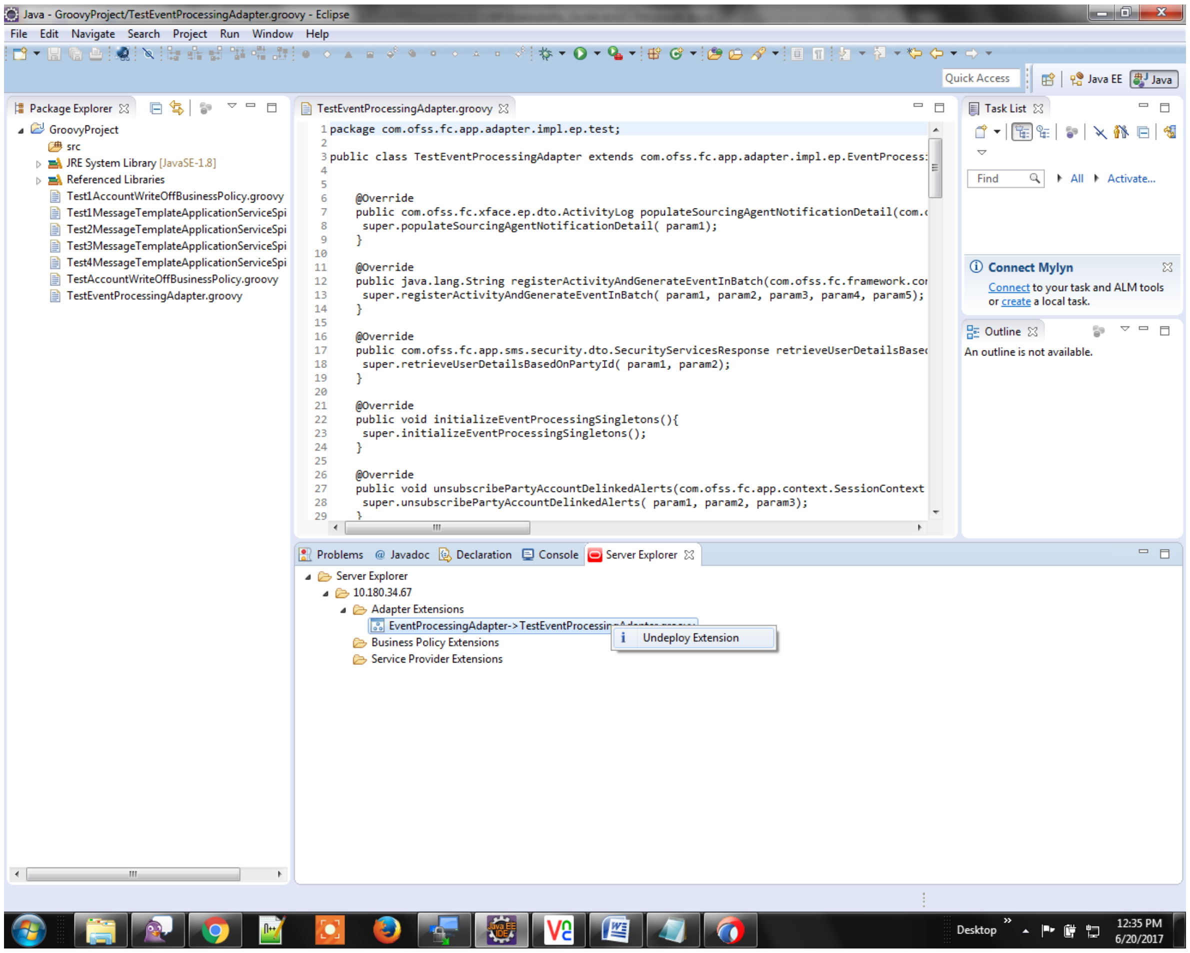
```
package com.ofss.fc.app.adapter.impl.ep.test;

public class TestEventProcessingAdapter extends com.ofss.fc.app.adapter.impl.ep.EventProcess:


    @Override
    public com.ofss.fc.xface.ep.dto.ActivityLog populateSourcingAgentNotificationDetail(com.c
     super.populateSourcingAgentNotificationDetail( param1);
    }

    @Override
    public java.lang.String registerActivityAndGenerateEventInBatch(com.ofss.fc.framework.cor
     super.registerActivityAndGenerateEventInBatch( param1, param2, param3, param4, param5);
    }

    @Override
    public com.ofss.fc.app.sms.security.dto.SecurityServicesResponse retrieveUserDetailsBase(
     super.retrieveUserDetailsBasedOnPartyId( param1, param2);
    }

    @Override
    public void initializeEventProcessingSingletons(){
     super.initializeEventProcessingSingletons();
    }

    @Override
    public void unsubscribePartyAccountDelinkedAlerts(com.ofss.fc.app.context.SessionContext
     super.unsubscribePartyAccountDelinkedAlerts( param1, param2, param3);
    }
```

Java - GroovyProject/TestEventProcessingAdapter.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
TestEventProcessingAdapter.groovy
TestEventProcessingAdapter.groovy
Task List
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems Javadoc Declaration Console Server Explorer
Server Explorer
10.180.34.67
Adapter Extensions
EventProcessingAdapter-> TestEventProcessingAdapter.groovy
Business Policy Extensions
Service Provider Extensions
Writable Insert 1 : 1

2. The extension gets saved in the project created to contain the Extension classes generated through the plugin. Also the Extension code opens up in the Code Editor.

8.4.6 Undeploying Adapter Extension from Server

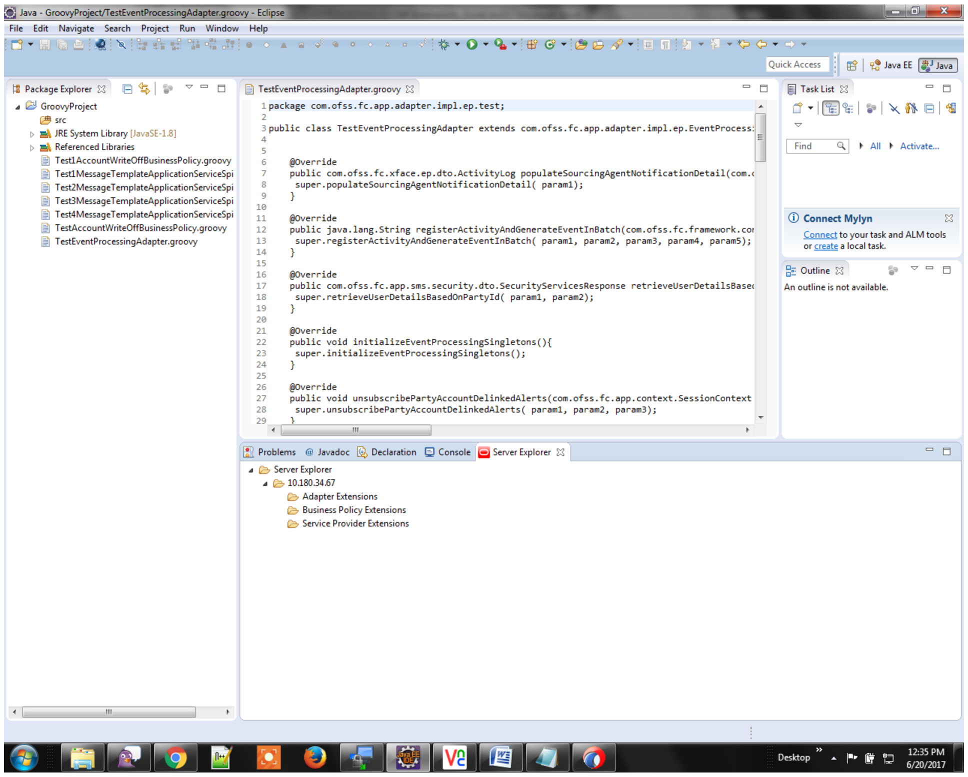
Figure 8–69 Undeploying Extension from Server


Java - GroovyProject/TestEventProcessingAdapter.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
TestEventProcessingAdapter.groovy
TestEventProcessingAdapter.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Server Explorer
10.180.34.67
Adapter Extensions
EventProcessingAdapter->TestEventProcessin
Undeploy Extension
Business Policy Extensions
Service Provider Extensions
Desktop
12:35 PM
6/20/2017

```
package com.ofss.fc.app.adapter.impl.ep.test;

public class TestEventProcessingAdapter extends com.ofss.fc.app.adapter.impl.ep.EventProcess:


    @Override
    public com.ofss.fc.xface.ep.dto.ActivityLog populateSourcingAgentNotificationDetail(com.c
     super.populateSourcingAgentNotificationDetail( param1);
    }

    @Override
    public java.lang.String registerActivityAndGenerateEventInBatch(com.ofss.fc.framework.cor
     super.registerActivityAndGenerateEventInBatch( param1, param2, param3, param4, param5);
    }

    @Override
    public com.ofss.fc.app.sms.security.dto.SecurityServicesResponse retrieveUserDetailsBasec
     super.retrieveUserDetailsBasedOnPartyId( param1, param2);
    }

    @Override
    public void initializeEventProcessingSingletons(){
     super.initializeEventProcessingSingletons();
    }

    @Override
    public void unsubscribePartyAccountDelinkedAlerts(com.ofss.fc.app.context.SessionContext
     super.unsubscribePartyAccountDelinkedAlerts( param1, param2, param3);
    }
```

1. For the purpose of undeploying the Extension from the Server, you right click on the specific Extension under the appropriate child of the Server configured under Server Explorer.

Figure 8–70 Extension Undeployed

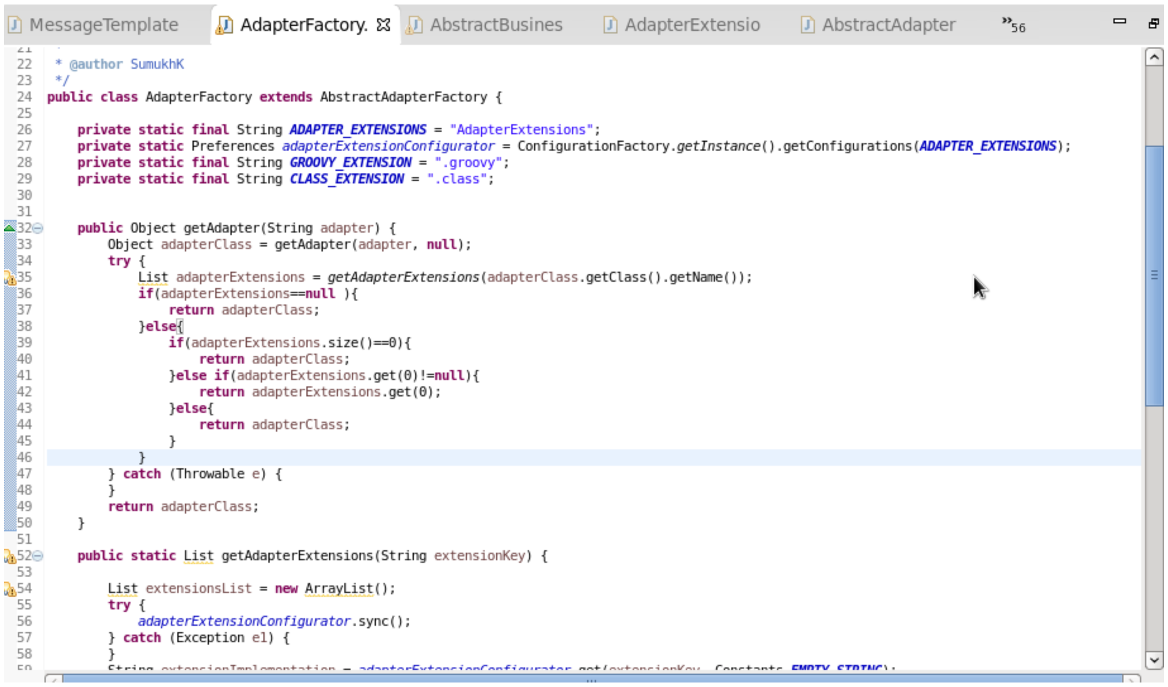

Java - GroovyProject/TestEventProcessingAdapter.groovy - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
GroovyProject
src
JRE System Library [JavaSE-1.8]
Referenced Libraries
Test1AccountWriteOffBusinessPolicy.groovy
Test1MessageTemplateApplicationServiceSpi
Test2MessageTemplateApplicationServiceSpi
Test3MessageTemplateApplicationServiceSpi
Test4MessageTemplateApplicationServiceSpi
TestAccountWriteOffBusinessPolicy.groovy
TestEventProcessingAdapter.groovy
TestEventProcessingAdapter.groovy
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Server Explorer
Server Explorer
10.180.34.67
Adapter Extensions
Business Policy Extensions
Service Provider Extensions
Desktop
12:35 PM
6/20/2017

```
package com.ofss.fc.app.adapter.impl.ep.test;

public class TestEventProcessingAdapter extends com.ofss.fc.app.adapter.impl.ep.EventProcess:


    @Override
    public com.ofss.fc.xface.ep.dto.ActivityLog populateSourcingAgentNotificationDetail(com.c
     super.populateSourcingAgentNotificationDetail( param1);
    }

    @Override
    public java.lang.String registerActivityAndGenerateEventInBatch(com.ofss.fc.framework.cor
     super.registerActivityAndGenerateEventInBatch( param1, param2, param3, param4, param5);
    }

    @Override
    public com.ofss.fc.app.sms.security.dto.SecurityServicesResponse retrieveUserDetailsBasec
     super.retrieveUserDetailsBasedOnPartyId( param1, param2);
    }

    @Override
    public void initializeEventProcessingSingletons(){
     super.initializeEventProcessingSingletons();
    }

    @Override
    public void unsubscribePartyAccountDelinkedAlerts(com.ofss.fc.app.context.SessionContext
     super.unsubscribePartyAccountDelinkedAlerts( param1, param2, param3);
    }
```

2. The specific extension under the appropriate child of the Server configured under “Server Explorer” disappears and gets undeployed from the Server.

8.4.7 Inclusion of Groovy Extension in Actual Code Flow

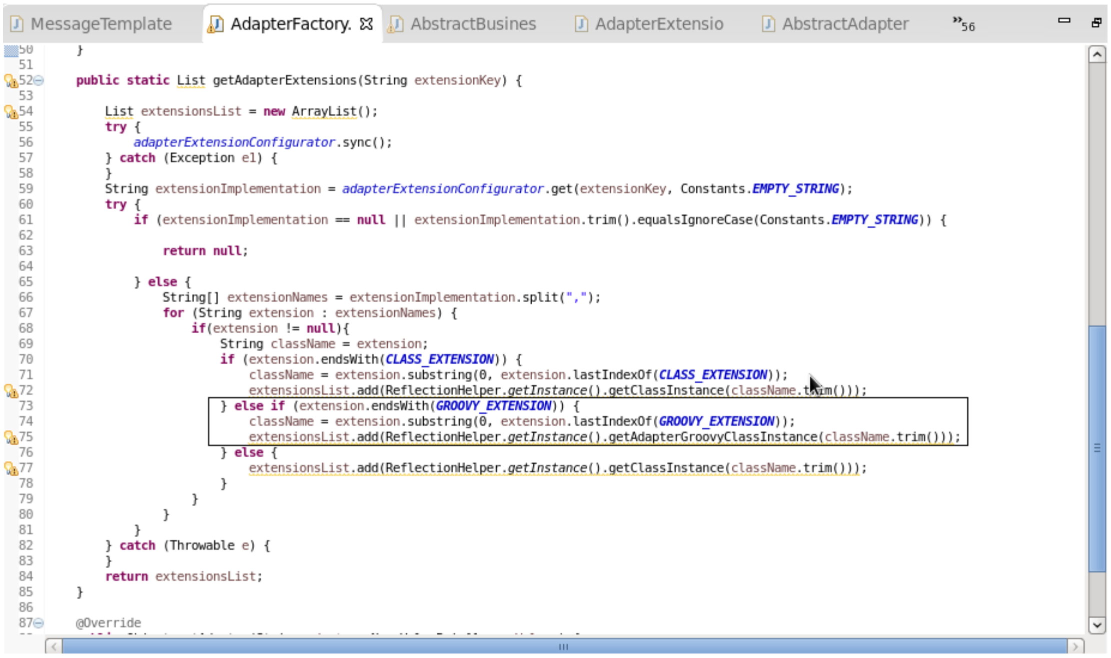
Figure 8–71 Groovy Extension in Code Flow



```
 * @author SumukhK
 */
public class AdapterFactory extends AbstractAdapterFactory {

    private static final String ADAPTER_EXTENSIONS = "AdapterExtensions";
    private static Preferences adapterExtensionConfigurator = ConfigurationFactory.getInstance().getConfigurations(ADAPTER_EXTENSIONS);
    private static final String GROOVY_EXTENSION = ".groovy";
    private static final String CLASS_EXTENSION = ".class";


    public Object getAdapter(String adapter) {
        Object adapterClass = getAdapter(adapter, null);
        try {
            List adapterExtensions = getAdapterExtensions(adapterClass.getClass().getName());
            if(adapterExtensions==null ){
                return adapterClass;
            }else{
                if(adapterExtensions.size()==0){
                    return adapterClass;
                }else if(adapterExtensions.get(0)!=null){
                    return adapterExtensions.get(0);
                }else{
                    return adapterClass;
                }
            }
        } catch (Throwable e) {
        }
        return adapterClass;
    }

    public static List getAdapterExtensions(String extensionKey) {

        List extensionsList = new ArrayList();
        try {
            adapterExtensionConfigurator.sync();
        } catch (Exception e1) {
        }
```

1. Whenever an AdapterFactory child class invokes getAdapter(String adapter) method the parent class' corresponding method gets invoked that deciphers the appropriate Adapter Class.

Figure 8–72 AdapterFactory



```
    }

    public static List getAdapterExtensions(String extensionKey) {

        List extensionsList = new ArrayList();
        try {
            adapterExtensionConfigurator.sync();
        } catch (Exception e1) {
        }
        String extensionImplementation = adapterExtensionConfigurator.get(extensionKey, Constants.EMPTY_STRING);
        try {
            if (extensionImplementation == null || extensionImplementation.trim().equalsIgnoreCase(Constants.EMPTY_STRING)) {

                return null;

            } else {
                String[] extensionNames = extensionImplementation.split(",");
                for (String extension : extensionNames) {
                    if(extension != null){
                        String className = extension;
                        if (extension.endsWith(CLASS_EXTENSION)) {
                            className = extension.substring(0, extension.lastIndexOf(CLASS_EXTENSION));
                            extensionsList.add(ReflectionHelper.getInstance().getClassInstance(className.trim()));
                        } else if (extension.endsWith(GROOVY_EXTENSION)) {
                            className = extension.substring(0, extension.lastIndexOf(GROOVY_EXTENSION));
                            extensionsList.add(ReflectionHelper.getInstance().getAdapterGroovyClassInstance(className.trim()));
                        } else {
                            extensionsList.add(ReflectionHelper.getInstance().getClassInstance(className.trim()));
                        }
                    }
                }
            }
        } catch (Throwable e) {
        }
        return extensionsList;
    }

    @Override
```

2. The name of this Adapter Class is then passed to “getAdapterExtensions(String extensionKey)” which then gets the extension name from FLX_FW_CONFIG_ALL_B based on the CATEGORY_ID=”AdapterExtensions” and PROP_ID=<Adapter Class Name>. This Extension is then used to get the Groovy Source code which is then parsed and compiled and returned as the appropriate Adapter Class.

9 Batch Framework Extensions

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

There are few primary categories in OBP Batch Processes:

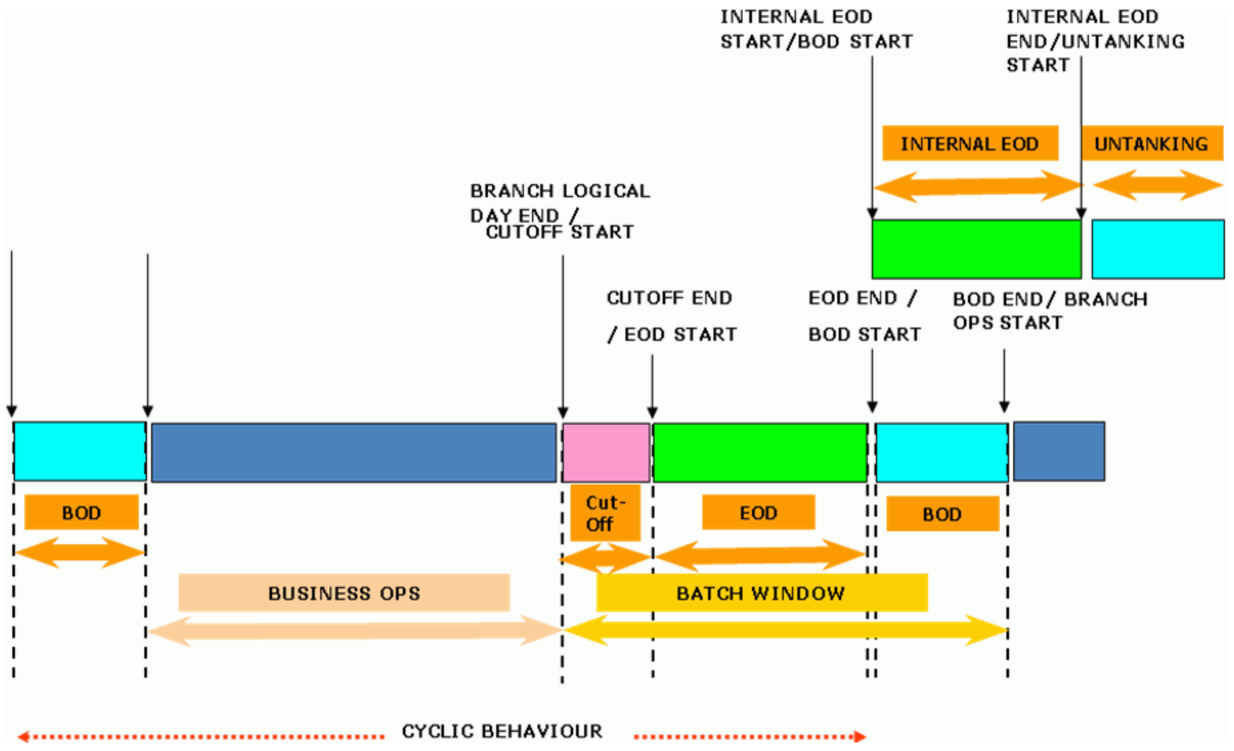
- Beginning of Day (BOD)
- Cut-off
- End of Day (EOD)
- Internal EOD
- Statement Generation
- Customer Communication

Additional categories can also be configured as per the requirement.

9.1 Typical Business Day in OBP

The following graphic describes a typical business day in OBP:

Figure 9–1 Business Day in OBP


INTERNAL EOD START/BOD START
INTERNAL EOD END/UNTANKING START
INTERNAL EOD
UNTANKING
BRANCH LOGICAL DAY END / CUTOFF START
CUTOFF END / EOD START
EOD END / BOD START
BOD END/ BRANCH OPS START
BOD
Cut-Off
EOD
BOD
BUSINESS OPS
BATCH WINDOW
CYCLIC BEHAVIOUR

9.2 Overview of Categories

This topic describes the categories in OBP Batch Processes.

9.2.1 Beginning of Day (BOD)

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

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

9.2.2 Cut-off

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

It also marks cut-off for the channel transactions.

9.2.3 End of Day (EOD)

Once all the operations for the current working date of the branch are completed and all the transactions are posted the Branch EOD batch is started. This batch executes a group of shells (programs) which are required to be performed before the Business Date of the branch is changed to the next working date. It marks the end of a business day. The EOD typically includes:

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

9.2.4 Internal EOD

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

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

9.2.5 Statement Generation

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

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

9.2.6 Customer Communication

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

- Regular Account Balance Notification On Specified Date

9.3 Batch Framework Architecture

This section describes the architecture of the Batch Framework.

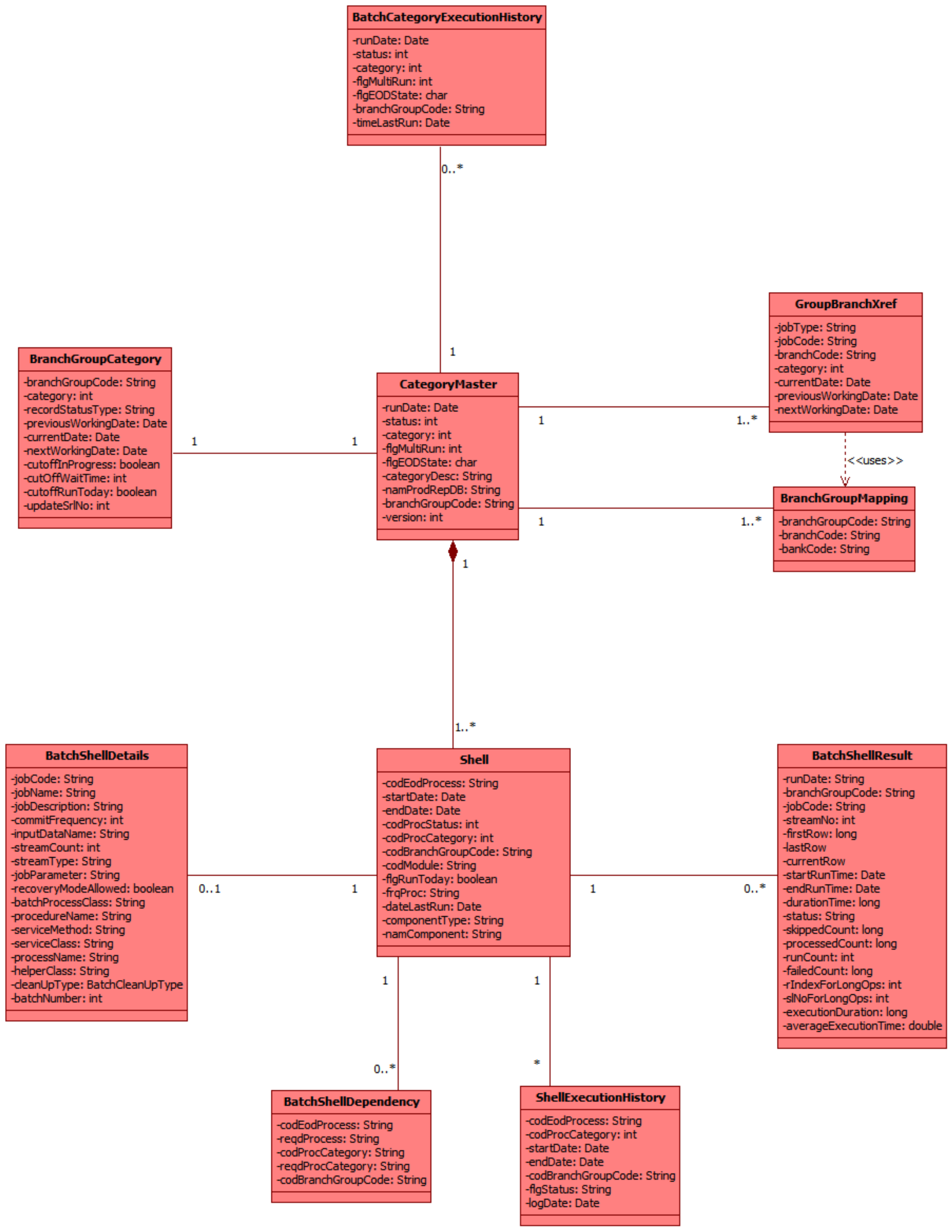
9.3.1 Static View

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

Class Diagram

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

Figure 9–2 Batch Framework Architecture - Static View



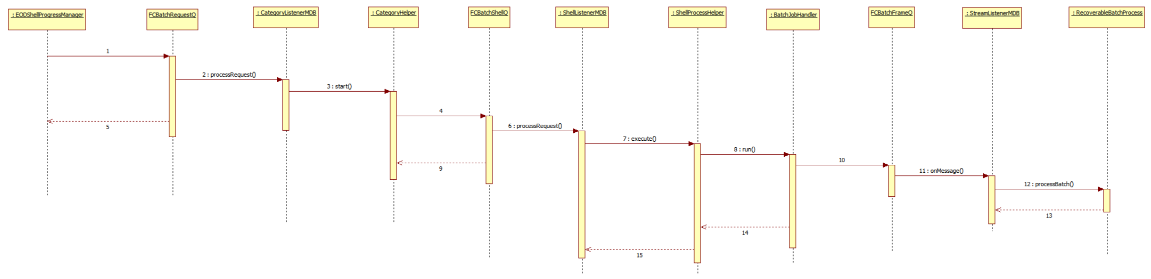
9.3.2 Dynamic View

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

Sequence Diagram

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

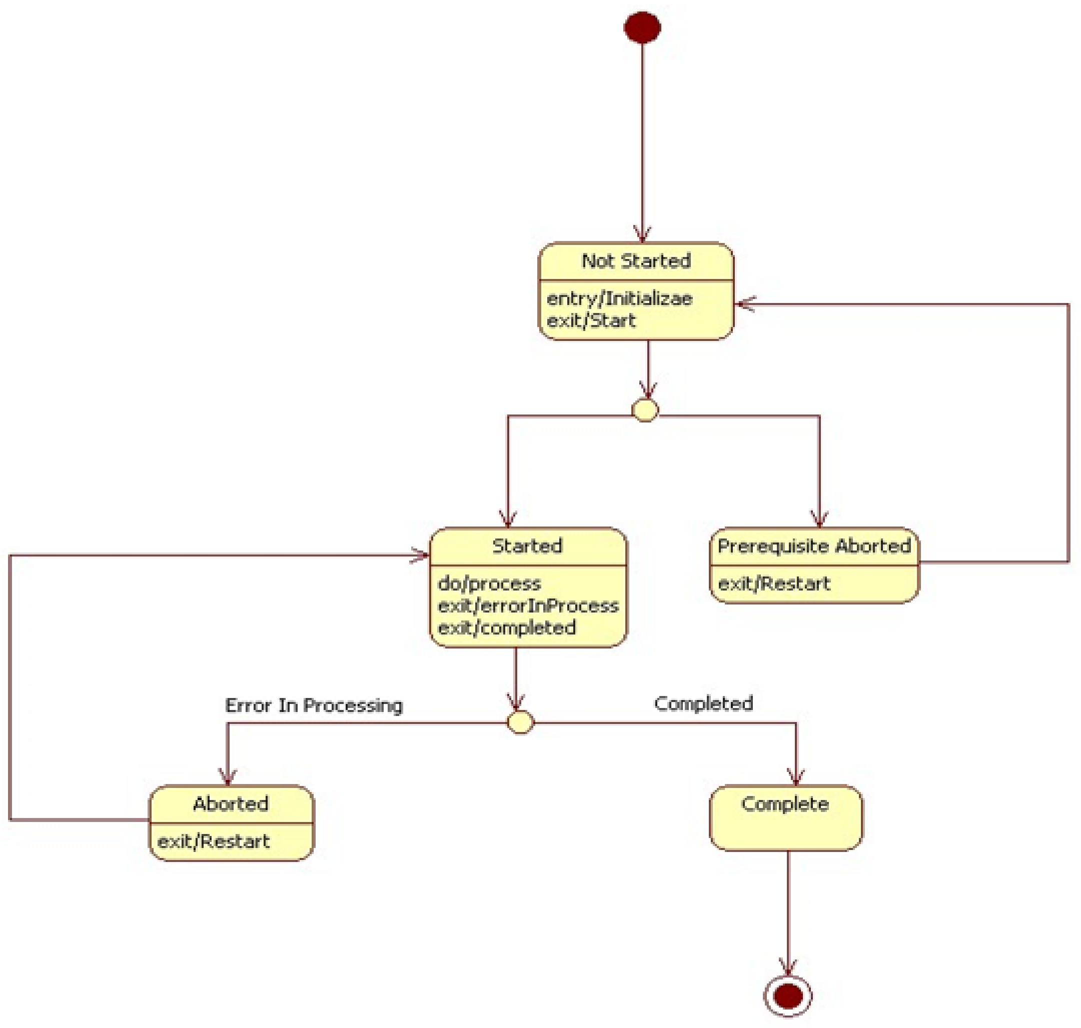
Figure 9–3 Dynamic View Sequence Diagram



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 9–4 State Diagram of a Shell



9.4 Batch Framework Components

This section describes the batch framework components.

9.4.1 Category Components

This section describes the category components.

CategoryListenerMDB

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

CategoryHelper

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

It will validate the input xml Request, validate the prerequisites for starting/restarting a category, get the list of shells that can be initiated on a category start/shell completion, prepare the Batch XML Message for each of the shell and send a message to FCBBatchShellQ for each Shell to be started.

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

9.4.2 Shell Components

This section describes the shell components.

ShellListenerMDB

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

ShellProcessHelper

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

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

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

ShellRootHelper

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

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

BatchFrameworkShellHelper

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

BatchJobHandler

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

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

BatchProcessManager

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

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

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

BatchJobHelper

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

ProcedureShellHelper

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

BatchReportShellBean

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

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

BatchReportRestartShellBean

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

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

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

9.4.3 Stream Components

This section describes the stream components.

StreamListenerMDB

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

StreamProcessHelper

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

BatchProcess

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

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

RecoverableBatchProcess

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

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

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

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

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

- Recoverable Batch Process with Recovery Mode ON: When a record fails, the previous records in the commit size will be committed and marked as success, the failed record will be marked as failed and the execution of batch process resumes from the record after the failed record. Hence in this mode all the successful records are committed and the failed records are marked as failed.
- Recoverable Batch Process with Recovery Mode OFF: In this mode, when a record fails the earlier records in the commit size are marked as skipped for the current run, the failed record is marked as failed and execution of batch process resumes from the record after the failed record.

Simple Batch Process

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

SimpleBatchProcess class is no longer used

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

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

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.

9.4.4 Database Components

The Database Server houses the following components:

Table 9–1 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.

9.5 Batch Configuration

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

9.5.1 Creation of New Category

The following steps explain the creation of new category:

1. Create an entry in **flx_batch_job_category_master**:

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

Table 9–2 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 9–3 FLX_BATCH_JOB_GRP_CATEGORY

Columns	**Description**
BRANCH_GROUP_ CODE	This column specifies the Branch Group Code.
COD_PROC_ CATEGORY	This column specifies the procedure category.
DAT_LAST_ PROCESS	This column specifies the date on which the category was last run.
DAT_PROCESS	This column specifies the current date of the category.
DAT_NEXT_ PROCESS	This column specifies the next date of the category.

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

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

Table 9–4 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.

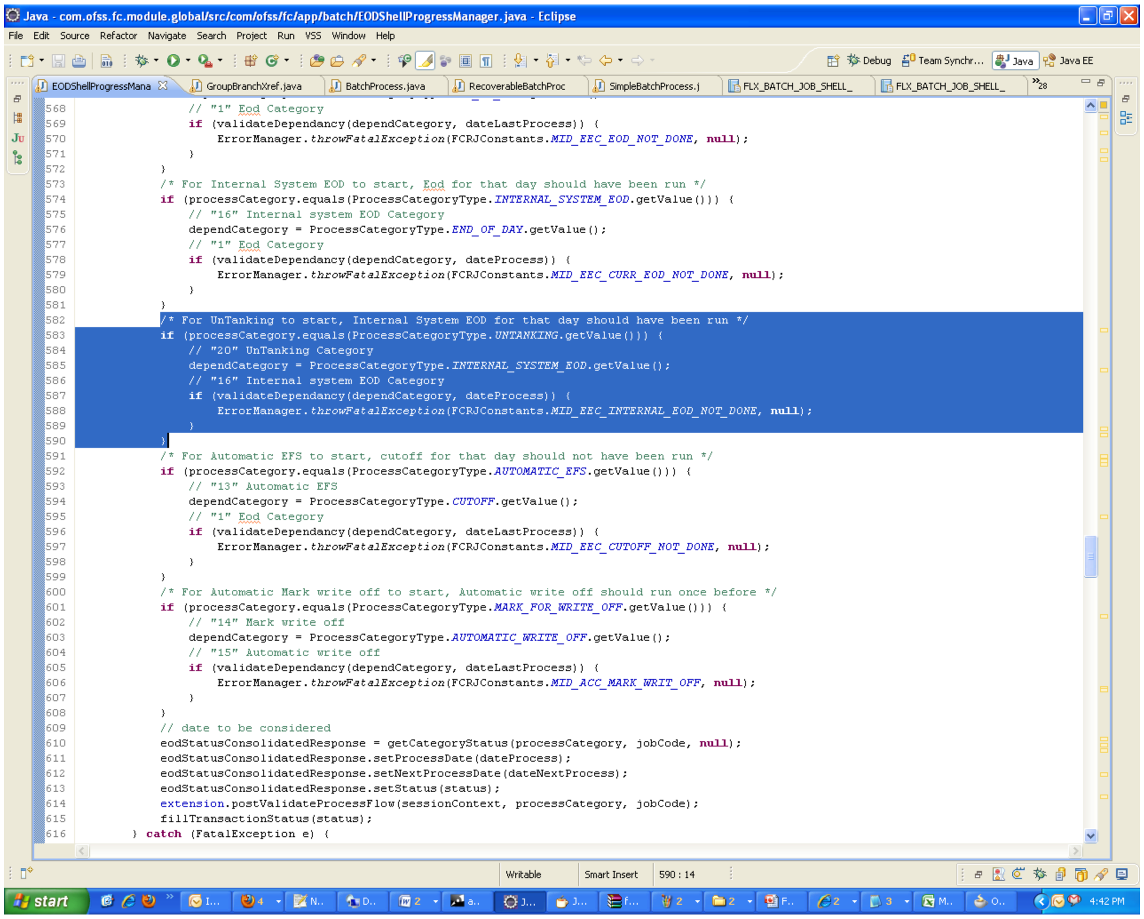
6. Add category code in the property file:

   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 9–5 Creation of New Category



```
                // "1" Eod Category
                if (validateDependancy(dependCategory, dateLastProcess)) {
                    ErrorManager.throwFatalException(FCRJConstants.MID_EEC_EOD_NOT_DONE, null);
                }
            }
            /* For Internal System EOD to start, Eod for that day should have been run */
            if (processCategory.equals(ProcessCategoryType.INTERNAL_SYSTEM_EOD.getValue())) {
                // "16" Internal system EOD Category
                dependCategory = ProcessCategoryType.END_OF_DAY.getValue();
                // "1" Eod Category
                if (validateDependancy(dependCategory, dateProcess)) {
                    ErrorManager.throwFatalException(FCRJConstants.MID_EEC_CURR_EOD_NOT_DONE, null);
                }
            }
            /* For UnTanking to start, Internal System EOD for that day should have been run */
            if (processCategory.equals(ProcessCategoryType.UNTANKING.getValue())) {
                // "20" UnTanking Category
                dependCategory = ProcessCategoryType.INTERNAL_SYSTEM_EOD.getValue();
                // "16" Internal system EOD Category
                if (validateDependancy(dependCategory, dateProcess)) {
                    ErrorManager.throwFatalException(FCRJConstants.MID_EEC_INTERNAL_EOD_NOT_DONE, null);
                }
            }
            /* For Automatic EFS to start, cutoff for that day should not have been run */
            if (processCategory.equals(ProcessCategoryType.AUTOMATIC_EFS.getValue())) {
                // "13" Automatic EFS
                dependCategory = ProcessCategoryType.CUTOFF.getValue();
                // "1" Eod Category
                if (validateDependancy(dependCategory, dateLastProcess)) {
                    ErrorManager.throwFatalException(FCRJConstants.MID_EEC_CUTOFF_NOT_DONE, null);
                }
            }
            /* For Automatic Mark write off to start, Automatic write off should run once before */
            if (processCategory.equals(ProcessCategoryType.MARK_FOR_WRITE_OFF.getValue())) {
                // "14" Mark write off
                dependCategory = ProcessCategoryType.AUTOMATIC_WRITE_OFF.getValue();
                // "15" Automatic write off
                if (validateDependancy(dependCategory, dateLastProcess)) {
                    ErrorManager.throwFatalException(FCRJConstants.MID_ACC_MARK_WRIT_OFF, null);
                }
            }
            // date to be considered
            eodStatusConsolidatedResponse = getCategoryStatus(processCategory, jobCode, null);
            eodStatusConsolidatedResponse.setProcessDate(dateProcess);
            eodStatusConsolidatedResponse.setNextProcessDate(dateNextProcess);
            eodStatusConsolidatedResponse.setStatus(status);
            extension.postValidateProcessFlow(sessionContext, processCategory, jobCode);
            fillTransactionStatus(status);
        } catch (FatalException e) {
```

9.5.2 Creation of Bean Based Shell

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

1. Create an entry for Shell Parameters in the table **FLX_BATCH_JOB_SHELL_MASTER**.

Table 9–5 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 9–6 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 9–7 FLX_BATCH_JOB_SHELL_DEPEND

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

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

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

Table 9–8 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 9–9 Actions Table

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

6. Create a procedure (the name of the proc prefixed with **ap_<Module Code>_pop_drv)** which would populate the data in the driver table, created above. This procedure would be called at the first time when the Batch shell is run. The procedure will have only three arguments branch group code, process date and next process date. 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.

9.5.3 Creation of Procedure Based Shell

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

1. Create an entry for **Shell Parameters** in the table **FLX_BATCH_JOB_SHELL_MASTER**. Same as described in the above section.
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);
```

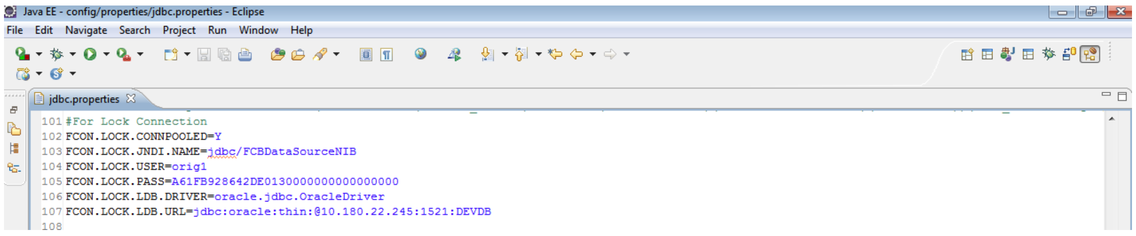
```
        RETURN 95;
        END;/
```

9.5.4 Population of Other Parameters

The following procedures describe the population of other parameters:

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

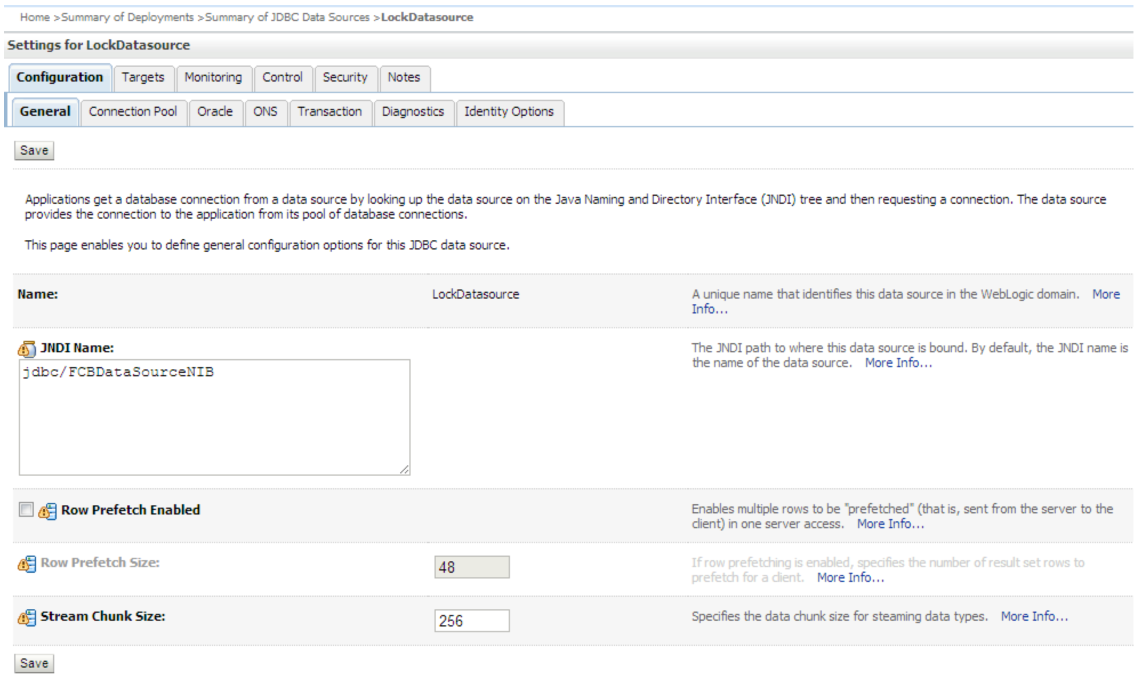
Figure 9–6 Population of Other Parameters


Java EE - config/properties/jdbc.properties - Eclipse
File Edit Navigate Search Project Run Window Help
jdbc.properties

```
#For Lock Connection
FCON.LOCK.CONNPOOLED=Y
FCON.LOCK.JNDI.NAME=jdbc/FCBDataSourceNIB
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
```

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

Figure 9–7 Population of Other Parameters - General Tab



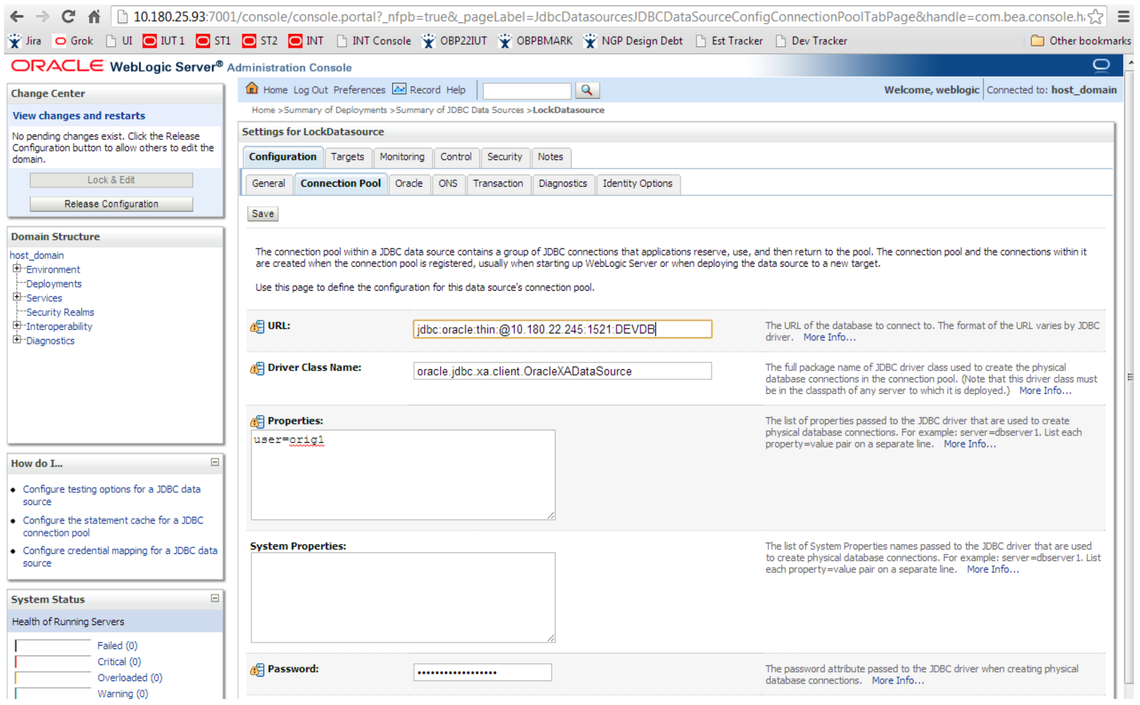
Settings for LockDatasource

Applications get a database connection from a data source by looking up the data source on the Java Naming and Directory Interface (JNDI) tree and then requesting a connection. The data source provides the connection to the application from its pool of database connections.

This page enables you to define general configuration options for this JDBC data source.

Name:	LockDatasource	A unique name that identifies this data source in the WebLogic domain. More Info...
JNDI Name:	jdbc/FCBDataSourceNIB	The JNDI path to where this data source is bound. By default, the JNDI name is the name of the data source. More Info...
☐ Row Prefetch Enabled		Enables multiple rows to be "prefetched" (that is, sent from the server to the client) in one server access. More Info...
Row Prefetch Size:	48	If row prefetching is enabled, specifies the number of result set rows to prefetch for a client. More Info...
Stream Chunk Size:	256	Specifies the data chunk size for steaming data types. More Info...

Figure 9–8 Population of Other Parameters - Connection Pool


10.180.25.93:7001/console/console.portal?_nfpb=true&_pageLabel=JdbcDatasourcesJDBCDataSourceConfigConnectionPoolTabPage&handle=com.bea.console.h
Jira
Grok
UI
IUT 1
ST1
ST2
INT
INT Console
OBP22IUT
OBPBMARK
NGP Design Debt
Est Tracker
Dev Tracker
Other bookmarks
ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic
Connected to: host_domain
Home >Summary of Deployments >Summary of JDBC Data Sources >LockDatasource
Change Center
View changes and restarts
Lock & Edit
Release Configuration
Domain Structure
host_domain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
Configure testing options for a JDBC data source
Configure the statement cache for a JDBC connection pool
Configure credential mapping for a JDBC data source
System Status
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
Settings for LockDatasource
Configuration
Targets
Monitoring
Control
Security
Notes
General
Connection Pool
Oracle
ONS
Transaction
Diagnostics
Identity Options
Save

No pending changes exist. Click the Release Configuration button to allow others to edit the domain.

The connection pool within a JDBC data source contains a group of JDBC connections that applications reserve, use, and then return to the pool. The connection pool and the connections within it are created when the connection pool is registered, usually when starting up WebLogic Server or when deploying the data source to a new target.

Use this page to define the configuration for this data source's connection pool.

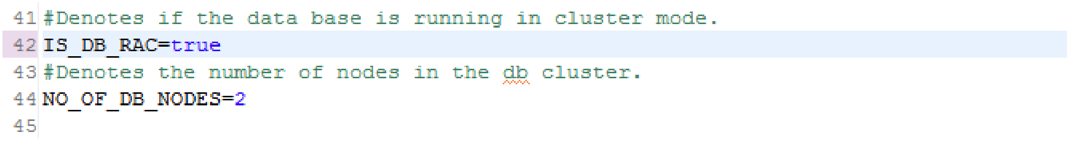
URL:	jdbc:oracle:thin:@10.180.22.245:1521:DEVDB	The URL of the database to connect to. The format of the URL varies by JDBC driver. More Info...
Driver Class Name:	oracle.jdbc.xa.client.OracleXADataSource	The full package name of JDBC driver class used to create the physical database connections in the connection pool. (Note that this driver class must be in the classpath of any server to which it is deployed.) More Info...
Properties:	user=orig1	The list of properties passed to the JDBC driver that are used to create physical database connections. For example: server=dbserver1. List each property=value pair on a separate line. More Info...
System Properties:		The list of System Properties names passed to the JDBC driver that are used to create physical database connections. For example: server=dbserver1. List each property=value pair on a separate line. More Info...
Password:	••••••••••••••••••	The password attribute passed to the JDBC driver when creating physical database connections. More Info...

3. Enable Node Affinity for Batch Processing (Optional)

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

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

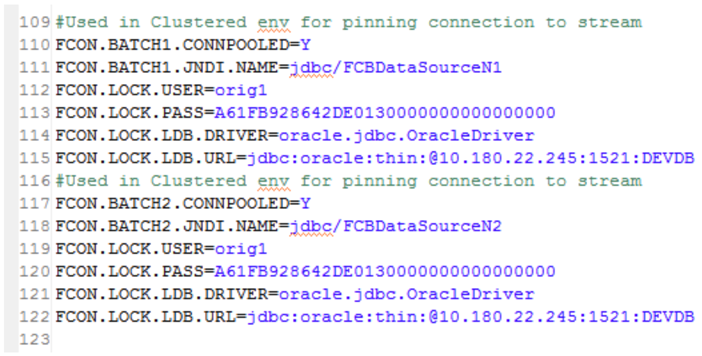
Figure 9–9 Population of Other Parameters - Set IS_DB_RAC



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

```

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

Figure 9–10 Population of Other Parameters - Specify Data



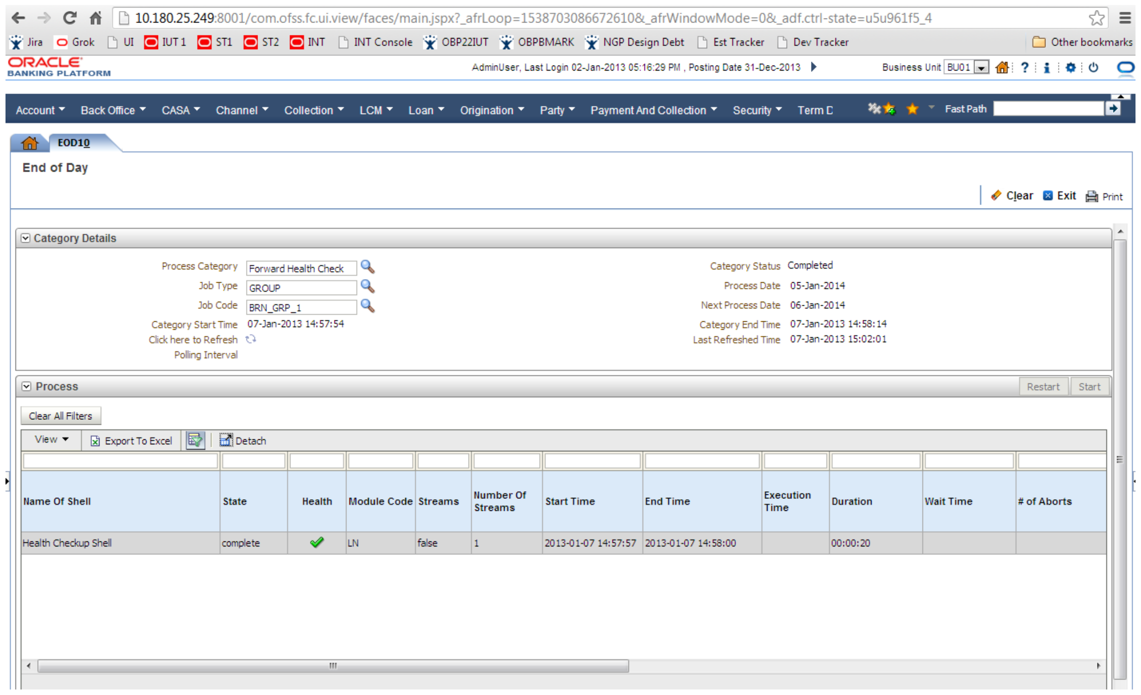
```
#Used in Clustered env for pinning connection to stream
FCON.BATCH1.CONNPOOLED=Y
FCON.BATCH1.JNDI.NAME=jdbc/FCBDataSourceN1
FCON.LOCK.USER=orig1
FCON.LOCK.PASS=A61FB928642DE0130000000000000000
FCON.LOCK.LDB.DRIVER=oracle.jdbc.OracleDriver
FCON.LOCK.LDB.URL=jdbc:oracle:thin:@10.180.22.245:1521:DEVDB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

9.6 Batch Execution

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

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

Figure 9–11 Batch Execution



Process Category	Forward Health Check	Category Status	Completed
Job Type	GROUP	Process Date	05-Jan-2014
Job Code	BRN_GRP_1	Next Process Date	06-Jan-2014
Category Start Time	07-Jan-2013 14:57:54	Category End Time	07-Jan-2013 14:58:14
Click here to Refresh		Last Refreshed Time	07-Jan-2013 15:02:01
Polling Interval			

Name Of Shell	State	Health	Module Code	Streams	Number Of Streams	Start Time	End Time	Execution Time	Duration	Wait Time	# of Aborts
Health Checkup Shell	complete	✓	LN	false	1	2013-01-07 14:57:57	2013-01-07 14:58:00		00:00:20		

10 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.

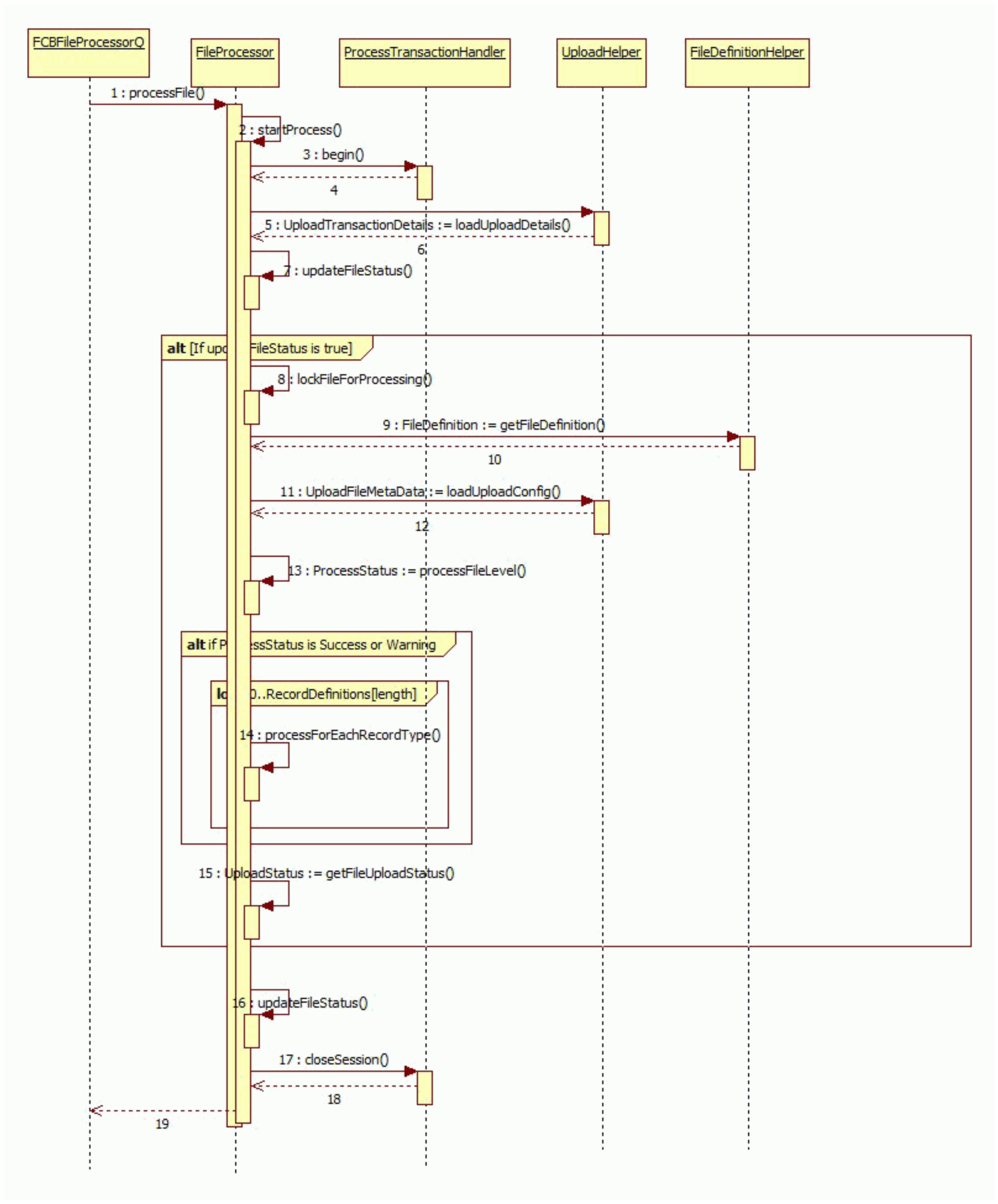
In OBP, file upload and file processing are two independent processes where the upload of file is done as one process and the processing on the uploaded data is done as another process. Every upload provides a unique file ID for the uploaded file. The processing is then done for each uploaded file and 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 10–1 Uploaded Data File Processing Framework


FCBFileProcessorQ
FileProcessor
ProcessTransactionHandler
UploadHelper
FileDefinitionHelper
1 : processFile()
2 : startProcess()
3 : begin()
4
5 : UploadTransactionDetails := loadUploadDetails()
6
7 : updateFileStatus()
alt [If updateFileStatus is true]
8 : lockFileForProcessing()
9 : FileDefinition := getFileDefinition()
10
11 : UploadFileMetaData := loadUploadConfig()
12
13 : ProcessStatus := processFileLevel()
alt if ProcessStatus is Success or Warning
loop 0..RecordDefinitions[length]
14 : processForEachRecordType()
15 : UploadStatus := getFileUploadStatus()
16 : updateFileStatus()
17 : closeSession()
18
19

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

10.1 Configuration

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

10.1.1 Database Tables and Setup

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

- FLX_EXT_FILE_UPLOAD_MAST

Table 10–1 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 10–2 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 10–3 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 10–4 FLX_BATCH_JOB_SHELL_DTLS

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

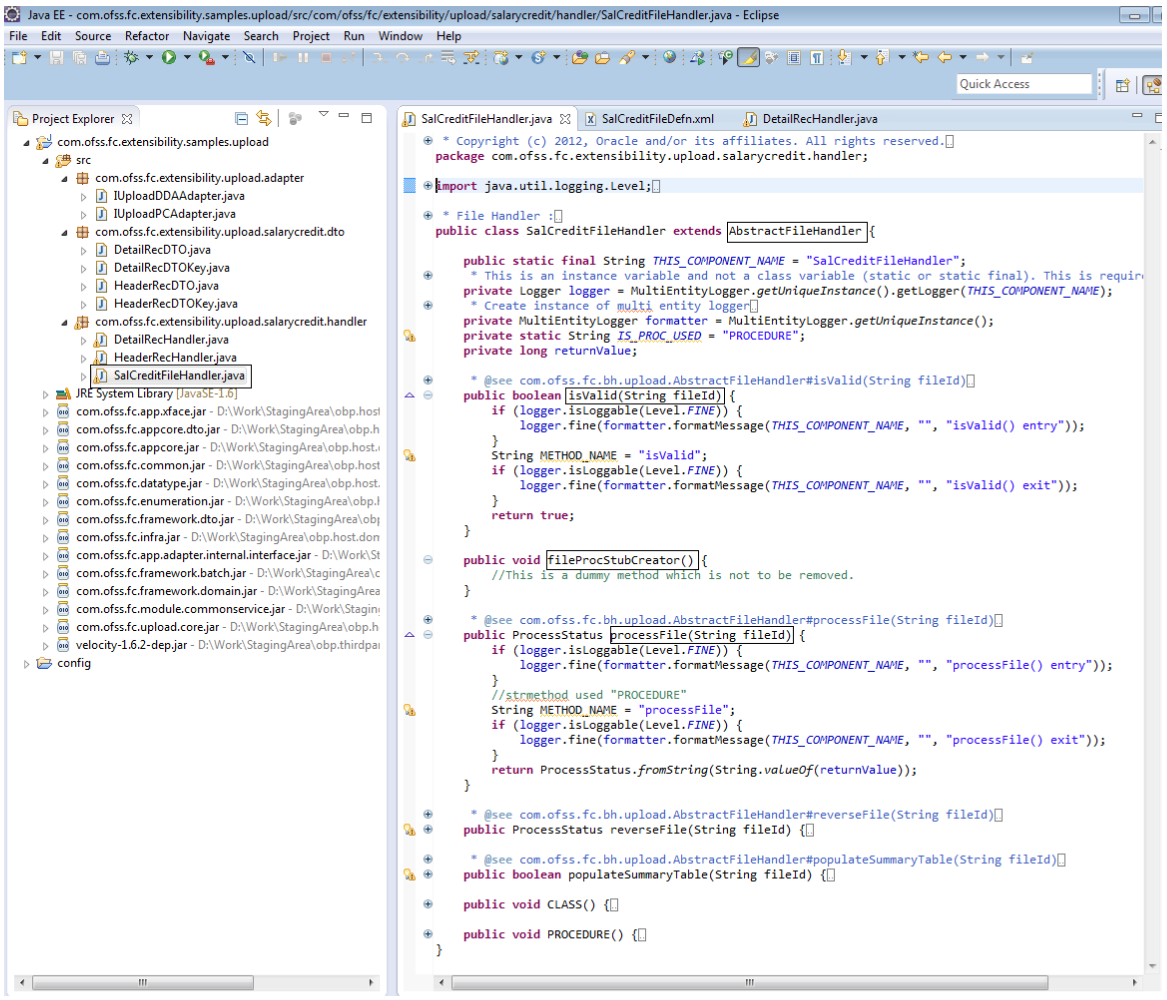
Columns	Description	Sample Values
STREAM_ COUNT	No of streams for the batch shell. Applicable only for StreamType as 'S' or 'R'	2
INPUT_DRV_ NAME	Fully classified class name mapped to the driver table	com.ofss.fc.entity.upload.AbstractRecordDTO
INPUT_SHELL_ PARAM	Name for the shell parameter	AbstractRecordDTO
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

10.1.2 File Handlers

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

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

Figure 10–2 File Handlers


Java EE - com.ofss.fc.extensibility.samples.upload/src/com/ofss/fc/extensibility/upload/salarycredit/handler/SalCreditFileHandler.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
SalCreditFileHandler.java
SalCreditFileDefn.xml
DetailRecHandler.java

```
com.ofss.fc.extensibility.samples.upload
  src
    com.ofss.fc.extensibility.upload.adapter
      IUploadDDAAdapter.java
      IUploadPCAdapter.java
    com.ofss.fc.extensibility.upload.salarycredit.dto
      DetailRecDTO.java
      DetailRecDTOKey.java
      HeaderRecDTO.java
      HeaderRecDTOKey.java
    com.ofss.fc.extensibility.upload.salarycredit.handler
      DetailRecHandler.java
      HeaderRecHandler.java
      SalCreditFileHandler.java
  JRE System Library [JavaSE-1.6]
  com.ofss.fc.app.xface.jar - D:\Work\StagingArea\obp.host
  com.ofss.fc.appcore.dto.jar - D:\Work\StagingArea\obp.h
  com.ofss.fc.appcore.jar - D:\Work\StagingArea\obp.host.
  com.ofss.fc.common.jar - D:\Work\StagingArea\obp.host
  com.ofss.fc.datatype.jar - D:\Work\StagingArea\obp.host.
  com.ofss.fc.enumeration.jar - D:\Work\StagingArea\obp.l
  com.ofss.fc.framework.dto.jar - D:\Work\StagingArea\obp
  com.ofss.fc.infra.jar - D:\Work\StagingArea\obp.host.dom
  com.ofss.fc.app.adapter.internal.interface.jar - D:\Work\St
  com.ofss.fc.framework.batch.jar - D:\Work\StagingArea\c
  com.ofss.fc.framework.domain.jar - D:\Work\StagingArea
  com.ofss.fc.module.commonservice.jar - D:\Work\Staging
  com.ofss.fc.upload.core.jar - D:\Work\StagingArea\obp.h
  velocity-1.6.2-dep.jar - D:\Work\StagingArea\obp.thirdpa
  config
```

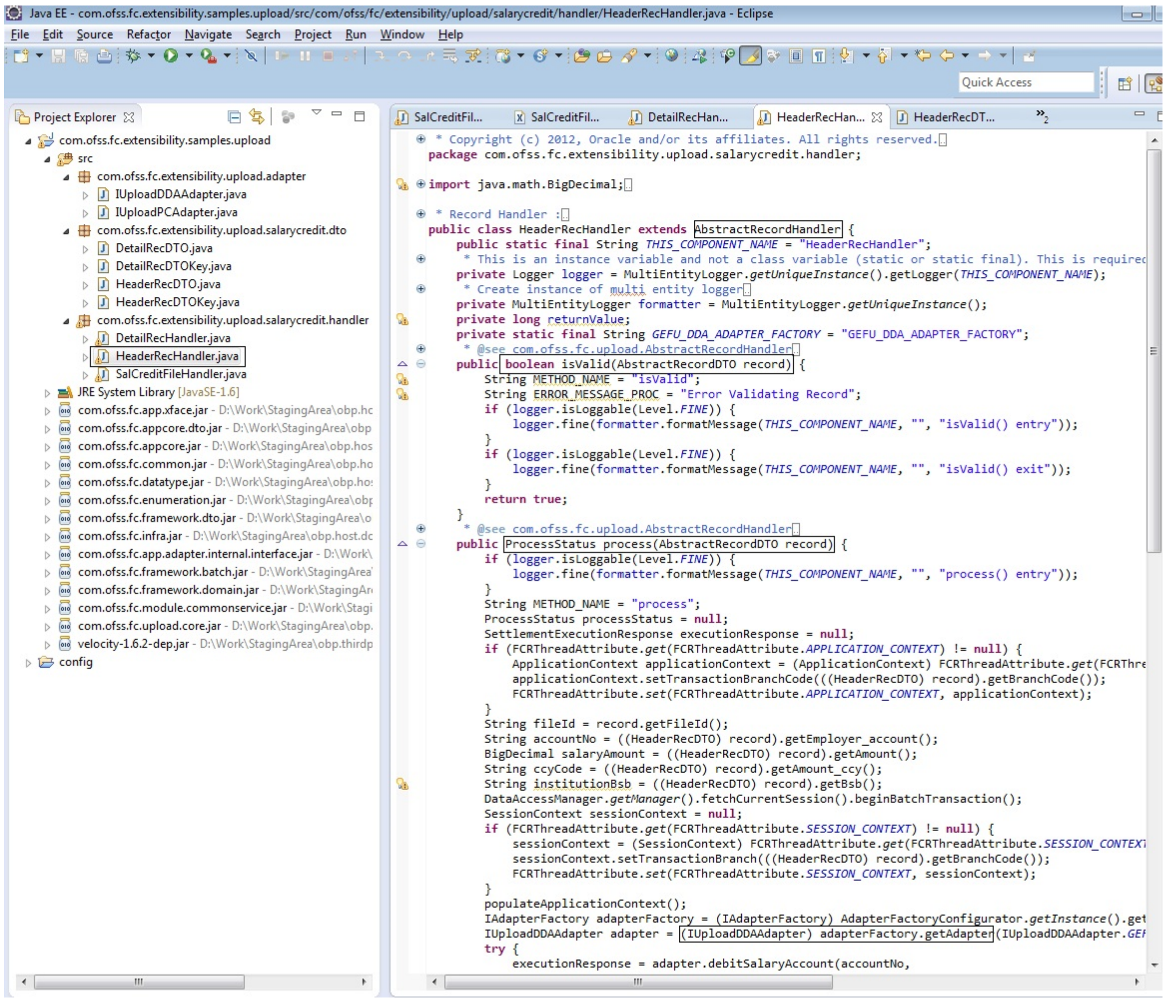
```
 * Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.extensibility.upload.salarycredit.handler;

import java.util.logging.Level;

 * File Handler :
public class SalCreditFileHandler extends AbstractFileHandler {

    public static final String THIS_COMPONENT_NAME = "SalCreditFileHandler";
     * This is an instance variable and not a class variable (static or static final). This is requir
    private Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(THIS_COMPONENT_NAME);
     * Create instance of multi entity logger
    private MultiEntityLogger formatter = MultiEntityLogger.getUniqueInstance();
    private static String IS_PROC_USED = "PROCEDURE";
    private long returnValue;

     * @see com.ofss.fc.bh.upload.AbstractFileHandler#isValid(String fileId)
    public boolean isValid(String fileId) {
        if (logger.isLoggable(Level.FINE)) {
            logger.fine(formatter.formatMessage(THIS_COMPONENT_NAME, "", "isValid() entry"));
        }
        String METHOD_NAME = "isValid";
        if (logger.isLoggable(Level.FINE)) {
            logger.fine(formatter.formatMessage(THIS_COMPONENT_NAME, "", "isValid() exit"));
        }
        return true;
    }

    public void fileProcStubCreator() {
        //This is a dummy method which is not to be removed.
    }

     * @see com.ofss.fc.bh.upload.AbstractFileHandler#processFile(String fileId)
    public ProcessStatus processFile(String fileId) {
        if (logger.isLoggable(Level.FINE)) {
            logger.fine(formatter.formatMessage(THIS_COMPONENT_NAME, "", "processFile() entry"));
        }
        //strmethod used "PROCEDURE"
        String METHOD_NAME = "processFile";
        if (logger.isLoggable(Level.FINE)) {
            logger.fine(formatter.formatMessage(THIS_COMPONENT_NAME, "", "processFile() exit"));
        }
        return ProcessStatus.fromString(String.valueOf(returnValue));
    }

     * @see com.ofss.fc.bh.upload.AbstractFileHandler#reverseFile(String fileId)
    public ProcessStatus reverseFile(String fileId) {

     * @see com.ofss.fc.bh.upload.AbstractFileHandler#populateSummaryTable(String fileId)
    public boolean populateSummaryTable(String fileId) {

    public void CLASS() {

    public void PROCEDURE() {
}
```

10.1.3 Record Handlers for Both Header and Details

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

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

Figure 10–3 Record Handlers for Both Header and Details


Java EE - com.ofss.fc.extensibility.samples.upload/src/com/ofss/fc/extensibility/upload/salarycredit/handler/HeaderRecHandler.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
SalCreditFil... SalCreditFil... DetailRecHan... HeaderRecHan... HeaderRecDT...

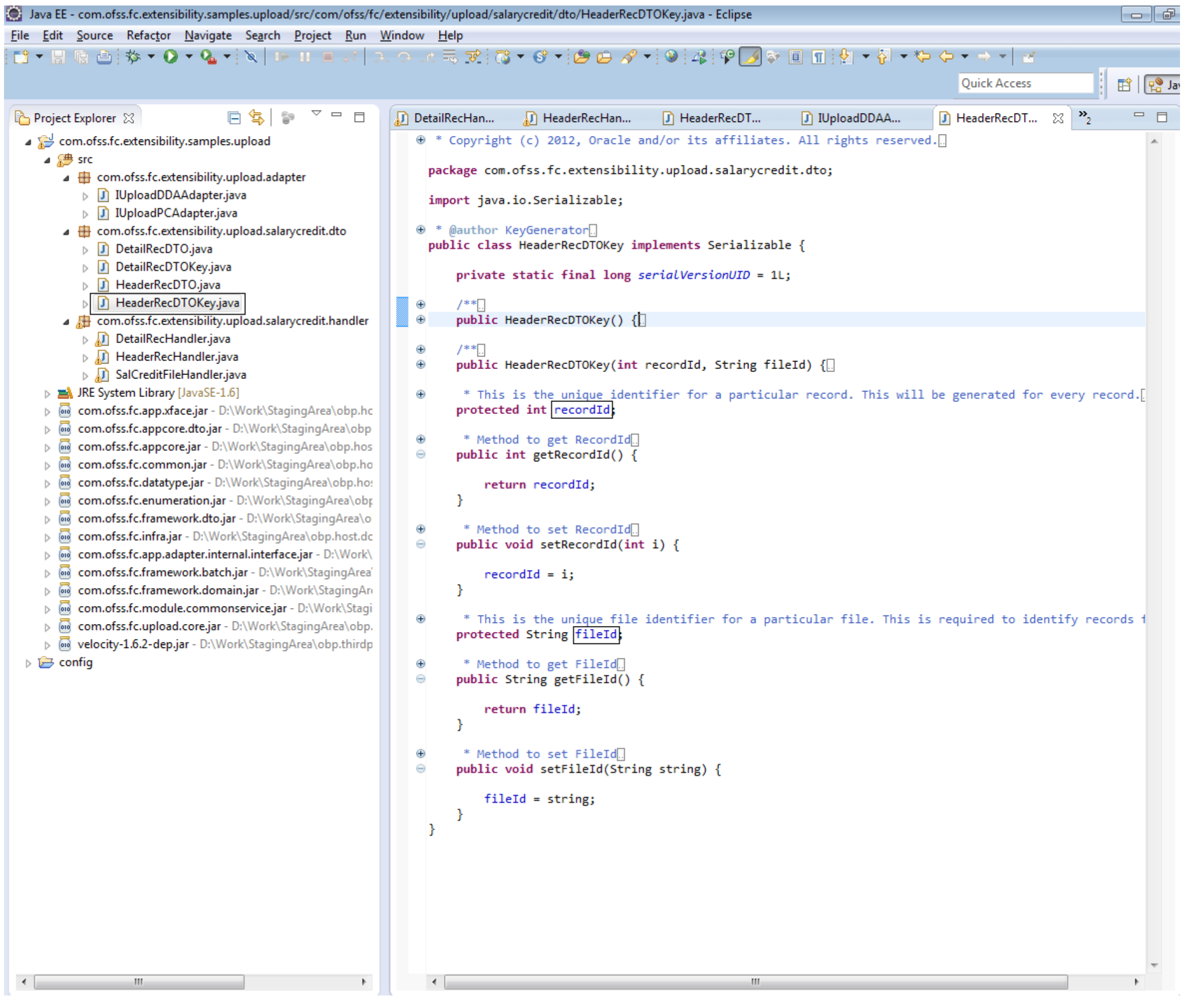
```
com.ofss.fc.extensibility.samples.upload
  src
    com.ofss.fc.extensibility.upload.adapter
      IUploadDDAAdapter.java
      IUploadPCAdapter.java
    com.ofss.fc.extensibility.upload.salarycredit.dto
      DetailRecDTO.java
      DetailRecDTOKey.java
      HeaderRecDTO.java
      HeaderRecDTOKey.java
    com.ofss.fc.extensibility.upload.salarycredit.handler
      DetailRecHandler.java
      HeaderRecHandler.java
      SalCreditFileHandler.java
  JRE System Library [JavaSE-1.6]
  com.ofss.fc.app.xface.jar - D:\Work\StagingArea\obp.hc
  com.ofss.fc.appcore.dto.jar - D:\Work\StagingArea\obp
  com.ofss.fc.appcore.jar - D:\Work\StagingArea\obp.hos
  com.ofss.fc.common.jar - D:\Work\StagingArea\obp.ho
  com.ofss.fc.datatype.jar - D:\Work\StagingArea\obp.ho:
  com.ofss.fc.enumeration.jar - D:\Work\StagingArea\obp
  com.ofss.fc.framework.dto.jar - D:\Work\StagingArea\o
  com.ofss.fc.infra.jar - D:\Work\StagingArea\obp.host.dc
  com.ofss.fc.app.adapter.internal.interface.jar - D:\Work\
  com.ofss.fc.framework.batch.jar - D:\Work\StagingArea'
  com.ofss.fc.framework.domain.jar - D:\Work\StagingAr
  com.ofss.fc.module.commonservice.jar - D:\Work\Stagi
  com.ofss.fc.upload.core.jar - D:\Work\StagingArea\obp.
  velocity-1.6.2-dep.jar - D:\Work\StagingArea\obp.thirdp
config
```

```java
 * Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.extensibility.upload.salarycredit.handler;

import java.math.BigDecimal;

 * Record Handler :
public class HeaderRecHandler extends AbstractRecordHandler {
    public static final String THIS_COMPONENT_NAME = "HeaderRecHandler";
     * This is an instance variable and not a class variable (static or static final). This is required
    private Logger logger = MultiEntityLogger.getUniqueInstance().getLogger(THIS_COMPONENT_NAME);
     * Create instance of multi entity logger
    private MultiEntityLogger formatter = MultiEntityLogger.getUniqueInstance();
    private long returnValue;
    private static final String GEFU_DDA_ADAPTER_FACTORY = "GEFU_DDA_ADAPTER_FACTORY";
     * @see com.ofss.fc.upload.AbstractRecordHandler
    public boolean isValid(AbstractRecordDTO record) {
        String METHOD_NAME = "isValid";
        String ERROR_MESSAGE_PROC = "Error Validating Record";
        if (logger.isLoggable(Level.FINE)) {
            logger.fine(formatter.formatMessage(THIS_COMPONENT_NAME, "", "isValid() entry"));
        }
        if (logger.isLoggable(Level.FINE)) {
            logger.fine(formatter.formatMessage(THIS_COMPONENT_NAME, "", "isValid() exit"));
        }
        return true;
    }
     * @see com.ofss.fc.upload.AbstractRecordHandler
    public ProcessStatus process(AbstractRecordDTO record) {
        if (logger.isLoggable(Level.FINE)) {
            logger.fine(formatter.formatMessage(THIS_COMPONENT_NAME, "", "process() entry"));
        }
        String METHOD_NAME = "process";
        ProcessStatus processStatus = null;
        SettlementExecutionResponse executionResponse = null;
        if (FCRThreadAttribute.get(FCRThreadAttribute.APPLICATION_CONTEXT) != null) {
            ApplicationContext applicationContext = (ApplicationContext) FCRThreadAttribute.get(FCRThre
            applicationContext.setTransactionBranchCode(((HeaderRecDTO) record).getBranchCode());
            FCRThreadAttribute.set(FCRThreadAttribute.APPLICATION_CONTEXT, applicationContext);
        }
        String fileId = record.getFileId();
        String accountNo = ((HeaderRecDTO) record).getEmployer_account();
        BigDecimal salaryAmount = ((HeaderRecDTO) record).getAmount();
        String ccyCode = ((HeaderRecDTO) record).getAmount_ccy();
        String institutionBsb = ((HeaderRecDTO) record).getBsb();
        DataAccessManager.getManager().fetchCurrentSession().beginBatchTransaction();
        SessionContext sessionContext = null;
        if (FCRThreadAttribute.get(FCRThreadAttribute.SESSION_CONTEXT) != null) {
            sessionContext = (SessionContext) FCRThreadAttribute.get(FCRThreadAttribute.SESSION_CONTEXT
            sessionContext.setTransactionBranch(((HeaderRecDTO) record).getBranchCode());
            FCRThreadAttribute.set(FCRThreadAttribute.SESSION_CONTEXT, sessionContext);
        }
        populateApplicationContext();
        IAdapterFactory adapterFactory = (IAdapterFactory) AdapterFactoryConfigurator.getInstance().get
        IUploadDDAAdapter adapter = (IUploadDDAAdapter) adapterFactory.getAdapter(IUploadDDAAdapter.GEF
        try {
            executionResponse = adapter.debitSalaryAccount(accountNo,
```

10.1.4 DTO and Keys Classes for Both Header and Details

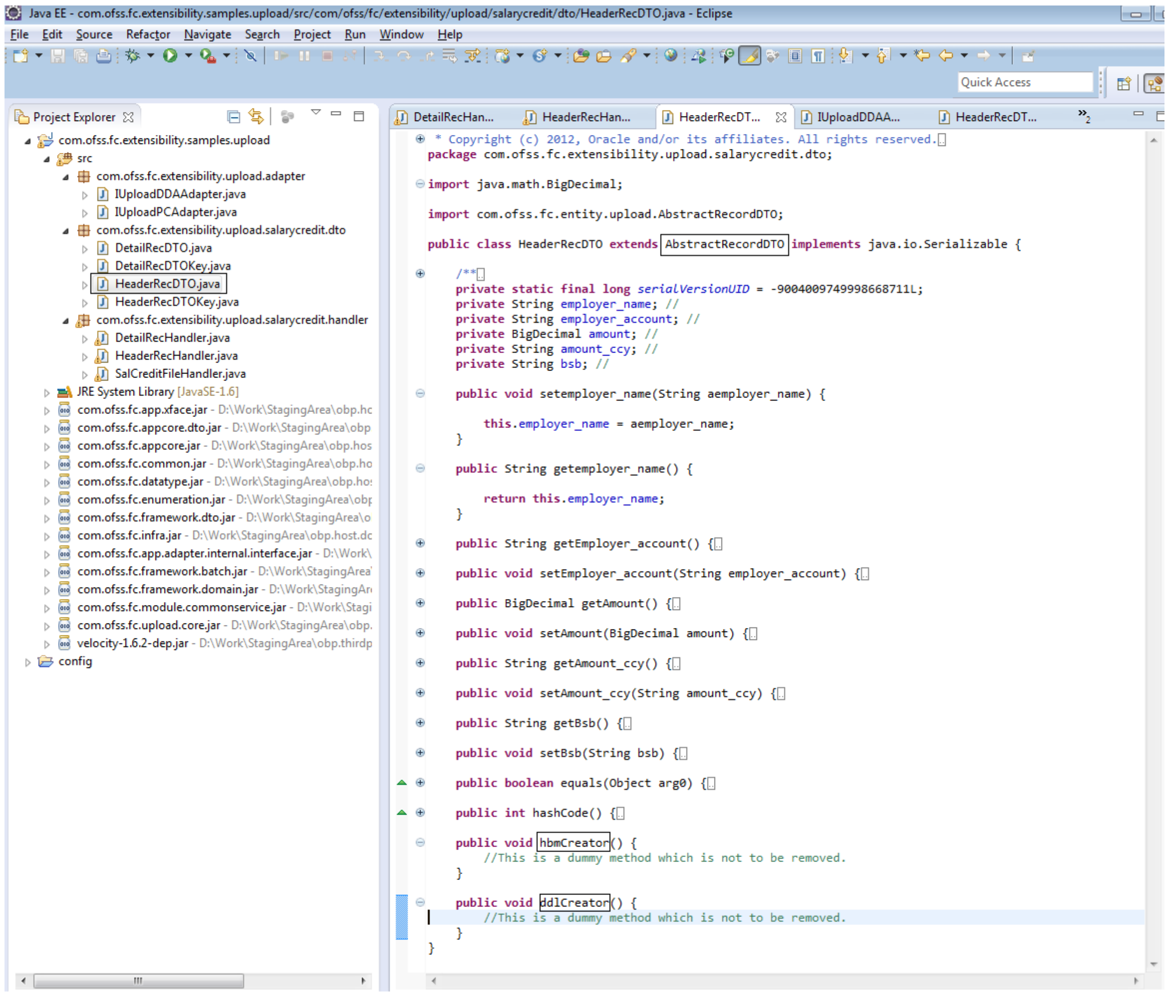
This is a persistent class for the particular process. This class provides the fields which represents the characteristics of the record data. This class is defined for each record type of a file.

Figure 10–4 DTO and Keys Classes for Both Header and Details - HeaderRecDTOKey


Java EE - com.ofss.fc.extensibility.samples.upload/src/com/ofss/fc/extensibility/upload/salarycredit/dto/HeaderRecDTOKey.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
com.ofss.fc.extensibility.samples.upload
src
com.ofss.fc.extensibility.upload.adapter
IUploadDDAAdapter.java
IUploadPCAdapter.java
com.ofss.fc.extensibility.upload.salarycredit.dto
DetailRecDTO.java
DetailRecDTOKey.java
HeaderRecDTO.java
HeaderRecDTOKey.java
com.ofss.fc.extensibility.upload.salarycredit.handler
DetailRecHandler.java
HeaderRecHandler.java
SalCreditFileHandler.java
JRE System Library [JavaSE-1.6]
com.ofss.fc.app.xface.jar - D:\Work\StagingArea\obp.hc
com.ofss.fc.appcore.dto.jar - D:\Work\StagingArea\obp
com.ofss.fc.appcore.jar - D:\Work\StagingArea\obp.hos
com.ofss.fc.common.jar - D:\Work\StagingArea\obp.ho
com.ofss.fc.datatype.jar - D:\Work\StagingArea\obp.ho:
com.ofss.fc.enumeration.jar - D:\Work\StagingArea\obp
com.ofss.fc.framework.dto.jar - D:\Work\StagingArea\o
com.ofss.fc.infra.jar - D:\Work\StagingArea\obp.host.dc
com.ofss.fc.app.adapter.internal.interface.jar - D:\Work\
com.ofss.fc.framework.batch.jar - D:\Work\StagingArea'
com.ofss.fc.framework.domain.jar - D:\Work\StagingAr
com.ofss.fc.module.commonservice.jar - D:\Work\Stagi
com.ofss.fc.upload.core.jar - D:\Work\StagingArea\obp.
velocity-1.6.2-dep.jar - D:\Work\StagingArea\obp.thirdp
config
DetailRecHan...
HeaderRecHan...
HeaderRecDT...
IUploadDDAA...
HeaderRecDT...

```
 * Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.

package com.ofss.fc.extensibility.upload.salarycredit.dto;

import java.io.Serializable;

 * @author KeyGenerator
public class HeaderRecDTOKey implements Serializable {

    private static final long serialVersionUID = 1L;

    /**
    public HeaderRecDTOKey() {

    /**
    public HeaderRecDTOKey(int recordId, String fileId) {

     * This is the unique identifier for a particular record. This will be generated for every record.
    protected int recordId;

     * Method to get RecordId
    public int getRecordId() {

        return recordId;
    }

     * Method to set RecordId
    public void setRecordId(int i) {

        recordId = i;
    }

     * This is the unique file identifier for a particular file. This is required to identify records
    protected String fileId;

     * Method to get FileId
    public String getFileId() {

        return fileId;
    }

     * Method to set FileId
    public void setFileId(String string) {

        fileId = string;
    }
}
```

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


Java EE - com.ofss.fc.extensibility.samples.upload/src/com/ofss/fc/extensibility/upload/salarycredit/dto/HeaderRecDTO.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
com.ofss.fc.extensibility.samples.upload
src
com.ofss.fc.extensibility.upload.adapter
IUploadDDAAdapter.java
IUploadPCAdapter.java
com.ofss.fc.extensibility.upload.salarycredit.dto
DetailRecDTO.java
DetailRecDTOKey.java
HeaderRecDTO.java
HeaderRecDTOKey.java
com.ofss.fc.extensibility.upload.salarycredit.handler
DetailRecHandler.java
HeaderRecHandler.java
SalCreditFileHandler.java
JRE System Library [JavaSE-1.6]
com.ofss.fc.app.xface.jar - D:\Work\StagingArea\obp.hc
com.ofss.fc.appcore.dto.jar - D:\Work\StagingArea\obp
com.ofss.fc.appcore.jar - D:\Work\StagingArea\obp.hos
com.ofss.fc.common.jar - D:\Work\StagingArea\obp.ho
com.ofss.fc.datatype.jar - D:\Work\StagingArea\obp.ho:
com.ofss.fc.enumeration.jar - D:\Work\StagingArea\obp
com.ofss.fc.framework.dto.jar - D:\Work\StagingArea\o
com.ofss.fc.infra.jar - D:\Work\StagingArea\obp.host.dc
com.ofss.fc.app.adapter.internal.interface.jar - D:\Work\
com.ofss.fc.framework.batch.jar - D:\Work\StagingArea
com.ofss.fc.framework.domain.jar - D:\Work\StagingAr
com.ofss.fc.module.commonservice.jar - D:\Work\Stagi
com.ofss.fc.upload.core.jar - D:\Work\StagingArea\obp.
velocity-1.6.2-dep.jar - D:\Work\StagingArea\obp.thirdp
config
DetailRecHan...
HeaderRecHan...
HeaderRecDT...
IUploadDDAA...
HeaderRecDT...

```
 * Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.extensibility.upload.salarycredit.dto;

import java.math.BigDecimal;

import com.ofss.fc.entity.upload.AbstractRecordDTO;

public class HeaderRecDTO extends AbstractRecordDTO implements java.io.Serializable {

    /**
    private static final long serialVersionUID = -9004009749998668711L;
    private String employer_name; //
    private String employer_account; //
    private BigDecimal amount; //
    private String amount_ccy; //
    private String bsb; //

    public void setemployer_name(String aemployer_name) {

        this.employer_name = aemployer_name;
    }

    public String getemployer_name() {

        return this.employer_name;
    }

    public String getEmployer_account() {

    public void setEmployer_account(String employer_account) {

    public BigDecimal getAmount() {

    public void setAmount(BigDecimal amount) {

    public String getAmount_ccy() {

    public void setAmount_ccy(String amount_ccy) {

    public String getBsb() {

    public void setBsb(String bsb) {

    public boolean equals(Object arg0) {

    public int hashCode() {

    public void hbmCreator() {
        //This is a dummy method which is not to be removed.
    }

    public void ddlCreator() {
        //This is a dummy method which is not to be removed.
    }
}
```

10.1.5 XFF File Definition XML

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

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

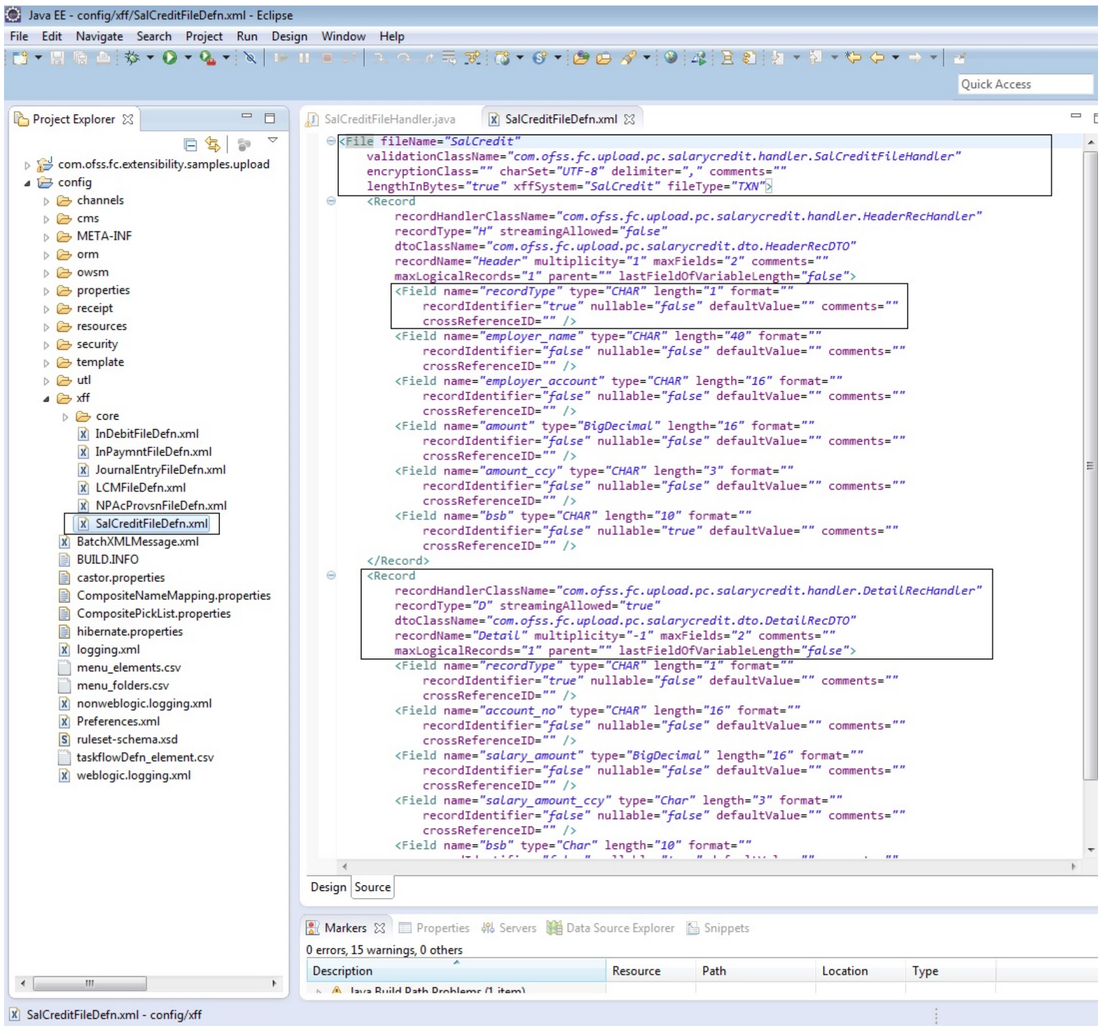
- One 'File' element can have many 'Record' elements, depending upon the number of recordType present for this upload.
- One 'Record' element can have many 'Field' elements, depending upon the number of fields present for this recordType of upload.

Table 10–5 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 10–6 XXF File Definition XML



```
<File fileName="SalCredit"
    validationClassName="com.ofss.fc.upload.pc.salarycredit.handler.SalCreditFileHandler"
    encryptionClass="" charSet="UTF-8" delimiter="," comments=""
    lengthInBytes="true" xffSystem="SalCredit" fileType="TXN">
    <Record
        recordHandlerClassName="com.ofss.fc.upload.pc.salarycredit.handler.HeaderRecHandler"
        recordType="H" streamingAllowed="false"
        dtoClassName="com.ofss.fc.upload.pc.salarycredit.dto.HeaderRecDTO"
        recordName="Header" multiplicity="1" maxFields="2" comments=""
        maxLogicalRecords="1" parent="" lastFieldOfVariableLength="false">
        <Field name="recordType" type="CHAR" length="1" format=""
            recordIdentifier="true" nullable="false" defaultValue="" comments=""
            crossReferenceID="" />
        <Field name="employer_name" type="CHAR" length="40" format=""
            recordIdentifier="false" nullable="false" defaultValue="" comments=""
            crossReferenceID="" />
        <Field name="employer_account" type="CHAR" length="16" format=""
            recordIdentifier="false" nullable="false" defaultValue="" comments=""
            crossReferenceID="" />
        <Field name="amount" type="BigDecimal" length="16" format=""
            recordIdentifier="false" nullable="false" defaultValue="" comments=""
            crossReferenceID="" />
        <Field name="amount_ccy" type="CHAR" length="3" format=""
            recordIdentifier="false" nullable="false" defaultValue="" comments=""
            crossReferenceID="" />
        <Field name="bsb" type="CHAR" length="10" format=""
            recordIdentifier="false" nullable="true" defaultValue="" comments=""
            crossReferenceID="" />
    </Record>
    <Record
        recordHandlerClassName="com.ofss.fc.upload.pc.salarycredit.handler.DetailRecHandler"
        recordType="D" streamingAllowed="true"
        dtoClassName="com.ofss.fc.upload.pc.salarycredit.dto.DetailRecDTO"
        recordName="Detail" multiplicity="-1" maxFields="2" comments=""
        maxLogicalRecords="1" parent="" lastFieldOfVariableLength="false">
        <Field name="recordType" type="CHAR" length="1" format=""
            recordIdentifier="true" nullable="false" defaultValue="" comments=""
            crossReferenceID="" />
        <Field name="account_no" type="CHAR" length="16" format=""
            recordIdentifier="false" nullable="false" defaultValue="" comments=""
            crossReferenceID="" />
        <Field name="salary_amount" type="BigDecimal" length="16" format=""
            recordIdentifier="false" nullable="false" defaultValue="" comments=""
            crossReferenceID="" />
        <Field name="salary_amount_ccy" type="Char" length="3" format=""
            recordIdentifier="false" nullable="false" defaultValue="" comments=""
            crossReferenceID="" />
        <Field name="bsb" type="Char" length="10" format=""
```

```
0 errors, 15 warnings, 0 others
Description                Resource    Path    Location    Type
Java Build Path Problems (1 item)
```

10.2 Processing

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

The processing of the business logic to different Service APIs of different modules are carried in the handler classes of the records. The processForRecordType() method of the FileProcessor invokes the respective handler classes that is, if the Header section is being processed, it invokes the HeaderHandler class.

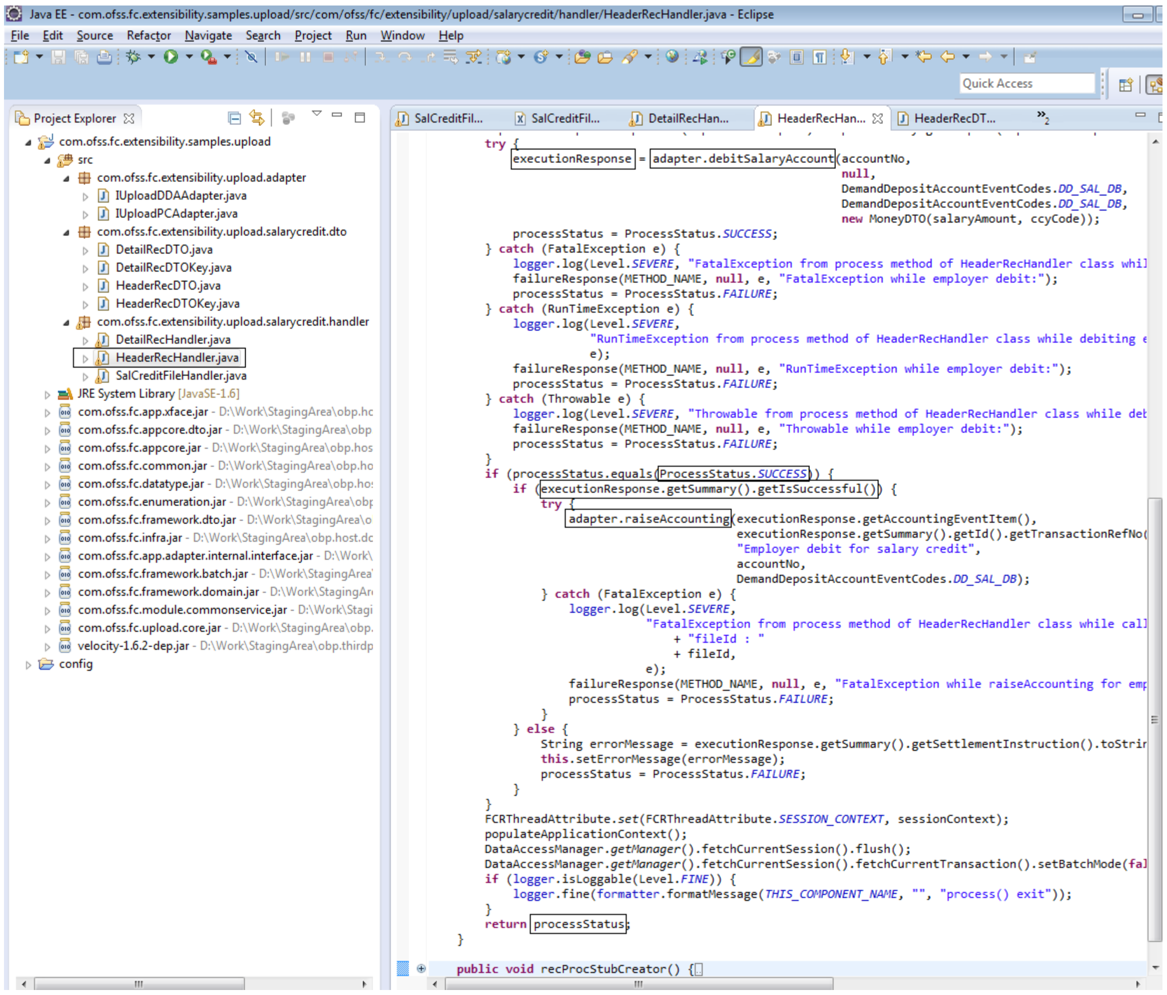
As per the process, the headers are processed first and then the details records. Each and every record is processed individually. As soon as a file is picked for processing, its status is changed to InProgress so that the same file is not picked by any other process for processing. Individual records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

10.2.1 API Calls in the Handlers

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

```
<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 10–7 API Calls in Adapters

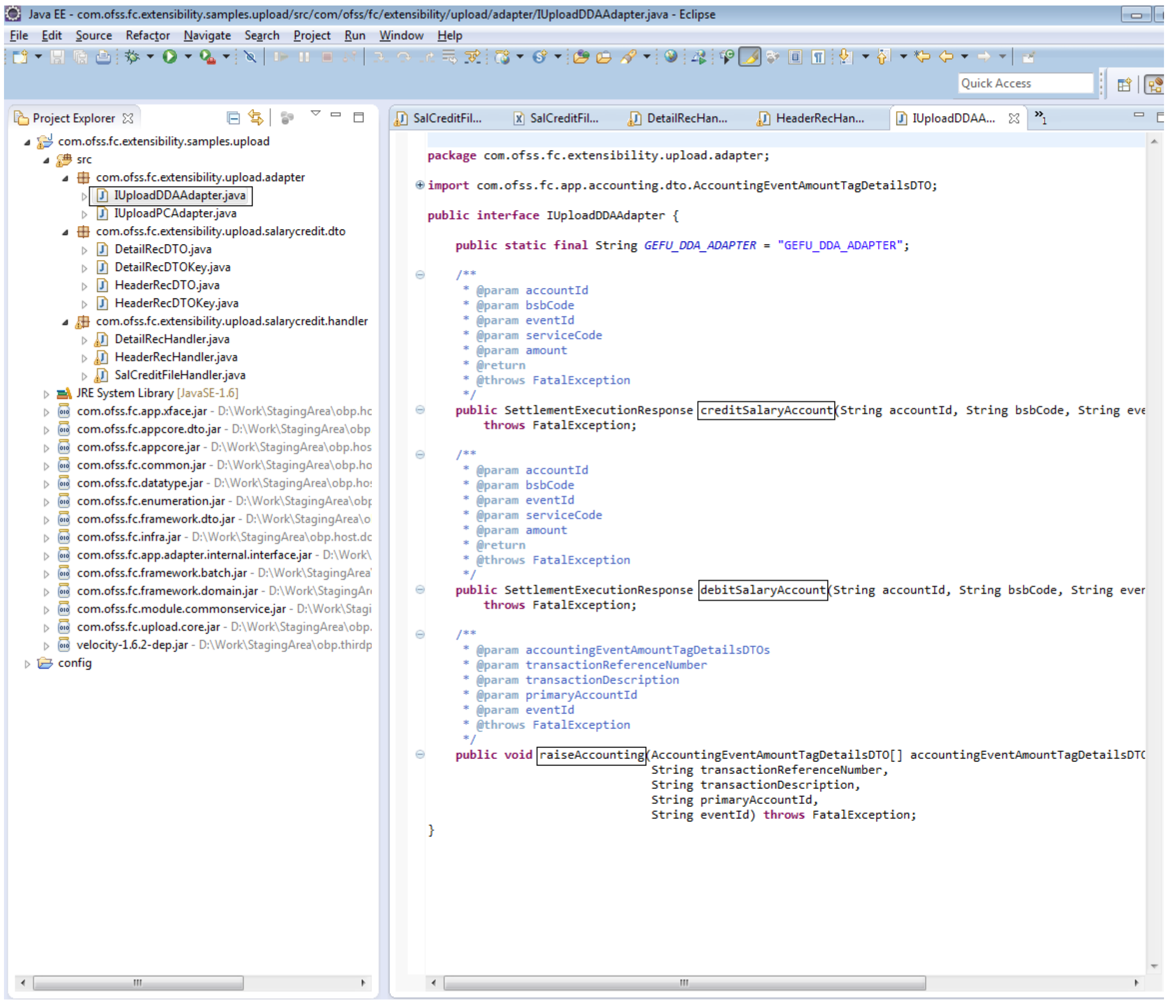

Java EE - com.ofss.fc.extensibility.samples.upload/src/com/ofss/fc/extensibility/upload/salarycredit/handler/HeaderRecHandler.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
com.ofss.fc.extensibility.samples.upload
src
com.ofss.fc.extensibility.upload.adapter
IUploadDDAAdapter.java
IUploadPCAdapter.java
com.ofss.fc.extensibility.upload.salarycredit.dto
DetailRecDTO.java
DetailRecDTOKey.java
HeaderRecDTO.java
HeaderRecDTOKey.java
com.ofss.fc.extensibility.upload.salarycredit.handler
DetailRecHandler.java
HeaderRecHandler.java
SalCreditFileHandler.java
JRE System Library [JavaSE-1.6]
com.ofss.fc.app.xface.jar - D:\Work\StagingArea\obp.hc
com.ofss.fc.appcore.dto.jar - D:\Work\StagingArea\obp
com.ofss.fc.appcore.jar - D:\Work\StagingArea\obp.hos
com.ofss.fc.common.jar - D:\Work\StagingArea\obp.ho
com.ofss.fc.datatype.jar - D:\Work\StagingArea\obp.ho:
com.ofss.fc.enumeration.jar - D:\Work\StagingArea\obp
com.ofss.fc.framework.dto.jar - D:\Work\StagingArea\o
com.ofss.fc.infra.jar - D:\Work\StagingArea\obp.host.dc
com.ofss.fc.app.adapter.internal.interface.jar - D:\Work\
com.ofss.fc.framework.batch.jar - D:\Work\StagingArea'
com.ofss.fc.framework.domain.jar - D:\Work\StagingAr
com.ofss.fc.module.commonservice.jar - D:\Work\Stagi
com.ofss.fc.upload.core.jar - D:\Work\StagingArea\obp.
velocity-1.6.2-dep.jar - D:\Work\StagingArea\obp.thirdp
config
SalCreditFil...
SalCreditFil...
DetailRecHan...
HeaderRecHan...
HeaderRecDT...

```
        try {
            executionResponse = adapter.debitSalaryAccount(accountNo,
                                                           null,
                                                           DemandDepositAccountEventCodes.DD_SAL_DB,
                                                           DemandDepositAccountEventCodes.DD_SAL_DB,
                                                           new MoneyDTO(salaryAmount, ccyCode));
            processStatus = ProcessStatus.SUCCESS;
        } catch (FatalException e) {
            logger.log(Level.SEVERE, "FatalException from process method of HeaderRecHandler class whil
            failureResponse(METHOD_NAME, null, e, "FatalException while employer debit:");
            processStatus = ProcessStatus.FAILURE;
        } catch (RunTimeException e) {
            logger.log(Level.SEVERE,
                       "RunTimeException from process method of HeaderRecHandler class while debiting e
                       e);
            failureResponse(METHOD_NAME, null, e, "RunTimeException while employer debit:");
            processStatus = ProcessStatus.FAILURE;
        } catch (Throwable e) {
            logger.log(Level.SEVERE, "Throwable from process method of HeaderRecHandler class while deb
            failureResponse(METHOD_NAME, null, e, "Throwable while employer debit:");
            processStatus = ProcessStatus.FAILURE;
        }
        if (processStatus.equals(ProcessStatus.SUCCESS)) {
            if (executionResponse.getSummary().getIsSuccessful()) {
                try {
                    adapter.raiseAccounting(executionResponse.getAccountingEventItem(),
                                            executionResponse.getSummary().getId().getTransactionRefNo(
                                            "Employer debit for salary credit",
                                            accountNo,
                                            DemandDepositAccountEventCodes.DD_SAL_DB);
                } catch (FatalException e) {
                    logger.log(Level.SEVERE,
                               "FatalException from process method of HeaderRecHandler class while call
                                   + "fileId : "
                                   + fileId,
                               e);
                    failureResponse(METHOD_NAME, null, e, "FatalException while raiseAccounting for emp
                    processStatus = ProcessStatus.FAILURE;
                }
            } else {
                String errorMessage = executionResponse.getSummary().getSettlementInstruction().toStrir
                this.setErrorMessage(errorMessage);
                processStatus = ProcessStatus.FAILURE;
            }
        }
        FCRThreadAttribute.set(FCRThreadAttribute.SESSION_CONTEXT, sessionContext);
        populateApplicationContext();
        DataAccessManager.getManager().fetchCurrentSession().flush();
        DataAccessManager.getManager().fetchCurrentSession().fetchCurrentTransaction().setBatchMode(fal
        if (logger.isLoggable(Level.FINE)) {
            logger.fine(formatter.formatMessage(THIS_COMPONENT_NAME, "", "process() exit"));
        }
        return processStatus;
    }

    public void recProcStubCreator() {
```

10.2.2 Processing Adapter

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

Figure 10–8 Processing Adapter


Java EE - com.ofss.fc.extensibility.samples.upload/src/com/ofss/fc/extensibility/upload/adapter/IUploadDDAAdapter.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
com.ofss.fc.extensibility.samples.upload
src
com.ofss.fc.extensibility.upload.adapter
IUploadDDAAdapter.java
IUploadPCAdapter.java
com.ofss.fc.extensibility.upload.salarycredit.dto
DetailRecDTO.java
DetailRecDTOKey.java
HeaderRecDTO.java
HeaderRecDTOKey.java
com.ofss.fc.extensibility.upload.salarycredit.handler
DetailRecHandler.java
HeaderRecHandler.java
SalCreditFileHandler.java
JRE System Library [JavaSE-1.6]
com.ofss.fc.app.xface.jar - D:\Work\StagingArea\obp.hc
com.ofss.fc.appcore.dto.jar - D:\Work\StagingArea\obp
com.ofss.fc.appcore.jar - D:\Work\StagingArea\obp.hos
com.ofss.fc.common.jar - D:\Work\StagingArea\obp.ho
com.ofss.fc.datatype.jar - D:\Work\StagingArea\obp.ho:
com.ofss.fc.enumeration.jar - D:\Work\StagingArea\obp
com.ofss.fc.framework.dto.jar - D:\Work\StagingArea\o
com.ofss.fc.infra.jar - D:\Work\StagingArea\obp.host.dc
com.ofss.fc.app.adapter.internal.interface.jar - D:\Work\
com.ofss.fc.framework.batch.jar - D:\Work\StagingArea'
com.ofss.fc.framework.domain.jar - D:\Work\StagingAr
com.ofss.fc.module.commonservice.jar - D:\Work\Stagi
com.ofss.fc.upload.core.jar - D:\Work\StagingArea\obp.
velocity-1.6.2-dep.jar - D:\Work\StagingArea\obp.thirdp
config
SalCreditFil...
SalCreditFil...
DetailRecHan...
HeaderRecHan...
IUploadDDAA...

```java
package com.ofss.fc.extensibility.upload.adapter;

import com.ofss.fc.app.accounting.dto.AccountingEventAmountTagDetailsDTO;

public interface IUploadDDAAdapter {

    public static final String GEFU_DDA_ADAPTER = "GEFU_DDA_ADAPTER";

    /**
     * @param accountId
     * @param bsbCode
     * @param eventId
     * @param serviceCode
     * @param amount
     * @return
     * @throws FatalException
     */
    public SettlementExecutionResponse creditSalaryAccount(String accountId, String bsbCode, String eve
        throws FatalException;

    /**
     * @param accountId
     * @param bsbCode
     * @param eventId
     * @param serviceCode
     * @param amount
     * @return
     * @throws FatalException
     */
    public SettlementExecutionResponse debitSalaryAccount(String accountId, String bsbCode, String ever
        throws FatalException;

    /**
     * @param accountingEventAmountTagDetailsDTOs
     * @param transactionReferenceNumber
     * @param transactionDescription
     * @param primaryAccountId
     * @param eventId
     * @throws FatalException
     */
    public void raiseAccounting(AccountingEventAmountTagDetailsDTO[] accountingEventAmountTagDetailsDTC
                                String transactionReferenceNumber,
                                String transactionDescription,
                                String primaryAccountId,
                                String eventId) throws FatalException;
}
```

10.3 Outcome

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

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

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

Table 10–6 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.

10.4 Failure/Exception Handling

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

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

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

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

11 Alerts Extension

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

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

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

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

11.1 Transaction as an Activity

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

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

11.1.1 Activity Record

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

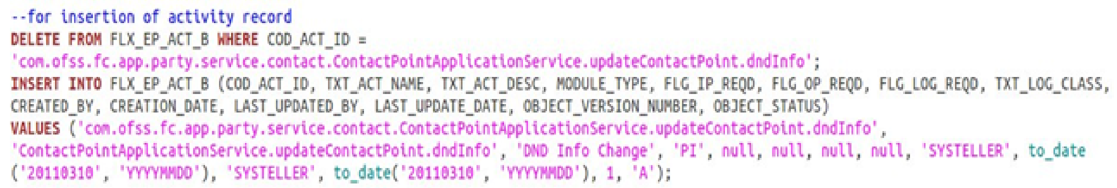
Table 11–1 FLX_EP_ACT_B

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 11–1 Sample script for Activity Record



```
--for insertion of activity record
DELETE FROM FLX_EP_ACT_B WHERE COD_ACT_ID =
'com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint.dndInfo';
INSERT INTO FLX_EP_ACT_B (COD_ACT_ID, TXT_ACT_NAME, TXT_ACT_DESC, MODULE_TYPE, FLG_IP_REQD, FLG_OP_REQD, FLG_LOG_REQD, TXT_LOG_CLASS,
CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATE_DATE, 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');
```

11.1.2 Attaching Events to Activity

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

11.1.3 Event Record

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

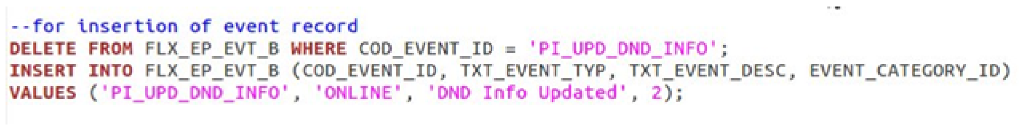
Table 11–2 FLX_EP_EVT_B

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 11–2 Sample script for Event Record



```
--for insertion of event record
DELETE FROM FLX_EP_EVT_B WHERE COD_EVENT_ID = 'PI_UPD_DND_INFO';
INSERT INTO FLX_EP_EVT_B (COD_EVENT_ID, TXT_EVENT_TYP, TXT_EVENT_DESC, EVENT_CATEGORY_ID)
VALUES ('PI_UPD_DND_INFO', 'ONLINE', 'DND Info Updated', 2);
```

11.1.4 Activity Event Mapping Record

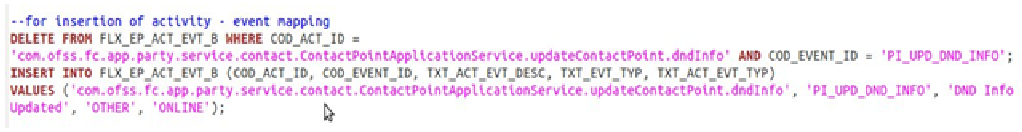
You will need to create an activity event mapping record in the table FLX_EP_ACT_EVT_B which stores the mapping between all activities and events. This table has the following columns:

Table 11–3 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 11–3 Activity Event Mapping Record



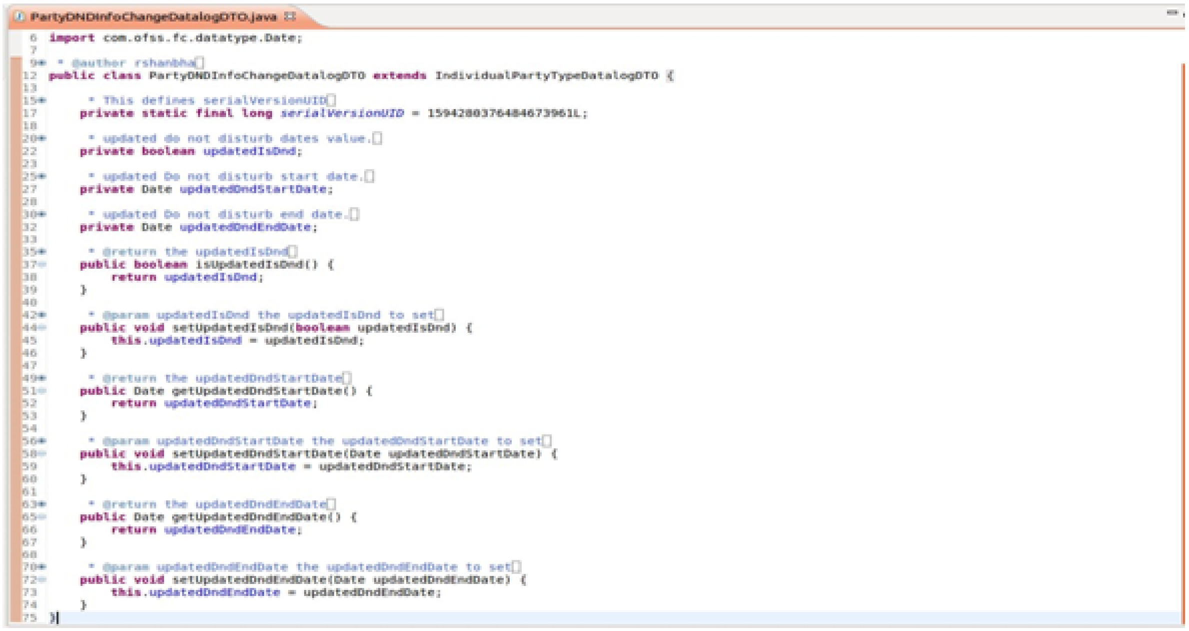
```
--for insertion of activity - event mapping
DELETE FROM FLX_EP_ACT_EVT_B WHERE COD_ACT_ID =
'com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint.dndInfo' AND COD_EVENT_ID = 'PI_UPD_DND_INFO';
INSERT INTO FLX_EP_ACT_EVT_B (COD_ACT_ID, COD_EVENT_ID, TXT_ACT_EVT_DESC,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

11.1.5 Activity Log DTO

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

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

Figure 11–4 Activity Log DTO



```
PartyDNDInfoChangeDatalogDTO.java
import com.ofss.fc.datatype.Date;

 * @author rshanbha
public class PartyDNDInfoChangeDatalogDTO extends IndividualPartyTypeDatalogDTO {

     * This defines serialVersionUID
    private static final long serialVersionUID = 1594280376484673961L;

     * updated do not disturb dates value.
    private boolean updatedIsDnd;

     * updated Do not disturb start date.
    private Date updatedDndStartDate;

     * updated Do not disturb end date.
    private Date updatedDndEndDate;

     * @return the updatedIsDnd
    public boolean isUpdatedIsDnd() {
        return updatedIsDnd;
    }

     * @param updatedIsDnd the updatedIsDnd to set
    public void setUpdatedIsDnd(boolean updatedIsDnd) {
        this.updatedIsDnd = updatedIsDnd;
    }

     * @return the updatedDndStartDate
    public Date getUpdatedDndStartDate() {
        return updatedDndStartDate;
    }

     * @param updatedDndStartDate the updatedDndStartDate to set
    public void setUpdatedDndStartDate(Date updatedDndStartDate) {
        this.updatedDndStartDate = updatedDndStartDate;
    }

     * @return the updatedDndEndDate
    public Date getUpdatedDndEndDate() {
        return updatedDndEndDate;
    }

     * @param updatedDndEndDate the updatedDndEndDate to set
    public void setUpdatedDndEndDate(Date updatedDndEndDate) {
        this.updatedDndEndDate = updatedDndEndDate;
    }
}
```

11.1.6 Alert Metadata Generation

This section describes the different types of alert metadata generation.

Metadata Generation

To generate metadata for alerts you need to have plugin.

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*.

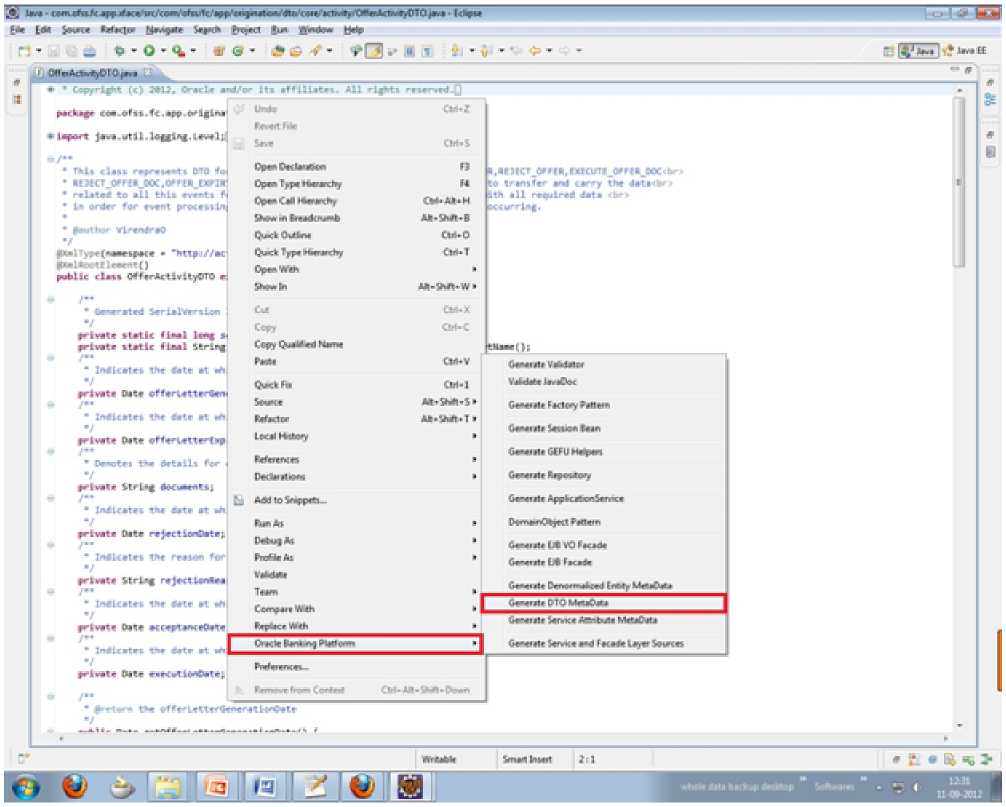
   This will generate the insert scripts for following two tables:

   - 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 11–5 Metadata Generation



```
 * Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.

package com.ofss.fc.app.origina

import java.util.logging.Level;

/**
 * This class represents DTO fo                R,REJECT_OFFER,EXECUTE_OFFER_DOC<br>
 * REJECT_OFFER_DOC,OFFER_EXPIR                to transfer and carry the data<br>
 * related to all this events f                ith all required data <br>
 * in order for event processin                occurring.
 *
 * @author VirendraD
 */
@XmlType(namespace = "http://ac
@XmlRootElement()
public class OfferActivityDTO e

    /**
     * Generated SerialVersion
     */
    private static final long s
    private static final String                tName();
    /**
     * Indicates the date at wh
     */
    private Date offerLetterGen
    /**
     * Indicates the date at wh
     */
    private Date offerLetterExp
    /**
     * Denotes the details for
     */
    private String documents;
    /**
     * Indicates the date at wh
     */
    private Date rejectionDate;
    /**
     * Indicates the reason for
     */
    private String rejectionRea
    /**
     * Indicates the date at wh
     */
    private Date acceptanceDate
    /**
     * Indicates the date at wh
     */
    private Date executionDate;

    /**
     * @return the offerLetterGenerationDate
     */
```

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.

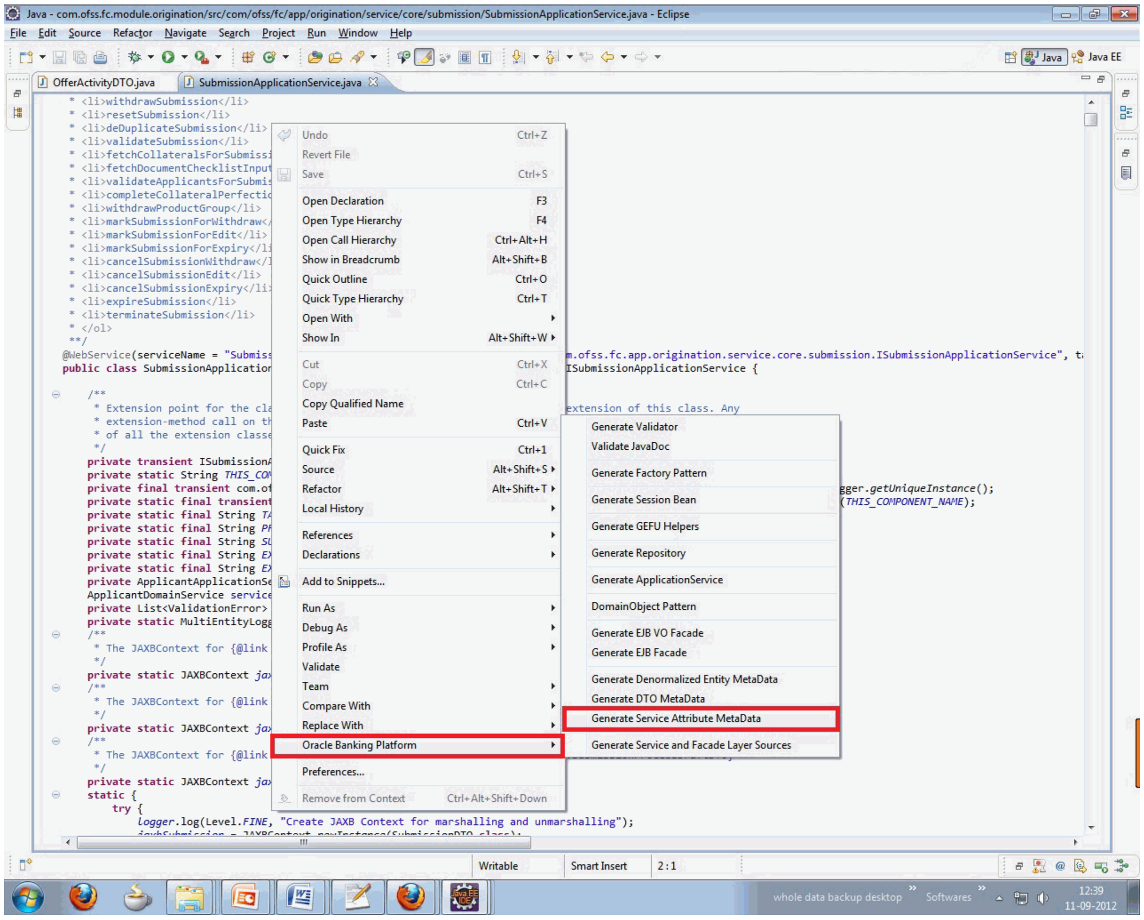
1. Right-click that service and select *Oracle Banking Platform -> Generate Service Attribute Metadata*.
2. In this case also insert scripts will be generate in same location as metadata attributes.

   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 11–6 Service Data Attribute Generation



```
     * <li>withdrawSubmission</li>
     * <li>resetSubmission</li>
     * <li>deDuplicateSubmission</li>
     * <li>validateSubmission</li>
     * <li>fetchCollateralsForSubmissi
     * <li>fetchDocumentChecklistInput
     * <li>validateApplicantsForSubmis
     * <li>completeCollateralPerfectic
     * <li>withdrawProductGroup</li>
     * <li>markSubmissionForWithdraw</
     * <li>markSubmissionForEdit</li>
     * <li>markSubmissionForExpiry</li
     * <li>cancelSubmissionWithdraw</l
     * <li>cancelSubmissionEdit</li>
     * <li>cancelSubmissionExpiry</li>
     * <li>expireSubmission</li>
     * <li>terminateSubmission</li>
     * </ol>
     **/
    @WebService(serviceName = "Submiss                              m.ofss.fc.app.origination.service.core.submission.ISubmissionApplicationService", t
    public class SubmissionApplicatior                              ISubmissionApplicationService {

        /**
         * Extension point for the cla                              extension of this class. Any
         * extension-method call on th
         * of all the extension classe
         */
        private transient ISubmissionA
        private static String THIS_COM                                                                gger.getUniqueInstance();
        private final transient com.of                                                                (THIS_COMPONENT_NAME);
        private static final transient
        private static final String TA
        private static final String PF
        private static final String SL
        private static final String EX
        private static final String EX
        private ApplicantApplicationSe
        ApplicantDomainService service
        private List<ValidationError>
        private static MultiEntityLogg
        /**
         * The JAXBContext for {@link
         */
        private static JAXBContext jax
        /**
         * The JAXBContext for {@link
         */
        private static JAXBContext jax
        /**
         * The JAXBContext for {@link
         */
        private static JAXBContext jax
        static {
            try {
                logger.log(Level.FINE, "Create JAXB Context for marshalling and unmarshalling");
```

FLX_MD_SERVICE_ATTR is used to map the alert activity attribute with the fact code and to map the alert activity attribute with the DTO field to extract the data from.

As an example, the key fields in FLX_MD_SERVICE_ATTR for an alert activity attribute have been listed below:

Table 11–4 Key Fields in FLX_MD_SERVICE_ATTR

Column	Description
COD_SERVICE_ATTR_ID	The Unique ID for the Attribute of any Activity configured for an alert. For example, com.ofss.fc.app.account.service.accountaddresslinkage. 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()

11.1.7 Alert Message Template Maintenance

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

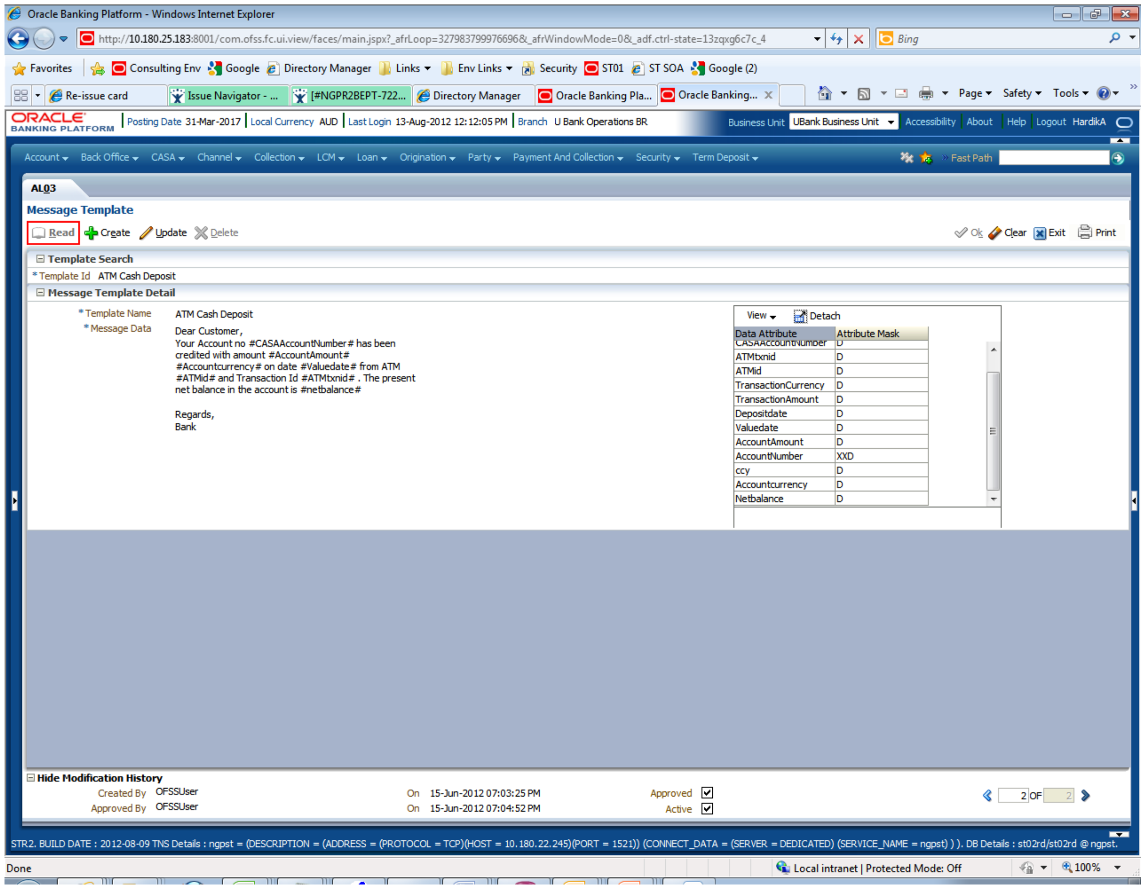
These messages need to be defined only if the same template is going to be used for multiple events. Else there is a provision to define the message template during the definition of the alert itself.

All data elements defined within the '#' symbol will be defaulted in the panel below as data attribute.

For example, your account Number #Acct No# has been credited with #currency# #transaction amount# being cash deposited.

The user can Mask certain digits in data elements that are preceded with '#' under the 'Attribute Mask' column.

Figure 11–7 Alert Message Template Maintenance



Posting Date 31-Mar-2017 | Local Currency AUD | Last Login 13-Aug-2012 12:12:05 PM | Branch U Bank Operations BR

Message Template

Template Search

*Template Id ATM Cash Deposit

Message Template Detail

*Template Name	ATM Cash Deposit
*Message Data	Dear Customer, Your Account no #CASAAccountNumber# has been credited with amount #AccountAmount# #Accountcurrency# on date #Valuedate# from ATM #ATMid# and Transaction Id #ATMtxnid# . The present net balance in the account is #netbalance# Regards, Bank

Data Attribute	Attribute Mask
CASAAccountNumber	D
ATMtxnid	D
ATMid	D
TransactionCurrency	D
TransactionAmount	D
Depositdate	D
Valuedate	D
AccountAmount	D
AccountNumber	XXD
ccy	D
Accountcurrency	D
Netbalance	D

Hide Modification History

Created By	OFSSUser	On	15-Jun-2012 07:03:25 PM	Approved	✓
Approved By	OFSSUser	On	15-Jun-2012 07:04:52 PM	Active	✓

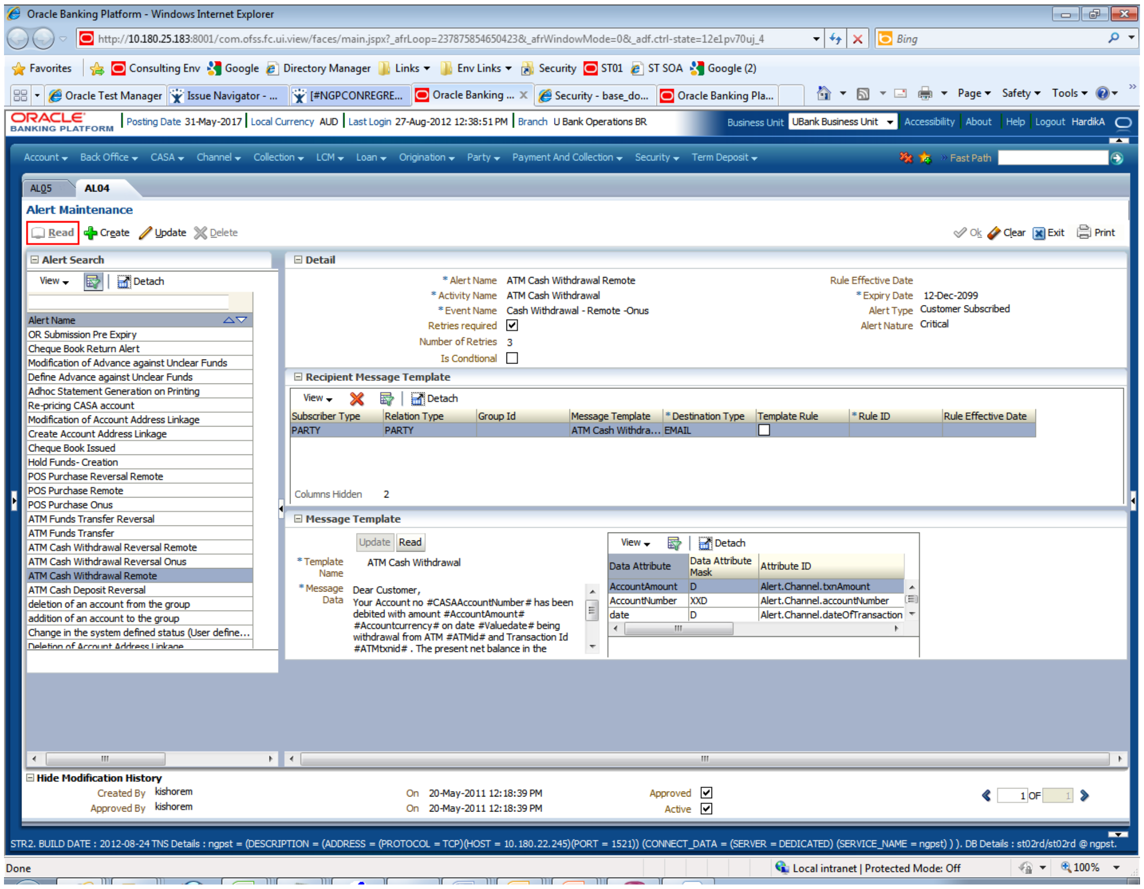
2 OF 2

```
STR2. BUILD DATE : 2012-08-09 TNS Details : ngpst = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = 10.180.22.245)(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = ngpst) ) ). DB Details : st02rd/st02rd @ ngpst.
```

11.1.8 Alert Maintenance

Given below is the Alert Maintenance screen.

Figure 11–8 Alert Maintenance



Oracle Banking Platform - Windows Internet Explorer

Posting Date 31-May-2017 | Local Currency AUD | Last Login 27-Aug-2012 12:38:51 PM | Branch U Bank Operations BR

Alert Maintenance

Read | Create | Update | Delete | Ok | Clear | Exit | Print

Alert Search

Alert Name
OR Submission Pre Expiry
Cheque Book Return Alert
Modification of Advance against Unclear Funds
Define Advance against Unclear Funds
Adhoc Statement Generation on Printing
Re-pricing CASA account
Modification of Account Address Linkage
Create Account Address Linkage
Cheque Book Issued
Hold Funds- Creation
POS Purchase Reversal Remote
POS Purchase Remote
POS Purchase Onus
ATM Funds Transfer Reversal
ATM Funds Transfer
ATM Cash Withdrawal Reversal Remote
ATM Cash Withdrawal Reversal Onus
ATM Cash Withdrawal Remote
ATM Cash Deposit Reversal
deletion of an account from the group
addition of an account to the group
Change in the system defined status (User define…
Deletion of Account Address Linkage

Detail

* Alert Name	ATM Cash Withdrawal Remote	Rule Effective Date	
* Activity Name	ATM Cash Withdrawal	* Expiry Date	12-Dec-2099
* Event Name	Cash Withdrawal - Remote -Onus	Alert Type	Customer Subscribed
Retries required	✓	Alert Nature	Critical
Number of Retries	3		
Is Condtional	☐		

Recipient Message Template

Subscriber Type	Relation Type	Group Id	Message Template	* Destination Type	Template Rule	* Rule ID	Rule Effective Date
PARTY	PARTY		ATM Cash Withdra…	EMAIL	☐		

Columns Hidden 2

Message Template

Update | Read

* Template Name: ATM Cash Withdrawal

* Message Data: Dear Customer,
Your Account no #CASAAccountNumber# has been debited with amount #AccountAmount# #Accountcurrency# on date #Valuedate# being withdrawal from ATM #ATMid# and Transaction Id #ATMtxnid# . The present net balance in the

Data Attribute	Data Attribute Mask	Attribute ID
AccountAmount	D	Alert.Channel.txnAmount
AccountNumber	XXD	Alert.Channel.accountNumber
date	D	Alert.Channel.dateOfTransaction

Hide Modification History

Created By	kishorem	On	20-May-2011 12:18:39 PM	Approved	✓
Approved By	kishorem	On	20-May-2011 12:18:39 PM	Active	✓

STR2. BUILD DATE : 2012-08-24 TNS Details : ngpst = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = 10.180.22.245)(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = ngpst))). DB Details : st02rd/st02rd @ ngpst.

We can define the alert name, expiry date, alert type (Customer Subscribed/ Mandatory) and link this with predefined activity and event. These entries are fed to table "flx_ep_act_evt_acn_b".

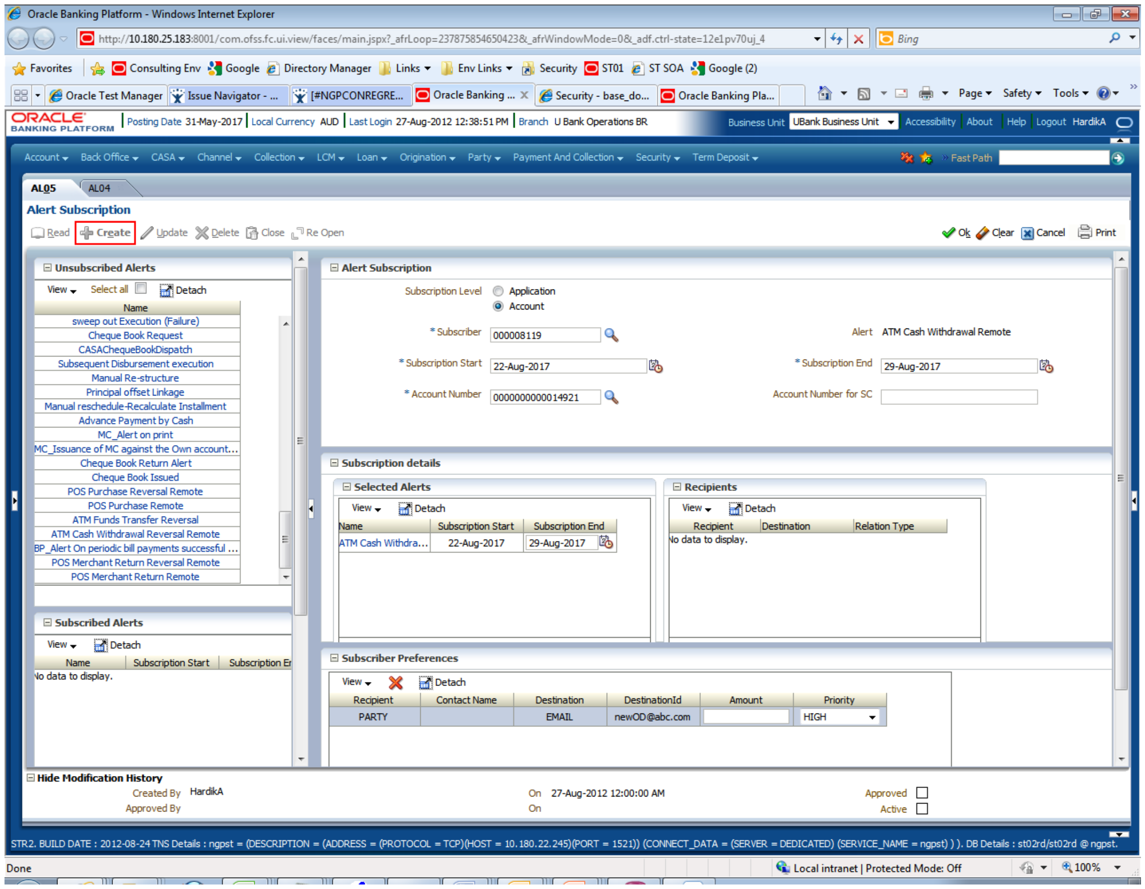
Now, we need to link a Recipient Message Template/s with this alert. For this we drag recipients from the Recipient Panel on to the Recipient Message Template Panel. In this setup, we define the kind of recipient and link this to predefined Message Template and Destination Types. The entry for this goes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

Finally, we need to complete the Message Template Mapping Configuration for each Recipient Message Template. For this, we map each data attribute of each Recipient Message Template with a corresponding attribute (Fact Code) from the drop down. This drop down populates fact codes configured for this activity id in the metadata table FLX_MD_SERVICE_ATTRIBUTE. The entry for this goes to table "flx_ep_msg_src_b"

11.2 Alert Subscription

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

Figure 11–9 Alert Subscription



Alert Subscription

Unsubscribed Alerts

Name
sweep out Execution (Failure)
Cheque Book Request
CASAChequeBookDispatch
Subsequent Disbursement execution
Manual Re-structure
Principal offset Linkage
Manual reschedule-Recalculate Installment
Advance Payment by Cash
MC_Alert on print
MC_Issuance of MC against the Own account...
Cheque Book Return Alert
Cheque Book Issued
POS Purchase Reversal Remote
POS Purchase Remote
ATM Funds Transfer Reversal
ATM Cash Withdrawal Reversal Remote
BP_Alert On periodic bill payments successful ...
POS Merchant Return Reversal Remote
POS Merchant Return Remote

Subscribed Alerts

Name	Subscription Start	Subscription End
No data to display.		

Alert Subscription

Subscription Level: ( ) Application [✓ Account]

* Subscriber: 000008119 — Alert: ATM Cash Withdrawal Remote

* Subscription Start: 22-Aug-2017 — * Subscription End: 29-Aug-2017

* Account Number: 0000000000014921 — Account Number for SC:

Subscription details

Selected Alerts

Name	Subscription Start	Subscription End
ATM Cash Withdra...	22-Aug-2017	29-Aug-2017

Recipients

Recipient	Destination	Relation Type
No data to display.		

Subscriber Preferences

Recipient	Contact Name	Destination	DestinationId	Amount	Priority
PARTY		EMAIL	newOD@abc.com		HIGH

Hide Modification History

Created By HardikA — On 27-Aug-2012 12:00:00 AM — Approved ☐

Approved By — On — Active ☐

STR2. BUILD DATE : 2012-08-24 TNS Details : ngpst = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = 10.180.22.245)(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = ngpst))). DB Details : st02rd/st02rd @ ngpst.

Oracle Banking Platform - Windows Internet Explorer
http://10.180.25.183:8001/com.ofss.fc.ui.view/faces/main.jspx?_afrLoop=2378758546504238&_afrWindowMode=0&_adf.ctrl-state=12e1pv70uj_4
Posting Date 31-May-2017 | Local Currency AUD | Last Login 27-Aug-2012 12:38:51 PM | Branch U Bank Operations BR
Business Unit UBank Business Unit | Accessibility | About | Help | Logout HardikA
Account
Back Office
CASA
Channel
Collection
LCM
Loan
Origination
Party
Payment And Collection
Security
Term Deposit
Fast Path
AL05
AL04
Read
Create
Update
Delete
Close
Re Open
Ok
Clear
Cancel
Print
View
Select all
Detach
Done
Local intranet | Protected Mode: Off
100%

11.2.1 Transaction API Changes

You will need to modify the transaction API to support the newly registered activities. This section gives an overview of how the developer needs to modify the transaction API.

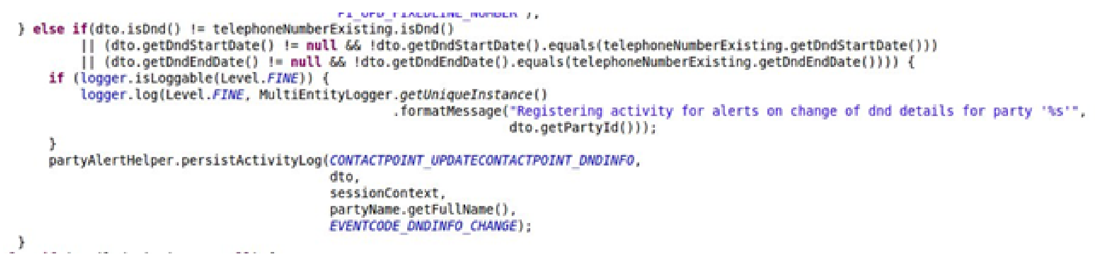
The entry point for activity business logic would be the service call for the transaction. In the aforementioned use case, the service call would be *com.ofss.fc.app.party.service.contact.ContactPointApplicationService.updateContactPoint(...).*

Figure 11–10 Transaction API Changes - Service Call

```
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.

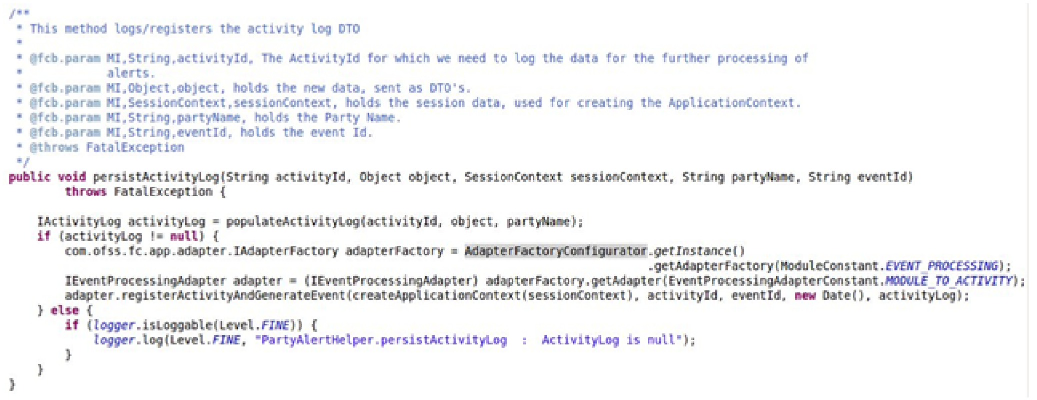
Figure 11–11 Transaction API Changes - Conditional Evaluation



```
} 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.

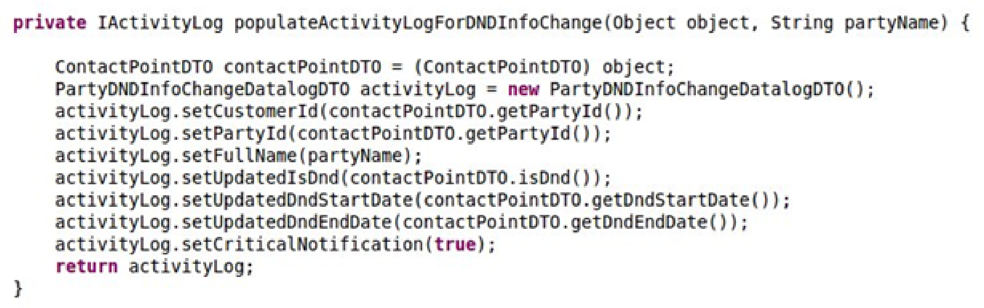
Figure 11–12 Transaction API Changes - persistActivityLog(..)



```
/**
 * 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");
        }
    }
}
```

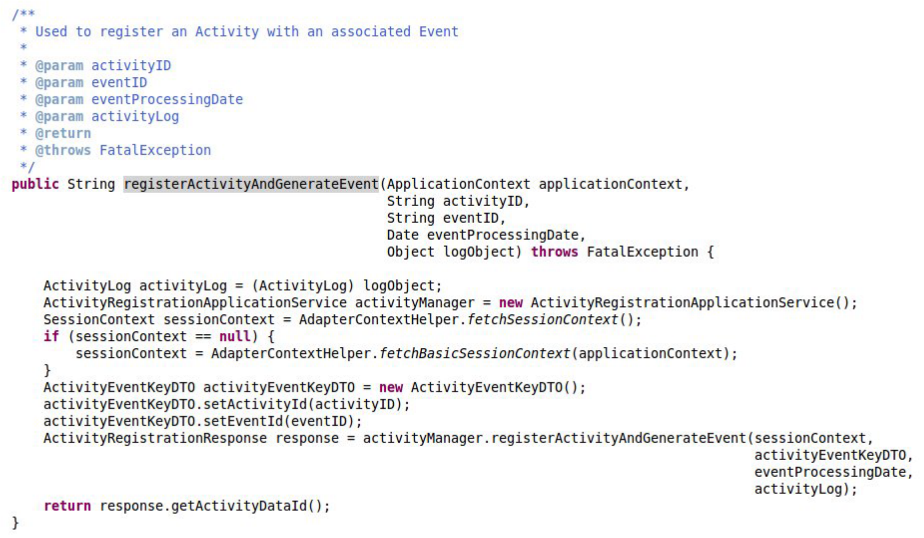
You will need to add the business logic to populate the activity log DTO with the data specific to the transaction and the activity. This logic can be present inside the activity helper class for the module. Module specific activity attributes can also be populated in this logic. Following code snippet shows the activity log DTO population with activity data for the aforementioned activity.

Figure 11–13 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 11–14 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.

11.3 Alert Processing Steps

For any modules the starting point is EventProcessingAdapter method named ’registerActivityAndGenerateEvent’.

Through this we call ’registerActivityAndGenerateEvent’ method of ActivityRegistrationApplicationService which marks actually registration of your activities and events.

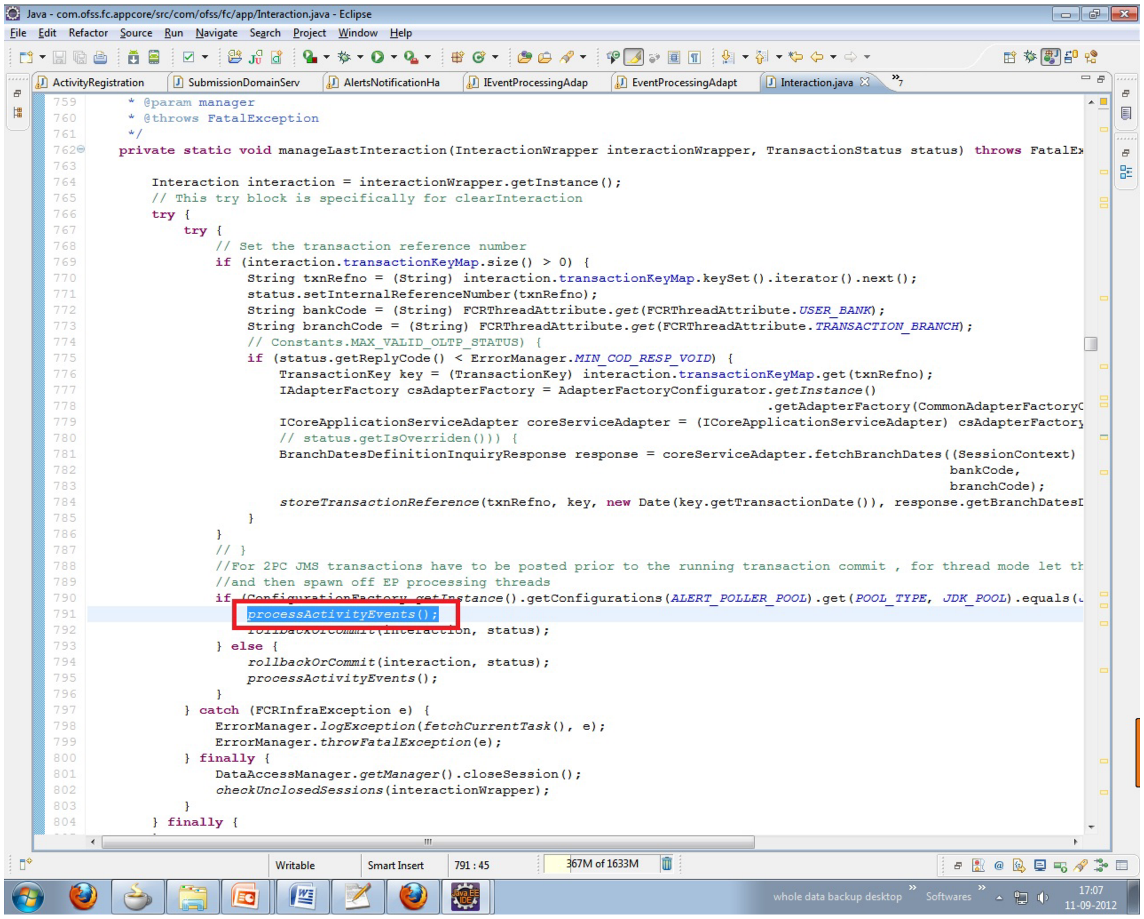
During this activity the entries are made in table FLX_EP_ACT_LOG_B and FLX_EP_EVT_LOG_B with appropriate comments depending on type of Alerts whether it is Mandatory (M) or Customer Subscribed (S).

There is one flag maintained in FLX_EP_EVT_LOG_B viz. FLG_PROCESS_STAT, which specifies status of event.

In this step various validations are also performed such as checking if email Id of recipient is mentioned and so on.

However, the final processing of alerts is managed in ’Interaction.java’ when it is about to close that is, call is made in ’manageLastInteraction’.

Figure 11–15 Alert Processing Steps



```
     * @param manager
     * @throws FatalException
     */
    private static void manageLastInteraction(InteractionWrapper interactionWrapper, TransactionStatus status) throws FatalEx

        Interaction interaction = interactionWrapper.getInstance();
        // This try block is specifically for clearInteraction
        try {
            try {
                // Set the transaction reference number
                if (interaction.transactionKeyMap.size() > 0) {
                    String txnRefno = (String) interaction.transactionKeyMap.keySet().iterator().next();
                    status.setInternalReferenceNumber(txnRefno);
                    String bankCode = (String) FCRThreadAttribute.get(FCRThreadAttribute.USER_BANK);
                    String branchCode = (String) FCRThreadAttribute.get(FCRThreadAttribute.TRANSACTION_BRANCH);
                    // Constants.MAX_VALID_OLTP_STATUS) {
                    if (status.getReplyCode() < ErrorManager.MIN_COD_RESP_VOID) {
                        TransactionKey key = (TransactionKey) interaction.transactionKeyMap.get(txnRefno);
                        IAdapterFactory csAdapterFactory = AdapterFactoryConfigurator.getInstance()
                                                                                .getAdapterFactory(CommonAdapterFactoryC
                        ICoreApplicationServiceAdapter coreServiceAdapter = (ICoreApplicationServiceAdapter) csAdapterFactory
                        // status.getIsOverriden())) {
                        BranchDatesDefinitionInquiryResponse response = coreServiceAdapter.fetchBranchDates((SessionContext)
                                                                                              bankCode,
                                                                                              branchCode);
                        storeTransactionReference(txnRefno, key, new Date(key.getTransactionDate()), response.getBranchDatesD
                    }
                }
                // }
                //For 2PC JMS transactions have to be posted prior to the running transaction commit , for thread mode let th
                //and then spawn off EP processing threads
                if (ConfigurationFactory.getInstance().getConfigurations(ALERT_POLLER_POOL).get(POOL_TYPE, JDK_POOL).equals(J
                    processActivityEvents();
                    rollbackOrCommit(interaction, status);
                } else {
                    rollbackOrCommit(interaction, status);
                    processActivityEvents();
                }
            } catch (FCRInfraException e) {
                ErrorManager.logException(fetchCurrentTask(), e);
                ErrorManager.throwFatalException(e);
            } finally {
                DataAccessManager.getManager().closeSession();
                checkUnclosedSessions(interactionWrapper);
            }
        } finally {
```

EventProcessStatusType

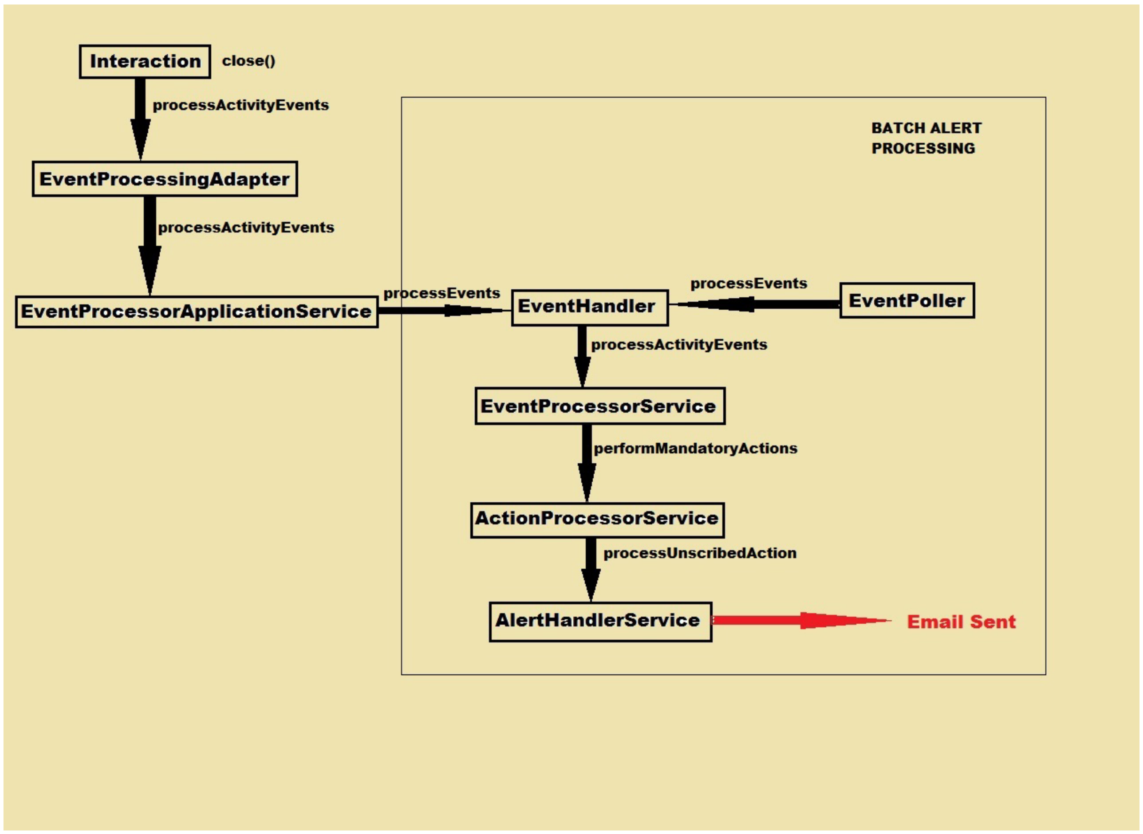
This shows status of event throughout cycle of event processing from Registration of event to Dispatch of Alert. (It is maintained in FLX_EP_EVT_LOG_B table as "flg_process_stat").

The various statuses of events are as follows:

- GENERATED("G“)
- COMPLETED("C“)
- NO_SUBSCRIPTION("N")
- ABORTED("A")
- INITIATED("I")
- REINITIATED("R")

For any event online or batch, when it is logged for first time it is marked as Generated "G" in flx_ep_evt_log_b table.

Figure 11–16 Event Processing Status Type

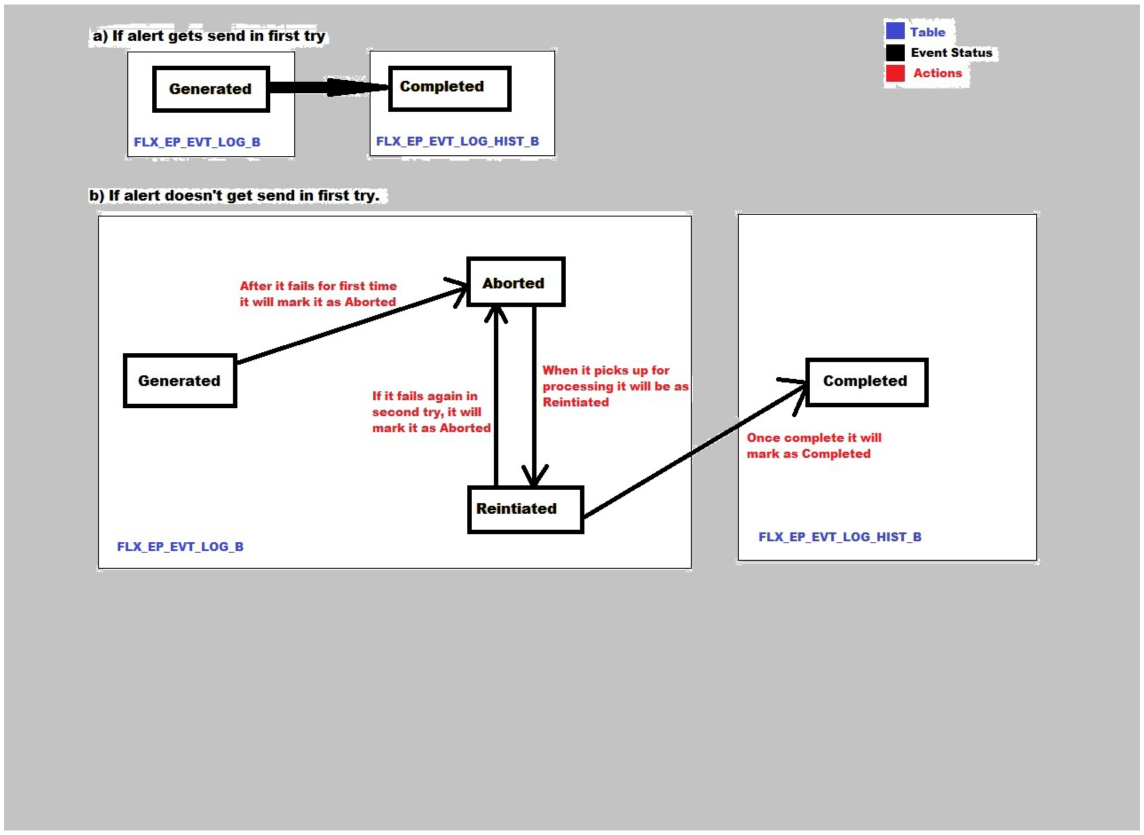


JMS (Java Messaging Service) is used for dispatch of alerts.

For Online Alerts:

- **Direct Approach**: If alert gets send in first try, flg_process_stat is as "G" in FLX_EP_EVT_LOG_B and alert is dispatched through JMS, and then entry for that event record is moved to FLX_EP_EVT_LOG_HIST_B and flg_process_stat is marked as "C".
- **EventPoller**: If alert gets failed in first retry it will mark status as "R". In this case EventPoller will pick the failed event and complete its processing and mark status as "A" and then entry for that event record is moved to FLX_EP_EVT_LOG_HIST_B and flg_process_stat is marked as "C".
- **For Batch Alerts**: In case of batch alerts as no Interaction.close() is called, the direct approach is not used in Batch Alerts. In this case only EventPoller approach is used.

Figure 11–17 Batch Alerts


a) If alert gets send in first try
Generated
Completed
FLX_EP_EVT_LOG_B
FLX_EP_EVT_LOG_HIST_B
Table
Event Status
Actions
b) If alert doesn't get send in first try.
After it fails for first time it will mark it as Aborted
Aborted
Generated
When it picks up for processing it will be as Reintiated
If it fails again in second try, it will mark it as Aborted
Reintiated
Completed
Once complete it will mark as Completed
FLX_EP_EVT_LOG_B
FLX_EP_EVT_LOG_HIST_B

11.4 Alert Dispatch Mechanism

The dispatch mechanism is triggered by the *AlertHandlerService* for dispatching subscribed actions of type *Alert*. The processing is implemented as part of the respective handlers. The handler services delegate the call to the *Dispatcher* based on the type of *DestinationType* configured in the *Recipient* at the time of *ActivityEventAction* maintenance which involves *RecipientMessageTemplate* setup.

The module provides definition of multiple dispatch detail configurations on the basis of *SubscriberType* and various configuration parameters like *UrgencyType*, *ImportantType* in the AlertTemplate.

The dispatcher uses the *DispatchDataConverter* to convert the data captured as part of activity registered in the system into data which can be dispatched to the target subscriber.

Figure 11–18 Alert Dispatch Mechanism

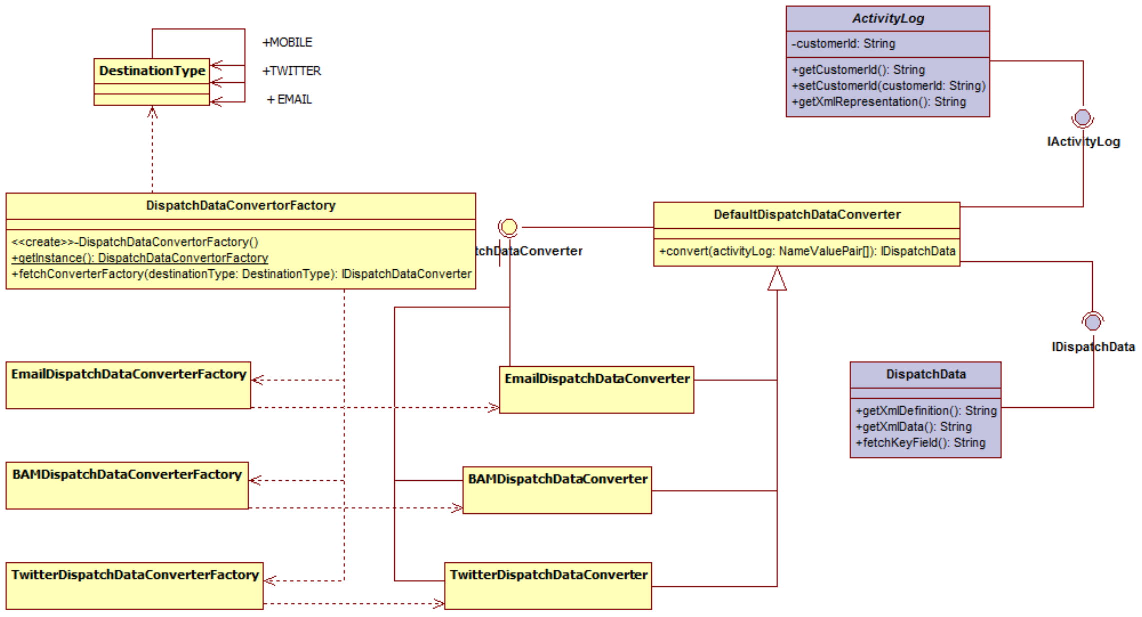

DestinationType
+MOBILE
+TWITTER
+ EMAIL
ActivityLog
-customerId: String
+getCustomerId(): String
+setCustomerId(customerId: String)
+getXmlRepresentation(): String
IActivityLog
DispatchDataConvertorFactory
<<create>>-DispatchDataConvertorFactory()
+getInstance(): DispatchDataConvertorFactory
+fetchConverterFactory(destinationType: DestinationType): IDispatchDataConverter
tchDataConverter
DefaultDispatchDataConverter
+convert(activityLog: NameValuePair[]): IDispatchData
IDispatchData
DispatchData
+getXmlDefinition(): String
+getXmlData(): String
+fetchKeyField(): String
EmailDispatchDataConverterFactory
EmailDispatchDataConverter
BAMDispatchDataConverterFactory
BAMDispatchDataConverter
TwitterDispatchDataConverterFactory
TwitterDispatchDataConverter

Figure 11–19 Alert Dispatch Mechanism - Dispatcher Factory

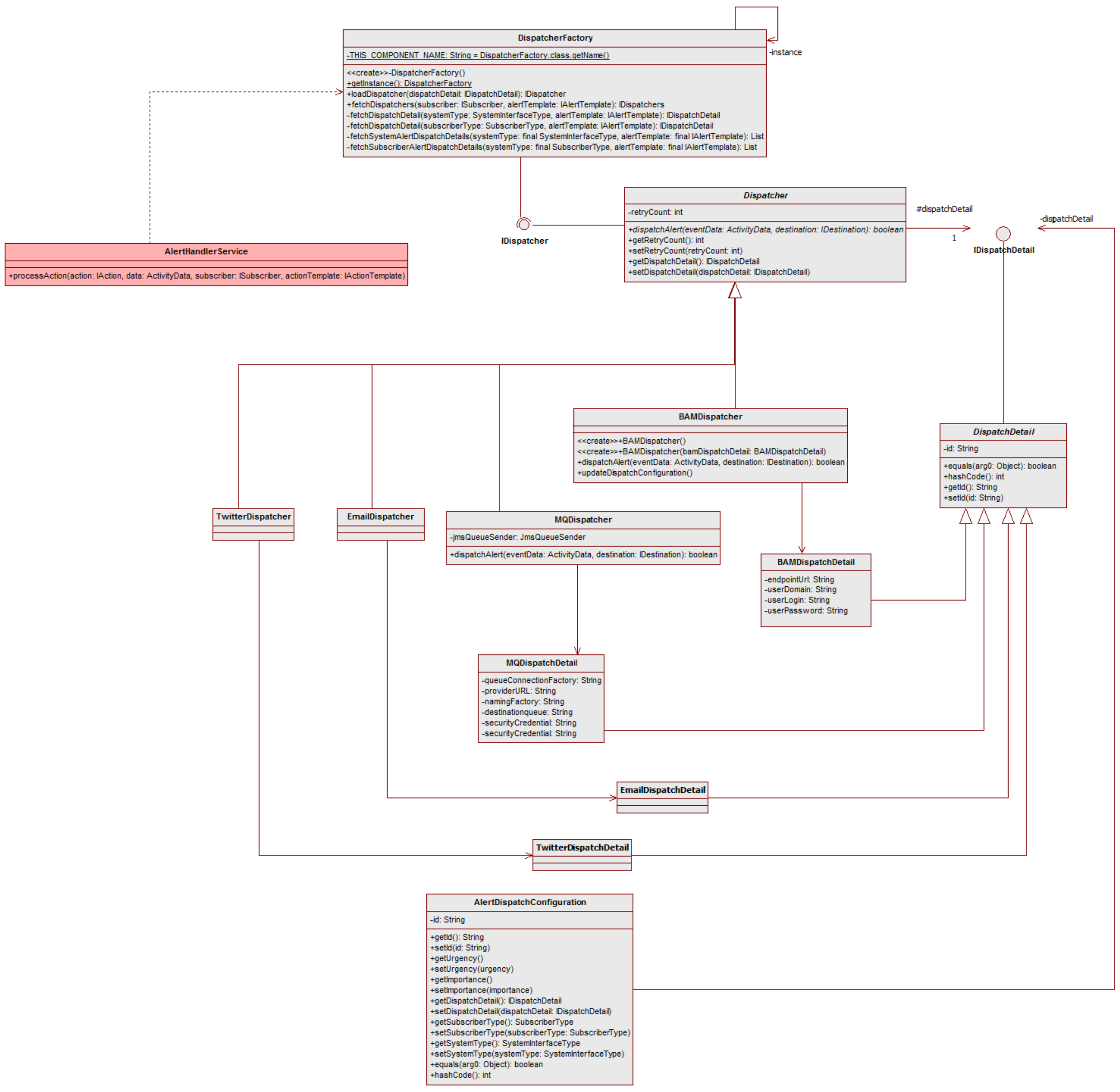
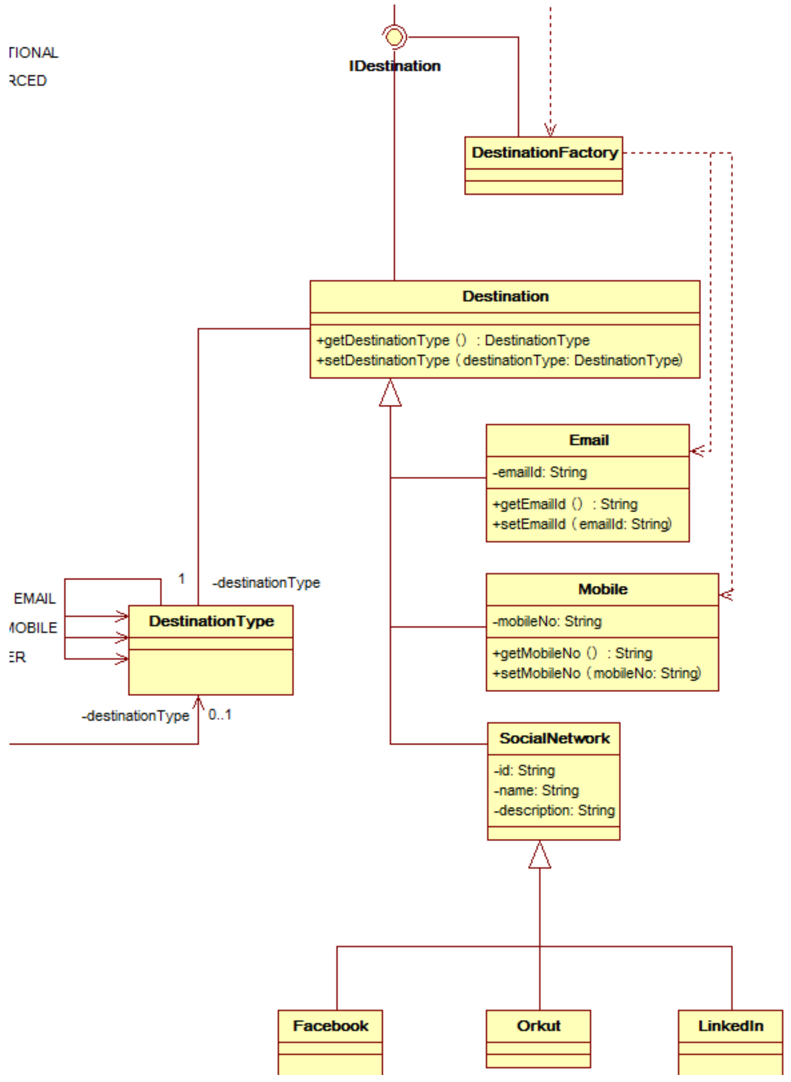

DispatcherFactory
-THIS_COMPONENT_NAME: String = DispatcherFactory.class.getName()
<<create>>-DispatcherFactory()
+getInstance(): DispatcherFactory
+loadDispatcher(dispatchDetail: IDispatchDetail): IDispatcher
+fetchDispatchers(subscriber: ISubscriber, alertTemplate: IAlertTemplate): IDispatchers
-fetchDispatchDetail(systemType: SystemInterfaceType, alertTemplate: IAlertTemplate): IDispatchDetail
-fetchDispatchDetail(subscriberType: SubscriberType, alertTemplate: IAlertTemplate): IDispatchDetail
-fetchSystemAlertDispatchDetails(systemType: final SystemInterfaceType, alertTemplate: final IAlertTemplate): List
-fetchSubscriberAlertDispatchDetails(systemType: final SubscriberType, alertTemplate: final IAlertTemplate): List
-instance
IDispatcher
Dispatcher
-retryCount: int
+dispatchAlert(eventData: ActivityData, destination: IDestination): boolean
+getRetryCount(): int
+setRetryCount(retryCount: int)
+getDispatchDetail(): IDispatchDetail
+setDispatchDetail(dispatchDetail: IDispatchDetail)
#dispatchDetail
1
IDispatchDetail
-dispatchDetail
AlertHandlerService
+processAction(action: IAction, data: ActivityData, subscriber: ISubscriber, actionTemplate: IActionTemplate)
BAMDispatcher
<<create>>+BAMDispatcher()
<<create>>+BAMDispatcher(bamDispatchDetail: BAMDispatchDetail)
+dispatchAlert(eventData: ActivityData, destination: IDestination): boolean
+updateDispatchConfiguration()
DispatchDetail
-id: String
+equals(arg0: Object): boolean
+hashCode(): int
+getId(): String
+setId(id: String)
TwitterDispatcher
EmailDispatcher
MQDispatcher
-jmsQueueSender: JmsQueueSender
+dispatchAlert(eventData: ActivityData, destination: IDestination): boolean
BAMDispatchDetail
-endpointUrl: String
-userDomain: String
-userLogin: String
-userPassword: String
MQDispatchDetail
-queueConnectionFactory: String
-providerURL: String
-namingFactory: String
-destinationqueue: String
-securityCredential: String
-securityCredential: String
EmailDispatchDetail
TwitterDispatchDetail
AlertDispatchConfiguration
-id: String
+getId(): String
+setId(id: String)
+getUrgency()
+setUrgency(urgency)
+getImportance()
+setImportance(importance)
+getDispatchDetail(): IDispatchDetail
+setDispatchDetail(dispatchDetail: IDispatchDetail)
+getSubscriberType(): SubscriberType
+setSubscriberType(subscriberType: SubscriberType)
+getSystemType(): SystemInterfaceType
+setSystemType(systemType: SystemInterfaceType)
+equals(arg0: Object): boolean
+hashCode(): int

Figure 11–20 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.

11.5 Adding New Alerts

To add a new alert:

1. Implement the Service Extension Interface for the application service of the method for which alert is to be raised.
2. Use either the preServiceMethod() or postServiceMethod() hook for the method in the implemented service extension class depending on the requirement.
3. The method should call the registerActivityAndGenerateEvent() of the EventProcessingAdapter class. In case a custom adapter is required the custom adapter method should call

registerActivityAndGenerateEvent() of ActivityRegistrationApplicationService.

4. New Activity ID, Event ID and implementation of IActivityLogDTO have to be created.

11.5.1 New Alert Example

This example will explain the above points in detail.

Use Case: A new alert has to be added after updating a party name.

The class PartyNameApplicationService has a method updateIndividualName() that does this activity.

Create the extension class, say PartyNameApplicationServiceExt, for this application service by implementing its extension interface IPartyNameApplicationServiceExt. Since the alert should be raised after updation of party name we will use the postUpdateIndividualName() method.

Within the method a call to registerActivityAndGenerateEvent() in EventProcessingAdapter should be made.

Code snippet for the call:

```
com.ofss.fc.app.adapter.IAdapterFactory adapterFactory =
AdapterFactoryConfigurator.getInstance().getAdapterFactory
(ModuleConstant.EVENT_PROCESSING);
IEventProcessingAdapter adapter = (IEventProcessingAdapter)
adapterFactory.getAdapter(EventProcessingAdapterConstant.MODULE_TO_
ACTIVITY);
adapter.registerActivityAndGenerateEvent(applicationContext,
activityId, eventId, new Date(), activityLog);
```

In case a new customer adapter has to be used, a call to registerActivityAndGenerateEvent() in ActivityRegistrationApplicationService should be made from within the adapter. A class called ActivityEventKeyDTO is used which captures the event ID and activity ID.

Code snippet for the call:

```
ActivityRegistrationApplicationService activityManager = new
ActivityRegistrationApplicationService();
ActivityEventKeyDTO activityEventKeyDTO = new ActivityEventKeyDTO
();
activityEventKeyDTO.setActivityId(activityID);
activityEventKeyDTO.setEventId(eventID);
ActivityRegistrationResponse response =
activityManager.registerActivityAndGenerateEvent
(sessionContext,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;
}
```

11.5.2 Testing New Alert

JUnit test cases can be used to test the alert created by supplying sample input data. The example below shows how the above new alert can be tested.

```
public void testPartyUpdateName() throws IOException {
String testCase = "PartyUpdateName";
ActivityRegistrationApplicationService
activityRegistrationApplicationService
= new ActivityRegistrationApplicationService();
ActivityEventKeyDTO activityEventKeyDTO = new ActivityEventKeyDTO
("com.ofss.fc.app.party.service.contact.
PartyNameApplicationService.updateIndividualName "," UPDATED_PARTY_
NAME");
Date date = new Date();
SessionContext sessionContext = getSessionContext();
com.ofss.fc.app.party.dto.alert.PartyNameChangeLogDTO activityLog
= new com.ofss.fc.app.party.dto.alert.PartyNameChangeLogDTO ();
activityLog.setUpdatedName("Andrew Matthew");
activityLog.setRegisteredOldName("Andy Matthew");
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.

11.6 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.

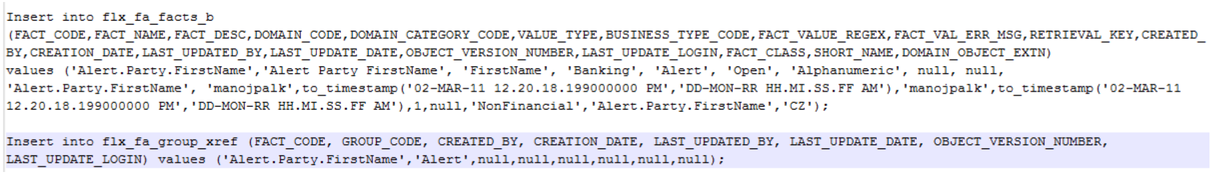
In Facts Module there is a provision to co-relate different Facts and derive the value of one Fact based on the value of the related Fact. This is done by maintaining the relationship in certain Fact tables.

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.

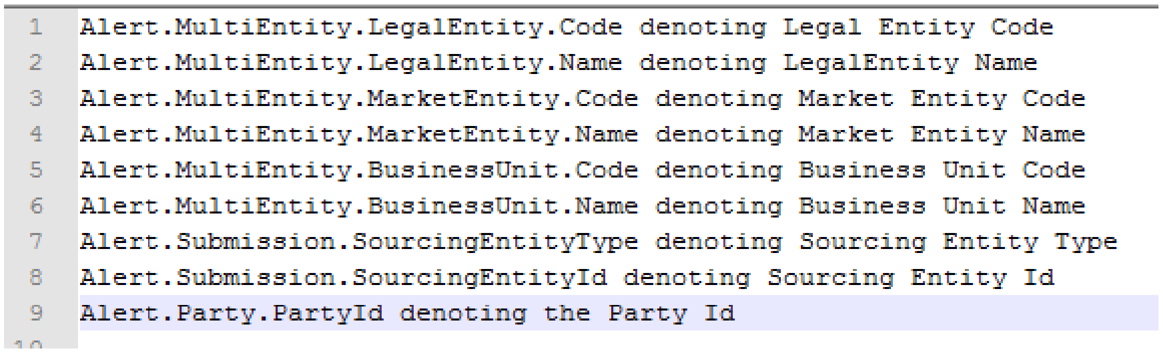
Figure 11–21 Alert.Party.FirstName



```
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 11–22 Facts in Alerts Framework



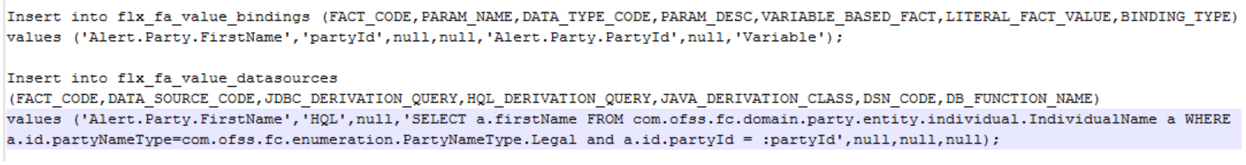
```
Alert.MultiEntity.LegalEntity.Code denoting Legal Entity Code
Alert.MultiEntity.LegalEntity.Name denoting LegalEntity Name
Alert.MultiEntity.MarketEntity.Code denoting Market Entity Code
Alert.MultiEntity.MarketEntity.Name denoting Market Entity Name
Alert.MultiEntity.BusinessUnit.Code denoting Business Unit Code
Alert.MultiEntity.BusinessUnit.Name denoting Business Unit Name
Alert.Submission.SourcingEntityType denoting Sourcing Entity Type
Alert.Submission.SourcingEntityId denoting Sourcing Entity Id
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 11–23 Alert.Party.PartyId

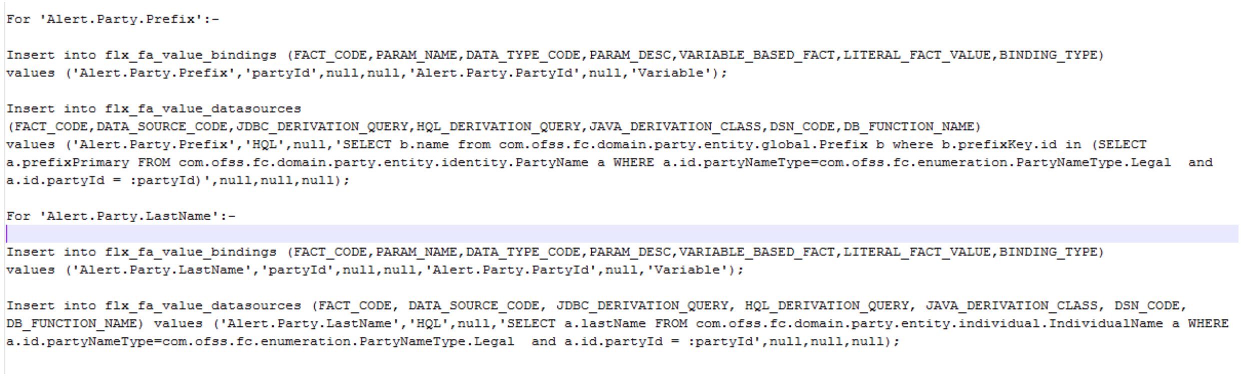


```
Insert into flx_fa_value_bindings (FACT_CODE,PARAM_NAME,DATA_TYPE_CODE,PARAM_DESC,VARIABLE_BASED_FACT,LITERAL_FACT_VALUE,BINDING_TYPE)
values ('Alert.Party.FirstName','partyId',null,null,'Alert.Party.PartyId',null,'Variable');

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.

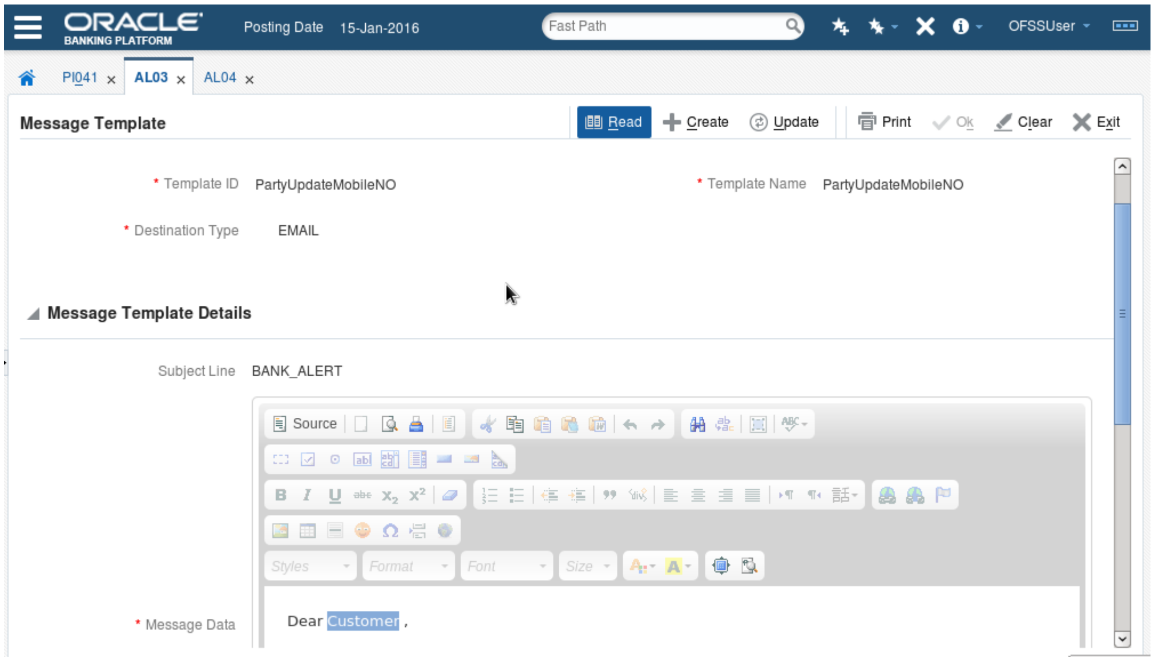
Figure 11–24 Alert.Party.Prefix and Alert.Party.LastName



```
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 11–25 Message Template (Fast Path: AL03)



Message Template

* Template ID PartyUpdateMobileNO

* Template Name PartyUpdateMobileNO

* Destination Type EMAIL

Message Template Details

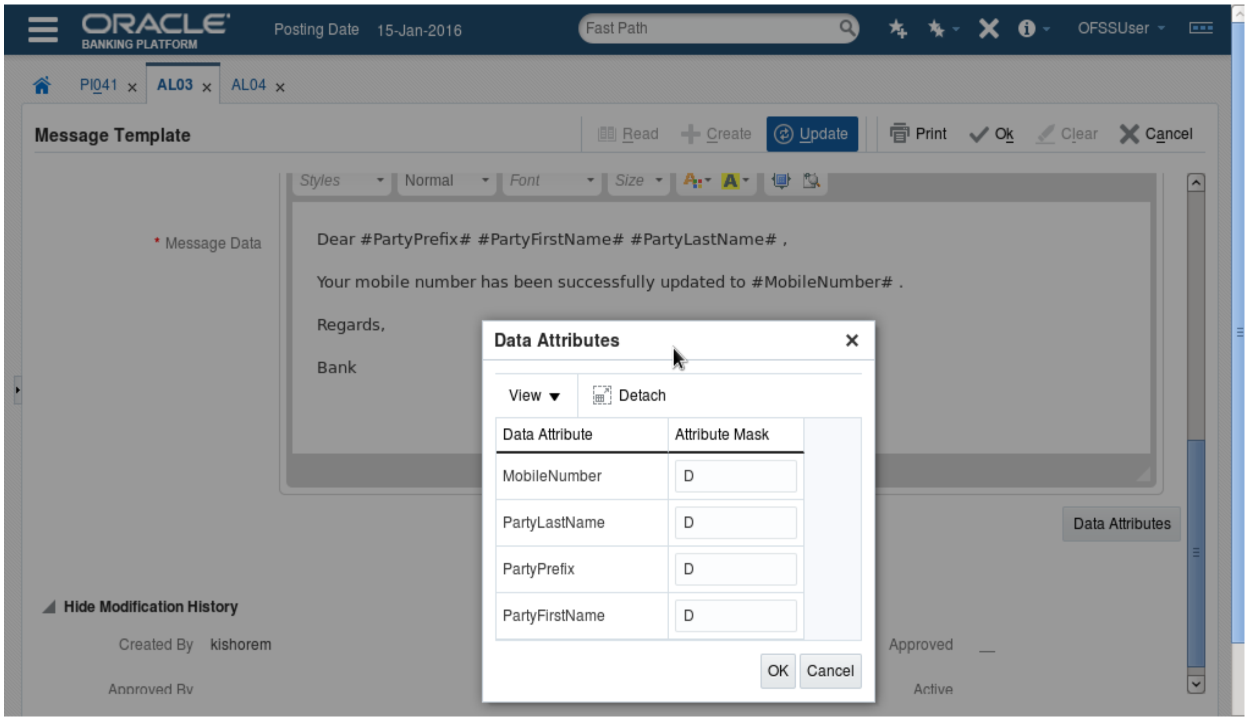
Subject Line BANK_ALERT

* Message Data

Dear Customer ,

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

Figure 11–26 Placeholder for Derived Facts



Message Template

* Message Data

Dear #PartyPrefix# #PartyFirstName# #PartyLastName# ,

Your mobile number has been successfully updated to #MobileNumber# .

Regards,

Bank

Data Attributes

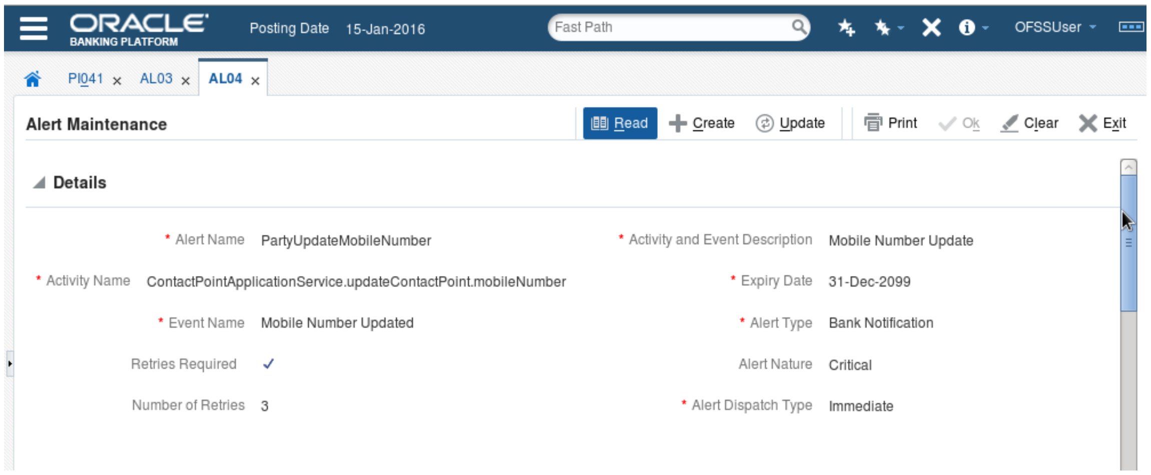
Data Attribute	Attribute Mask
MobileNumber	D
PartyLastName	D
PartyPrefix	D
PartyFirstName	D

Created By kishorem

Approved By

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 11–27 Alert Maintenance (Fast Path: AL04)

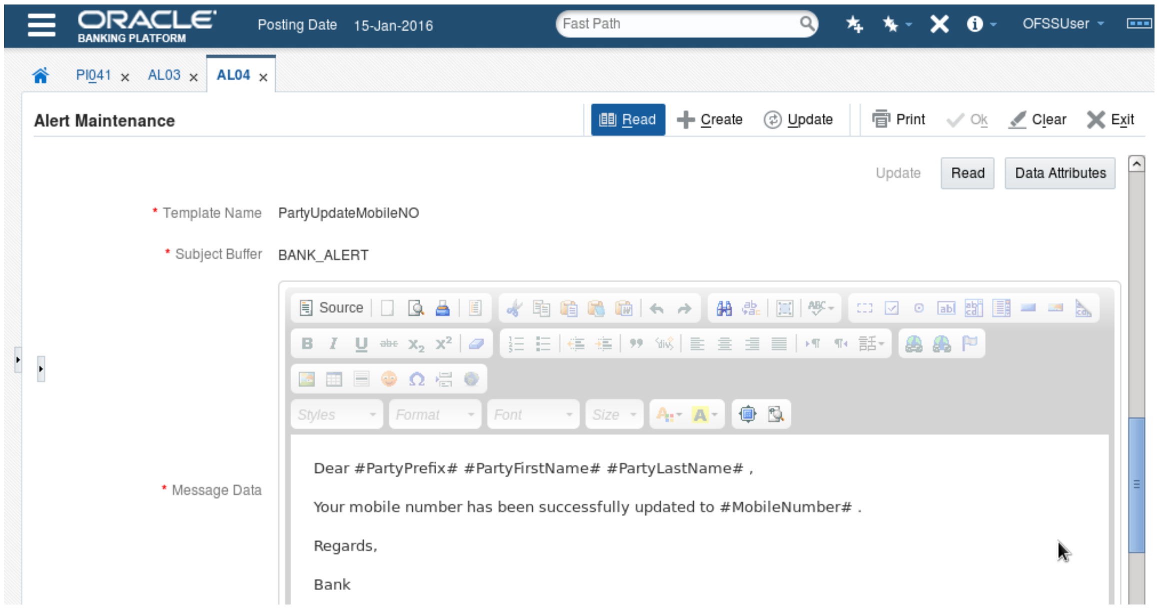


Alert Maintenance

Details

* Alert Name	PartyUpdateMobileNumber	* Activity and Event Description	Mobile Number Update
* Activity Name	ContactPointApplicationService.updateContactPoint.mobileNumber	* Expiry Date	31-Dec-2099
* Event Name	Mobile Number Updated	* Alert Type	Bank Notification
Retries Required	✓	Alert Nature	Critical
Number of Retries	3	* Alert Dispatch Type	Immediate

Figure 11–28 Alert Maintenance - Map the New Message Template Placeholders



Alert Maintenance

* Template Name PartyUpdateMobileNO
* Subject Buffer BANK_ALERT
* Message Data

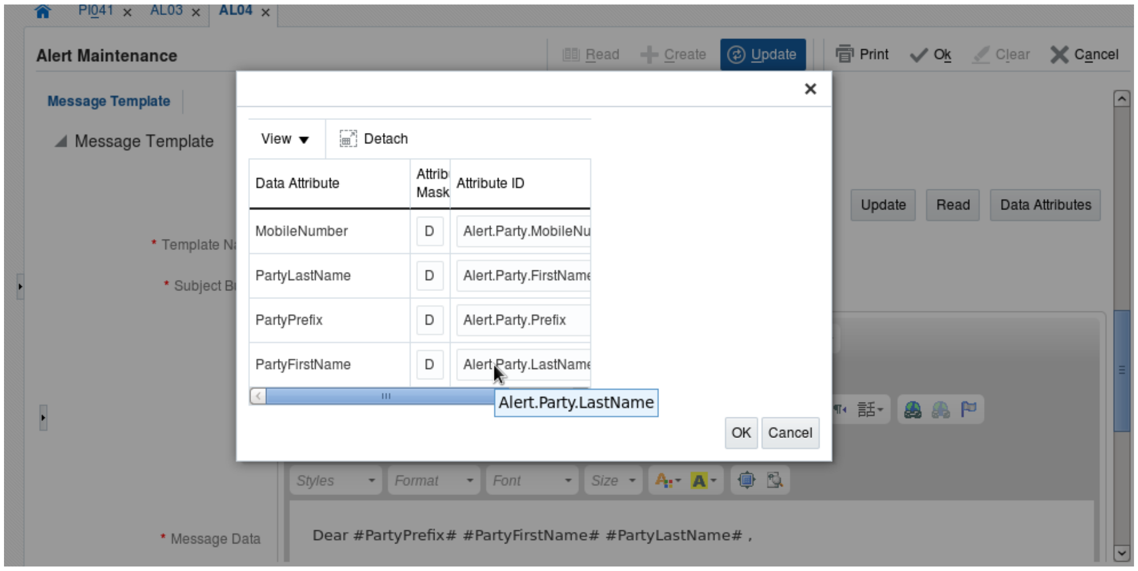
Dear #PartyPrefix# #PartyFirstName# #PartyLastName# ,

Your mobile number has been successfully updated to #MobileNumber# .

Regards,

Bank

Figure 11–29 Alert Maintenance - Facts List



Alert Maintenance

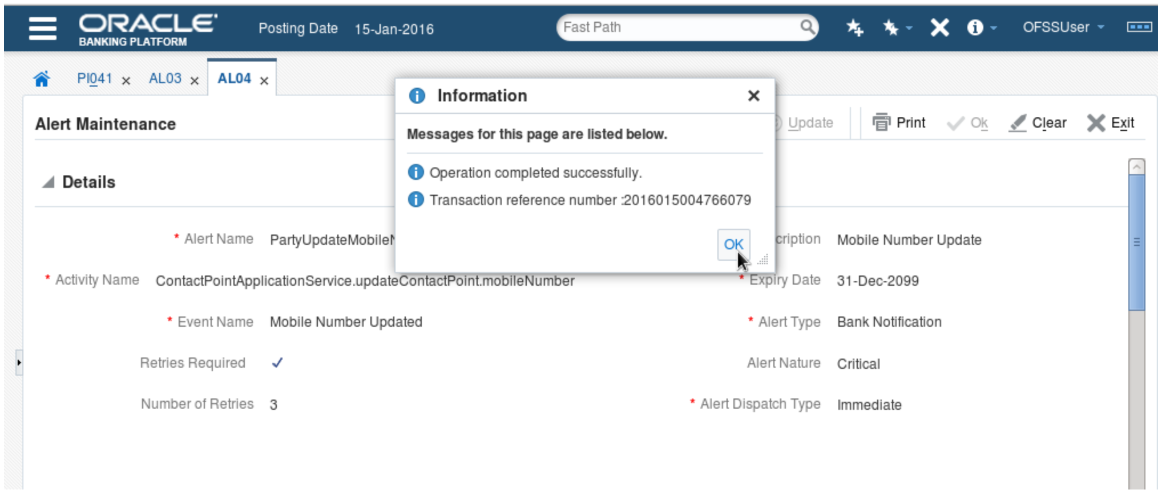
Data Attribute	Attrib Mask	Attribute ID
MobileNumber	D	Alert.Party.MobileNu
PartyLastName	D	Alert.Party.FirstName
PartyPrefix	D	Alert.Party.Prefix
PartyFirstName	D	Alert.Party.LastName

Alert.Party.LastName

* Template N
* Subject B
* Message Data

Dear #PartyPrefix# #PartyFirstName# #PartyLastName# ,

Figure 11–30 Alert Maintenance - Mapping Completed



Alert Maintenance

Details

Information

Messages for this page are listed below.

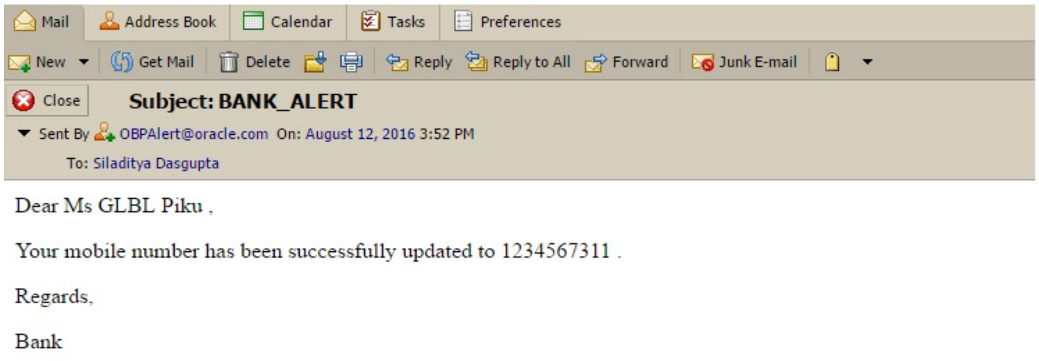
Operation completed successfully.

Transaction reference number :2016015004766079

* Alert Name	PartyUpdateMobileN	cription	Mobile Number Update
* Activity Name	ContactPointApplicationService.updateContactPoint.mobileNumber	* Expiry Date	31-Dec-2099
* Event Name	Mobile Number Updated	* Alert Type	Bank Notification
Retries Required	✓	Alert Nature	Critical
Number of Retries	3	* Alert Dispatch Type	Immediate

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 11–31 Alert Mail on Mobile Number Update in Contact Point screen



Subject: BANK_ALERT

Sent By OBPAlert@oracle.com On: August 12, 2016 3:52 PM

To: Siladitya Dasgupta

Dear Ms GLBL Piku ,

Your mobile number has been successfully updated to 1234567311 .

Regards,

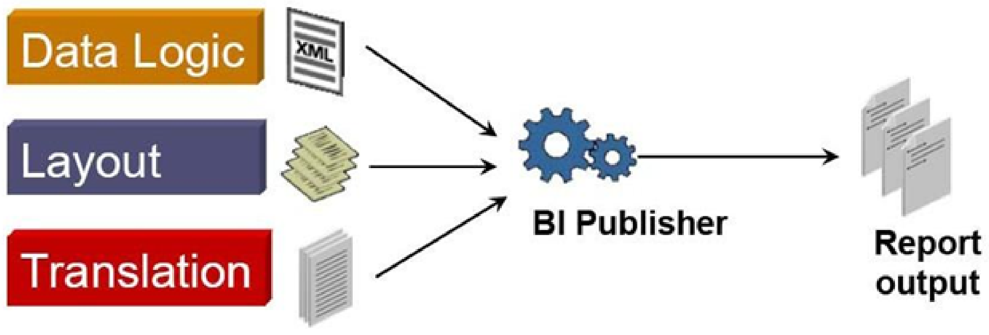
Bank

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.

12 Creating New Reports

Oracle's Business Intelligence Publisher Enterprise is a standalone reporting and document output management solution that allows companies to lower the cost of ownership for reporting solutions. BI Publisher Enterprise's (hereafter known as BIP) strength is that it separates the data model from the actual report formatting/layout. BIP relies on 2 fundamental components to create reports, XML data and a template that represents the look and feel of the report. The XML data can be generated from any number of sources and BIP makes accessing data in the proper format easy. Templates can be created in Microsoft Word and Adobe Acrobat allowing almost anyone familiar with these desktop applications the ability to create reports.

Figure 12–1 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.

12.1 Data Objects for the Report

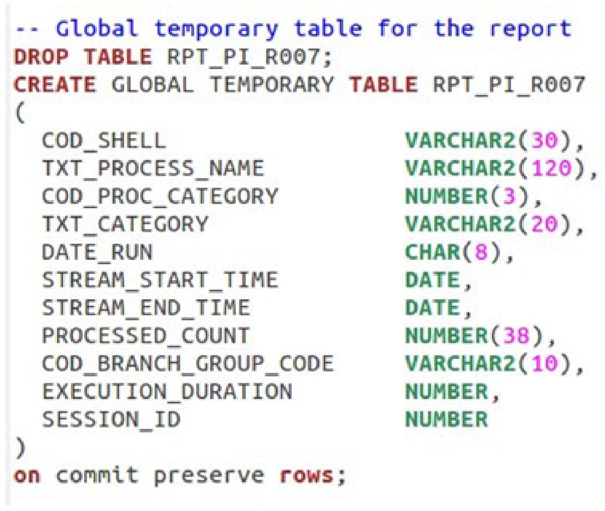
The *Data Model* of the report invokes the database to fetch the data for the report through certain data objects that we will need to create. The primary data objects needed for the reports are as follows:

Global Temporary Table

You will need to create a *Global Temporary Table* based on the fields required for the report data. This table should mandatory have the field *SESSION_ID* of *NUMBER* type. The naming convention followed in OBP for the global temporary table's name is *RPT_<Module_Code>_R<Report_Number>*.

For the aforementioned use case, the script for creating the global temporary table would be as shown below.

Figure 12–2 Global Temporary Table



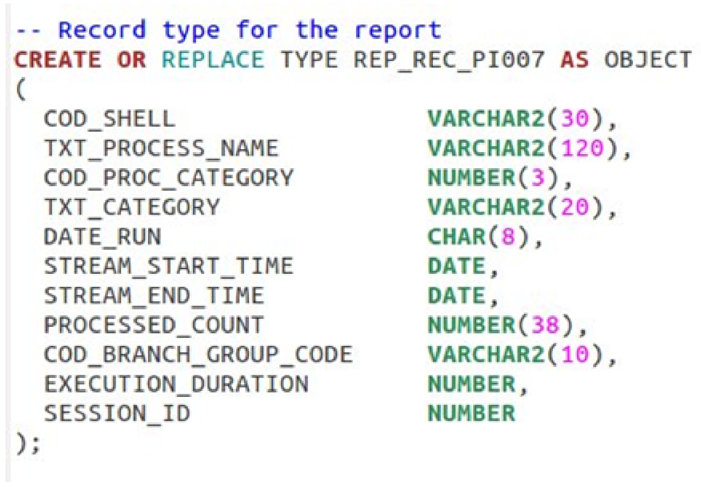
```
-- Global temporary table for the report
DROP TABLE RPT_PI_R007;
CREATE GLOBAL TEMPORARY TABLE RPT_PI_R007
(
  COD_SHELL                  VARCHAR2(30),
  TXT_PROCESS_NAME           VARCHAR2(120),
  COD_PROC_CATEGORY          NUMBER(3),
  TXT_CATEGORY               VARCHAR2(20),
  DATE_RUN                   CHAR(8),
  STREAM_START_TIME          DATE,
  STREAM_END_TIME            DATE,
  PROCESSED_COUNT            NUMBER(38),
  COD_BRANCH_GROUP_CODE      VARCHAR2(10),
  EXECUTION_DURATION         NUMBER,
  SESSION_ID                 NUMBER
)
on commit preserve rows;
```

Report Record Type

You will need to create a *Type* object with the fields present in the global temporary table. This type will represent a single row of data for the report. The naming convention followed in OBP for the report record type's name is *REP_REC_<Report_Id>*.

For the aforementioned use case, the script for creating the report record type would be as shown below.

Figure 12–3 Report Record Type



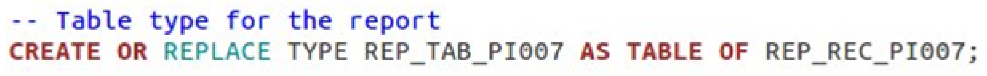
```
-- Record type for the report
CREATE OR REPLACE TYPE REP_REC_PI007 AS OBJECT
(
  COD_SHELL                  VARCHAR2(30),
  TXT_PROCESS_NAME           VARCHAR2(120),
  COD_PROC_CATEGORY          NUMBER(3),
  TXT_CATEGORY               VARCHAR2(20),
  DATE_RUN                   CHAR(8),
  STREAM_START_TIME          DATE,
  STREAM_END_TIME            DATE,
  PROCESSED_COUNT            NUMBER(38),
  COD_BRANCH_GROUP_CODE      VARCHAR2(10),
  EXECUTION_DURATION         NUMBER,
  SESSION_ID                 NUMBER
);
```

Report Table Type

You will need to create a *Type* object which will be a table of the previously created report record type. This type will represent the set of rows of data for the report. The naming convention followed in OBP for the report table type's name is *REC_TAB_<Report_Id>*.

For the aforementioned use case, the script for creating the report table type would be as shown below.

Figure 12–4 Report Table Type



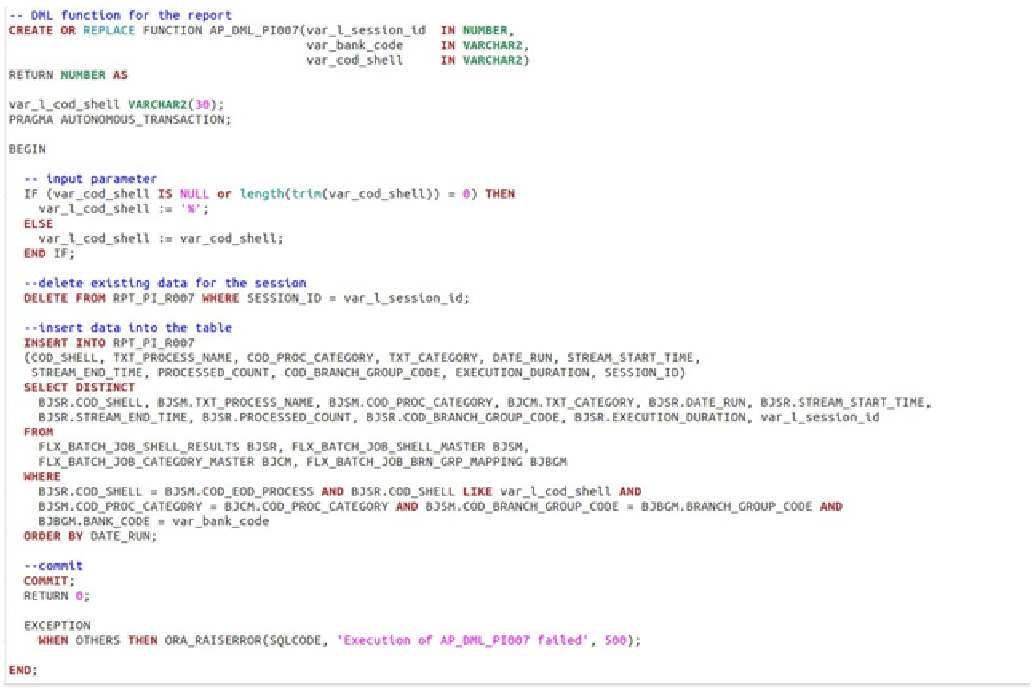
```
-- Table type for the report
CREATE OR REPLACE TYPE REP_TAB_PI007 AS TABLE OF REP_REC_PI007;
```

Report DML Function

You will need to create a DML function which will be invoked to populate the previously created global temporary table with the data required to be displayed in the report. This function can have parameters as per the developer’s requirements with filtering the data or inserting additional data. The naming convention followed in OBP for the report DML function's name is *AP_DML_<Report_Id>.*

For the aforementioned use case, the script for the report DML function would be as shown below.

Figure 12–5 Report DML Function



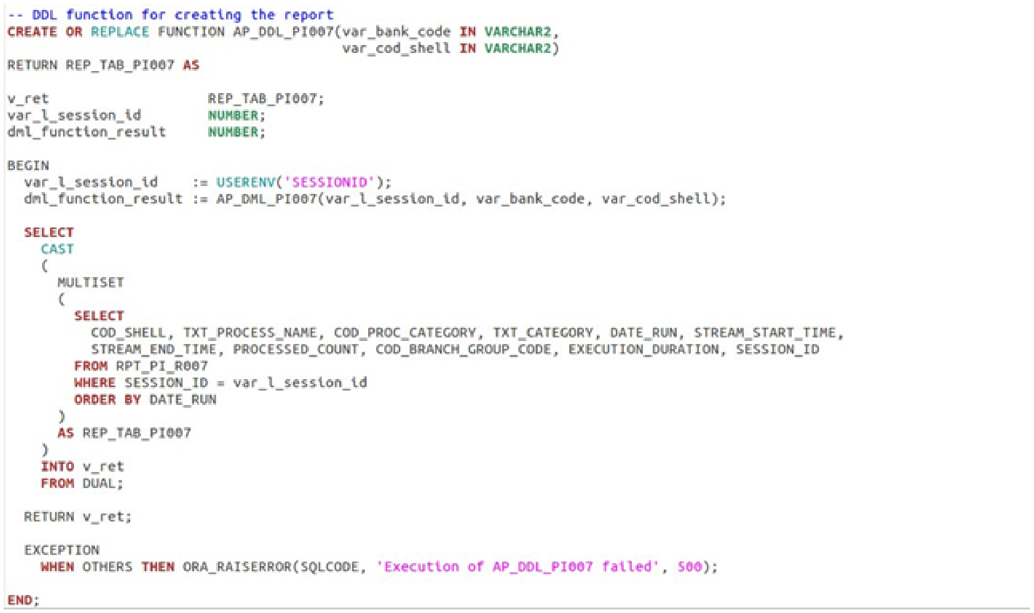
```
-- DML function for the report
CREATE OR REPLACE FUNCTION AP_DML_PI007(var_l_session_id  IN NUMBER,
                                       var_bank_code     IN VARCHAR2,
                                       var_cod_shell     IN VARCHAR2)
RETURN NUMBER AS

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 12–6 Report DDL Function



```
-- DDL function for creating the report
CREATE OR REPLACE FUNCTION AP_DDL_PI007(var_bank_code IN VARCHAR2,
                                        var_cod_shell IN VARCHAR2)
RETURN REP_TAB_PI007 AS

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.

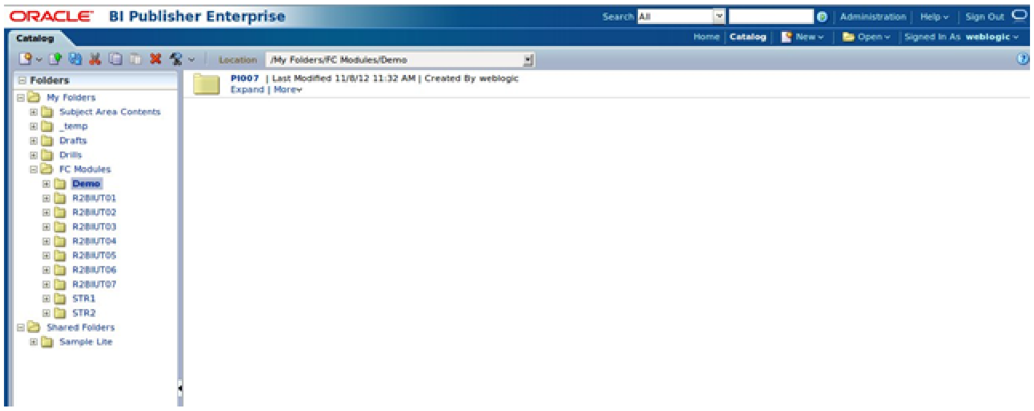
You can log in to the BIP application deployed on http: //<IP ADDRESS><PORT>/xmlpserver/ with the credentials *weblogic/weblogic1*.

12.2 Catalog Folder

Before creating the data model or the layout for the report, you should create a folder to save the model and layout. You can find the link for the Catalog tab on the home screen. Click it and create a folder for your report at an appropriate location.

For the aforementioned use case, you can create a folder *PI007* at the location */My Folders/FC Module/Demo* as shown below.

Figure 12–7 Catalog Folder



Location /My Folders/FC Modules/Demo

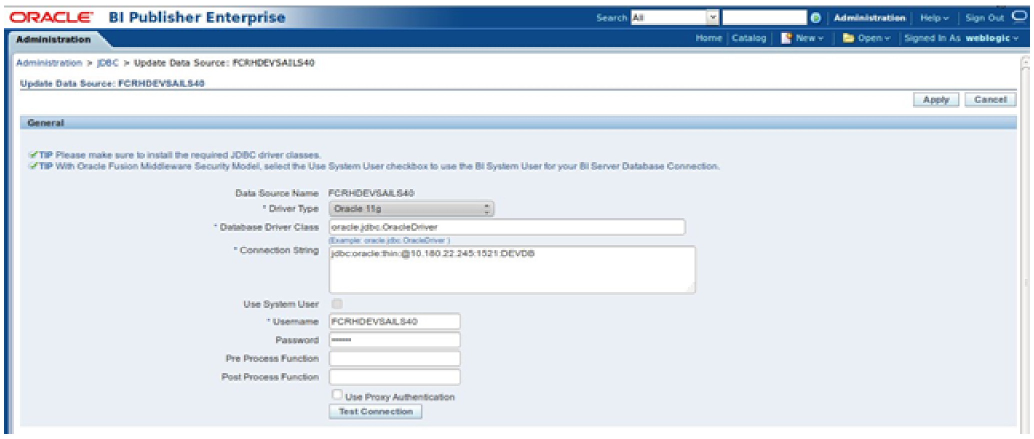
PI007 | Last Modified 11/8/12 11:32 AM | Created By weblogic

12.3 Data Source

You will need to add the data source from which the data will be fetched to be displayed in the report. The data source can be a *JDBC Connection*, *JNDI Connection*, *File*, *LDAP Connection* and so on. You can find the link for the *Administration* tab on the home screen. Click it and choose the appropriate data source connection type. Enter the required parameter values and validate the connection. Save the data source with an appropriate name.

For the aforementioned use case, you can add the JDBC Connection data source as show below.

Figure 12–8 Data Source


ORACLE BI Publisher Enterprise
Search All
Administration | Help | Sign Out
Administration
Home | Catalog | New | Open | Signed In As weblogic
Apply
Cancel
General
Test Connection

Administration > JDBC > Update Data Source: FCRHDEVSAILS40

Update Data Source: FCRHDEVSAILS40

TIP Please make sure to install the required JDBC driver classes.
TIP With Oracle Fusion Middleware Security Model, select the Use System User checkbox to use the BI System User for your BI Server Database Connection.

Data Source Name	FCRHDEVSAILS40
* Driver Type	Oracle 11g
* Database Driver Class	oracle.jdbc.OracleDriver (Example: oracle.jdbc.OracleDriver)
* Connection String	jdbc:oracle:thin:@10.180.22.245:1521:DEVDB
Use System User	☐
* Username	FCRHDEVSAILS40
Password	•••••••
Pre Process Function	
Post Process Function	
	☐ Use Proxy Authentication

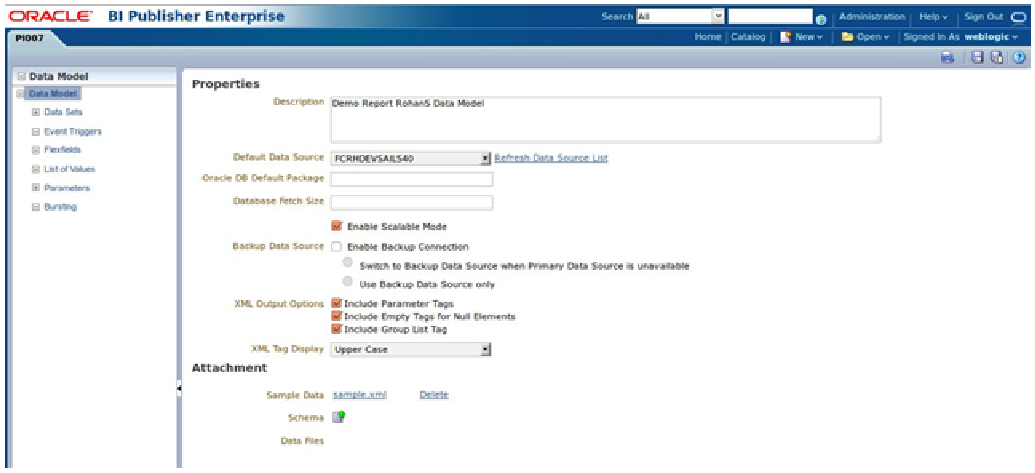
12.4 Data Model

You will need to create a data model to back the report. This data model represents the report data fetched using the data objects and formatted into XML data. You can find the link to *Create Data Model* on the home screen of BIP. Click it and follow these steps:

1. Enter an appropriate *description* for the data model.
2. Choose the previously created *data source* from the list displayed.
3. Check the Enable Scalable Model option.
4. Check the Include Parameter Tags option.
5. Check the Include Empty Tags for Null Elements option.
6. Check the Include Group List Tags option.
7. You can leave the rest of the options to default.

For the aforementioned use case, you can create data model as shown below.

Figure 12–9 Data Model


ORACLE BI Publisher Enterprise
Search All
Administration | Help | Sign Out
Home | Catalog | New | Open | Signed In As weblogic
PI007
Data Model
Data Model
Data Sets
Event Triggers
Flexfields
List of Values
Parameters
Bursting

Properties

Description: Demo Report RohanS Data Model

Default Data Source: FCRHDEVSAILS40 Refresh Data Source List

Oracle DB Default Package:

Database Fetch Size:

[✓] Enable Scalable Mode

Backup Data Source: [ ] Enable Backup Connection

○ Switch to Backup Data Source when Primary Data Source is unavailable

○ Use Backup Data Source only

XML Output Options: [✓] Include Parameter Tags
[✓] Include Empty Tags for Null Elements
[✓] Include Group List Tag

XML Tag Display: Upper Case

Attachment

Sample Data: sample.xml Delete

Schema

Data Files

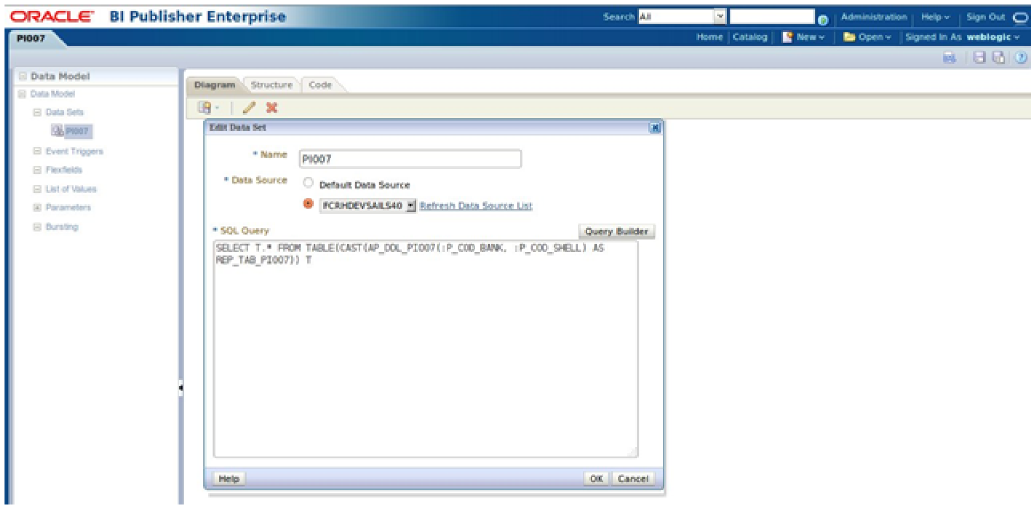
Data Set

After creating the data model, you will need to create a data set of the fields required to be displayed in the report. You can find the link for *Data Sets* on the left side pane of the screen. To create the data set, follow these steps:

1. In the Create Data Set icon, choose the option Create Data Set from SQL Query.
2. Enter an appropriate *name* for the data set.
3. Choose the previously created *data source* from the list displayed.
4. Enter the SQL query which will be used to fetch the data for the report. The results returned should be of the *Report Table Type* previously created.

For the aforementioned use case, you can create the data set as shown below.

Figure 12–10 Data Set



* Name: PI007

* Data Source: ○ Default Data Source
● FCRHDEVSAILS40 Refresh Data Source List

* SQL Query

```
SELECT T.* FROM TABLE(CAST(AP_DDL_PI007(:P_COD_BANK, :P_COD_SHELL) AS
REP_TAB_PI007)) T
```

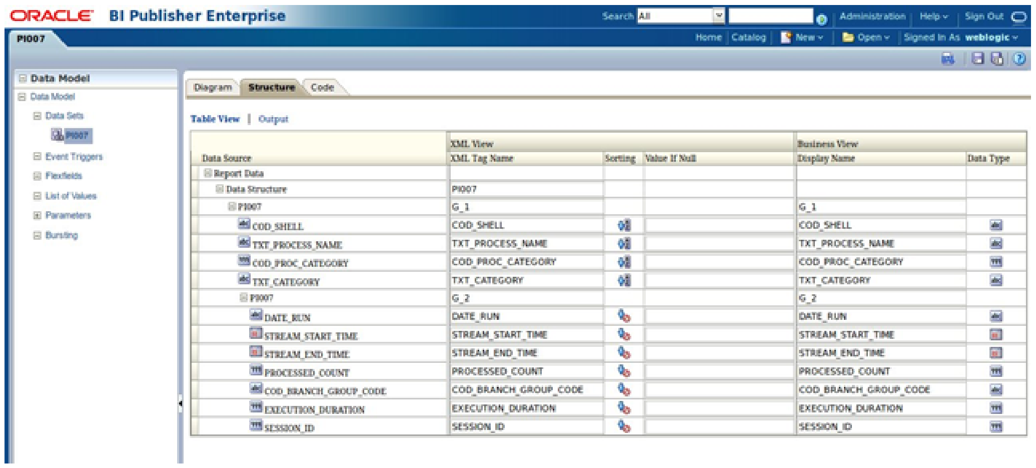
On click of OK, a data set will be created with all the fields as defined in the previously created *Report Record Type*.

You can group the fields as per the requirements of the report:

1. Select the field on which you want to group and choose *Group By*.
2. After creating a group, you can move fields between the groups.
3. You can also set field which will be used to sort the data displayed in a group.

For the aforementioned use case, you can group the fields as shown below.

Figure 12–11 Group Fields

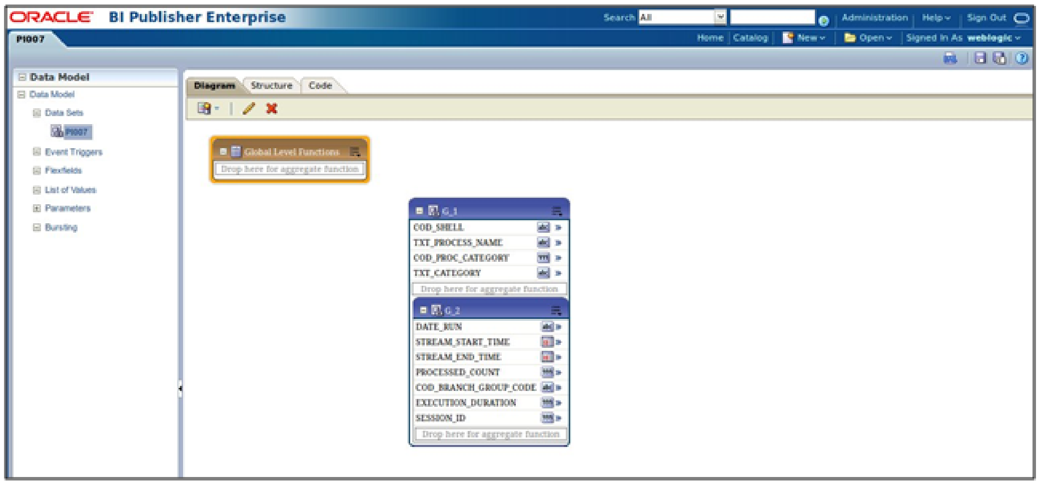


	XML View			Business View	
Data Source	XML Tag Name	Sorting	Value If Null	Display Name	Data Type
Report Data					
Data Structure	PI007				
PI007	G_1			G_1	
COD_SHELL	COD_SHELL			COD_SHELL	
TXT_PROCESS_NAME	TXT_PROCESS_NAME			TXT_PROCESS_NAME	
COD_PROC_CATEGORY	COD_PROC_CATEGORY			COD_PROC_CATEGORY	
TXT_CATEGORY	TXT_CATEGORY			TXT_CATEGORY	
PI007	G_2			G_2	
DATE_RUN	DATE_RUN			DATE_RUN	
STREAM_START_TIME	STREAM_START_TIME			STREAM_START_TIME	
STREAM_END_TIME	STREAM_END_TIME			STREAM_END_TIME	
PROCESSED_COUNT	PROCESSED_COUNT			PROCESSED_COUNT	
COD_BRANCH_GROUP_CODE	COD_BRANCH_GROUP_CODE			COD_BRANCH_GROUP_CODE	
EXECUTION_DURATION	EXECUTION_DURATION			EXECUTION_DURATION	
SESSION_ID	SESSION_ID			SESSION_ID	

You can view and edit the XML structure and labels of the report data in the *Structure* tab in a tabular format.

For the aforementioned use case, the structure would be as shown below:

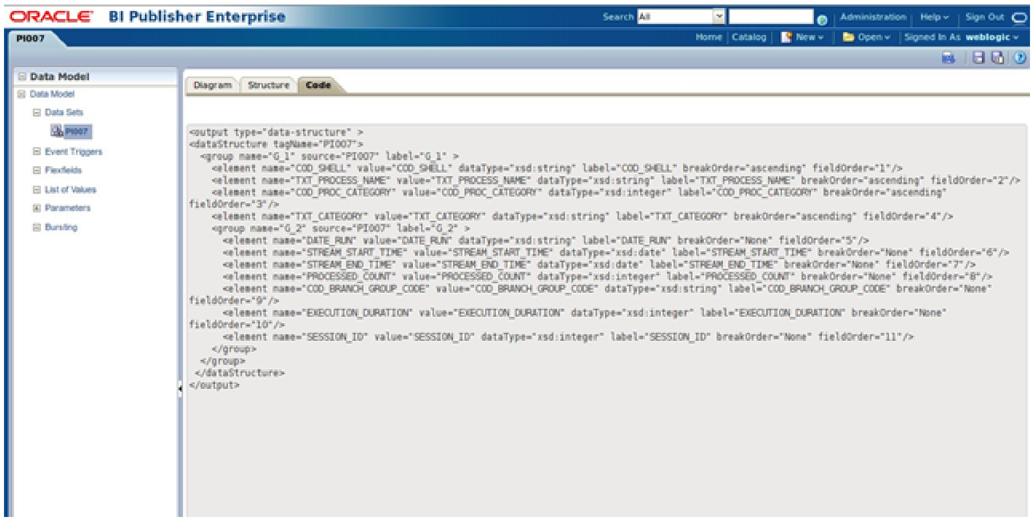
Figure 12–12 XML Structure and Labels



You can view the actual XML code in the *Code* tab.

For the aforementioned use case, the XML code would be as shown below.

Figure 12–13 XML Code



```
<output type="data-structure" >
<dataStructure tagName="PI007">
  <group name="G_1" source="PI007" label="G_1" >
    <element name="COD_SHELL" value="COD_SHELL" dataType="xsd:string" label="COD_SHELL" breakOrder="ascending" fieldOrder="1"/>
    <element name="TXT_PROCESS_NAME" value="TXT_PROCESS_NAME" dataType="xsd:string" label="TXT_PROCESS_NAME" breakOrder="ascending" fieldOrder="2"/>
    <element name="COD_PROC_CATEGORY" value="COD_PROC_CATEGORY" dataType="xsd:integer" label="COD_PROC_CATEGORY" breakOrder="ascending"
fieldOrder="3"/>
    <element name="TXT_CATEGORY" value="TXT_CATEGORY" dataType="xsd:string" label="TXT_CATEGORY" breakOrder="ascending" fieldOrder="4"/>
    <group name="G_2" source="PI007" label="G_2" >
      <element name="DATE_RUN" value="DATE_RUN" dataType="xsd:string" label="DATE_RUN" breakOrder="None" fieldOrder="5"/>
      <element name="STREAM_START_TIME" value="STREAM_START_TIME" dataType="xsd:date" label="STREAM_START_TIME" breakOrder="None" fieldOrder="6"/>
      <element name="STREAM_END_TIME" value="STREAM_END_TIME" dataType="xsd:date" label="STREAM_END_TIME" breakOrder="None" fieldOrder="7"/>
      <element name="PROCESSED_COUNT" value="PROCESSED_COUNT" dataType="xsd:integer" label="PROCESSED_COUNT" breakOrder="None" fieldOrder="8"/>
      <element name="COD_BRANCH_GROUP_CODE" value="COD_BRANCH_GROUP_CODE" dataType="xsd:string" label="COD_BRANCH_GROUP_CODE" breakOrder="None"
fieldOrder="9"/>
      <element name="EXECUTION_DURATION" value="EXECUTION_DURATION" dataType="xsd:integer" label="EXECUTION_DURATION" breakOrder="None"
fieldOrder="10"/>
      <element name="SESSION_ID" value="SESSION_ID" dataType="xsd:integer" label="SESSION_ID" breakOrder="None" fieldOrder="11"/>
    </group>
  </group>
 </dataStructure>
</output>
```

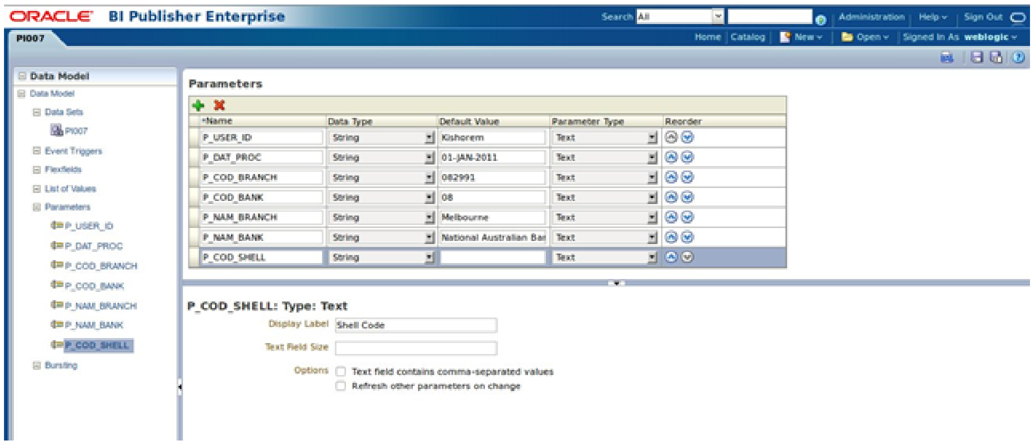
Input Parameters

You can define the *Input Parameters* required by the report in the *Parameters* tab present on the left hand side pane of the screen. To define input parameters, follow these steps:

1. In the **Parameters** tab, click the icon for *Add Parameter*.
2. Enter the name, type, display label and default value for the parameter.
3. Repeat the above steps to define as many parameters as required.

For the aforementioned use case, you can add parameters as shown below:

Figure 12–14 Add Input Parameters



Parameters

*Name	Data Type	Default Value	Parameter Type
P_USER_ID	String	Kishorem	Text
P_DAT_PROC	String	01-JAN-2011	Text
P_COD_BRANCH	String	082991	Text
P_COD_BANK	String	08	Text
P_NAM_BRANCH	String	Melbourne	Text
P_NAM_BANK	String	National Australian Bar	Text
P_COD_SHELL	String		Text

P_COD_SHELL: Type: Text

Display Label: Shell Code

Text Field Size:

Options: ☐ Text field contains comma-separated values ☐ Refresh other parameters on change

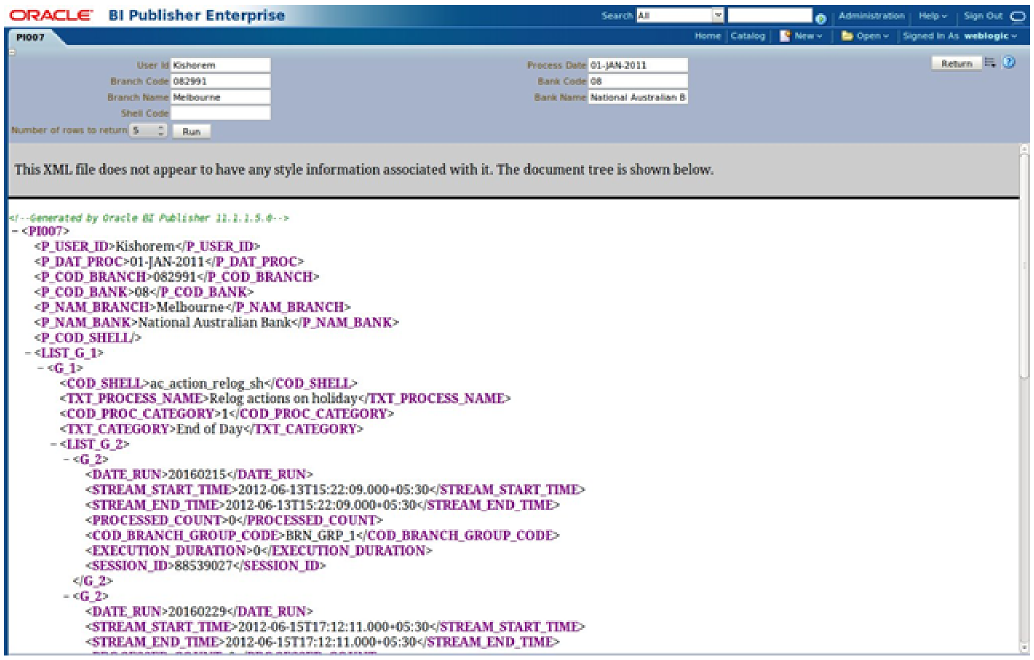
12.5 XML View of Report

After following the above steps, save the data model in the previously created catalog folder with an appropriate name. You can view the report without the layout in the XML form by clicking on the icon for *XML View*.

In the XML view, you will see input fields for the previously defined *input parameters*. Enter appropriate values in those fields and click *Run*. You will be able to see the XML representation of the report data.

For the aforementioned use case, the XML representation of the report data would be as shown below.

Figure 12–15 XML View of Report



User Id	Kishorem	Process Date	01-JAN-2011
Branch Code	082991	Bank Code	08
Branch Name	Melbourne	Bank Name	National Australian B
Shell Code			
Number of rows to return	5		

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!--Generated by Oracle BI Publisher 11.1.1.5.0-->
- <PI007>
    <P_USER_ID>Kishorem</P_USER_ID>
    <P_DAT_PROC>01-JAN-2011</P_DAT_PROC>
    <P_COD_BRANCH>082991</P_COD_BRANCH>
    <P_COD_BANK>08</P_COD_BANK>
    <P_NAM_BRANCH>Melbourne</P_NAM_BRANCH>
    <P_NAM_BANK>National Australian Bank</P_NAM_BANK>
    <P_COD_SHELL/>
  - <LIST_G_1>
    - <G_1>
        <COD_SHELL>ac_action_relog_sh</COD_SHELL>
        <TXT_PROCESS_NAME>Relog actions on holiday</TXT_PROCESS_NAME>
        <COD_PROC_CATEGORY>1</COD_PROC_CATEGORY>
        <TXT_CATEGORY>End of Day</TXT_CATEGORY>
      - <LIST_G_2>
        - <G_2>
            <DATE_RUN>20160215</DATE_RUN>
            <STREAM_START_TIME>2012-06-13T15:22:09.000+05:30</STREAM_START_TIME>
            <STREAM_END_TIME>2012-06-13T15:22:09.000+05:30</STREAM_END_TIME>
            <PROCESSED_COUNT>0</PROCESSED_COUNT>
            <COD_BRANCH_GROUP_CODE>BRN_GRP_1</COD_BRANCH_GROUP_CODE>
            <EXECUTION_DURATION>0</EXECUTION_DURATION>
            <SESSION_ID>88539027</SESSION_ID>
          </G_2>
        - <G_2>
            <DATE_RUN>20160229</DATE_RUN>
            <STREAM_START_TIME>2012-06-15T17:12:11.000+05:30</STREAM_START_TIME>
            <STREAM_END_TIME>2012-06-15T17:12:11.000+05:30</STREAM_END_TIME>
```

12.6 Layout of the Report

A report needs to be presented in an appropriate format. The format can vary from report to report and client to client. BIP separates the data model from the layout making it convenient for the developer.

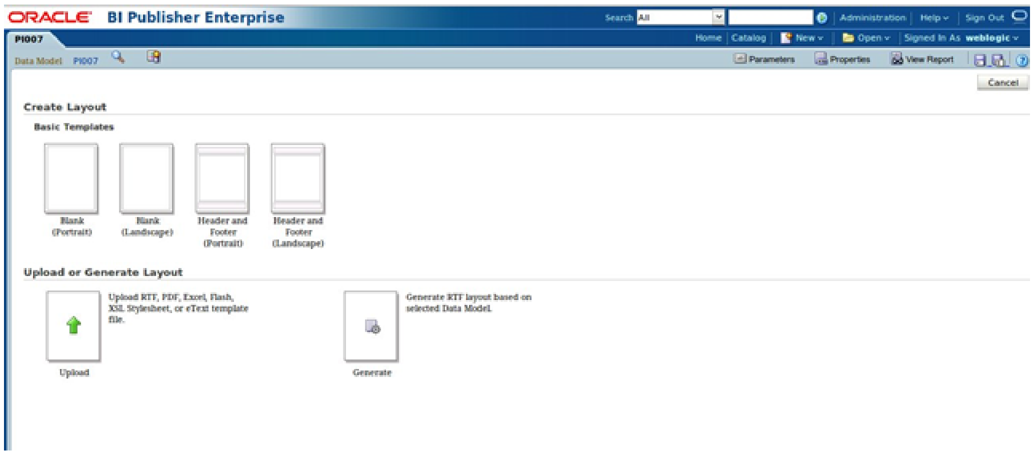
Anybody familiar with using Microsoft Word or Adobe Acrobat can use the corresponding plug-ins for these tools to create a layout for a report. You can create a rich layout using these standalone applications with BIP plug-ins and then upload them to the BIP application for use in your report.

The BIP application can generate a very basic layout for your report from the data set. You can download the generated layout, modify it as per your layout requirements and upload it to the BIP application for use in your report.

The BIP application also allows the user to create a layout on the web. It has a rich set of tools to with drag and drop features and a ready link to the data set fields. You can create a layout in this fashion and use it in your report.

You can find the link to *Add New Layout* on the right side of the screen. Click it to get the options to *create*, *generate* or *upload* a layout.

Figure 12–16 Layout of the Report - Create Layout

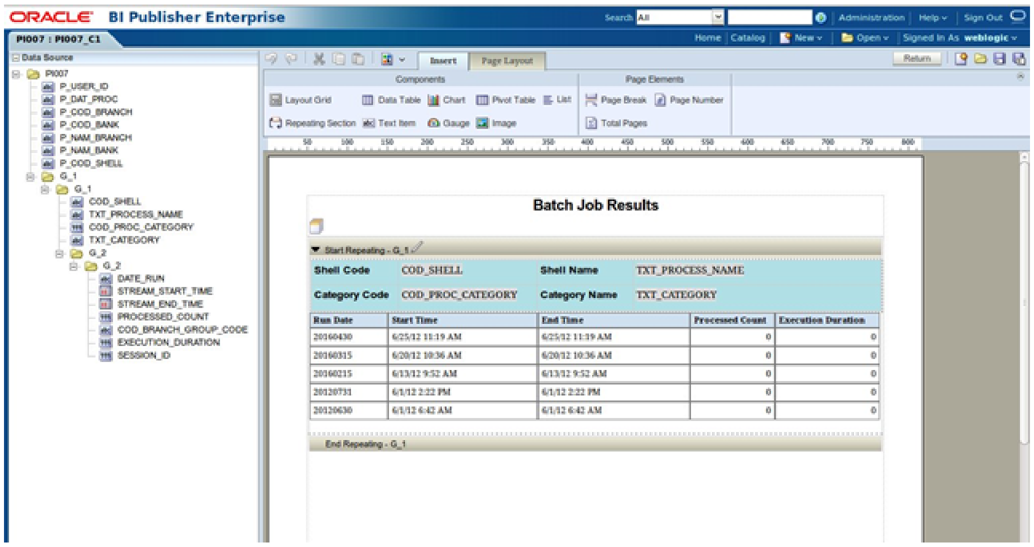

ORACLE BI Publisher Enterprise
Create Layout
Basic Templates
Blank (Portrait)
Blank (Landscape)
Header and Footer (Portrait)
Header and Footer (Landscape)
Upload or Generate Layout
Upload RTF, PDF, Excel, Flash, XSL Stylesheet, or eText template file.
Upload
Generate RTF layout based on selected Data Model.
Generate
Cancel

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 12–17 Layout of the Report - Batch Job Results



Batch Job Results

Start Repeating - G_1

Shell Code	COD_SHELL	Shell Name	TXT_PROCESS_NAME
Category Code	COD_PROC_CATEGORY	Category Name	TXT_CATEGORY

Run Date	Start Time	End Time	Processed Count	Execution Duration
20160430	6/25/12 11:19 AM	6/25/12 11:19 AM	0	0
20160315	6/20/12 10:36 AM	6/20/12 10:36 AM	0	0
20160215	6/13/12 9:52 AM	6/13/12 9:52 AM	0	0
20120731	6/1/12 2:22 PM	6/1/12 2:22 PM	0	0
20120630	6/1/12 6:42 AM	6/1/12 6:42 AM	0	0

End Repeating - G_1

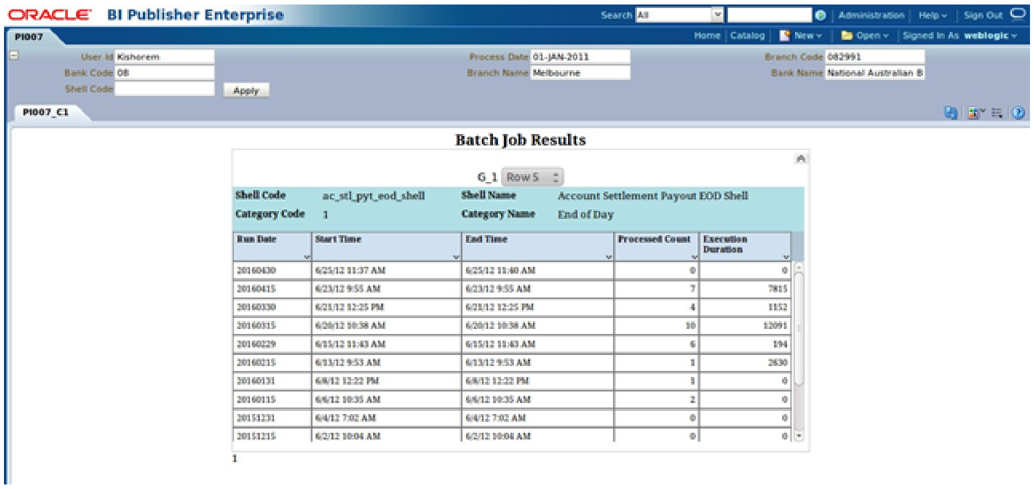
12.7 View Report in BIP

After saving the *Data Model* and *Layout*, you can view the report in BIP. Click the **View Report** link on the top right corner of the screen to open the report screen.

You will be able to see the input fields for the input parameters defined for the report. Enter appropriate values in these fields and click **Apply**. The report will be generated and displayed on the screen with the applicable data returned by the previously created *Data Model* and formatted as per the previously created *Layout*.

For the aforementioned use case, the final report would be as shown below.

Figure 12–18 View Report in BIP


ORACLE BI Publisher Enterprise
Search All
Administration | Help | Sign Out
PI007
Home | Catalog | New | Open | Signed In As weblogic
Apply
PI007_C1

User Id Kishorem
Bank Code 08
Shell Code
Process Date 01-JAN-2011
Branch Name Melbourne
Branch Code 082991
Bank Name National Australian B

Batch Job Results

G_1 Row 5

Shell Code	ac_stl_pyt_eod_shell	**Shell Name**	Account Settlement Payout EOD Shell
Category Code	1	**Category Name**	End of Day

Run Date	Start Time	End Time	Processed Count	Execution Duration
20160430	6/25/12 11:37 AM	6/25/12 11:40 AM	0	0
20160415	6/23/12 9:55 AM	6/23/12 9:55 AM	7	7815
20160330	6/21/12 12:25 PM	6/21/12 12:25 PM	4	1152
20160315	6/20/12 10:38 AM	6/20/12 10:38 AM	10	12091
20160229	6/15/12 11:43 AM	6/15/12 11:43 AM	6	194
20160215	6/13/12 9:53 AM	6/13/12 9:53 AM	1	2630
20160131	6/8/12 12:22 PM	6/8/12 12:22 PM	1	0
20160115	6/6/12 10:35 AM	6/6/12 10:35 AM	2	0
20151231	6/4/12 7:02 AM	6/4/12 7:02 AM	0	0
20151215	6/2/12 10:04 AM	6/2/12 10:04 AM	0	0

1

You can export the report in *HTML, PDF, Excel, RTF* or *PowerPoint* formats by clicking on the icon for *Export* on the right top corner of the screen and choosing the corresponding export option.

12.8 OBP Batch Report Configuration - Define the Batch Reports

Entries are required in three tables as given below to generate reports during EOD.

```
insert into FLX_BATCH_JOB_SHELL_MASTER (COD_EOD_PROCESS, TXT_
PROCESS, TXT_PROCESS_NAME, FRQ_PROC, DAT_LAST_RUN, DAT_SCHEDULED_
RUN, TXT_PROC_PARAM, COD_PROC_STATUS, NUM_PROC_ERROR, FLG_RUN_
TODAY, COD_PROC_CATEGORY, FLG_MONTH_END, FLG_MNT_STATUS, COD_MNT_
ACTION, COD_LAST_MNT_MAKERID, COD_LAST_MNT_CHKRID, DAT_LAST_MNT,
CTR_UPDAT_SRLNO, COD_MODULE, DAT_PROC_START, DAT_PROC_END, TXN_KEY,
SERVICE_KEY, NAM_COMPONENT, TYPE_COMPONENT, NAM_DBINSTANCE, RETRY_
COUNTER, NON_RETRY_COUNTER, COD_UNSTREAMED_PROCESS, COD_BRANCH_
GROUP_CODE)
values ('ch_eod_report_shell', 'CASA EOD Reports', 'CASA EOD
Reports', '1', to_date('15-02-2012', 'dd-mm-yyyy'), to_date('15-12-
2007', 'dd-mm-yyyy'), '99', 0, 0, 'Y', 1, 0, 'A', ' ', 'SETUP1',
'SETUP2', to_date('09-02-2002', 'dd-mm-yyyy'), 2, 'CH', to_date
('21-08-2008 09:54:57', 'dd-mm-yyyy hh24:mi:ss'), to_date('28-02-
2011 05:02:41', 'dd-mm-yyyy hh24:mi:ss'), 'DUMMY', 'execute',
'com.ofss.fc.bh.batch.BatchReportShellBean', 'B', 'PROD', 0, 0,
'ch_eod_report_shell', 'BRN_GRP_1');
```

Cod_proc_category = **1**, for EOD; 2, for BOD and 16 for Internal System EOD

Nam_component is the same for all report shells.

Also we are using Branch_Group_Category ='BRN_GRP_1' for all these report shells.

12.9 OBP Batch Report Configuration - Define the Batch Report Shell

```
Insert into FLX_BATCH_JOB_SHELL_DEPEND (COD_EOD_PROCESS, COD_REQD_
PROCESS, COD_PROC_CATEGORY, COD_REQD_PROC_CAT, FLG_MNT_STATUS, COD_
MNT_ACTION, COD_LAST_MNT_MAKERID, COD_LAST_MNT_CHKRID, DAT_LAST_
MNT, CTR_UPDAT_SRLNO, COD_BRANCH_GROUP_CODE)
Values ('ch_eod_report_shell', 'dd_eod_action', 1, 1, 'A', ' ',
'SETUP', 'SETUP', to_date('30-06-1995', 'dd-mm-yyyy'),2, 'BRN_GRP_
1');
```

Here, in the first column is the report shell name and second is the name of the shell after which this shell should run. So 'ch_bod_report_shell' runs after 'dd_bod_action'. The remaining columns are self explanatory.

```
COD_PROC_CATEGORY=1 , for EOD; 2, for BOD and 16 for Internal
System EOD
COD_REQD_PROC_CAT=1, for EOD; 2, for BOD and 16 for Internal System
EOD
```

Also we are using Branch_Group_Category = 'BRN_GRP_1' for all these report shells.

12.10 OBP Batch Report Configuration - Define the Batch Report Shell Dependencies

```
Insert into flx_ba_report_ctrl (COD_REPORT_ID, FLG_REP_ADV, COD_
MODULE, NAM_REPORT, TYP_REPORT, FRQ_REPORT, FLG_PRINT, FLG_DELETE,
CTR_REP_COPIES, COD_PRIORITY, COD_ACCESS_LVL, COD_FILEID, BUF_INV_
VAR1, BUF_INV_VAR2, BUF_INV_VAR3, BUF_INV_VAR4, BUF_INV_VAR5, FLG_
MNT_STATUS, COD_MNT_ACTION, COD_LAST_MNT_MAKERID, COD_LAST_MNT_
CHKRID, DAT_LAST_MNT, CTR_UPDAT_SRLNO, FLG_SOURCE, FLG_SPLIT, FLG_
PROD_REP, COD_REPORT_DB_PREFIX, FLG_APPLY_SC, REF_UDF_NO, XPATH,
FLG_REPORT_SERVER)
values ('CH318', 'R', 'CH', 'CASA BALANCE LISTING', 'E', '1', '1',
'0', 1, 0, 0, 10047, ' ', ' ', ' ', ' ', ' ', 'A', ' ', 'PHASE_2',
'PHASE_2', to_date('01-11-1999', 'dd-mm-yyyy'), 2, 'P', 'Y', 'P',
'PROD', '', '', '', 'B');
```

Entry for each report should be here with typ_report = 'I' for Internal System EOD; 'E' for EOD and 'B' for BOD.

Currently, for EOD and BOD eod_report_shell and bod_report_shell will take care of all non CASA and TD EOD and BOD reports respectively.

No separate module specific shell is required during EOD and BOD. That is to mention Entry 3 alone is sufficient during EOD and BOD categories for any module. However, entries are needed for all three entries for batch report generation during any other category.

12.11 OBP Batch Report Configuration

This section describes the OBP batch report configuration.

12.11.1 Batch Report Generation for a Branch Group Code

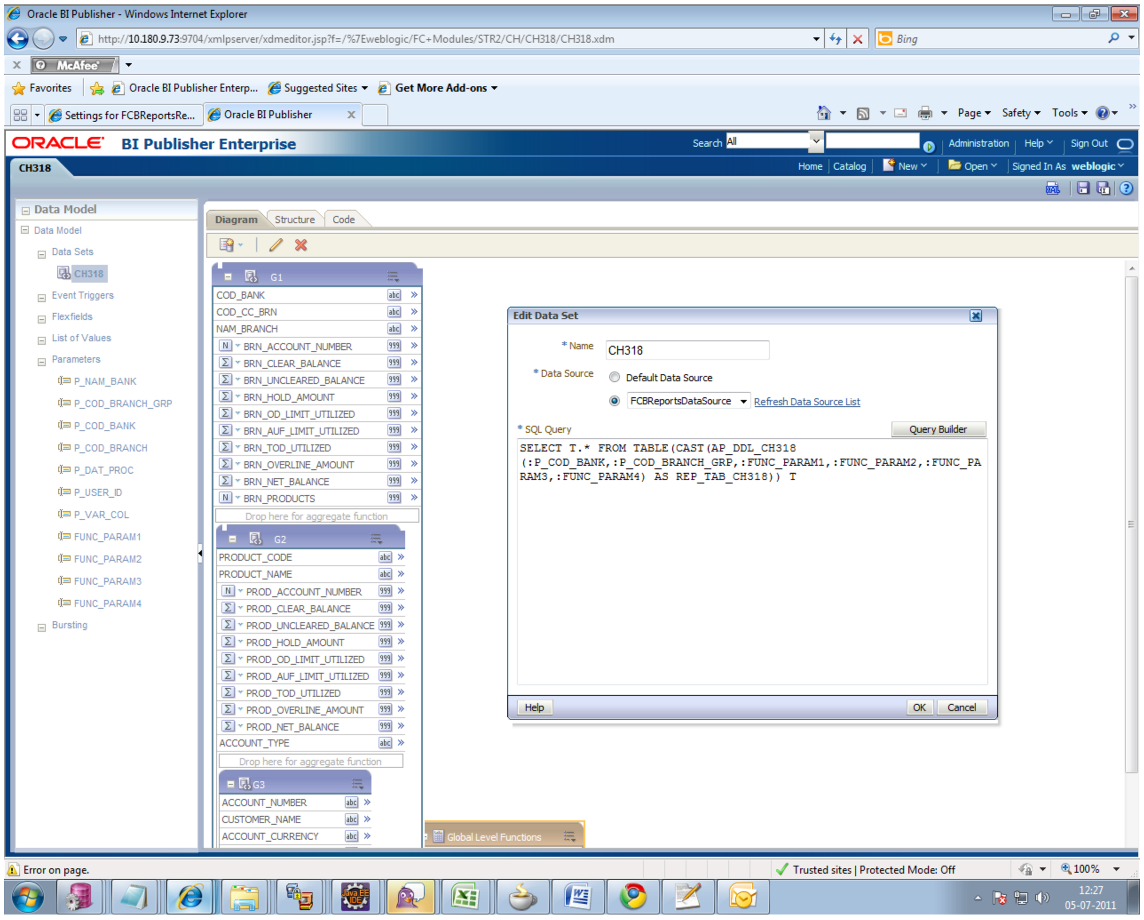
During Batch Process, a report should be generated for all branches linked to the respective Branch Group Code.

For any Batch Report to make use of the Branch Group Code getting passed by the application, a parameter 'P_COD_BRANCH_GRP' has to be defined in the Data Model.

The Data Model should pass this parameter to the Report Related DDL Function.

The Report Related DML Function filters all branch codes from FLX_BATCH_JOB_RESULTS_FILTERED that belong to the same Branch Group Code.

Figure 12–19 Batch Report Generation for a Branch Group Code



Edit Data Set

* Name: CH318

* Data Source: ( ) Default Data Source [✓ FCBReportsDataSource] Refresh Data Source List

* SQL Query

```
SELECT T.* FROM TABLE(CAST(AP_DDL_CH318
(:P_COD_BANK,:P_COD_BRANCH_GRP,:FUNC_PARAM1,:FUNC_PARAM2,:FUNC_PA
RAM3,:FUNC_PARAM4) AS REP_TAB_CH318)) T
```

12.11.2 Batch Report Generation Status

At the end of all batch processes BA_REPORT_RESTART gets logged with the generated report status as D -> Done or F->Failed.

12.11.3 Batch Report Generation Path

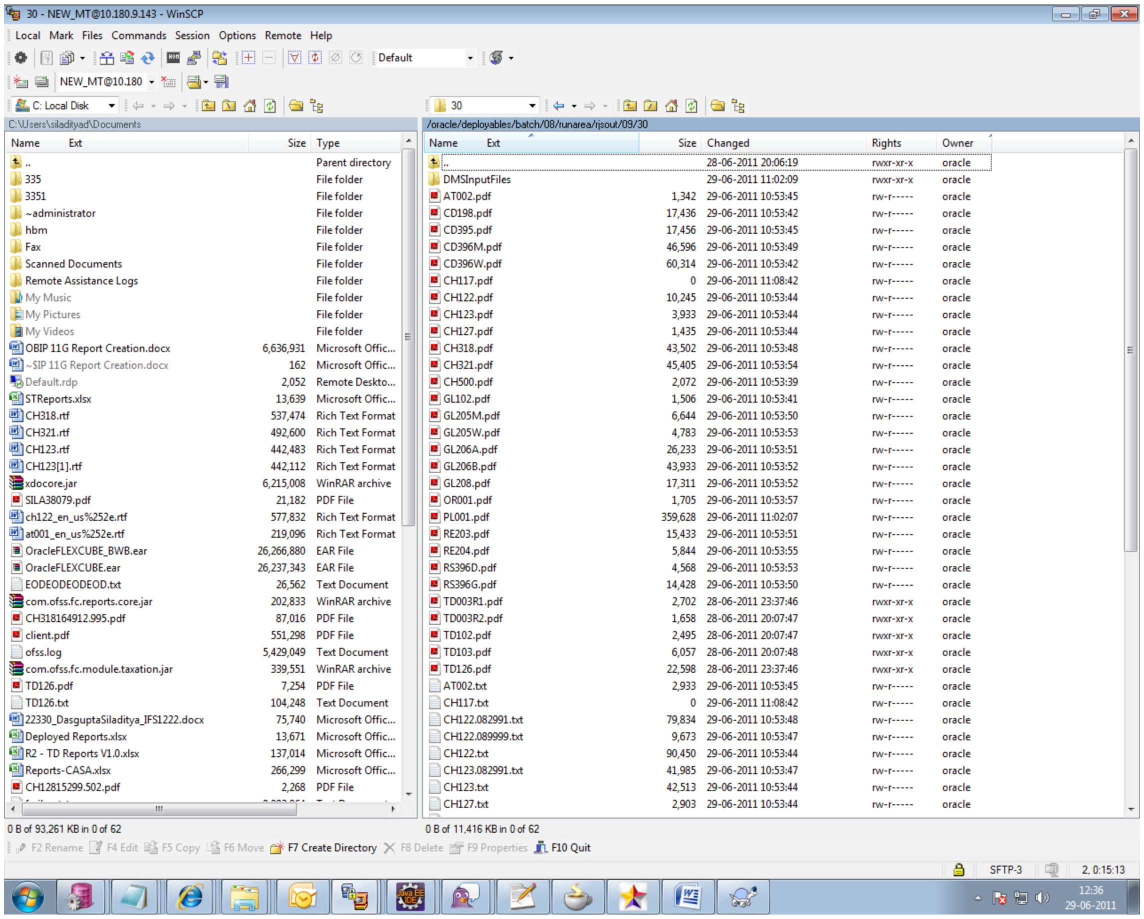
The reports (for example, 30th September 2008) will be generated as shown in the host side screen-shot.

Locate these reports at this location in the host server.

/oracle/deployables/batch/08/runarea/rjsout/09/30 which actually is of the format

/config/../<BankCode>/runarea/rjsout/<MM>/<DD>

Figure 12–20 Batch Report Generation Path


30 - NEW_MT@10.180.9.143 - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
NEW_MT@10.180
C: Local Disk
30
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3
2, 0:15:13
12:36
29-06-2011

C:\Users\siladityad\Documents

Name Ext	Size	Type
..		Parent directory
335		File folder
3351		File folder
~administrator		File folder
hbm		File folder
Fax		File folder
Scanned Documents		File folder
Remote Assistance Logs		File folder
My Music		File folder
My Pictures		File folder
My Videos		File folder
OBIP 11G Report Creation.docx	6,636,931	Microsoft Offic...
~$IP 11G Report Creation.docx	162	Microsoft Offic...
Default.rdp	2,052	Remote Deskto...
STReports.xlsx	13,639	Microsoft Offic...
CH318.rtf	537,474	Rich Text Format
CH321.rtf	492,600	Rich Text Format
CH123.rtf	442,483	Rich Text Format
CH123[1].rtf	442,112	Rich Text Format
xdocore.jar	6,215,008	WinRAR archive
SILA38079.pdf	21,182	PDF File
ch122_en_us%252e.rtf	577,832	Rich Text Format
at001_en_us%252e.rtf	219,096	Rich Text Format
OracleFLEXCUBE_BWB.ear	26,266,880	EAR File
OracleFLEXCUBE.ear	26,237,343	EAR File
EODEODEODEOD.txt	26,562	Text Document
com.ofss.fc.reports.core.jar	202,833	WinRAR archive
CH318164912.995.pdf	87,016	PDF File
client.pdf	551,298	PDF File
ofss.log	5,429,049	Text Document
com.ofss.fc.module.taxation.jar	339,551	WinRAR archive
TD126.pdf	7,254	PDF File
TD126.txt	104,248	Text Document
22330_DasguptaSiladitya_IFS1222.docx	75,740	Microsoft Offic...
Deployed Reports.xlsx	13,671	Microsoft Offic...
R2 - TD Reports V1.0.xlsx	137,014	Microsoft Offic...
Reports-CASA.xlsx	266,299	Microsoft Offic...
CH12815299.502.pdf	2,268	PDF File

0 B of 93,261 KB in 0 of 62

/oracle/deployables/batch/08/runarea/rjsout/09/30

Name Ext	Size	Changed	Rights	Owner
..		28-06-2011 20:06:19	rwxr-xr-x	oracle
DMSInputFiles		29-06-2011 11:02:09	rwxr-xr-x	oracle
AT002.pdf	1,342	29-06-2011 10:53:45	rw-r-----	oracle
CD198.pdf	17,436	29-06-2011 10:53:42	rw-r-----	oracle
CD395.pdf	17,456	29-06-2011 10:53:45	rw-r-----	oracle
CD396M.pdf	46,596	29-06-2011 10:53:49	rw-r-----	oracle
CD396W.pdf	60,314	29-06-2011 10:53:42	rw-r-----	oracle
CH117.pdf	0	29-06-2011 11:08:42	rw-r-----	oracle
CH122.pdf	10,245	29-06-2011 10:53:44	rw-r-----	oracle
CH123.pdf	3,933	29-06-2011 10:53:44	rw-r-----	oracle
CH127.pdf	1,435	29-06-2011 10:53:44	rw-r-----	oracle
CH318.pdf	43,502	29-06-2011 10:53:48	rw-r-----	oracle
CH321.pdf	45,405	29-06-2011 10:53:54	rw-r-----	oracle
CH500.pdf	2,072	29-06-2011 10:53:39	rw-r-----	oracle
GL102.pdf	1,506	29-06-2011 10:53:41	rw-r-----	oracle
GL205M.pdf	6,644	29-06-2011 10:53:50	rw-r-----	oracle
GL205W.pdf	4,783	29-06-2011 10:53:53	rw-r-----	oracle
GL206A.pdf	26,233	29-06-2011 10:53:51	rw-r-----	oracle
GL206B.pdf	43,933	29-06-2011 10:53:52	rw-r-----	oracle
GL208.pdf	17,311	29-06-2011 10:53:52	rw-r-----	oracle
OR001.pdf	1,705	29-06-2011 10:53:57	rw-r-----	oracle
PL001.pdf	359,628	29-06-2011 11:02:07	rw-r-----	oracle
RE203.pdf	15,433	29-06-2011 10:53:51	rw-r-----	oracle
RE204.pdf	5,844	29-06-2011 10:53:55	rw-r-----	oracle
RS396D.pdf	4,568	29-06-2011 10:53:53	rw-r-----	oracle
RS396G.pdf	14,428	29-06-2011 10:53:50	rw-r-----	oracle
TD003R1.pdf	2,702	28-06-2011 23:37:46	rwxr-xr-x	oracle
TD003R2.pdf	1,658	28-06-2011 20:07:47	rwxr-xr-x	oracle
TD102.pdf	2,495	28-06-2011 20:07:47	rwxr-xr-x	oracle
TD103.pdf	6,057	28-06-2011 20:07:48	rwxr-xr-x	oracle
TD126.pdf	22,598	28-06-2011 23:37:46	rwxr-xr-x	oracle
AT002.txt	2,933	29-06-2011 10:53:45	rw-r-----	oracle
CH117.txt	0	29-06-2011 11:08:42	rw-r-----	oracle
CH122.082991.txt	79,834	29-06-2011 10:53:48	rw-r-----	oracle
CH122.089999.txt	9,673	29-06-2011 10:53:47	rw-r-----	oracle
CH122.txt	90,450	29-06-2011 10:53:44	rw-r-----	oracle
CH123.082991.txt	41,985	29-06-2011 10:53:47	rw-r-----	oracle
CH123.txt	42,513	29-06-2011 10:53:44	rw-r-----	oracle
CH127.txt	2,903	29-06-2011 10:53:44	rw-r-----	oracle

0 B of 11,416 KB in 0 of 62

12.12 OBP Adhoc Report Configuration

This section describes the OBP adhoc report configuration.

12.12.1 Define the Adhoc Reports

Define the adhoc reports as follows:

```
Insert into flx_ba_report_ctrl (COD_REPORT_ID, FLG_REP_ADV, COD_
MODULE, NAM_REPORT, TYP_REPORT, FRQ_REPORT, FLG_PRINT, FLG_DELETE,
CTR_REP_COPIES, COD_PRIORITY, COD_ACCESS_LVL, COD_FILEID, BUF_INV_
VAR1, BUF_INV_VAR2, BUF_INV_VAR3, BUF_INV_VAR4, BUF_INV_VAR5, FLG_
MNT_STATUS, COD_MNT_ACTION, COD_LAST_MNT_MAKERID, COD_LAST_MNT_
CHKRID, DAT_LAST_MNT, CTR_UPDAT_SRLNO, FLG_SOURCE, FLG_SPLIT, FLG_
PROD_REP, COD_REPORT_DB_PREFIX, FLG_APPLY_SC, REF_UDF_NO, XPATH,
FILE_DESC, FLG_REPORT_SERVER)
values ('CH318', 'R', 'CH', 'CASA BALANCE LISTING', 'A', '1', '1',
'0', 1, 0, 0, 10047, ' ', ' ', ' ', ' ', ' ', 'A', ' ', 'PHASE_2',
'PHASE_2', to_date('01-11-1999', 'dd-mm-yyyy'), 2, 'P', 'Y', 'P',
'PROD', '', '', '', 'Savings Listing Reports', 'B');
```

12.12.2 Define the Adhoc Report Parameters

Define the adhoc report parameters as follows:

```
INSERT INTO flx_ba_report_params (COD_REPORT_ID,FLG_REP_ADV,COD_
SERIAL,NAM_PROMPT, COD_FLD_TYP,LEN_FLD,FLG_DELETE,DAT_LAST_MNT,NAM_
VAL_ROUTINE,REQD_DESC) VALUES ('CH318','R',1,'Branch
Code',0,0,'N','01-NOV-99','','Y')
/
INSERT INTO flx_ba_report_params (COD_REPORT_ID,FLG_REP_ADV,COD_
SERIAL,NAM_PROMPT, COD_FLD_TYP,LEN_FLD,FLG_DELETE,DAT_LAST_MNT,NAM_
VAL_ROUTINE,REQD_DESC) VALUES ('CH318','R',2,'Product
Code',0,0,'N','01-NOV-99','','Y')
/
INSERT INTO flx_ba_report_params (COD_REPORT_ID,FLG_REP_ADV,COD_
SERIAL,NAM_PROMPT, COD_FLD_TYP,LEN_FLD,FLG_DELETE,DAT_LAST_MNT,NAM_
VAL_ROUTINE,REQD_DESC) VALUES ('CH318','R',3,'From Date(DD-MMM-
YYYY)',8,0,'N','01-NOV-99','','Y')
/
```

Also COD_FLD_TYP = 8 will ensures the host side date format validations.

COD_FLD_TYP = 0 is for string type parameters.

Corresponding to each of the above sequence of parameters appearing in screen, a mandatory parameter 'FUNC_PARAM<Parameter Sequence Number>' should be defined in BIP Data Model. So the input parameter 'FUNC_PARAM2' defined in data model should correspond to Product Code as defined above.

12.12.3 Define the Adhoc Reports to be listed in Screen

Define the group name as follows:

For Adhoc Report, column FILE_DESC of report master table FLX_BA_REPORT_CTRL contains the name of the group under which the report will be listed in 7775 screen.

12.12.4 Adding Screen Tab for Report Module

For adding a Screen Tab do the following:

```
com.ofss.fc.ui.view.brop.jar@
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

12.13 Adhoc Report Generation – Screen 7775

The adhoc report can be generated using the following screen:

Figure 12–21 Adhoc Report Generation - Report Request

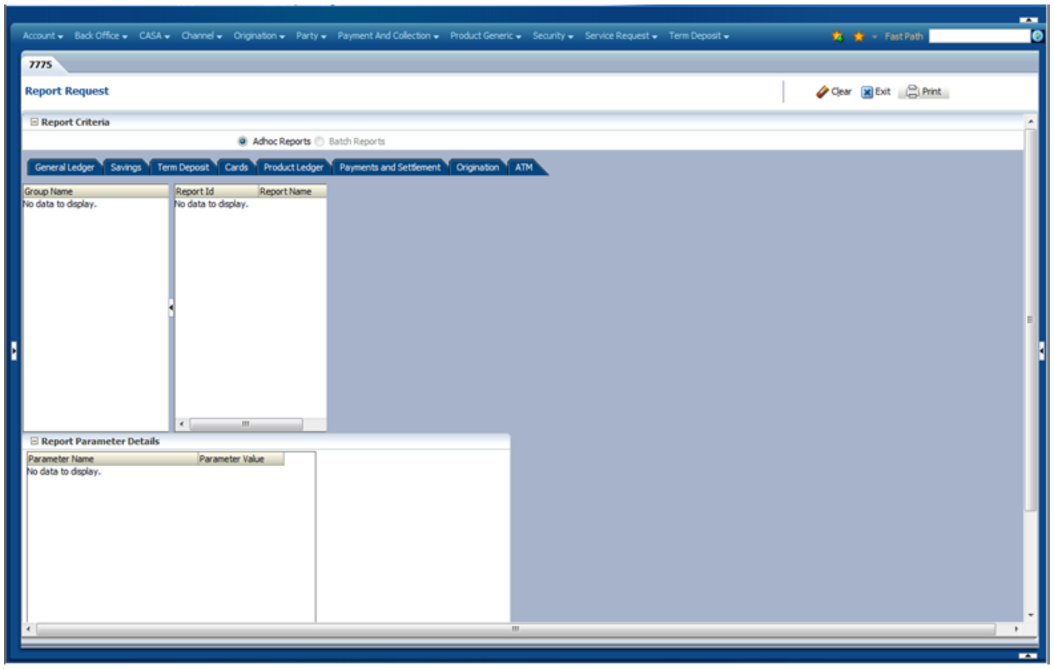
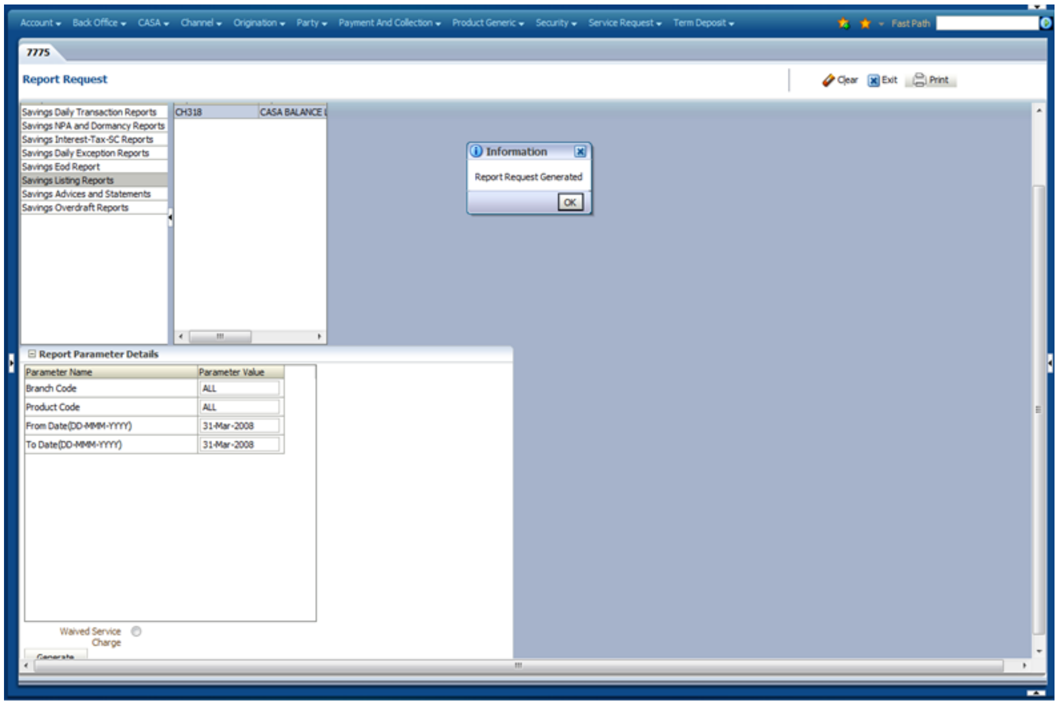


Figure 12–22 Adhoc Report Generation - Report Generated



Report Request

Savings Daily Transaction Reports
Savings NPA and Dormancy Reports
Savings Interest-Tax-SC Reports
Savings Daily Exception Reports
Savings Eod Report
Savings Listing Reports
Savings Advices and Statements
Savings Overdraft Reports

CH318 CASA BALANCE

Information

Report Request Generated

OK

Report Parameter Details

Parameter Name	Parameter Value
Branch Code	ALL
Product Code	ALL
From Date(DD-MMM-YYYY)	31-Mar-2008
To Date(DD-MMM-YYYY)	31-Mar-2008

Waived Service Charge

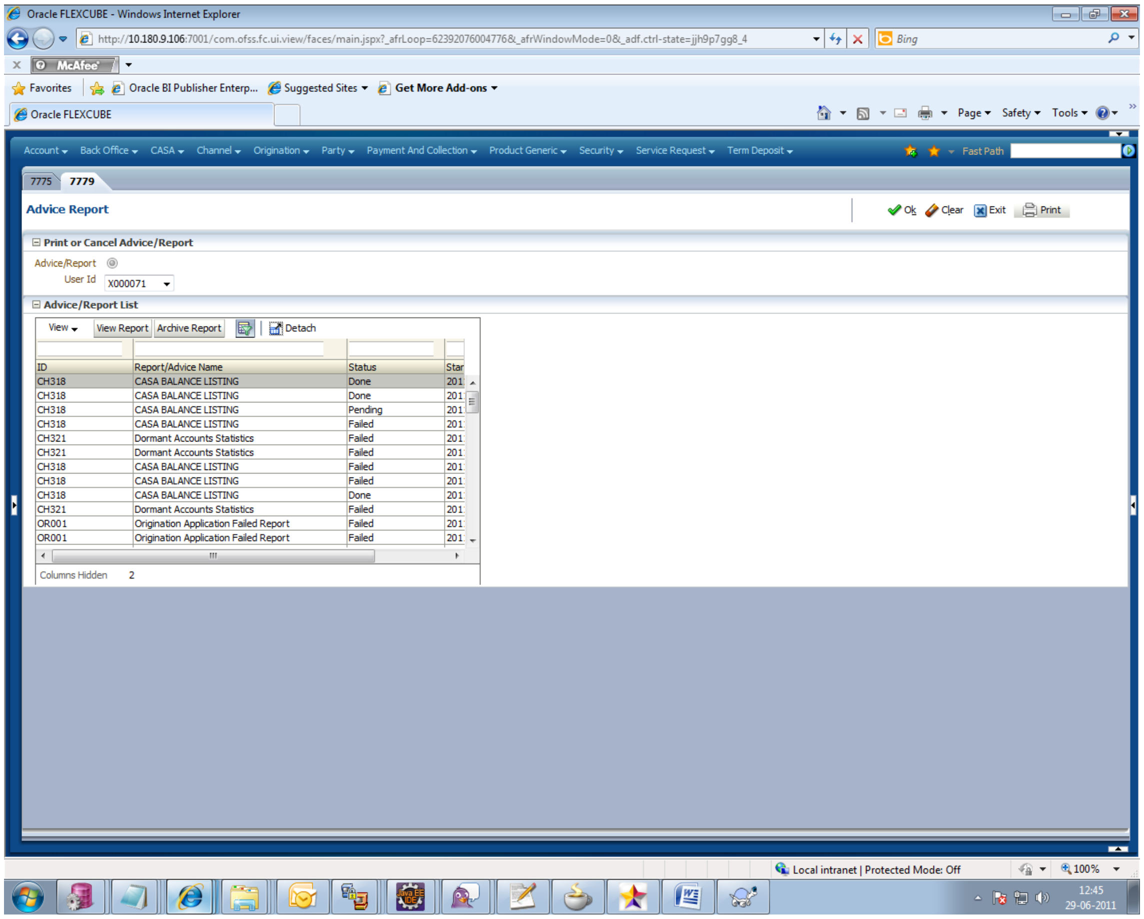
On filling the parameters and clicking on 'Generate' the report request gets successfully posted.

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

12.14 Adhoc Report Viewing – Screen 7779

The adhoc report can be viewed using the following screen:

Figure 12–23 Advice Report



Advice Report

Print or Cancel Advice/Report

Advice/Report

User Id: X000071

Advice/Report List

ID	Report/Advice Name	Status	Star
CH318	CASA BALANCE LISTING	Done	201
CH318	CASA BALANCE LISTING	Done	201
CH318	CASA BALANCE LISTING	Pending	201
CH318	CASA BALANCE LISTING	Failed	201
CH321	Dormant Accounts Statistics	Failed	201
CH321	Dormant Accounts Statistics	Failed	201
CH318	CASA BALANCE LISTING	Failed	201
CH318	CASA BALANCE LISTING	Failed	201
CH318	CASA BALANCE LISTING	Done	201
CH321	Dormant Accounts Statistics	Failed	201
OR001	Origination Application Failed Report	Failed	201
OR001	Origination Application Failed Report	Failed	201

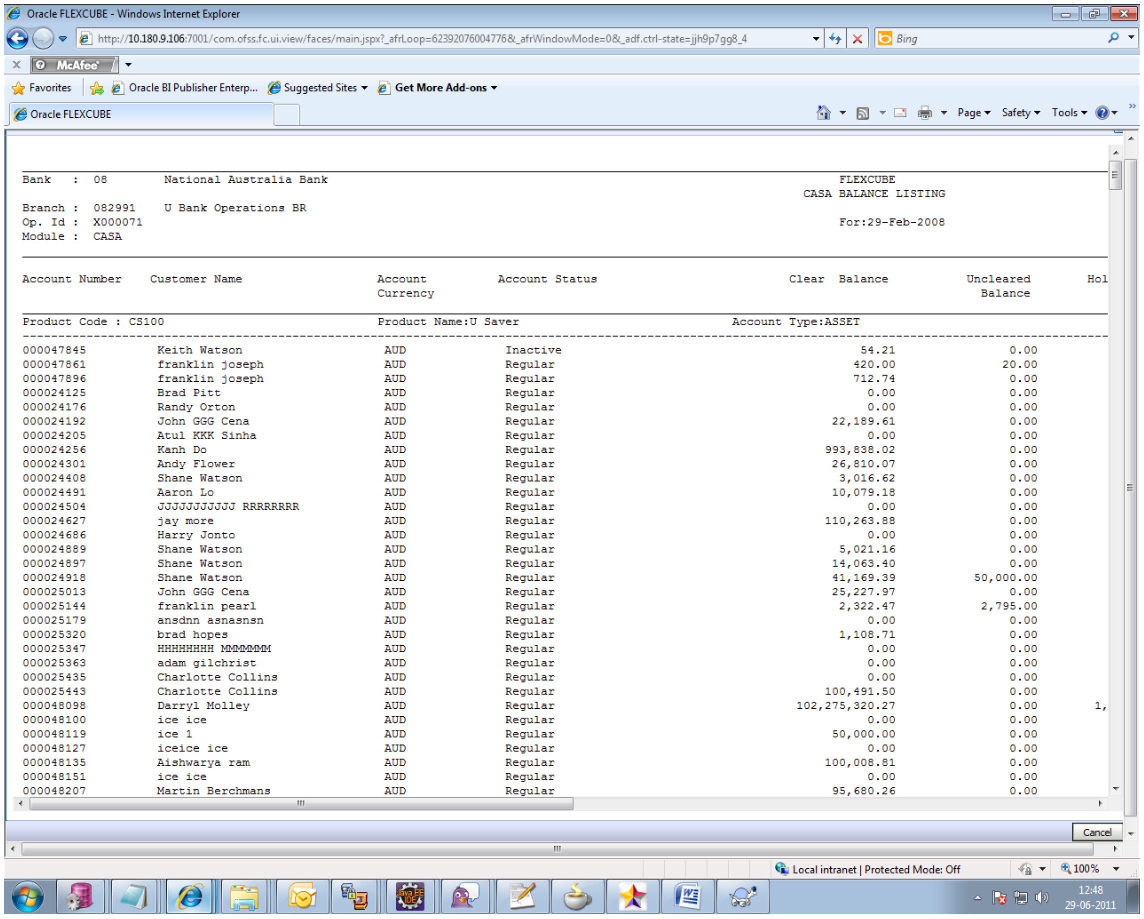
Columns Hidden 2

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 12–24 View Generated Adhoc Report



```
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          Account Status                         Clear  Balance         Uncleared       Hol
                                                  Currency                                                                         Balance

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          JJJJJJJJJJ 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 MMMMMMM                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
```

Oracle FLEXCUBE - Windows Internet Explorer
http://10.180.9.106:7001/com.ofss.fc.ui.view/faces/main.jspx?_afrLoop=62392076004776&_afrWindowMode=0&_adf.ctrl-state=jjh9p7gg8_4
Bing
McAfee
Favorites
Oracle BI Publisher Enterp...
Suggested Sites
Get More Add-ons
Oracle FLEXCUBE
Page
Safety
Tools
Cancel
Local intranet | Protected Mode: Off
100%
12:48
29-06-2011

The report is rendered in the front end.

13 Security Customizations

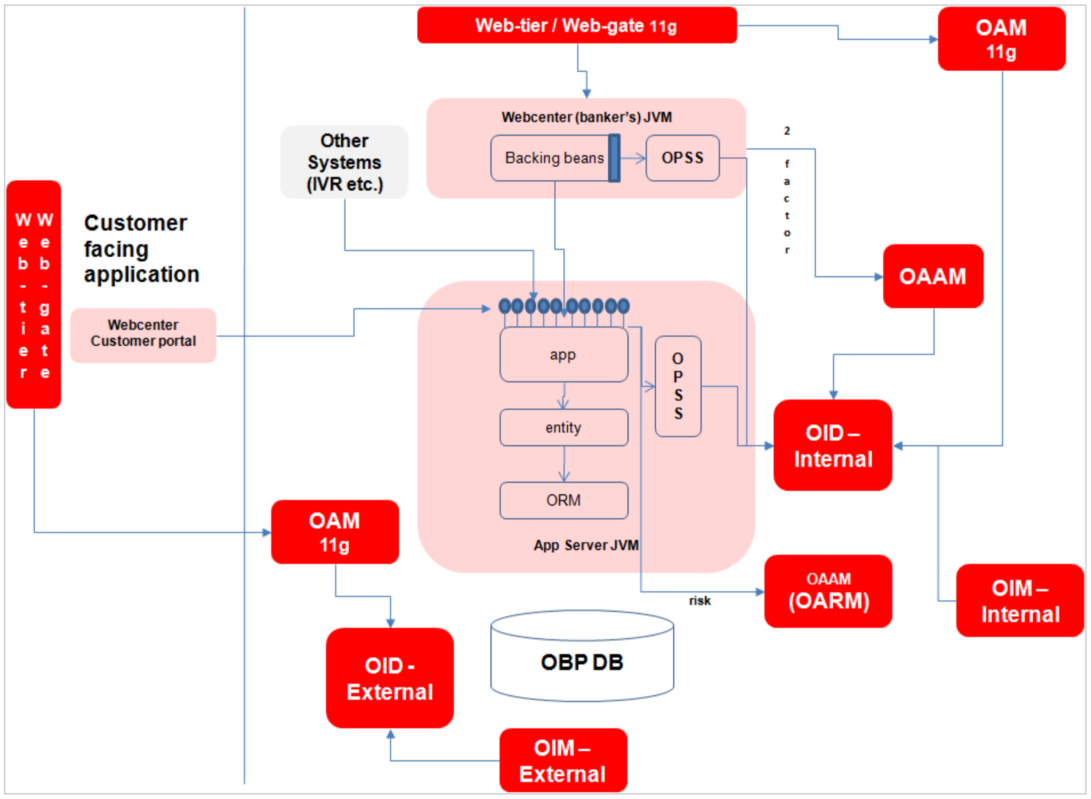
OBP comprising of several modules has to interface with various systems in an enterprise to transfer or share data which is generated during business activity that takes place during teller operations or processing. While managing the transactions that are within OBP's domain, it is needed to consider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

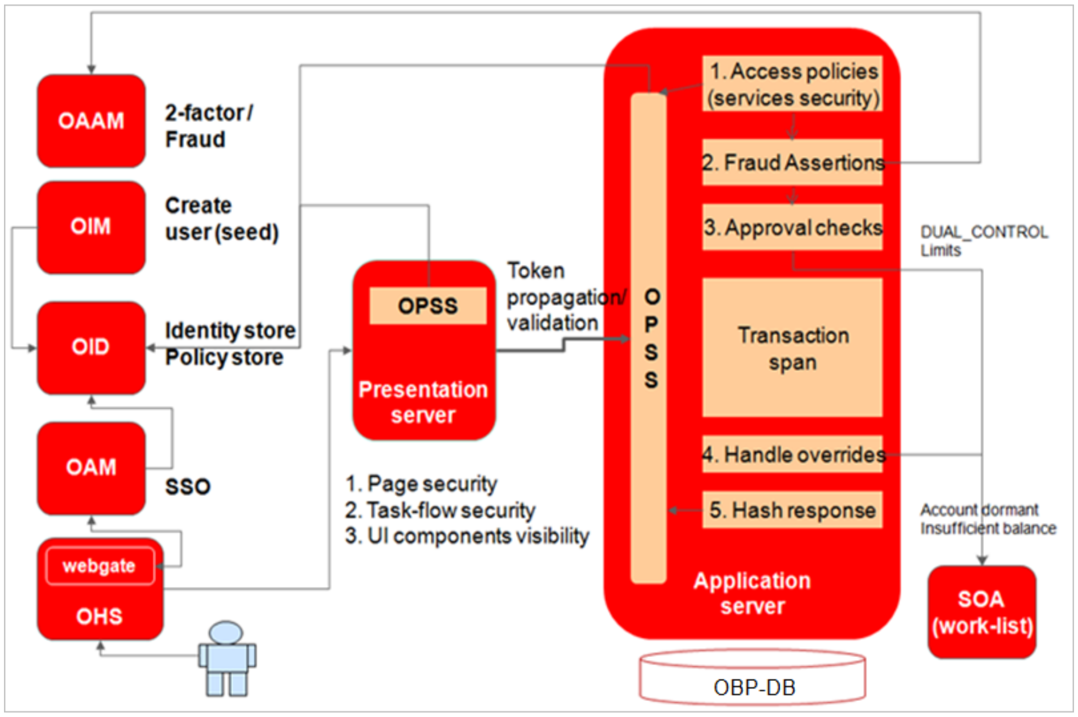
Figure 13–1 Security Customizations Interface



- 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.
- Oracle Internet Directory (OID) is used as the identity/policy store.

A high-level security use case has the following access checks and assertions.

Figure 13–2 Security Use Case with Access Checks and Assertions



13.1 OPSS Access Policies – Adding Attributes

OBP uses OPSS to assert role-based access policies. Access policies are rules-based to give more flexibility.

Example of an access policy rule:

```
Grant
Role = RetailBranchOperationsExecutive
Service=com.ofss.fc.app.dda.service.transaction.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
```

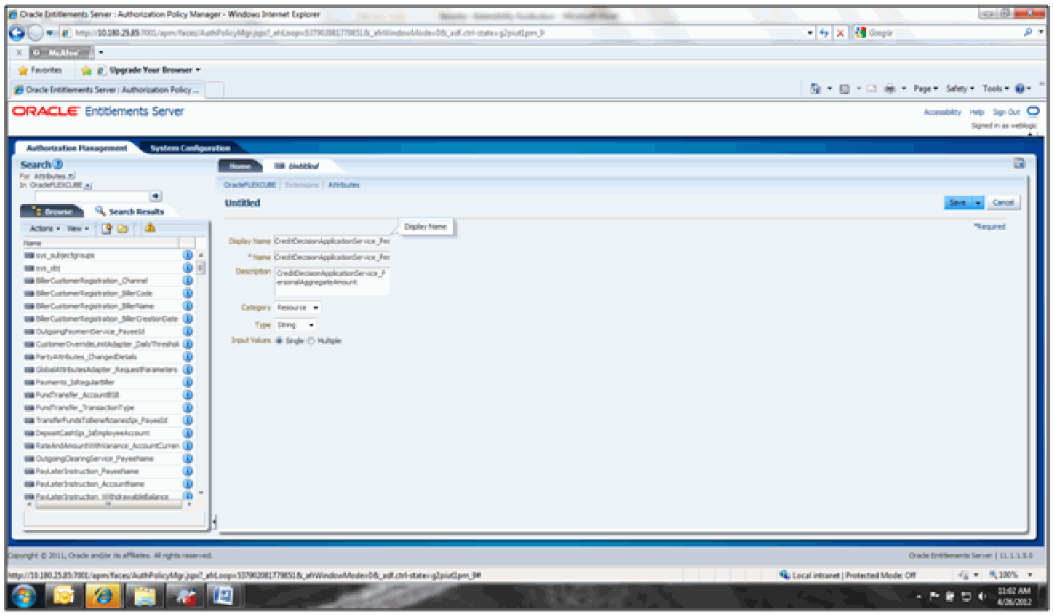
The security framework allows for addition to the facts that can be used in rules. The steps to do this are mentioned in the next section.

13.1.1 Steps

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

1. Add attribute in OID under the 'Attributes' entry.

Figure 13–3 Add Attributes to Access Policy Rule



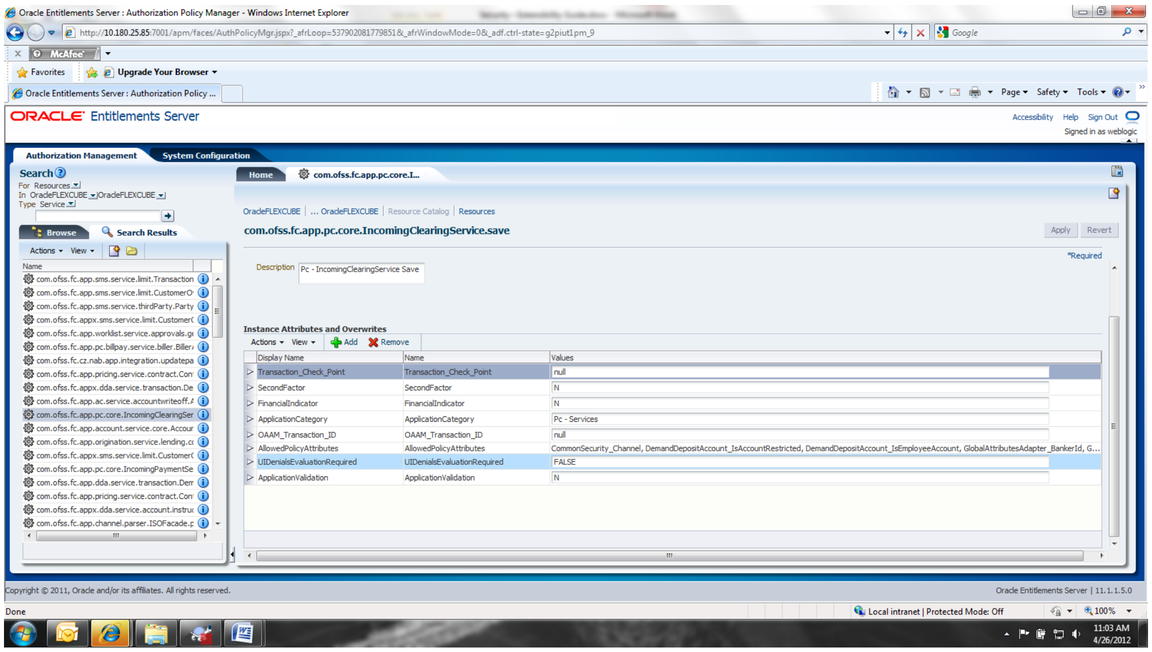
Untitled

Display Name	CreditDecisionApplicationService_Per
* Name	CreditDecisionApplicationService_Per
Description	CreditDecisionApplicationService_PersonalAggregateAmount
Category	Resource
Type	String
Input Values	[✓ Single] Multiple

This can be done directly in OID or by using APM, as shown above.

2. Add the attribute under 'AllowedPolicyAttributes' against the particular resource.

Figure 13–4 Attribute to Access Policy Rule - Authorization Management



com.ofss.fc.app.pc.core.IncomingClearingService.save

Description: Pc - IncomingClearingService Save

Instance Attributes and Overwrites

Display Name	Name	Values
Transaction_Check_Point	Transaction_Check_Point	null
SecondFactor	SecondFactor	N
FinancialIndicator	FinancialIndicator	N
ApplicationCategory	ApplicationCategory	Pc - Services
OAAM_Transaction_ID	OAAM_Transaction_ID	null
AllowedPolicyAttributes	AllowedPolicyAttributes	CommonSecurity_Channel, DemandDepositAccount_IsAccountRestricted, DemandDepositAccount_IsEmployeeAccount, GlobalAttributesAdapter_BankerId, G...
UIDenialsEvaluationRequired	UIDenialsEvaluationRequired	FALSE
ApplicationValidation	ApplicationValidation	N

This can be done directly in OID or by using APM, as shown above. Adding this attribute under 'AllowedPolicyAttributes' ensures that the security framework executes a specified adapter to fetch the attribute value and make it available to the execution context.

3. Develop custom adapter to retrieve attribute value. Attribute should be structured along similar lines as the other adapters used for the same purpose.

```
Example -
Attribute - CreditDecisionMatrix_
OverallAggregateApplicationAmount
Adapter -
public
com.ofss.fc.app.adapter.impl.sms.CreditDecisionAttributesAdap
ter {
public String getOverallAggregateApplicationAmount () {
//Logic to fetch overall aggregate amount
}
}
```

Note

The naming convention of the attribute should be as follows:

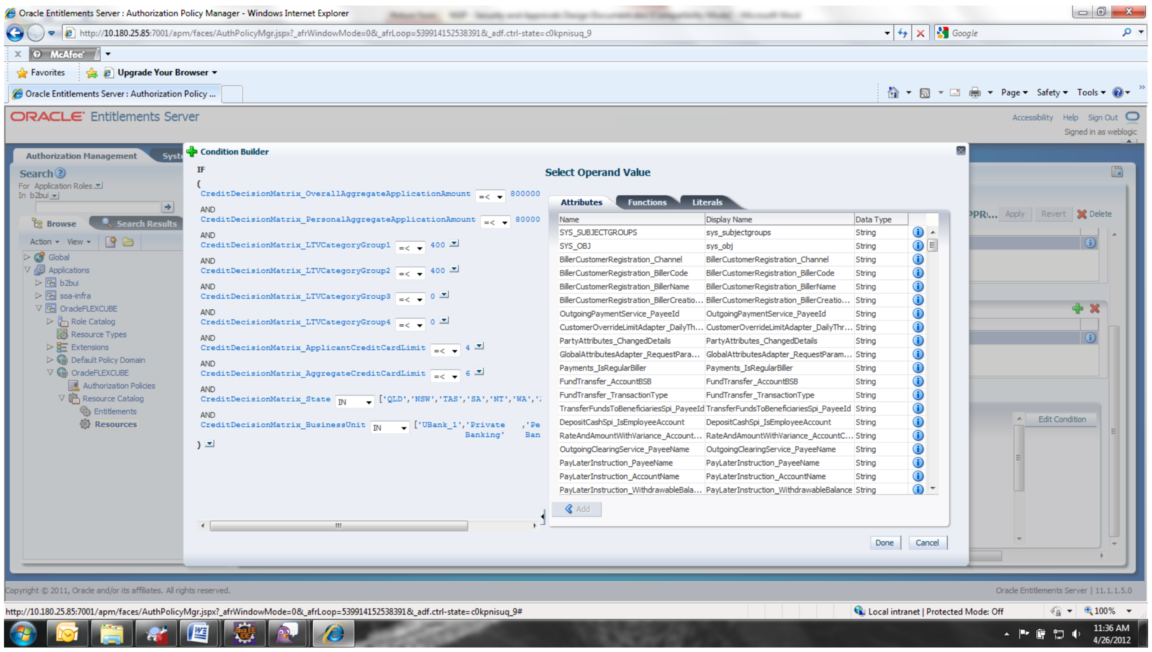
The first part of the attribute till the '-' delimiter identifies the transaction. The remaining part with CamelCase is prefixed with a 'get' to form the method in the adapter.

4. Add entry in ConstraintAttributeHelper.properties to link the attribute to the adapter.

```
CreditDecisionMatrix_OverallAggregateApplicationAmount=
com.ofss.fc.app.adapter.impl.sms.CreditDecisionAttributesAdap
ter
```

5. Add/Modify access policy/rule in APM to use the extra attribute.

Figure 13–5 Add or Modify Access Policy Rule



```
IF
(
CreditDecisionMatrix_OverallAggregateApplicationAmount =< 800000
AND
CreditDecisionMatrix_PersonalAggregateApplicationAmount =< 80000
AND
CreditDecisionMatrix_LTVCategoryGroup1 =< 400
AND
CreditDecisionMatrix_LTVCategoryGroup2 =< 400
AND
CreditDecisionMatrix_LTVCategoryGroup3 =< 0
AND
CreditDecisionMatrix_LTVCategoryGroup4 =< 0
AND
CreditDecisionMatrix_ApplicantCreditCardLimit =< 4
AND
CreditDecisionMatrix_AggregateCreditCardLimit =< 6
AND
CreditDecisionMatrix_State IN ['QLD','NSW','TAS','SA','NT','WA','
AND
CreditDecisionMatrix_BusinessUnit IN ['UBank_1','Private    ,'Pe
                                                 Banking'    Ban
)
```

Name	Display Name	Data Type
SYS_SUBJECTGROUPS	sys_subjectgroups	String
SYS_OBJ	sys_obj	String
BillerCustomerRegistration_Channel	BillerCustomerRegistration_Channel	String
BillerCustomerRegistration_BillerCode	BillerCustomerRegistration_BillerCode	String
BillerCustomerRegistration_BillerName	BillerCustomerRegistration_BillerName	String
BillerCustomerRegistration_BillerCreatio...	BillerCustomerRegistration_BillerCreatio...	String
OutgoingPaymentService_PayeeId	OutgoingPaymentService_PayeeId	String
CustomerOverrideLimitAdapter_DailyTh...	CustomerOverrideLimitAdapter_DailyThr...	String
PartyAttributes_ChangedDetails	PartyAttributes_ChangedDetails	String
GlobalAttributesAdapter_RequestPara...	GlobalAttributesAdapter_RequestParam...	String
Payments_IsRegularBiller	Payments_IsRegularBiller	String
FundTransfer_AccountBSB	FundTransfer_AccountBSB	String
FundTransfer_TransactionType	FundTransfer_TransactionType	String
TransferFundsToBeneficiariesSpi_PayeeId	TransferFundsToBeneficiariesSpi_PayeeId	String
DepositCashSpi_IsEmployeeAccount	DepositCashSpi_IsEmployeeAccount	String
RateAndAmountWithVariance_Account...	RateAndAmountWithVariance_AccountC...	String
OutgoingClearingService_PayeeName	OutgoingClearingService_PayeeName	String
PayLaterInstruction_PayeeName	PayLaterInstruction_PayeeName	String
PayLaterInstruction_AccountName	PayLaterInstruction_AccountName	String
PayLaterInstruction_WithdrawableBala...	PayLaterInstruction_WithdrawableBalance	String

13.2 OAAM Fraud Assertions – Adding Attributes

OBP uses OAAM to assert fraud policies consisting of rules to identify potentially fraudulent transactions.

Attributes used in fraud identification rules:

```
payee_id, account_number
```

The security framework allows for addition to this list of facts. The steps to do this are mentioned in the next section.

13.2.1 Steps

Following steps are needed to add an attribute to an existing OAAM transaction:

1. Add the attribute under ‘AllowedPolicyAttributes’ against the particular resource.
2. Add attribute in OID under the ‘Attributes’ entry.
3. Develop custom adapter to retrieve attribute value.
4. Add entry in ConstraintAttributeHelper.properties to link the attribute to the adapter.

The above steps are exactly the same as mentioned in the previous section.

1. Add seed data in the following tables to persist the mapping between OID attributes and OAAM attributes.

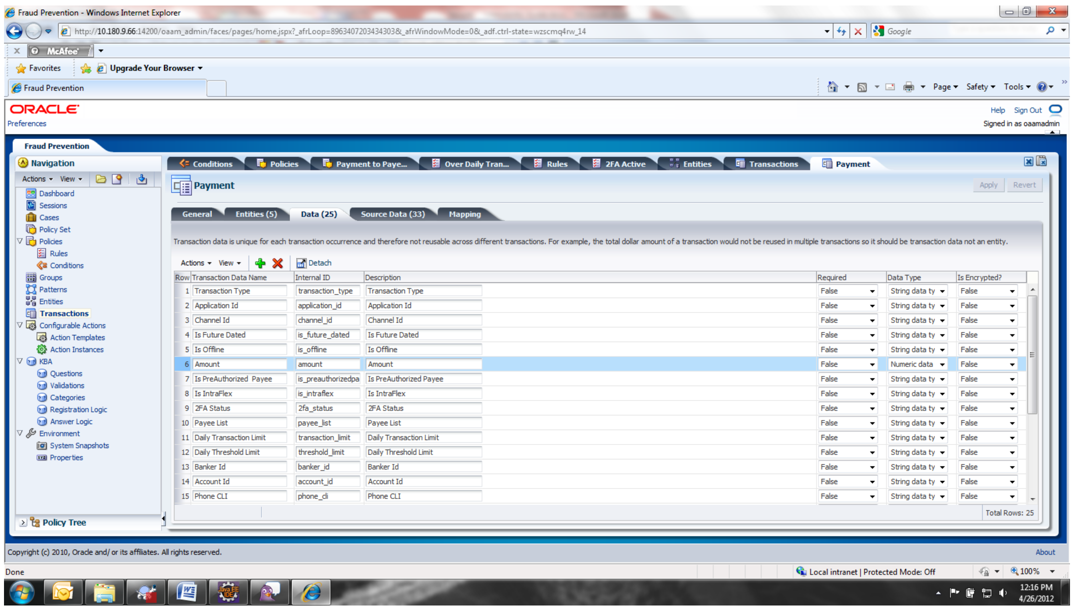
   flx_sm_fraud_txn_attributes (stores OAAM transaction key to OAAM attribute mapping) and

   flx_sm_fraud_assert_attributes (stores OBP attributeName - oaamAttributeName mapping.

```
Example -
insert into Flx_Sm_Fraud_Txn_Attributes (TRANSACTION_KEY,
ATTRIBUTE_NAME)
values ('payment', 'is_2fa_completed')
/
insert into flx_sm_fraud_assert_attributes (ATTRIBUTE_KEY,
FRAUD_ATTRIBUTE_NAME)
values (OutgoiOBPaymentService_Is2FACompleted', 'is_2fa_
completed')
/
```

2. Add/Modify fraud rules in OAAM to use the extra attribute

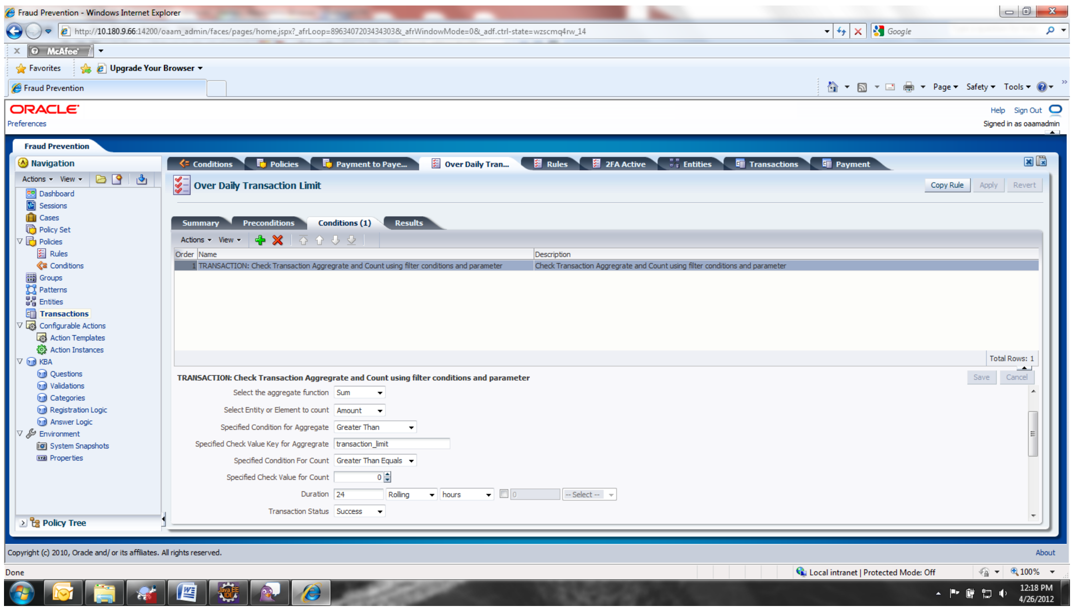
Figure 13–6 Add or Modify Fraud Rules in OAAM - Data Tab


Fraud Prevention - Windows Internet Explorer
http://10.180.9.66:14200/oaam_admin/faces/pages/home.jspx?_afrLoop=8963407203434303&_afrWindowMode=0&_adf.ctrl-state=wzscmq4rw_14
ORACLE
Preferences
Help Sign Out
Signed in as oaamadmin
Fraud Prevention
Navigation
Dashboard
Sessions
Cases
Policy Set
Policies
Rules
Conditions
Groups
Patterns
Entities
Transactions
Configurable Actions
Action Templates
Action Instances
KBA
Questions
Validations
Categories
Registration Logic
Answer Logic
Environment
System Snapshots
Properties
Policy Tree
Conditions | Policies | Payment to Paye... | Over Daily Tran... | Rules | 2FA Active | Entities | Transactions | Payment
Payment
Apply Revert
General | Entities (5) | Data (25) | Source Data (33) | Mapping
Actions View Detach
Total Rows: 25
Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
About
Local intranet | Protected Mode: Off
12:16 PM 4/26/2012

Transaction data is unique for each transaction occurrence and therefore not reusable across different transactions. For example, the total dollar amount of a transaction would not be reused in multiple transactions so it should be transaction data not an entity.

Row	Transaction Data Name	Internal ID	Description	Required	Data Type	Is Encrypted?
1	Transaction Type	transaction_type	Transaction Type	False	String data ty	False
2	Application Id	application_id	Application Id	False	String data ty	False
3	Channel Id	channel_id	Channel Id	False	String data ty	False
4	Is Future Dated	is_future_dated	Is Future Dated	False	String data ty	False
5	Is Offline	is_offline	Is Offline	False	String data ty	False
6	Amount	amount	Amount	False	Numeric data	False
7	Is PreAuthorized Payee	is_preauthorizedpa	Is PreAuthorized Payee	False	String data ty	False
8	Is IntraFlex	is_intraflex	Is IntraFlex	False	String data ty	False
9	2FA Status	2fa_status	2FA Status	False	String data ty	False
10	Payee List	payee_list	Payee List	False	String data ty	False
11	Daily Transaction Limit	transaction_limit	Daily Transaction Limit	False	String data ty	False
12	Daily Threshold Limit	threshold_limit	Daily Threshold Limit	False	String data ty	False
13	Banker Id	banker_id	Banker Id	False	String data ty	False
14	Account Id	account_id	Account Id	False	String data ty	False
15	Phone CLI	phone_cli	Phone CLI	False	String data ty	False

Figure 13–7 Add or Modify Fraud Rules in OAAM - Conditions Tab



Over Daily Transaction Limit

Order	Name	Description
1	TRANSACTION: Check Transaction Aggregate and Count using filter conditions and parameter	Check Transaction Aggregate and Count using filter conditions and parameter

TRANSACTION: Check Transaction Aggregate and Count using filter conditions and parameter

Select the aggregate function	Sum
Select Entity or Element to count	Amount
Specified Condition for Aggregate	Greater Than
Specified Check Value Key for Aggregate	transaction_limit
Specified Condition For Count	Greater Than Equals
Specified Check Value for Count	0
Duration	24 Rolling hours
Transaction Status	Success

13.3 Matrix Based Approvals – Adding Attributes

OBP uses OPSS to assert matrix-based approvals. The matrix comprises of various facts.

Example of a matrix-based rule:

```
Grant
Role = CreditAnalyst
Service=com.ofss.fc.app.origination.service.lending.core.credit.de
cision.CreditDecisionApplicationService.approveDecision
Action = performWithoutApprovals
IF CreditDecisionMatrix_Margin > 1
AND CreditDecisionMatrix_CustomerExposure > 10000000
```

In the above example, the following facts (attributes) make up the access policy rule:

```
CreditDecisionMatrix_Margin
CreditDecisionMatrix_CustomerExposure
```

The security framework allows for addition to the facts that can be used in rules.

The steps to add facts are same as described in above section.

Note

The only difference between the policy semantics in the example mentioned under this and last action is the 'Action'. ['perform' versus 'performWithoutApprovals']

13.4 Security Validators

In addition to OPSS access policies, there are additional validators that perform security checks. The sole purpose of these validators was to give hooks to enable site specific security logic that would be complicated enough and hence cannot be provisioned as rules.

Note

These additional validators come into effect only when the following property is set.

APPLICATION_SECURITY_VALIDATOR=true

The role, channel, service and the attributes available in the execution context are passed to the validators.

The validators implement the interface com.ofss.fc.app.adapter.impl.sms.validator.IExtendableApplicationValidator

There are three types of security-validation categories:

- Customer validators
- Account validators
- Business unit validators

There can be multiple validator classes contributing to each individual category.

The package structure of the validators is required to be:

```
'com.ofss.fc.app.adapter.impl.sms.validator'
```

13.4.1 Customer Validators

This validator returns a Boolean signifying whether the logged-in user can perform a transaction on the customer.

Step 1
Add property in ApplicationValidators.properties

```
com.ofss.fc.app.dda.service.account.core.DDAInquiryApplicationServ
ice.fetchBasicDetails.CustomerValidators=RestrictedAccountApplicat
ionValidator,EmployeeAccountApplicationValidator
```

Step 2
Develop custom validator along the lines of existing adapters.

13.4.2 Account Validators

This validator returns a Boolean signifying whether the logged-in user can perform a transaction on the account.

Step 1
Add property in ApplicationValidators.properties

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

Step 2

Develop custom validator along the lines of existing adapters.

13.4.3 Business Unit Validators

This validator returns a Boolean signifying whether the logged-in user can perform a transaction on the business unit.

Step 1

Add property in ApplicationValidators.properties

```
APPLY_BUSINESS_UNIT_VALIDATION_TO_ALL_SERVICES=false
com.ofss.fc.app.dda.service.account.core.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

13.5 Customizing User Search

OBP application services use SessionContext as an input parameter to set the context of the user interacting with the system. The session-context is populated out of the user's details in OID. Across implementations, the user metadata (objectclasses, attributes) is expected to be different resulting in the requirements to have a custom user search capability.

The security framework provides an extension point to inject a custom search. The steps are given in the next section.

13.5.1 Steps

SecurityConstants.properties contains attributes that enable custom user searches.

Step 1

Add properties in SecurityConstants.properties.

```
CUSTOM_SEARCH_
CLASS=com.ofss.fc.domain.ixface.sms.service.utils.CustomUserSearch
Adapter.retrieveUserUsingExtendableAttributes
CUSTOM_SEARCH_PARAM=nagactualaccessid
```

Step 2
Develop custom user search adapter.

13.6 Customizing One-Time-Password (OTP) Processing Logic

OBP uses OAAM for step-up authentication and fraud assertions. Customer is asked to enter a one-time password (OTP) if OAAM suspects the transaction to be fraudulent. The logic to send or validate an OTP is implemented using a custom hook. Details of the extension are given in the next section.

13.6.1 Steps

OAAM.properties contains a property that provides an extension for second factor password generation / dispatch.

Steps:

1. Add property for the class implementing 2FA in OAAM.properties

```
TWO_FACTOR_AUTH_
SERVICE=com.ofss.fc.domain.ixface.oaam.TwoFactorAuthService
```

2. Develop custom class.

13.7 Customizing Role Evaluation

OPSS can be configured to add a user in multiple groups (enterprise roles), as a result of which a user can have multiple application roles. OBP uses the most significant role amongst this list to query the user's severity configuration.

The default role-evaluator can be overridden to provide custom role evaluation logic. The steps to do this are given in the next section.

13.7.1 Steps

SecurityConstants.properties contains an attribute that provides an extension for a custom role evaluator.

Step 1
Replace property value in SecurityConstants.properties

```
ROLE_
EVALUATOR=com.ofss.fc.domain.sms.entity.user.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.

13.8 Customizing Limits Exclusions

OBP application services evaluate transaction limits for various services. The assertion logic excludes limits checks for certain conditions. Example, if the customer is transferring funds to his own accounts. Banks have site-specific requirements to exclude transactions from limits checks. The security framework provides an extension point to inject a custom limits exclusions adapter. The steps are given in the next section.

13.8.1 Steps

LimitsExclusions.properties contains a property that enables custom limit exclusions logic for a particular service.

Step 1

Add properties in LimitsExclusions.properties

```
EXCLUSION_PACKAGE_NAME=com.ofss.fc.app.adapter.impl.sms.exclusions
com.ofss.fc.app.dda.service.transaction.DemandDepositFundsTransfer
Service.
transferFundsToBeneficiaries=TransferFundsExclusionValidator
```

Step 2

Develop custom limits exclusions adapter.

13.9 Customizing Business Rules

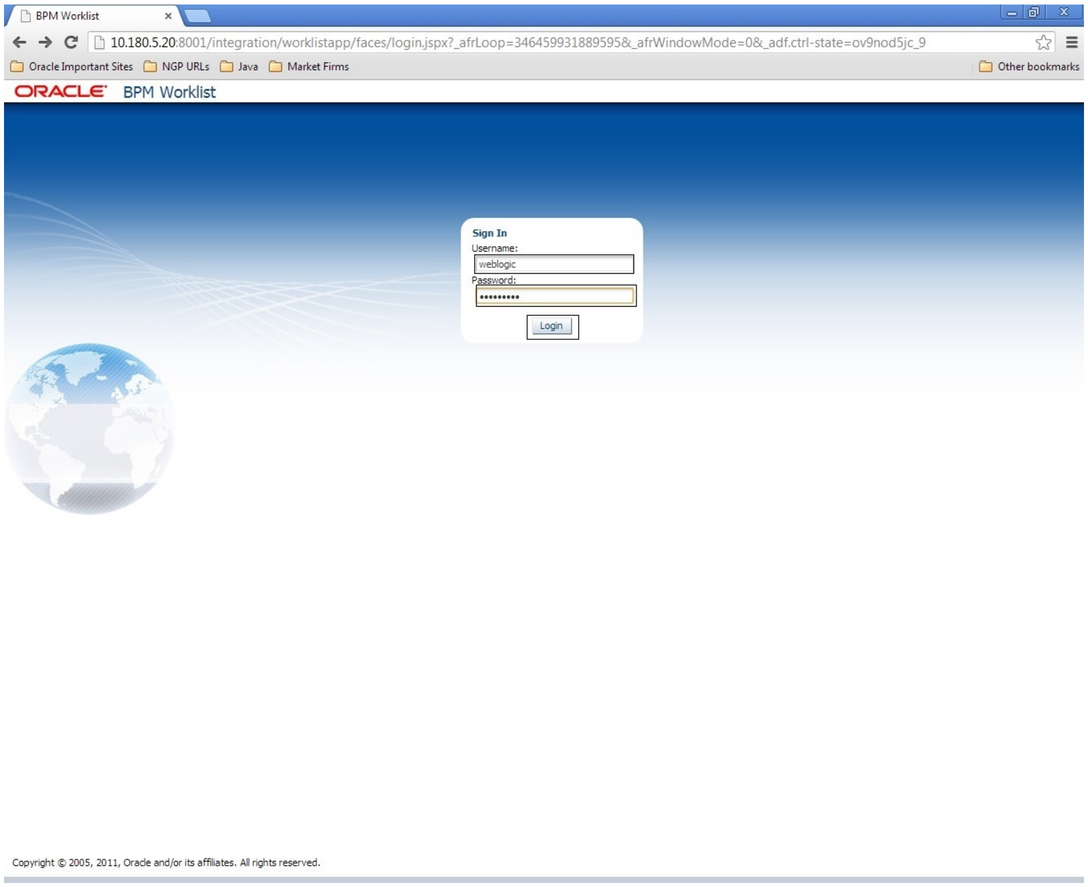
BPEL approval process business rules can be configured and it is based on input authorizations raised during transaction processing at OBP host. The steps for configuring the business rules of the approvals are given in the below section.

13.9.1 Steps to Update the Business Rules by Browser

Following are the steps to update the business rules by browser.

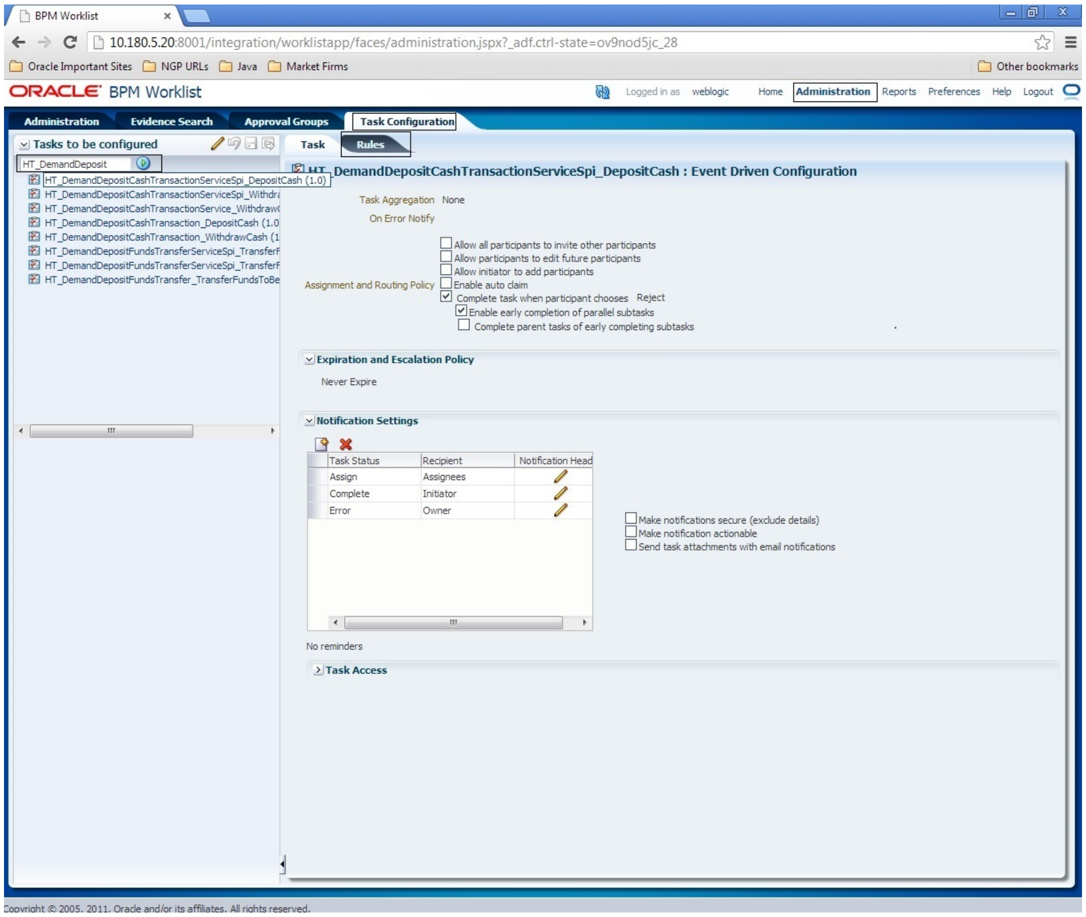
1. Log in to BPM Worklist application of the OBP.

Figure 13–8 Log in to BPM Worklist Application screen


BPM Worklist
10.180.5.20:8001/integration/worklistapp/faces/login.jspx?_afrLoop=346459931889595&_afrWindowMode=0&_adf.ctrl-state=ov9nod5jc_9
Oracle Important Sites
NGP URLs
Java
Market Firms
Other bookmarks
ORACLE BPM Worklist
Sign In
Username:
weblogic
Password:
Login
Copyright © 2005, 2011, Oracle and/or its affiliates. All rights reserved.

2. Select the 'Task' in the select box from the 'Task Configuration' tab in 'Administration'.

Figure 13–9 Task Configuration tab



BPM Worklist

10.180.5.20:8001/integration/worklistapp/faces/administration.jspx?_adf.ctrl-state=ov9nod5jc_28

Oracle Important Sites | NGP URLs | Java | Market Firms | Other bookmarks

ORACLE BPM Worklist — Logged in as weblogic — Home | Administration | Reports | Preferences | Help | Logout

Administration | Evidence Search | Approval Groups | Task Configuration

Tasks to be configured

HT_DemandDeposit

- HT_DemandDepositCashTransactionServiceSpi_DepositCash (1.0)
- HT_DemandDepositCashTransactionServiceSpi_Withdra
- HT_DemandDepositCashTransactionService_Withdraw
- HT_DemandDepositCashTransaction_DepositCash (1.0
- HT_DemandDepositCashTransaction_WithdrawCash (1
- HT_DemandDepositFundsTransferServiceSpi_TransferF
- HT_DemandDepositFundsTransferServiceSpi_TransferF
- HT_DemandDepositFundsTransfer_TransferFundsToBe

Task | Rules

HT_DemandDepositCashTransactionServiceSpi_DepositCash : Event Driven Configuration

Task Aggregation None

On Error Notify

Assignment and Routing Policy
- ☐ Allow all participants to invite other participants
- ☐ Allow participants to edit future participants
- ☐ Allow initiator to add participants
- ☐ Enable auto claim
- ☑ Complete task when participant chooses Reject
  - ☑ Enable early completion of parallel subtasks
  - ☐ Complete parent tasks of early completing subtasks

Expiration and Escalation Policy

Never Expire

Notification Settings

Task Status	Recipient	Notification Head
Assign	Assignees	✎
Complete	Initiator	✎
Error	Owner	✎

- ☐ Make notifications secure (exclude details)
- ☐ Make notification actionable
- ☐ Send task attachments with email notifications

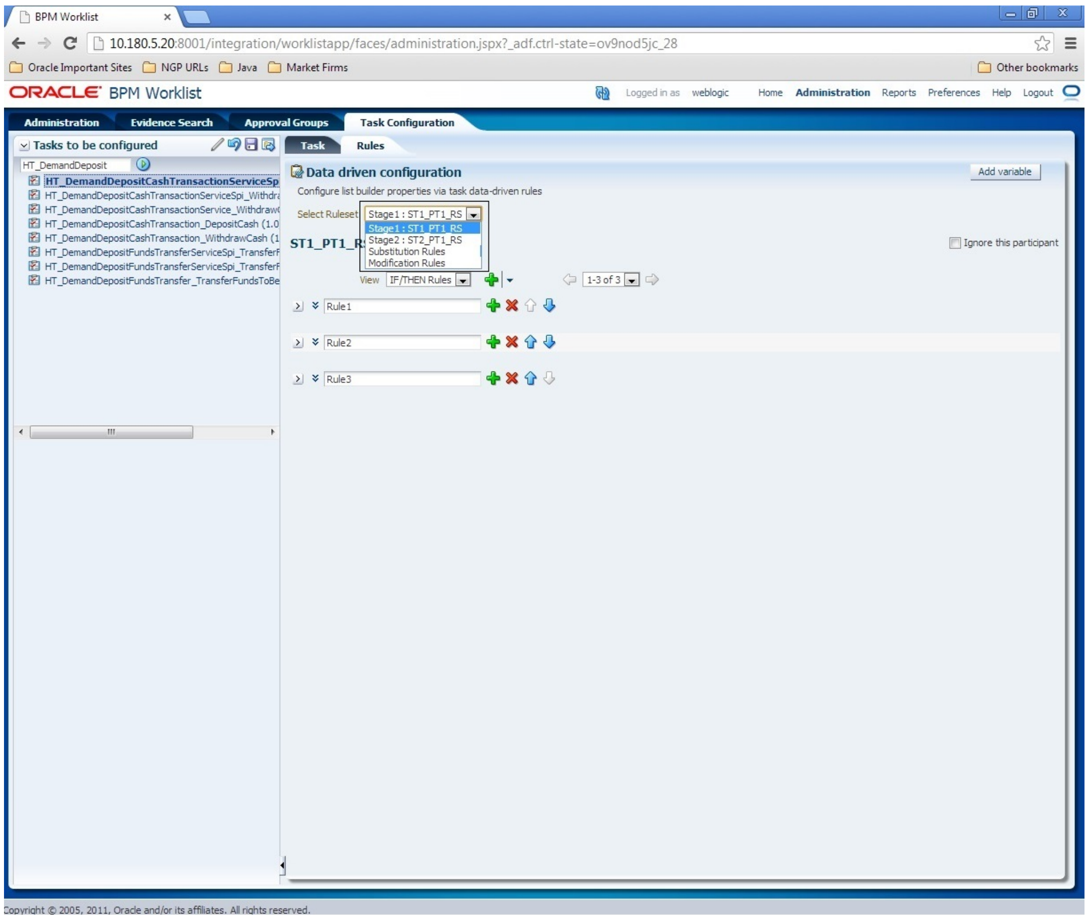
No reminders

Task Access

Copyright © 2005, 2011, Oracle and/or its affiliates. All rights reserved.

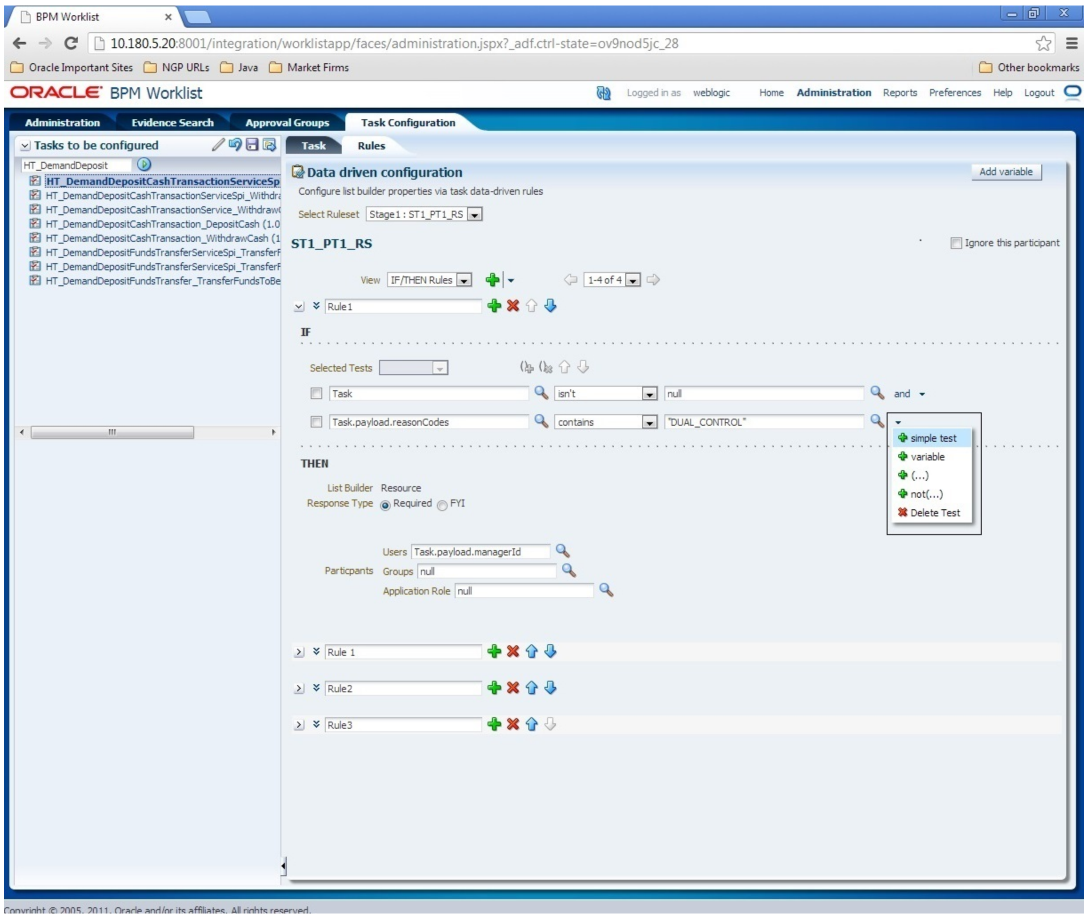
3. In the 'Rules' tab of the 'Task Configuration' screen, select the stages of approval where the condition in rule is required to be updated.

Figure 13–10 Stages of Approval



4. After stage selection, select the specific rule where the condition needs to be updated. The existing condition can be updated or the new test condition (simple/variable) can be added.

Figure 13–11 Select Test Condition



10.180.5.20:8001/integration/worklistapp/faces/administration.jspx?_adf.ctrl-state=ov9nod5jc_28

Logged in as weblogic

Tasks to be configured: HT_DemandDeposit

- HT_DemandDepositCashTransactionServiceSp
- HT_DemandDepositCashTransactionServiceSpi_Withdra
- HT_DemandDepositCashTransactionService_Withdraw
- HT_DemandDepositCashTransaction_DepositCash (1.0
- HT_DemandDepositCashTransaction_WithdrawCash (1
- HT_DemandDepositFundsTransferServiceSpi_TransferF
- HT_DemandDepositFundsTransferServiceSpi_TransferF
- HT_DemandDepositFundsTransfer_TransferFundsToBe

Data driven configuration

Configure list builder properties via task data-driven rules

Select Ruleset: Stage1 : ST1_PT1_RS

ST1_PT1_RS

View: IF/THEN Rules 1-4 of 4

Rule1

IF

Task	isn't	null	and
Task.payload.reasonCodes	contains	"DUAL_CONTROL"	

THEN

List Builder: Resource

Response Type: [✓ Required] FYI

Particpants

Users: Task.payload.managerId

Groups: null

Application Role: null

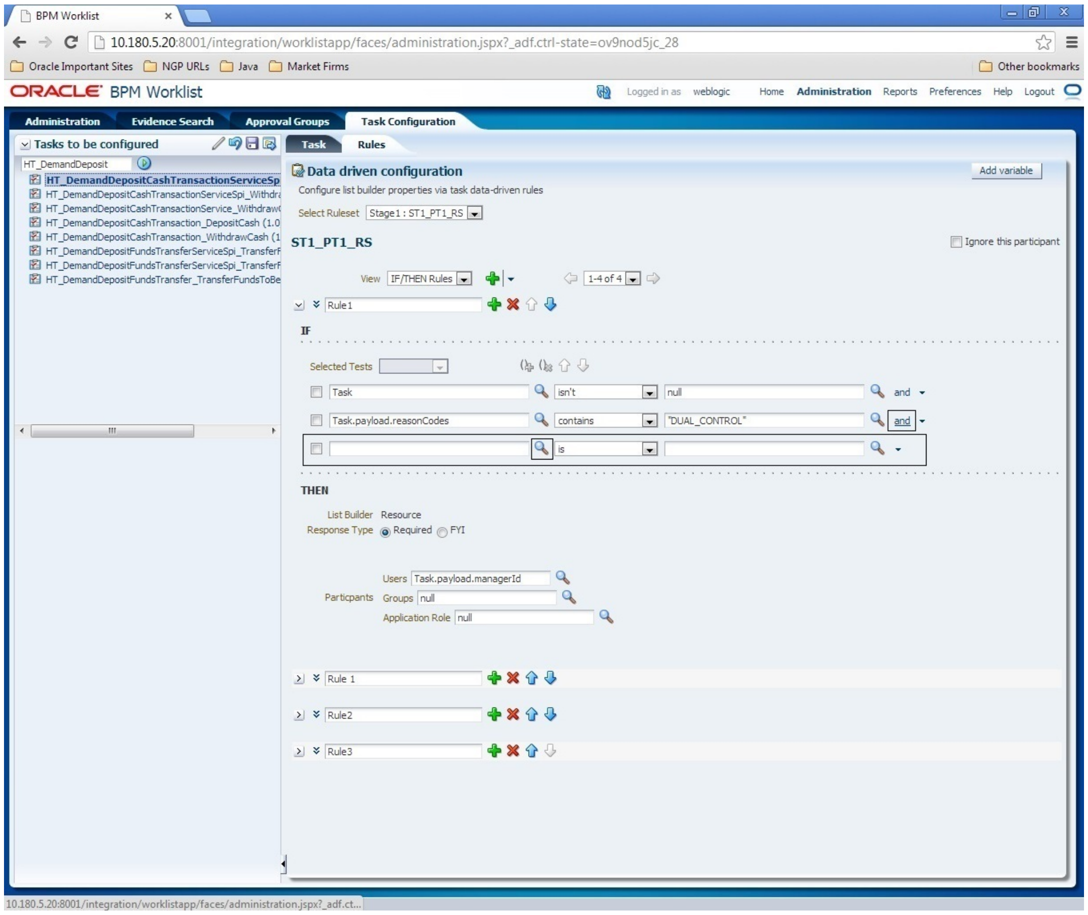
Rule 1

Rule2

Rule3

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

Figure 13–12 Select Values



HT_DemandDeposit

- HT_DemandDepositCashTransactionServiceSp
- HT_DemandDepositCashTransactionServiceSpi_Withdr
- HT_DemandDepositCashTransactionService_Withdraw
- HT_DemandDepositCashTransaction_DepositCash (1.0
- HT_DemandDepositCashTransaction_WithdrawCash (1
- HT_DemandDepositFundsTransferServiceSpi_TransferF
- HT_DemandDepositFundsTransferServiceSpi_TransferF
- HT_DemandDepositFundsTransfer_TransferFundsToBe

Data driven configuration

Configure list builder properties via task data-driven rules

Select Ruleset: Stage1 : ST1_PT1_RS

ST1_PT1_RS

Ignore this participant

View: IF/THEN Rules 1-4 of 4

Rule1

IF

Selected Tests

Task	isn't	null	and
Task.payload.reasonCodes	contains	"DUAL_CONTROL"	and
	is		

THEN

List Builder: Resource

Response Type: [✓ Required] FYI

Particpants

Users: Task.payload.managerId

Groups: null

Application Role: null

Rule 1

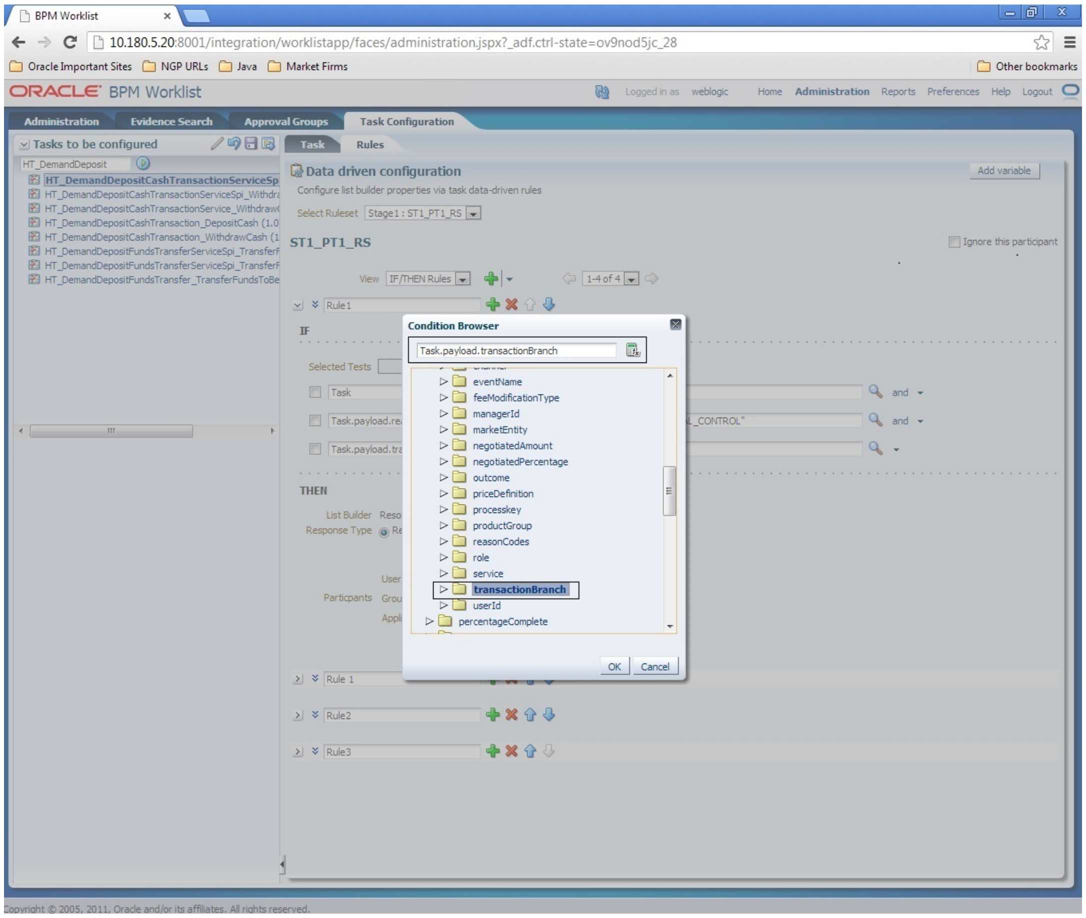
Rule2

Rule3

10.180.5.20:8001/integration/worklistapp/faces/administration.jspx?_adf.ct...

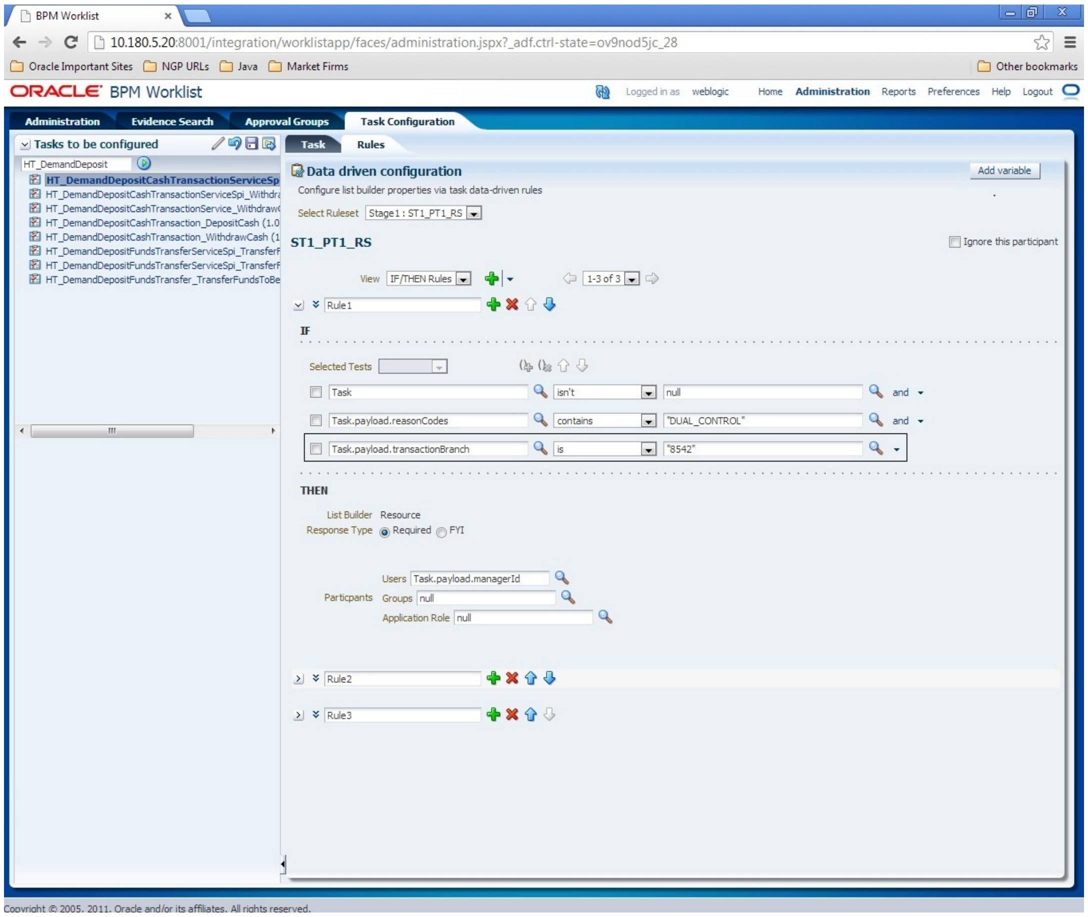
6. On selection of the search button next to the variable select box, the condition browser opens where the specific task can be selected.

Figure 13–13 Select Specific Task



7. Update the variable, operator and value of the expression in a row.

Figure 13–14 Update Values



HT_DemandDeposit
- HT_DemandDepositCashTransactionServiceSp
- HT_DemandDepositCashTransactionServiceSpi_Withdra
- HT_DemandDepositCashTransactionService_Withdraw
- HT_DemandDepositCashTransaction_DepositCash (1.0
- HT_DemandDepositCashTransaction_WithdrawCash (1
- HT_DemandDepositFundsTransferServiceSpi_TransferF
- HT_DemandDepositFundsTransferServiceSpi_TransferF
- HT_DemandDepositFundsTransfer_TransferFundsToBe

Data driven configuration

Configure list builder properties via task data-driven rules

Select Ruleset Stage1 : ST1_PT1_RS

ST1_PT1_RS

Rule1

IF

Selected Tests

Task	isn't	null	and
Task.payload.reasonCodes	contains	"DUAL_CONTROL"	and
Task.payload.transactionBranch	is	"8542"	

THEN

List Builder Resource

Response Type [✓ Required] FYI

Particpants

Users Task.payload.managerId

Groups null

Application Role null

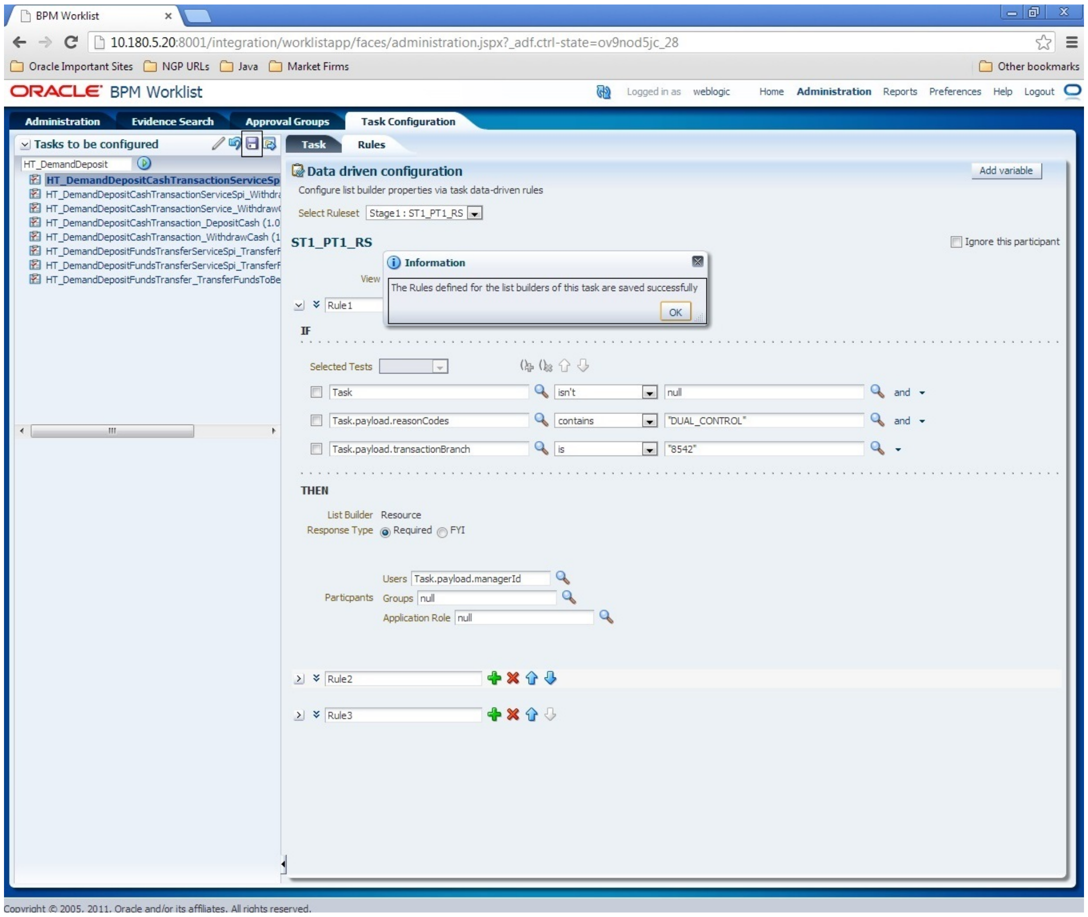
Rule2

Rule3

Copyright © 2005, 2011, Oracle and/or its affiliates. All rights reserved.

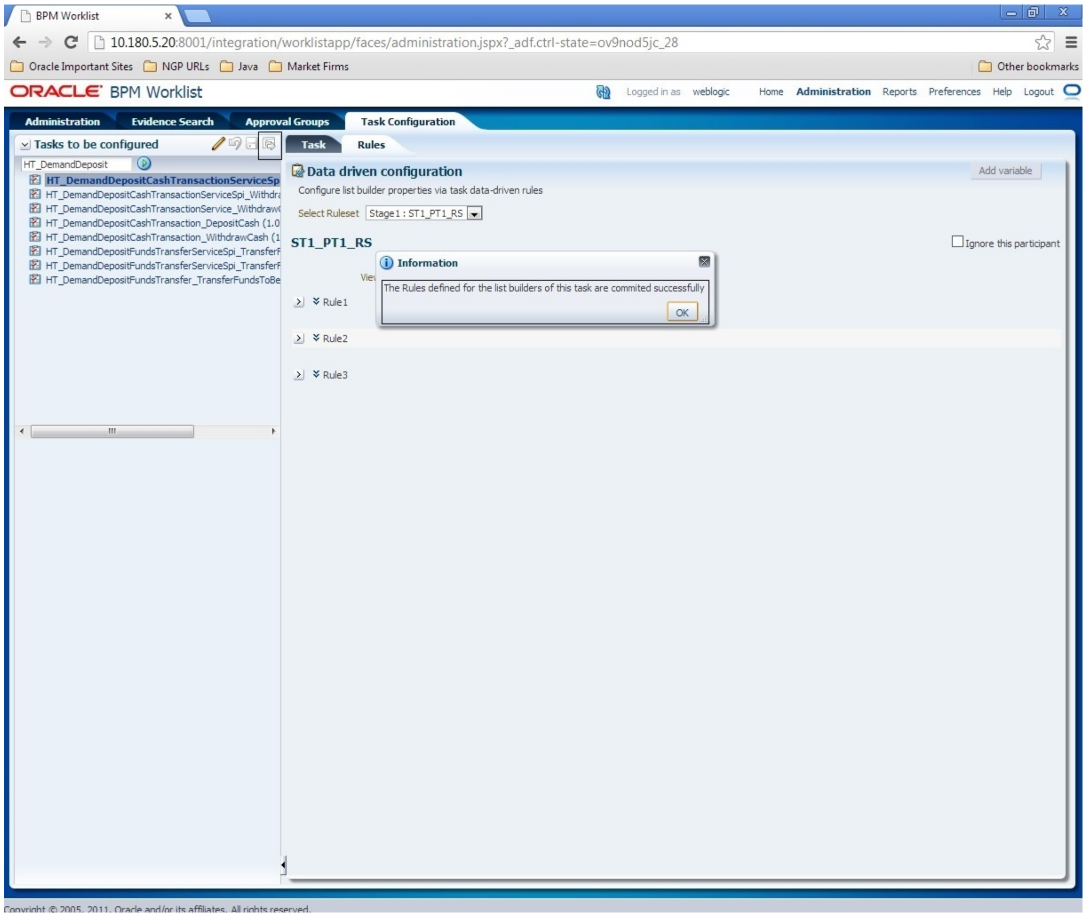
8. Save the updated rule using the save button in the left side menu.

Figure 13–15 Save the Updated Rule


BPM Worklist
10.180.5.20:8001/integration/worklistapp/faces/administration.jspx?_adf.ctrl-state=ov9nod5jc_28
Oracle Important Sites
NGP URLs
Java
Market Firms
Other bookmarks
ORACLE BPM Worklist
Logged in as weblogic
Home
Administration
Reports
Preferences
Help
Logout
Administration
Evidence Search
Approval Groups
Task Configuration
Tasks to be configured
HT_DemandDeposit
HT_DemandDepositCashTransactionServiceSp
HT_DemandDepositCashTransactionServiceSpi_Withdra
HT_DemandDepositCashTransactionService_Withdraw
HT_DemandDepositCashTransaction_DepositCash (1.0
HT_DemandDepositCashTransaction_WithdrawCash (1
HT_DemandDepositFundsTransferServiceSpi_TransferF
HT_DemandDepositFundsTransferServiceSpi_Transfer
HT_DemandDepositFundsTransfer_TransferFundsToBe
Task
Rules
Data driven configuration
Add variable
Configure list builder properties via task data-driven rules
Select Ruleset Stage1 : ST1_PT1_RS
ST1_PT1_RS
Ignore this participant
Information
The Rules defined for the list builders of this task are saved successfully
OK
View
Rule1
IF
Selected Tests
Task isn't null and
Task.payload.reasonCodes contains "DUAL_CONTROL" and
Task.payload.transactionBranch is "8542"
THEN
List Builder Resource
Response Type Required FYI
Users Task.payload.managerId
Particpants Groups null
Application Role null
Rule2
Rule3
Copyright © 2005, 2011, Oracle and/or its affiliates. All rights reserved.

9. Commit the changes in the rule by clicking the commit button.

Figure 13–16 Commit the Changes



10.180.5.20:8001/integration/worklistapp/faces/administration.jspx?_adf.ctrl-state=ov9nod5jc_28

HT_DemandDeposit

HT_DemandDepositCashTransactionServiceSp
HT_DemandDepositCashTransactionServiceSpi_Withdra
HT_DemandDepositCashTransactionService_Withdraw
HT_DemandDepositCashTransaction_DepositCash (1.0
HT_DemandDepositCashTransaction_WithdrawCash (1
HT_DemandDepositFundsTransferServiceSpi_TransferF
HT_DemandDepositFundsTransferServiceSpi_TransferF
HT_DemandDepositFundsTransfer_TransferFundsToBe

Configure list builder properties via task data-driven rules

Select Ruleset Stage1 : ST1_PT1_RS

ST1_PT1_RS

The Rules defined for the list builders of this task are commited successfully

Copyright © 2005, 2011, Oracle and/or its affiliates. All rights reserved.

Note

'Ignore this participant' check box is available on the screen for ignoring the specific stage. The particular stage is then ignored while consideration of the rules implementation in the approval process.

13.9.2 Steps to Update the Business Rules in JDeveloper

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

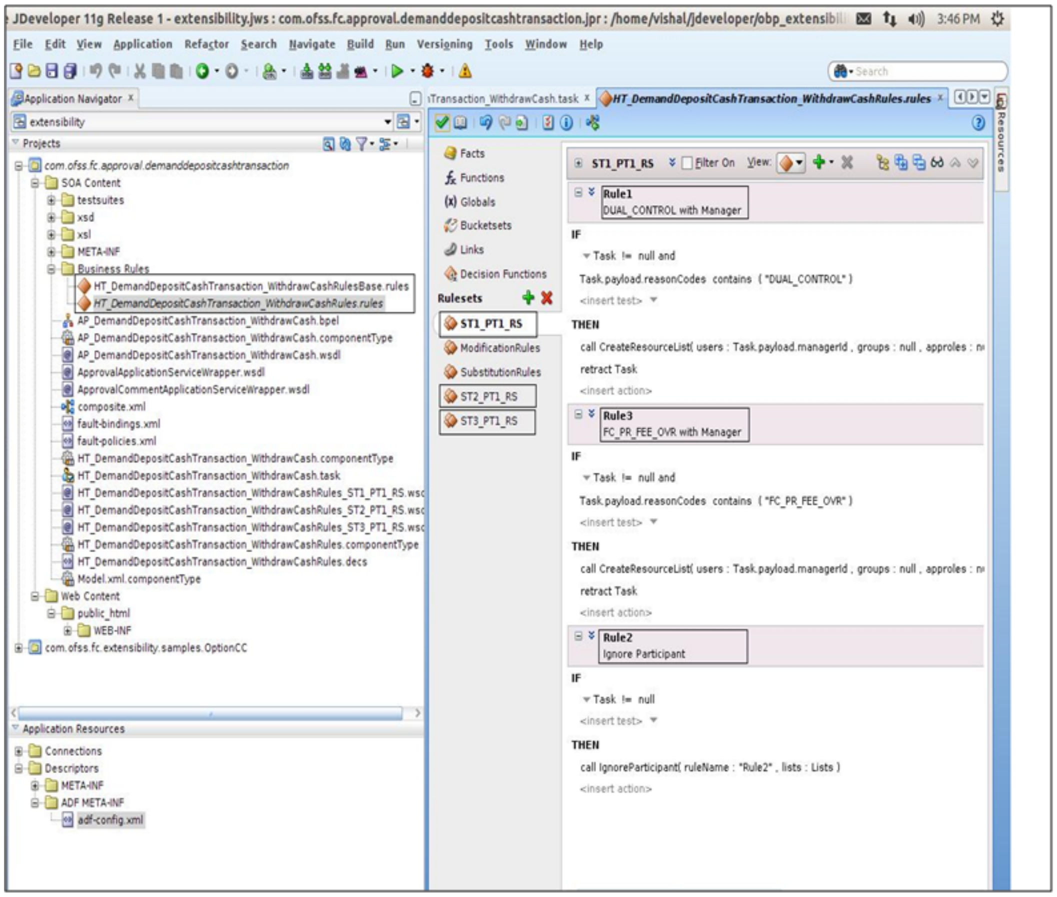
Step 1

Configure the JDeveloper in the customization option and the particular process jar import as the SOA project in the customizable mode. The details of this step are explained in this document in the section SOA customization.

Step 2

Expand the Business Rules folder of your project. You will see two .rules files in it. One will be <<HumanTaskName>>Rules.rules file and the other will be <<HumanTaskName>>RulesBase.rules file. Double Click and open the <<HumanTaskName>>Rules.rules file. The existing approval stages and rulesets will be available in the file.

Figure 13–17 Expand Business Rules



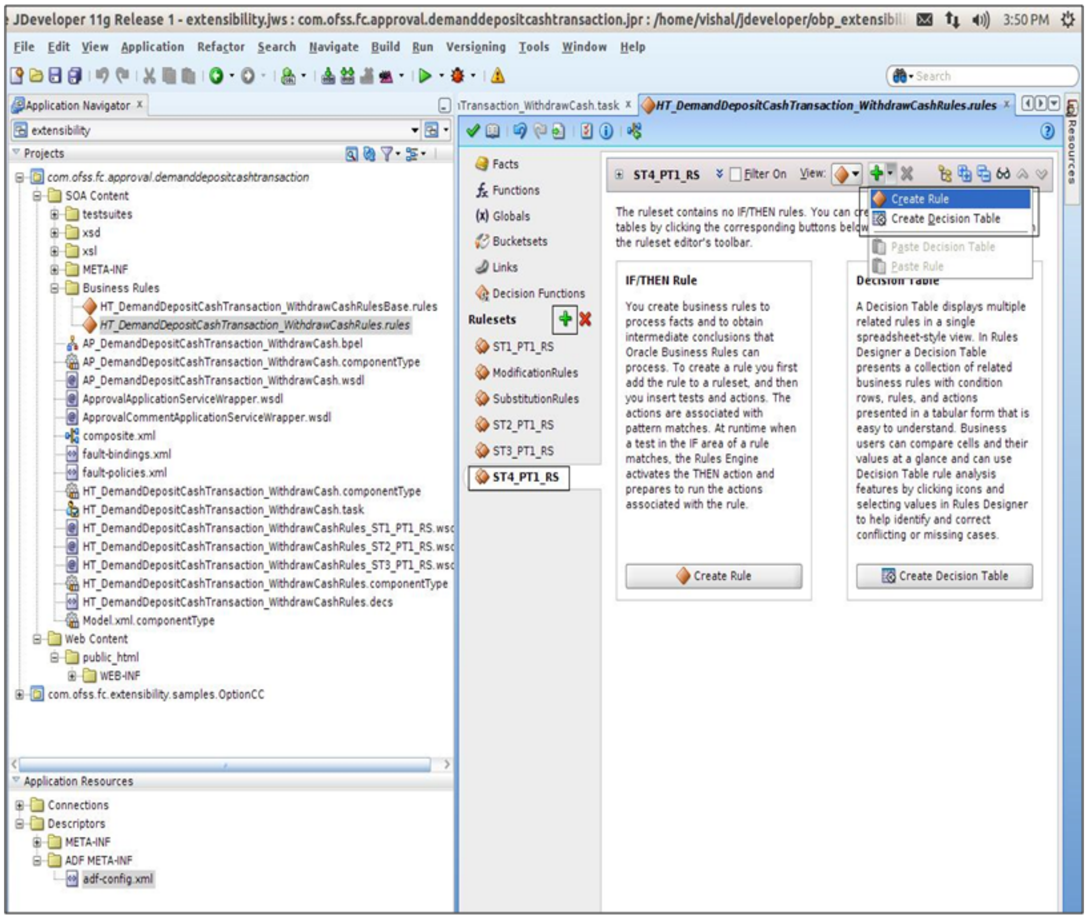
```
JDeveloper 11g Release 1 - extensibility.jws : com.ofss.fc.approval.demanddepositcashtransaction.jpr : /home/vishal/jdeveloper/obp_extensibi

Projects
com.ofss.fc.approval.demanddepositcashtransaction
  SOA Content
    testsuites
    xsd
    xsl
    META-INF
    Business Rules
      HT_DemandDepositCashTransaction_WithdrawCashRulesBase.rules
      HT_DemandDepositCashTransaction_WithdrawCashRules.rules
    AP_DemandDepositCashTransaction_WithdrawCash.bpel
    AP_DemandDepositCashTransaction_WithdrawCash.componentType
    AP_DemandDepositCashTransaction_WithdrawCash.wsdl
    ApprovalApplicationServiceWrapper.wsdl
    ApprovalCommentApplicationServiceWrapper.wsdl
    composite.xml
    fault-bindings.xml
    fault-policies.xml
    HT_DemandDepositCashTransaction_WithdrawCash.componentType
    HT_DemandDepositCashTransaction_WithdrawCash.task
    HT_DemandDepositCashTransaction_WithdrawCashRules_ST1_PT1_RS.wsc
    HT_DemandDepositCashTransaction_WithdrawCashRules_ST2_PT1_RS.wsc
    HT_DemandDepositCashTransaction_WithdrawCashRules_ST3_PT1_RS.wsc
    HT_DemandDepositCashTransaction_WithdrawCashRules.componentType
    HT_DemandDepositCashTransaction_WithdrawCashRules.decs
    Model.xml.componentType
  Web Content
    public_html
      WEB-INF
com.ofss.fc.extensibility.samples.OptionCC

Application Resources
  Connections
  Descriptors
    META-INF
    ADF META-INF
      adf-config.xml
```

```
HT_DemandDepositCashTransaction_WithdrawCashRules.rules

Rulesets: ST1_PT1_RS, ModificationRules, SubstitutionRules, ST2_PT1_RS, ST3_PT1_RS

ST1_PT1_RS

Rule1
DUAL_CONTROL with Manager
IF
  Task != null and
  Task.payload.reasonCodes contains ( "DUAL_CONTROL" )
  <insert test>
THEN
  call CreateResourceList( users : Task.payload.managerId , groups : null , approles : n
  retract Task
  <insert action>

Rule3
FC_PR_FEE_OVR with Manager
IF
  Task != null and
  Task.payload.reasonCodes contains ( "FC_PR_FEE_OVR" )
  <insert test>
THEN
  call CreateResourceList( users : Task.payload.managerId , groups : null , approles : n
  retract Task
  <insert action>

Rule2
Ignore Participant
IF
  Task != null
  <insert test>
THEN
  call IgnoreParticipant( ruleName : "Rule2" , lists : Lists )
  <insert action>
```

Step 3

Create a new stage in the format 'ST<Stage Number>_PT1_RS' by clicking the '+' button in the Rulesets. The new rules/decision table can be added to a stage.

Figure 13–18 Create New Stage


JDeveloper 11g Release 1 - extensibility.jws : com.ofss.fc.approval.demanddepositcashtransaction.jpr : /home/vishal/jdeveloper/obp_extensibil
File Edit View Application Refactor Search Navigate Build Run Versioning Tools Window Help
Application Navigator
extensibility
Projects
com.ofss.fc.approval.demanddepositcashtransaction
SOA Content
testsuites
xsd
xsl
META-INF
Business Rules
HT_DemandDepositCashTransaction_WithdrawCashRulesBase.rules
HT_DemandDepositCashTransaction_WithdrawCashRules.rules
AP_DemandDepositCashTransaction_WithdrawCash.bpel
AP_DemandDepositCashTransaction_WithdrawCash.componentType
AP_DemandDepositCashTransaction_WithdrawCash.wsdl
ApprovalApplicationServiceWrapper.wsdl
ApprovalCommentApplicationServiceWrapper.wsdl
composite.xml
fault-bindings.xml
fault-policies.xml
HT_DemandDepositCashTransaction_WithdrawCash.componentType
HT_DemandDepositCashTransaction_WithdrawCash.task
HT_DemandDepositCashTransaction_WithdrawCashRules_ST1_PT1_RS.wsc
HT_DemandDepositCashTransaction_WithdrawCashRules_ST2_PT1_RS.wsc
HT_DemandDepositCashTransaction_WithdrawCashRules_ST3_PT1_RS.wsc
HT_DemandDepositCashTransaction_WithdrawCashRules.componentType
HT_DemandDepositCashTransaction_WithdrawCashRules.decs
Model.xml.componentType
Web Content
public_html
WEB-INF
com.ofss.fc.extensibility.samples.OptionCC
Application Resources
Connections
Descriptors
META-INF
ADF META-INF
adf-config.xml
Transaction_WithdrawCash.task
HT_DemandDepositCashTransaction_WithdrawCashRules.rules
Facts
Functions
Globals
Bucketsets
Links
Decision Functions
Rulesets
ST1_PT1_RS
ModificationRules
SubstitutionRules
ST2_PT1_RS
ST3_PT1_RS
ST4_PT1_RS
Filter On
View:
Create Rule
Create Decision Table
Paste Decision Table
Paste Rule
IF/THEN Rule
Decision Table
Create Rule
Create Decision Table
Search
Resources

The ruleset contains no IF/THEN rules. You can cre tables by clicking the corresponding buttons belo the ruleset editor's toolbar.

IF/THEN Rule

You create business rules to process facts and to obtain intermediate conclusions that Oracle Business Rules can process. To create a rule you first add the rule to a ruleset, and then you insert tests and actions. The actions are associated with pattern matches. At runtime when a test in the IF area of a rule matches, the Rules Engine activates the THEN action and prepares to run the actions associated with the rule.

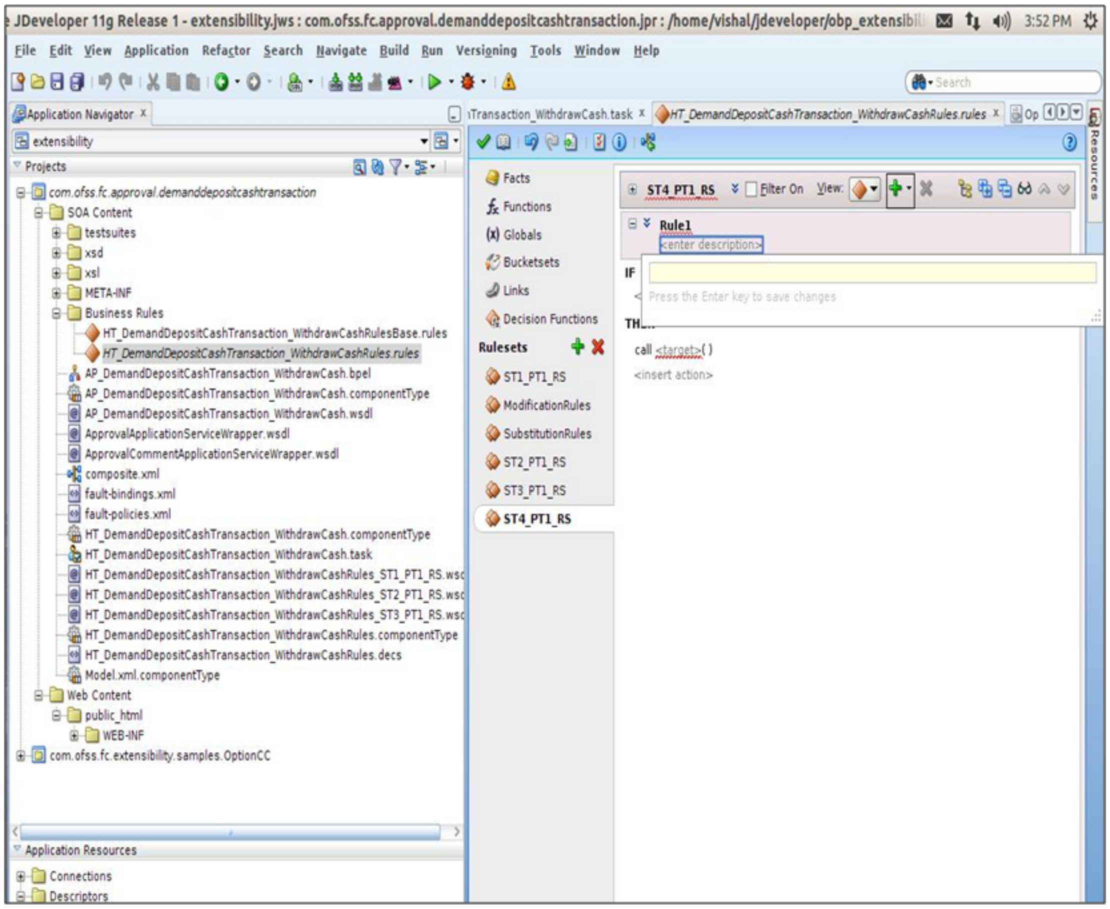
Decision Table

A Decision Table displays multiple related rules in a single spreadsheet-style view. In Rules Designer a Decision Table presents a collection of related business rules with condition rows, rules, and actions presented in a tabular form that is easy to understand. Business users can compare cells and their values at a glance and can use Decision Table rule analysis features by clicking icons and selecting values in Rules Designer to help identify and correct conflicting or missing cases.

Step 4

Add the new rule by clicking the '+' button on the stage. The existing rule can also be added/modified in the existing stage.

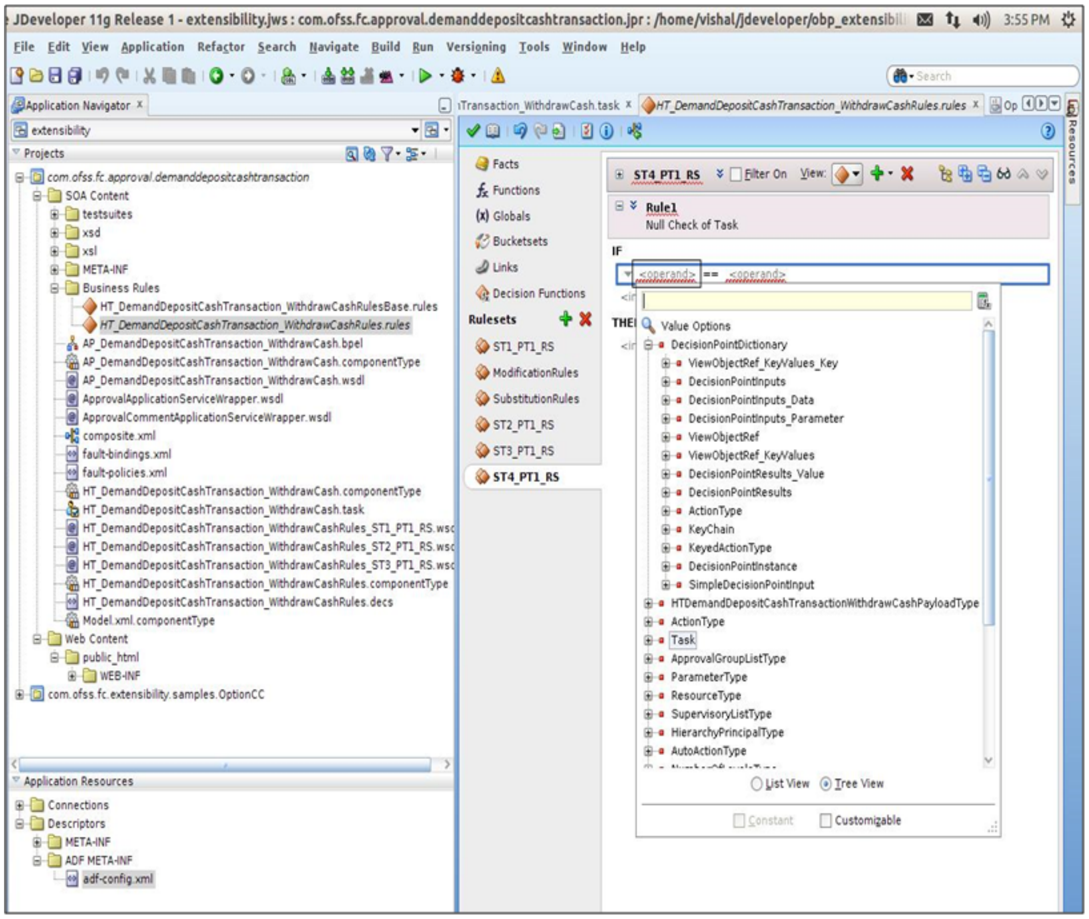
Figure 13–19 Add New Rule



Step 5

Populate the rule with the conditions in 'if' and 'then' block.

Figure 13–20 Populate the New Rule


JDeveloper 11g Release 1 - extensibility.jws : com.ofss.fc.approval.demanddepositcashtransaction.jpr : /home/vishal/jdeveloper/obp_extensibil
File Edit View Application Refactor Search Navigate Build Run Versioning Tools Window Help
Application Navigator
extensibility
Projects
com.ofss.fc.approval.demanddepositcashtransaction
SOA Content
testsuites
xsd
xsl
META-INF
Business Rules
HT_DemandDepositCashTransaction_WithdrawCashRulesBase.rules
HT_DemandDepositCashTransaction_WithdrawCashRules.rules
AP_DemandDepositCashTransaction_WithdrawCash.bpel
AP_DemandDepositCashTransaction_WithdrawCash.componentType
AP_DemandDepositCashTransaction_WithdrawCash.wsdl
ApprovalApplicationServiceWrapper.wsdl
ApprovalCommentApplicationServiceWrapper.wsdl
composite.xml
fault-bindings.xml
fault-policies.xml
HT_DemandDepositCashTransaction_WithdrawCash.componentType
HT_DemandDepositCashTransaction_WithdrawCash.task
HT_DemandDepositCashTransaction_WithdrawCashRules_ST1_PT1_RS.wsc
HT_DemandDepositCashTransaction_WithdrawCashRules_ST2_PT1_RS.wsc
HT_DemandDepositCashTransaction_WithdrawCashRules_ST3_PT1_RS.wsc
HT_DemandDepositCashTransaction_WithdrawCashRules.componentType
HT_DemandDepositCashTransaction_WithdrawCashRules.decs
Model.xml.componentType
Web Content
public_html
WEB-INF
com.ofss.fc.extensibility.samples.OptionCC
Application Resources
Connections
Descriptors
META-INF
ADF META-INF
adf-config.xml
Facts
Functions
Globals
Bucketsets
Links
Decision Functions
Rulesets
ST1_PT1_RS
ModificationRules
SubstitutionRules
ST2_PT1_RS
ST3_PT1_RS
ST4_PT1_RS
Filter On
View:
Rule1
Null Check of Task
IF
<operand> == <operand>
THEN
Value Options
DecisionPointDictionary
ViewObjectRef_KeyValues_Key
DecisionPointInputs
DecisionPointInputs_Data
DecisionPointInputs_Parameter
ViewObjectRef
ViewObjectRef_KeyValues
DecisionPointResults_Value
DecisionPointResults
ActionType
KeyChain
KeyedActionType
DecisionPointInstance
SimpleDecisionPointInput
HTDemandDepositCashTransactionWithdrawCashPayloadType
ActionType
Task
ApprovalGroupListType
ParameterType
ResourceType
SupervisoryListType
HierarchyPrincipalType
AutoActionType
List View
Tree View
Constant
Customizable
Resources

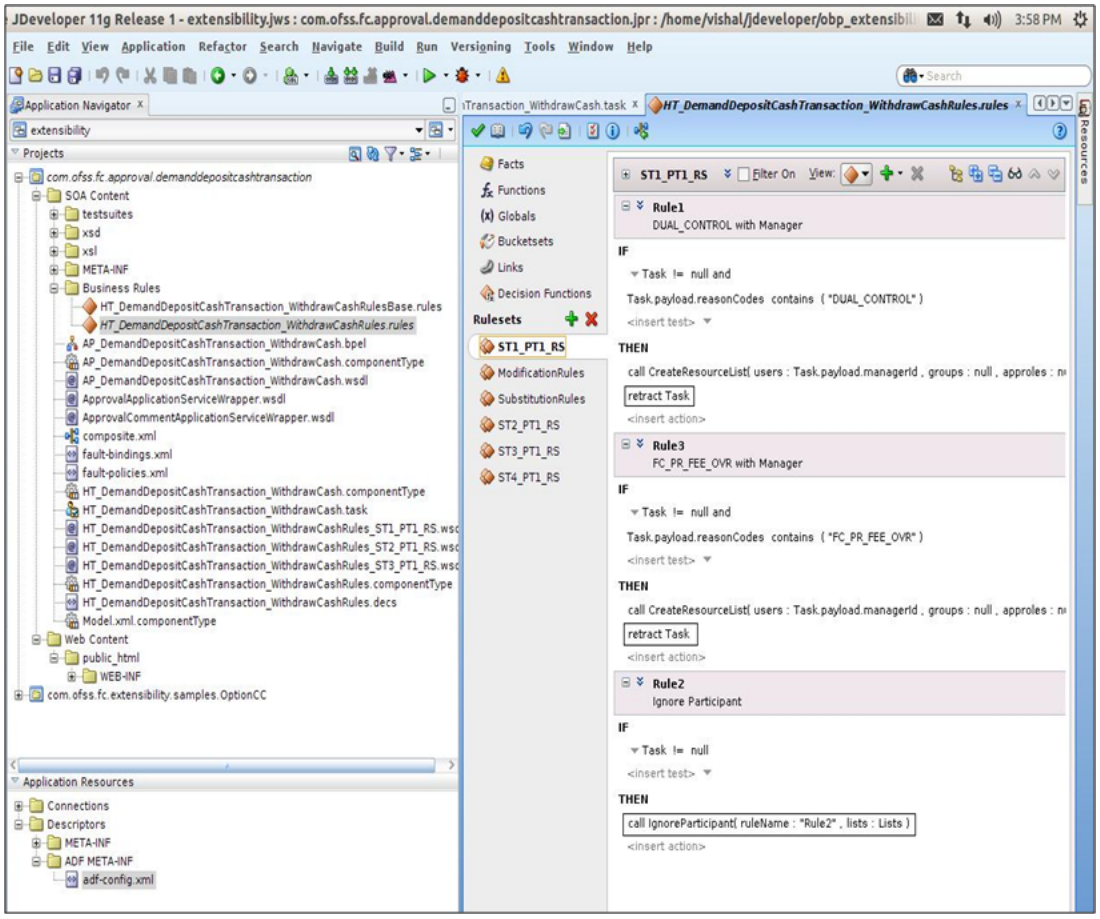
Step 6

Deploy the project jar as explained in this document in the section SOA customization.

Note

All the rules should have the final 'THEN' statement with the return type as 'retract Task'. 'retract Task' makes sure that if the condition of the rule is satisfied then the second rule should not be evaluated else the flow will execute the entire ruleset. It is also mandatory to have the last rule with the final 'THEN' statement as 'call IgnoreParticipant'. This is done to bring the control out of the ruleset.

Figure 13–21 Deploy Project Jar



JDeveloper 11g Release 1 - extensibility.jws : com.ofss.fc.approval.demanddepositcashtransaction.jpr : /home/vishal/jdeveloper/obp_extensibil 3:58 PM

File Edit View Application Refactor Search Navigate Build Run Versioning Tools Window Help

Application Navigator

extensibility

Projects

```
com.ofss.fc.approval.demanddepositcashtransaction
  SOA Content
    testsuites
    xsd
    xsl
    META-INF
    Business Rules
      HT_DemandDepositCashTransaction_WithdrawCashRulesBase.rules
      HT_DemandDepositCashTransaction_WithdrawCashRules.rules
    AP_DemandDepositCashTransaction_WithdrawCash.bpel
    AP_DemandDepositCashTransaction_WithdrawCash.componentType
    AP_DemandDepositCashTransaction_WithdrawCash.wsdl
    ApprovalApplicationServiceWrapper.wsdl
    ApprovalCommentApplicationServiceWrapper.wsdl
    composite.xml
    fault-bindings.xml
    fault-policies.xml
    HT_DemandDepositCashTransaction_WithdrawCash.componentType
    HT_DemandDepositCashTransaction_WithdrawCash.task
    HT_DemandDepositCashTransaction_WithdrawCashRules_ST1_PT1_RS.wsc
    HT_DemandDepositCashTransaction_WithdrawCashRules_ST2_PT1_RS.wsc
    HT_DemandDepositCashTransaction_WithdrawCashRules_ST3_PT1_RS.wsc
    HT_DemandDepositCashTransaction_WithdrawCashRules.componentType
    HT_DemandDepositCashTransaction_WithdrawCashRules.decs
    Model.xml.componentType
  Web Content
    public_html
      WEB-INF
com.ofss.fc.extensibility.samples.OptionCC
```

Application Resources

```
Connections
Descriptors
  META-INF
  ADF META-INF
    adf-config.xml
```

HT_DemandDepositCashTransaction_WithdrawCashRules.rules

ST1_PT1_RS Filter On View:

```
Rule1
DUAL_CONTROL with Manager
IF
  Task != null and
  Task.payload.reasonCodes contains ( "DUAL_CONTROL" )
  <insert test>
THEN
  call CreateResourceList( users : Task.payload.managerId , groups : null , approles : n
  retract Task
  <insert action>

Rule3
FC_PR_FEE_OVR with Manager
IF
  Task != null and
  Task.payload.reasonCodes contains ( "FC_PR_FEE_OVR" )
  <insert test>
THEN
  call CreateResourceList( users : Task.payload.managerId , groups : null , approles : n
  retract Task
  <insert action>

Rule2
Ignore Participant
IF
  Task != null
  <insert test>
THEN
  call IgnoreParticipant( ruleName : "Rule2" , lists : Lists )
  <insert action>
```

14 Loan Schedule Computation Algorithm

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

14.1 Adding a New Algorithm

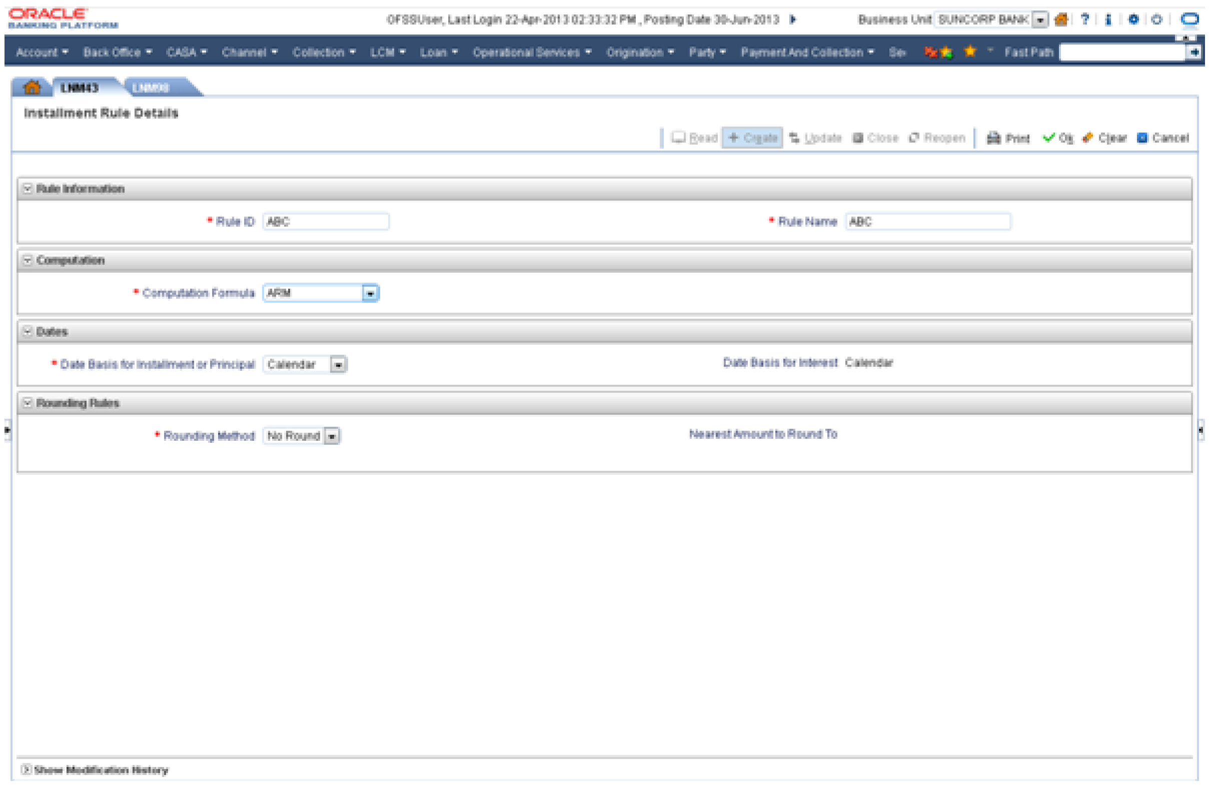
For adding a new algorithm following additions need to be done.

LoanCalculationMethodType.properties contains list of available loan stage algorithms in the system in the form of key-value pairs. For example, ARM=ARM

This list is used as part of screen LNM43 to populate a drop down called Computation Formula.

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

Figure 14–1 Add New Algorithm



OFSSUser, Last Login 22-Apr-2013 02:33:32 PM , Posting Date 30-Jun-2013

Business Unit SUNCORP BANK

Installment Rule Details

Rule Information

* Rule ID ABC

* Rule Name ABC

Computation

* Computation Formula ARM

Dates

* Date Basis for Installment or Principal Calendar

Date Basis for Interest Calendar

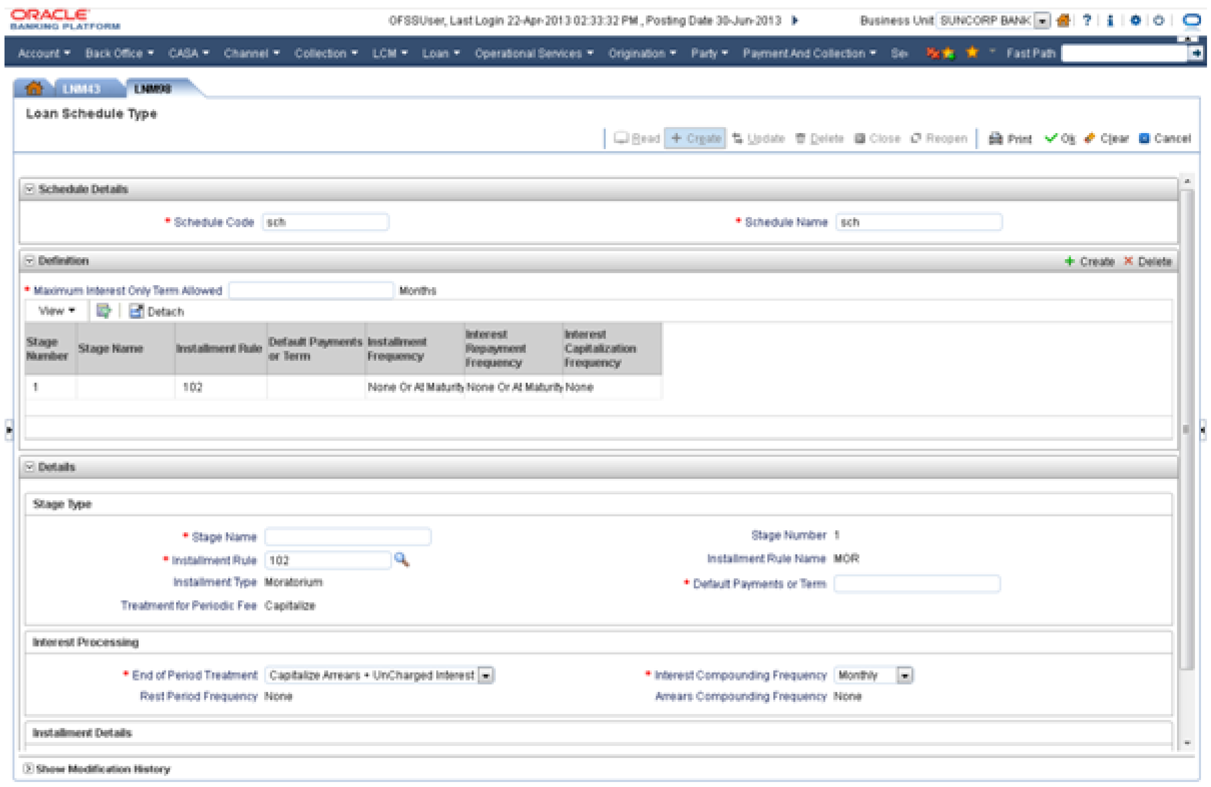
Rounding Rules

* Rounding Method No Round

Nearest Amount to Round To

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

Figure 14–2 Create New Installment



OFSSUser, Last Login 22-Apr-2013 02:33:32 PM , Posting Date 30-Jun-2013

Business Unit SUNCORP BANK

Loan Schedule Type

Schedule Details

* Schedule Code	sch	* Schedule Name	sch

Definition

* Maximum Interest Only Term Allowed ______ Months

Stage Number	Stage Name	Installment Rule	Default Payments or Term	Installment Frequency	Interest Repayment Frequency	Interest Capitalization Frequency
1		102		None Or At Maturity	None Or At Maturity	None

Details

Stage Type

* Stage Name		Stage Number	1
* Installment Rule	102	Installment Rule Name	MOR
Installment Type	Moratorium	* Default Payments or Term	
Treatment for Periodic Fee	Capitalize		

Interest Processing

* End of Period Treatment	Capitalize Arrears + UnCharged Interest	* Interest Compounding Frequency	Monthly
Rest Period Frequency	None	Arrears Compounding Frequency	None

Installment Details

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())
```

14.2 Consuming Third Party Schedules

As mentioned above the PrincipalRepaymentScheduleGeneratorFactory reads the property file ScheduleCalculator.properties which has entry for the schedule generator class to be used for a schedule code. For using third party schedule algorithms, an entry in this file has to be made against the required schedule codes.

```
IOI-EIPI-PMI_IntOnly-Month_Pr-Month_Ann=
com.ofss.external.ScheduleAlgoExt.XYZScheduleGenerator;
```

15 Facts and Rules Configuration

This chapter explains the facts and rules configuration details.

15.1 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.

15.1.1 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.

15.1.2 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.

15.1.3 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.

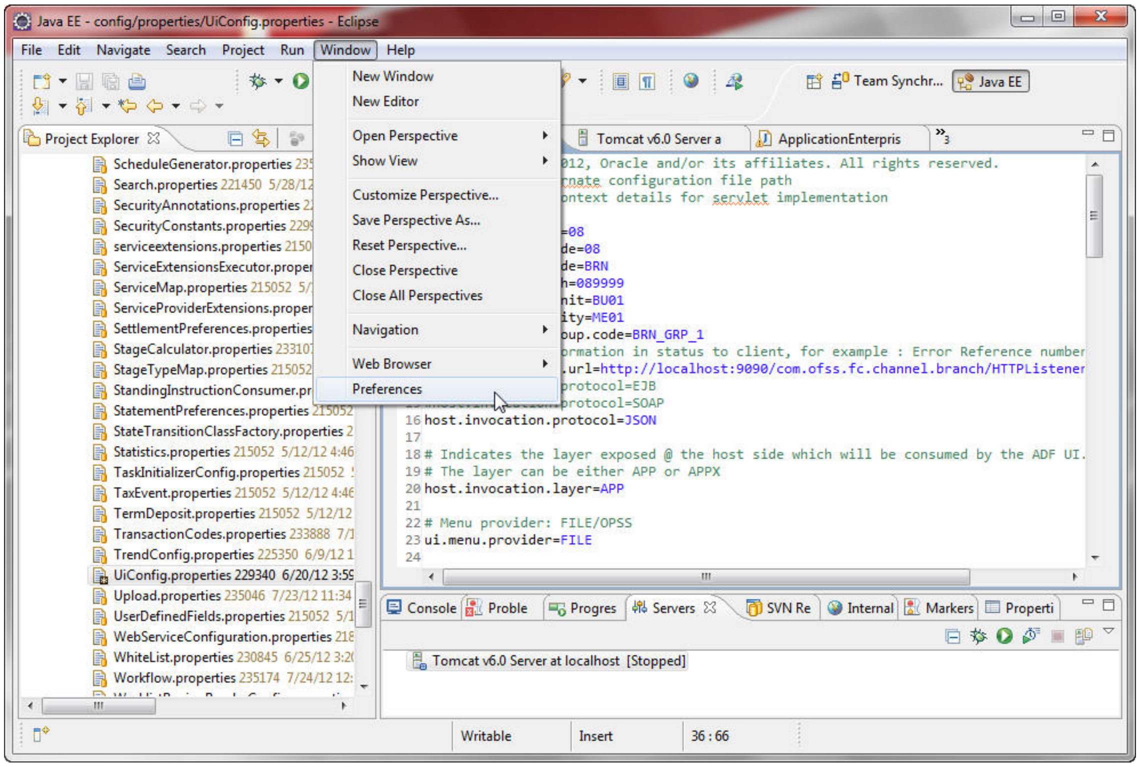
Get the Fact Plugin from the development team.

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.

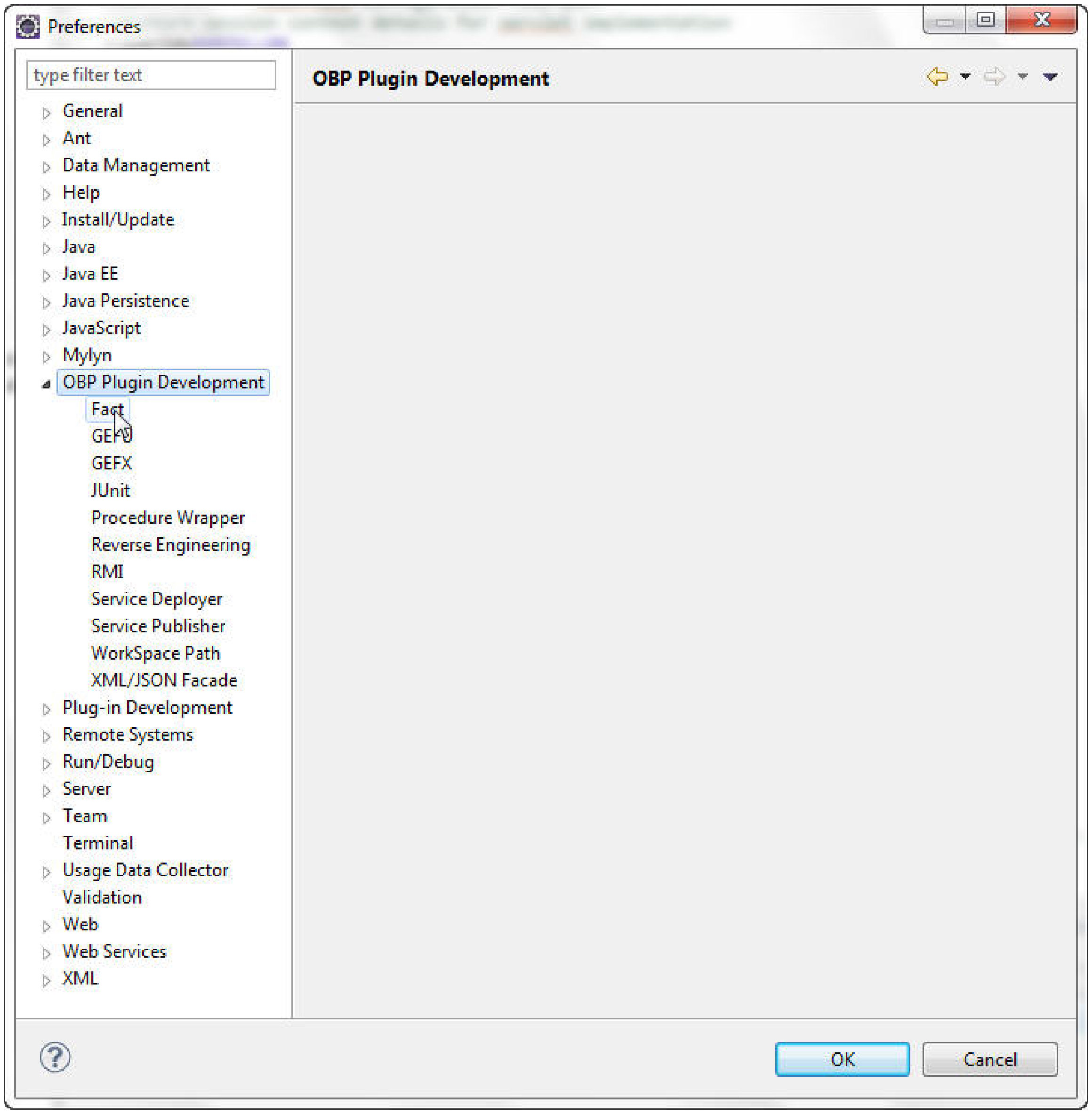
Figure 15–1 Select Window Preferences



```
012, Oracle and/or its affiliates. All rights reserved.
rnate configuration file path
ontext details for servlet implementation

=08
de=08
de=BRN
h=089999
nit=BU01
ity=ME01
oup.code=BRN_GRP_1
ormation in status to client, for example : Error Reference number
.url=http://localhost:9090/com.ofss.fc.channel.branch/HTTPListener
protocol=EJB
protocol=SOAP
host.invocation.protocol=JSON

# Indicates the layer exposed @ the host side which will be consumed by the ADF UI.
# The layer can be either APP or APPX
host.invocation.layer=APP

# Menu provider: FILE/OPSS
ui.menu.provider=FILE
```

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

Figure 15–2 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

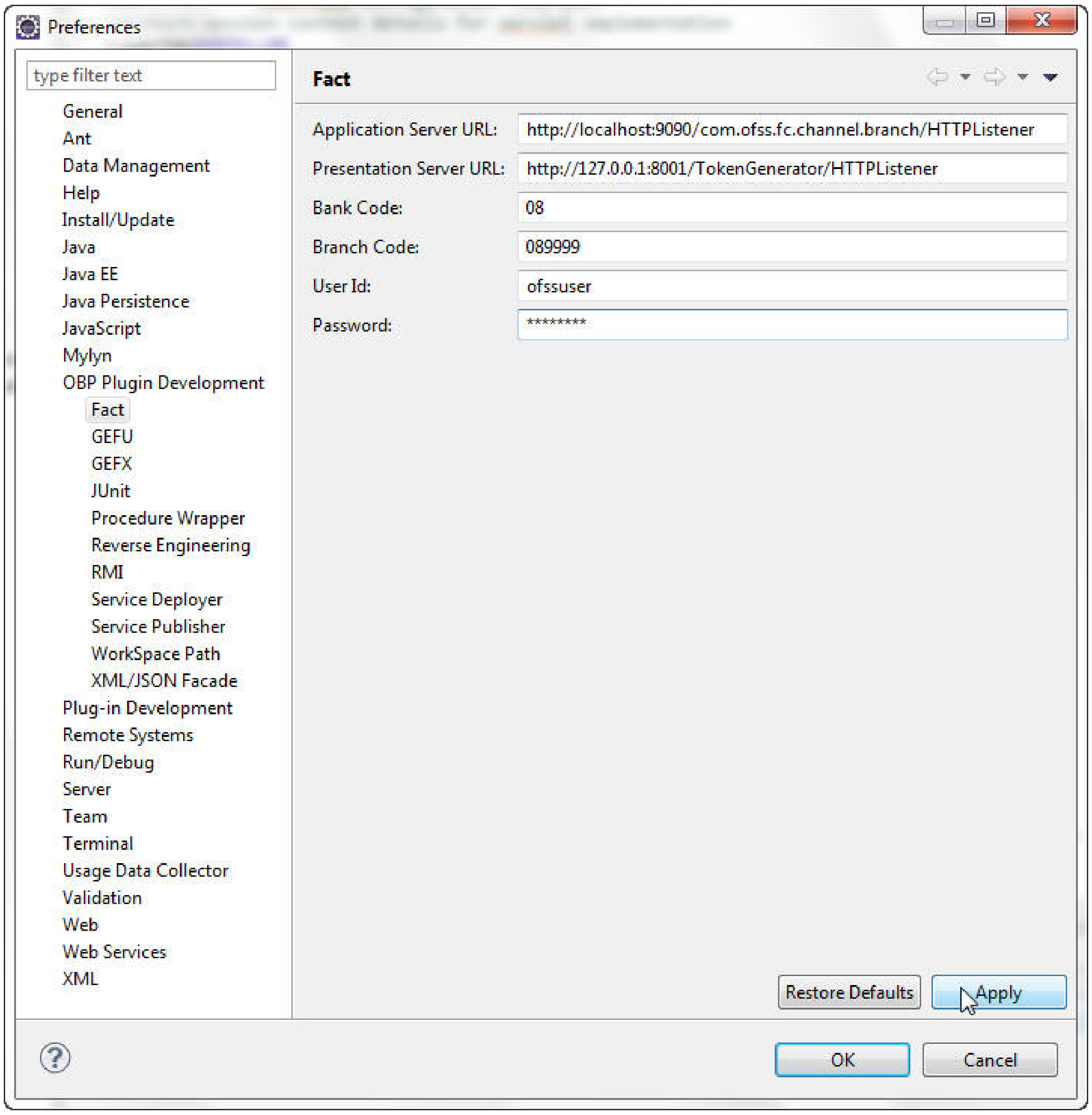
     Example: http: //127.0.0.1:8001/TokenGenerator/HTTPListener

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 15–3 Enter the Preferences Fact values



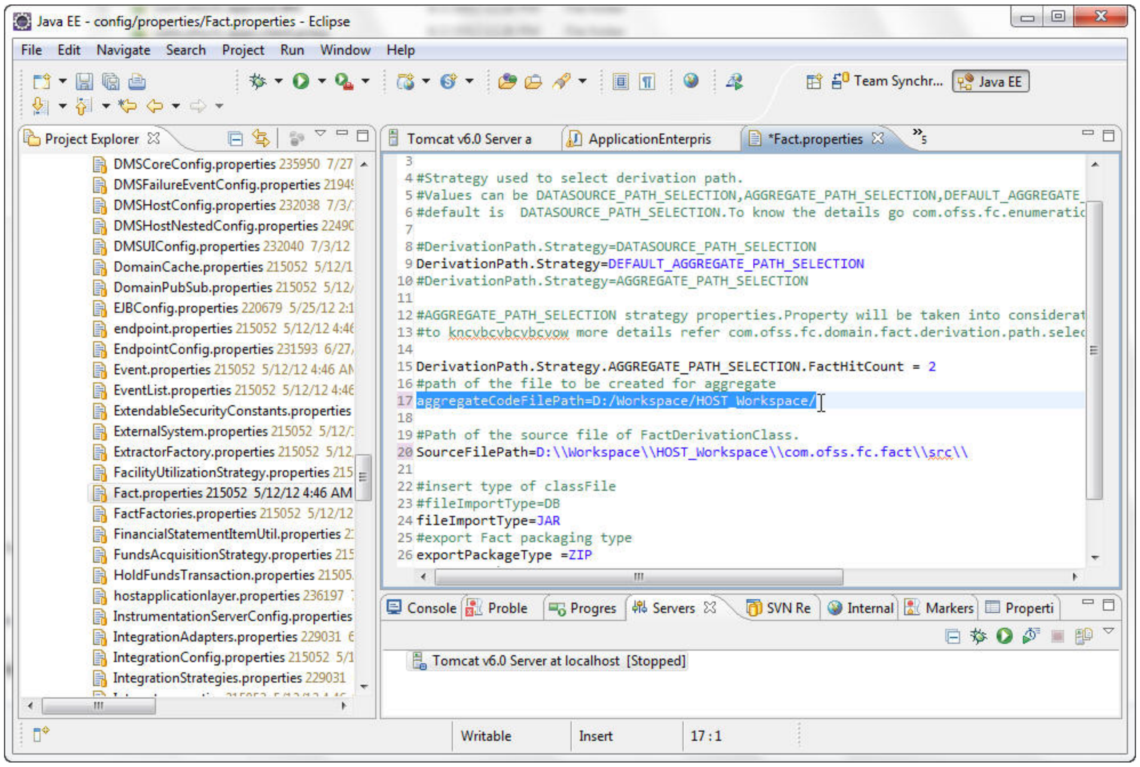
Application Server URL: http://localhost:9090/com.ofss.fc.channel.branch/HTTPListener
Presentation Server URL: http://127.0.0.1:8001/TokenGenerator/HTTPListener
Bank Code: 08
Branch Code: 089999
User Id: ofssuser
Password: ********

4. Now click **Apply**, and then click **Ok**.

5. Open Fact.properties and modify:

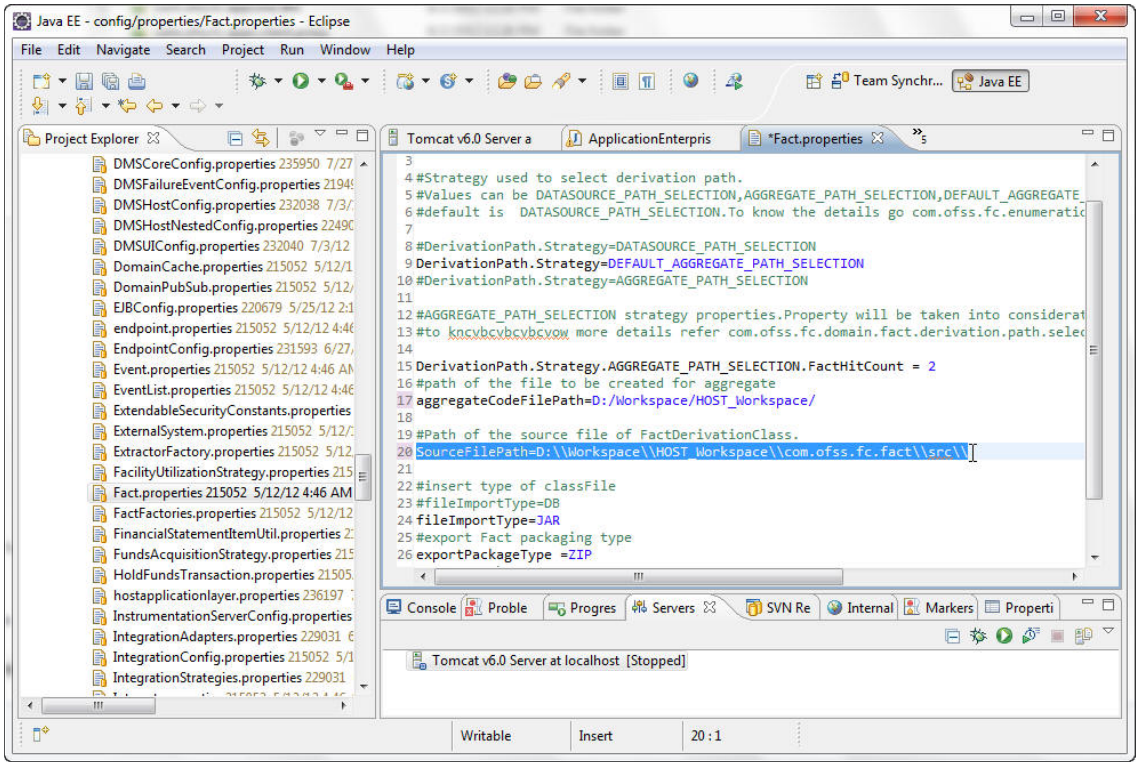
   - **aggregateCodeFilePath**: The location of host workspace.
   - **sourceFilePath**: The location of src directory of com.ofss.fc.fact project.

Figure 15–4 Fact Properties - aggregateCodeFilePath



```

#Strategy used to select derivation path.
#Values can be DATASOURCE_PATH_SELECTION,AGGREGATE_PATH_SELECTION,DEFAULT_AGGREGATE_
#default is  DATASOURCE_PATH_SELECTION.To know the details go com.ofss.fc.enumeratic

#DerivationPath.Strategy=DATASOURCE_PATH_SELECTION
DerivationPath.Strategy=DEFAULT_AGGREGATE_PATH_SELECTION
#DerivationPath.Strategy=AGGREGATE_PATH_SELECTION

#AGGREGATE_PATH_SELECTION strategy properties.Property will be taken into considerat
#to kncvbcvbcvbcvow more details refer com.ofss.fc.domain.fact.derivation.path.selec

DerivationPath.Strategy.AGGREGATE_PATH_SELECTION.FactHitCount = 2
#path of the file to be created for aggregate
aggregateCodeFilePath=D:/Workspace/HOST_Workspace/

#Path of the source file of FactDerivationClass.
SourceFilePath=D:\\Workspace\\HOST_Workspace\\com.ofss.fc.fact\\src\\

#insert type of classFile
#fileImportType=DB
fileImportType=JAR
#export Fact packaging type
exportPackageType =ZIP
```

Figure 15–5 Fact Properties - sourceFilePath

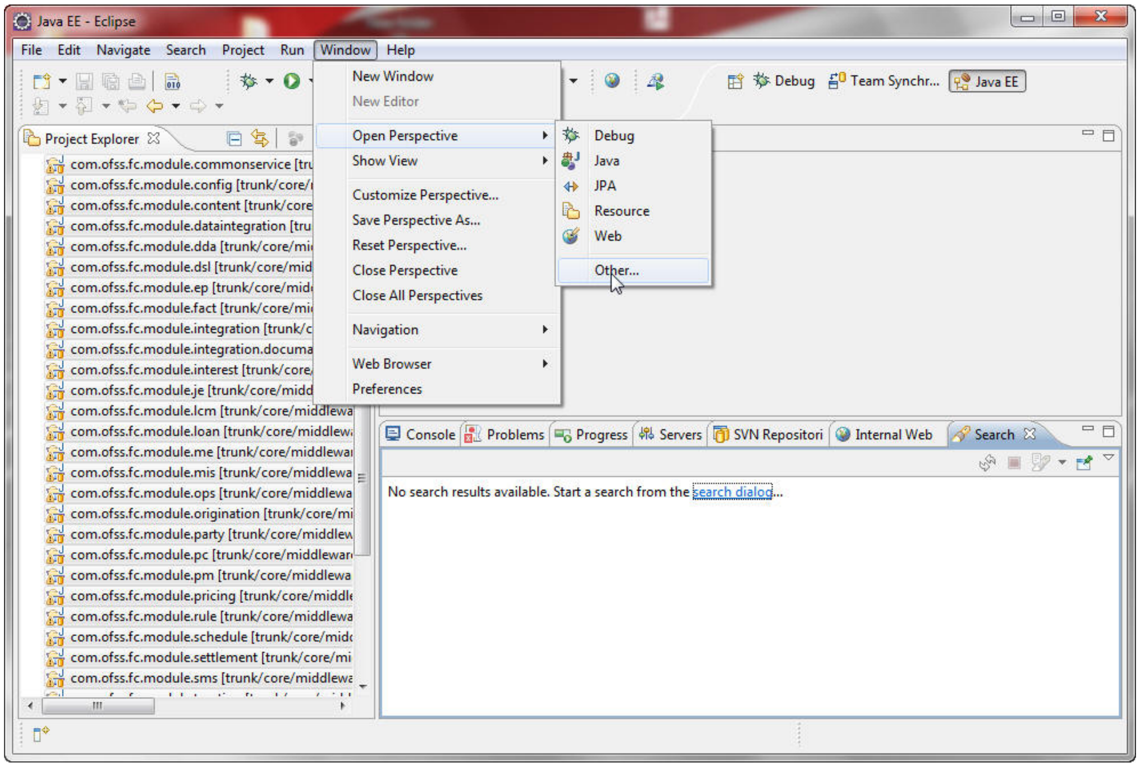


```

#Strategy used to select derivation path.
#Values can be DATASOURCE_PATH_SELECTION,AGGREGATE_PATH_SELECTION,DEFAULT_AGGREGATE_
#default is  DATASOURCE_PATH_SELECTION.To know the details go com.ofss.fc.enumeratic

#DerivationPath.Strategy=DATASOURCE_PATH_SELECTION
DerivationPath.Strategy=DEFAULT_AGGREGATE_PATH_SELECTION
#DerivationPath.Strategy=AGGREGATE_PATH_SELECTION

#AGGREGATE_PATH_SELECTION strategy properties.Property will be taken into considerat
#to kncvbcvbcvbcvow more details refer com.ofss.fc.domain.fact.derivation.path.selec

DerivationPath.Strategy.AGGREGATE_PATH_SELECTION.FactHitCount = 2
#path of the file to be created for aggregate
aggregateCodeFilePath=D:/Workspace/HOST_Workspace/

#Path of the source file of FactDerivationClass.
SourceFilePath=D:\\Workspace\\HOST_Workspace\\com.ofss.fc.fact\\src\\

#insert type of classFile
#fileImportType=DB
fileImportType=JAR
#export Fact packaging type
exportPackageType =ZIP
```

6. Now start the Host server.

7. In eclipse, go to Window -> Open Perspective -> Other.

Figure 15–6 Start Host Server

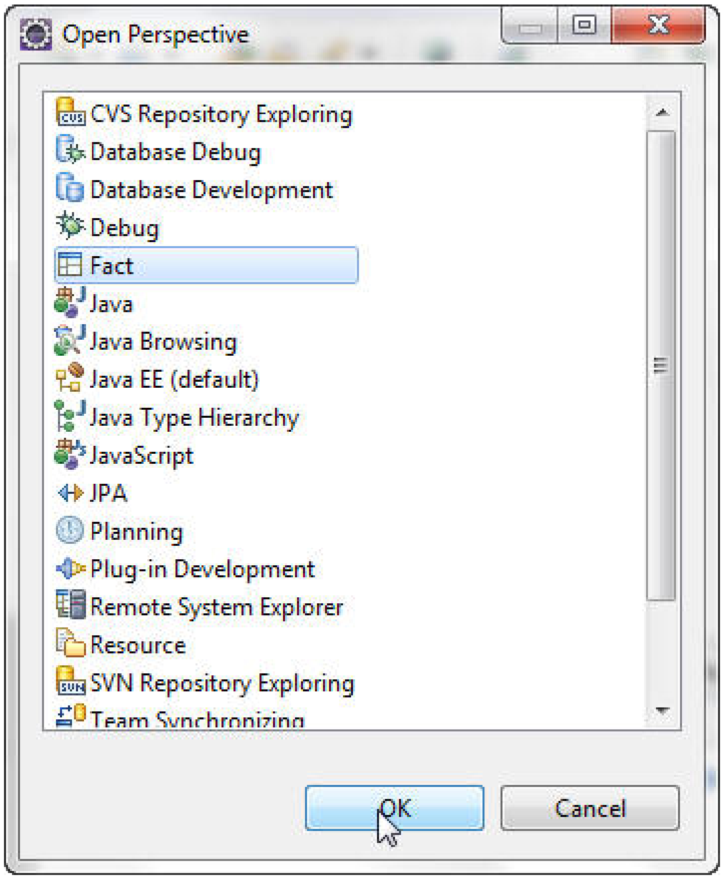


com.ofss.fc.module.commonservice [tru
com.ofss.fc.module.config [trunk/core/
com.ofss.fc.module.content [trunk/core
com.ofss.fc.module.dataintegration [tru
com.ofss.fc.module.dda [trunk/core/mi
com.ofss.fc.module.dsl [trunk/core/mid
com.ofss.fc.module.ep [trunk/core/mid
com.ofss.fc.module.fact [trunk/core/mi
com.ofss.fc.module.integration [trunk/c
com.ofss.fc.module.integration.documa
com.ofss.fc.module.interest [trunk/core/
com.ofss.fc.module.je [trunk/core/midd
com.ofss.fc.module.lcm [trunk/core/middlewa
com.ofss.fc.module.loan [trunk/core/middlew
com.ofss.fc.module.me [trunk/core/middlewa
com.ofss.fc.module.mis [trunk/core/middlewa
com.ofss.fc.module.ops [trunk/core/middlewa
com.ofss.fc.module.origination [trunk/core/mi
com.ofss.fc.module.party [trunk/core/middlew
com.ofss.fc.module.pc [trunk/core/middlewar
com.ofss.fc.module.pm [trunk/core/middlewa
com.ofss.fc.module.pricing [trunk/core/middle
com.ofss.fc.module.rule [trunk/core/middlewa
com.ofss.fc.module.schedule [trunk/core/mid
com.ofss.fc.module.settlement [trunk/core/mi
com.ofss.fc.module.sms [trunk/core/middlewa

No search results available. Start a search from the search dialog...

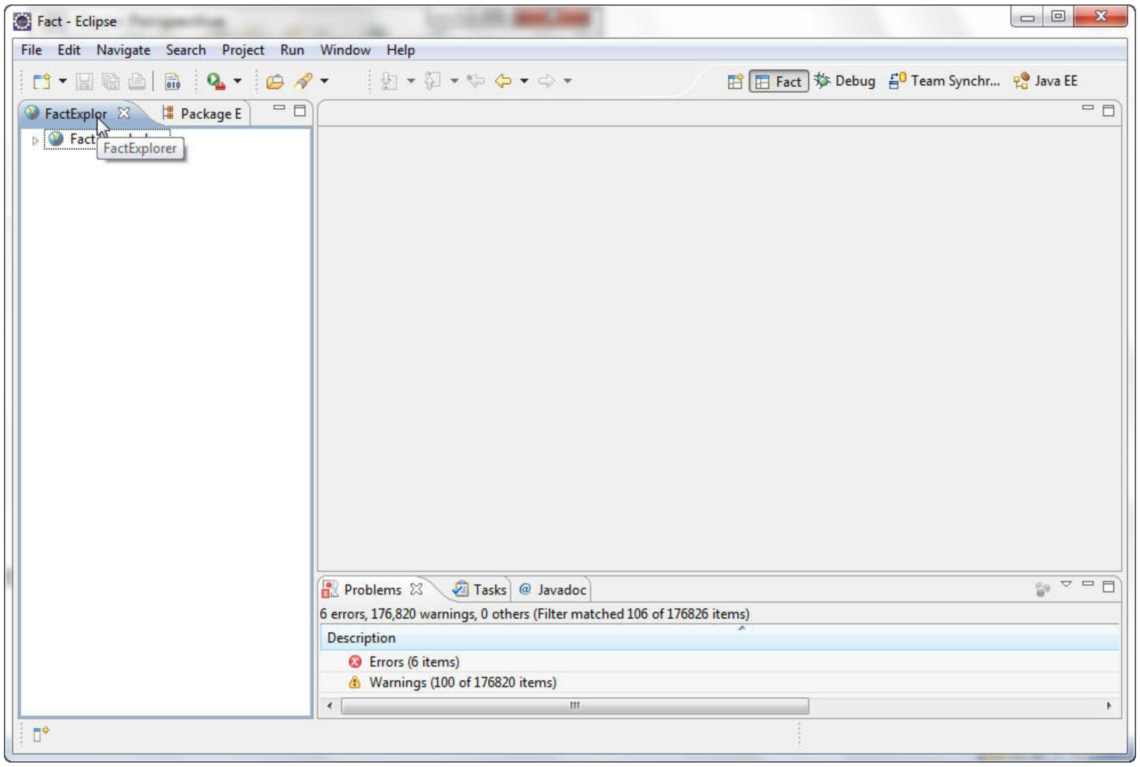
8. Now in Open Perspective window select **Fact**.
9. Click **Ok**.

Figure 15–7 Select Open Perspective value



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

Figure 15–8 Fact Explorer

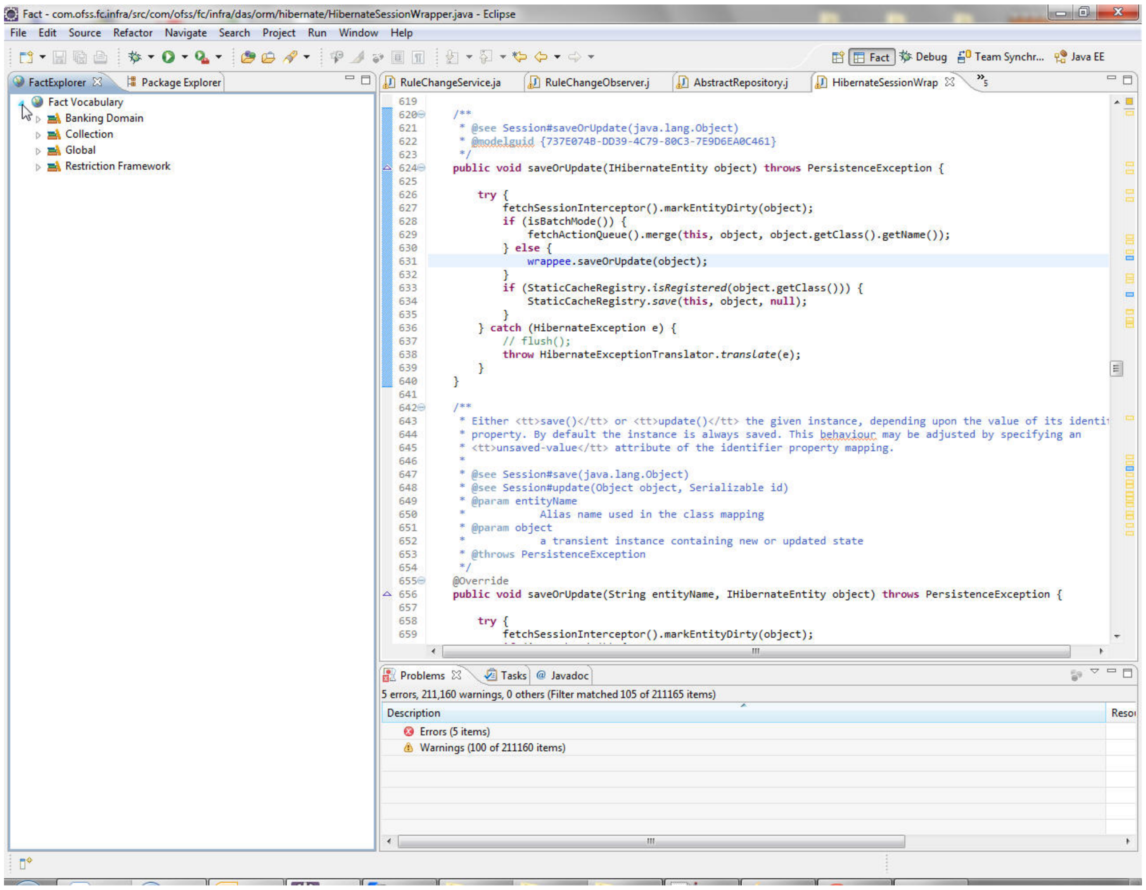


6 errors, 176,820 warnings, 0 others (Filter matched 106 of 176826 items)

Description
Errors (6 items)
Warnings (100 of 176820 items)

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

Figure 15–9 Fact Vocabulary


Fact - com.ofss.fc.infra/src/com/ofss/fc/infra/das/orm/hibernate/HibernateSessionWrapper.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Fact Debug Team Synchr... Java EE
FactExplorer
Package Explorer
Fact Vocabulary
Banking Domain
Collection
Global
Restriction Framework
RuleChangeService.ja
RuleChangeObserver.j
AbstractRepository.j
HibernateSessionWrap
Problems
Tasks
Javadoc

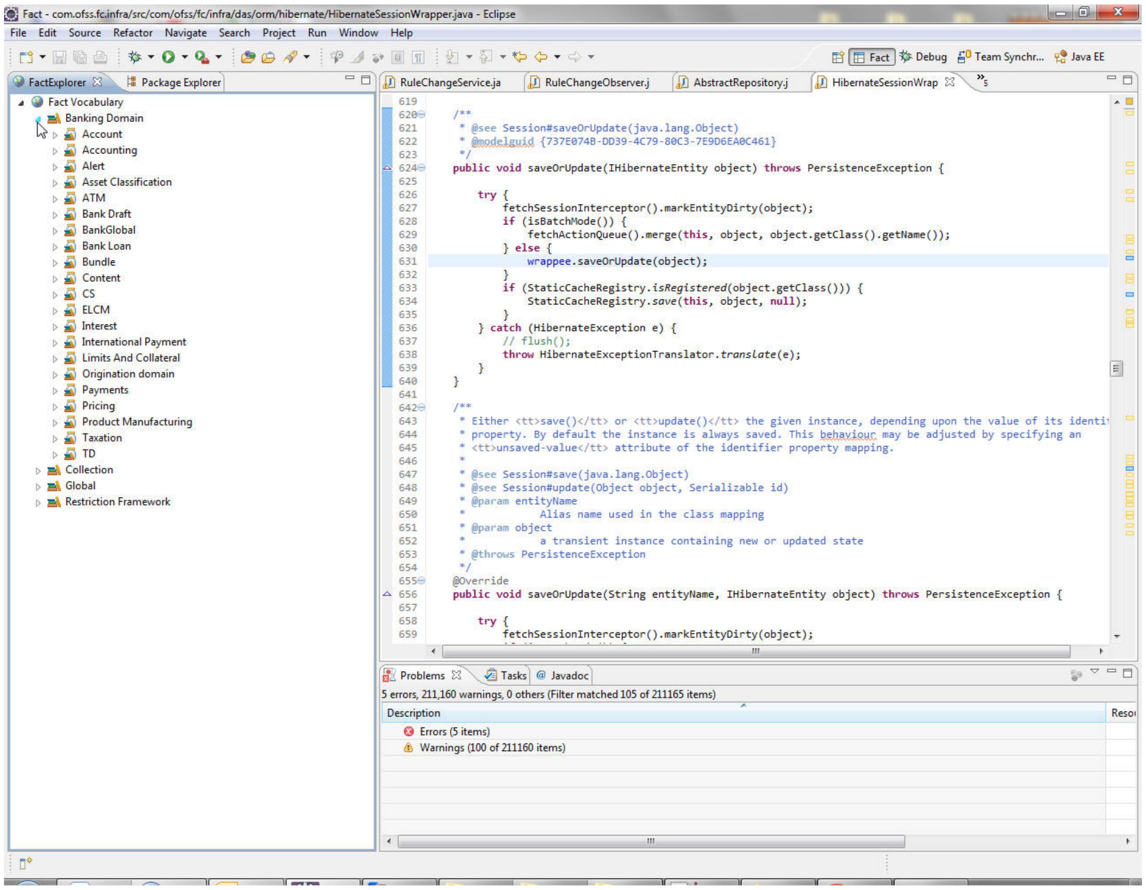
```
    /**
     * @see Session#saveOrUpdate(java.lang.Object)
     * @modelguid {737E074B-DD39-4C79-80C3-7E9D6EA0C461}
     */
    public void saveOrUpdate(IHibernateEntity object) throws PersistenceException {

        try {
            fetchSessionInterceptor().markEntityDirty(object);
            if (isBatchMode()) {
                fetchActionQueue().merge(this, object, object.getClass().getName());
            } else {
                wrappee.saveOrUpdate(object);
            }
            if (StaticCacheRegistry.isRegistered(object.getClass())) {
                StaticCacheRegistry.save(this, object, null);
            }
        } catch (HibernateException e) {
            // flush();
            throw HibernateExceptionTranslator.translate(e);
        }
    }

    /**
     * Either <tt>save()</tt> or <tt>update()</tt> the given instance, depending upon the value of its identi
     * property. By default the instance is always saved. This behaviour may be adjusted by specifying an
     * <tt>unsaved-value</tt> attribute of the identifier property mapping.
     *
     * @see Session#save(java.lang.Object)
     * @see Session#update(Object object, Serializable id)
     * @param entityName
     *            Alias name used in the class mapping
     * @param object
     *            a transient instance containing new or updated state
     * @throws PersistenceException
     */
    @Override
    public void saveOrUpdate(String entityName, IHibernateEntity object) throws PersistenceException {

        try {
            fetchSessionInterceptor().markEntityDirty(object);
```

5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)

Description	Resou
Errors (5 items)	
Warnings (100 of 211160 items)	

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

Figure 15–10 Domain Category



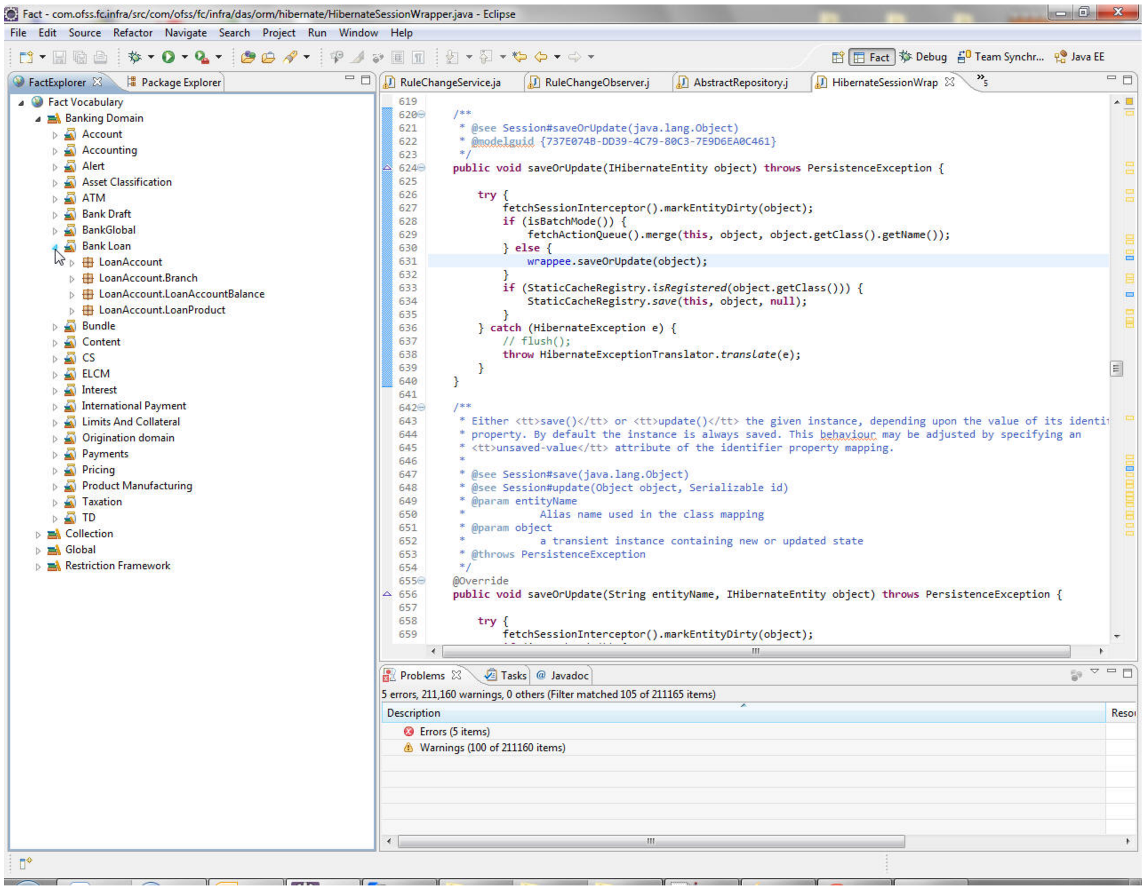
- Fact Vocabulary
  - Banking Domain
    - Account
    - Accounting
    - Alert
    - Asset Classification
    - ATM
    - Bank Draft
    - BankGlobal
    - Bank Loan
    - Bundle
    - Content
    - CS
    - ELCM
    - Interest
    - International Payment
    - Limits And Collateral
    - Origination domain
    - Payments
    - Pricing
    - Product Manufacturing
    - Taxation
    - TD
  - Collection
  - Global
  - Restriction Framework

```java

    /**
     * @see Session#saveOrUpdate(java.lang.Object)
     * @modelguid {737E074B-DD39-4C79-80C3-7E9D6EA0C461}
     */
    public void saveOrUpdate(IHibernateEntity object) throws PersistenceException {

        try {
            fetchSessionInterceptor().markEntityDirty(object);
            if (isBatchMode()) {
                fetchActionQueue().merge(this, object, object.getClass().getName());
            } else {
                wrappee.saveOrUpdate(object);
            }
            if (StaticCacheRegistry.isRegistered(object.getClass())) {
                StaticCacheRegistry.save(this, object, null);
            }
        } catch (HibernateException e) {
            // flush();
            throw HibernateExceptionTranslator.translate(e);
        }
    }

    /**
     * Either <tt>save()</tt> or <tt>update()</tt> the given instance, depending upon the value of its identi
     * property. By default the instance is always saved. This behaviour may be adjusted by specifying an
     * <tt>unsaved-value</tt> attribute of the identifier property mapping.
     *
     * @see Session#save(java.lang.Object)
     * @see Session#update(Object object, Serializable id)
     * @param entityName
     *            Alias name used in the class mapping
     * @param object
     *            a transient instance containing new or updated state
     * @throws PersistenceException
     */
    @Override
    public void saveOrUpdate(String entityName, IHibernateEntity object) throws PersistenceException {

        try {
            fetchSessionInterceptor().markEntityDirty(object);
```

```
5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)
Errors (5 items)
Warnings (100 of 211160 items)
```

Each Domain category contain its **Fact Groups**

Figure 15–11 Fact Groups



Fact - com.ofss.fc.infra/src/com/ofss/fc/infra/das/orm/hibernate/HibernateSessionWrapper.java - Eclipse

- Fact Vocabulary
  - Banking Domain
    - Account
    - Accounting
    - Alert
    - Asset Classification
    - ATM
    - Bank Draft
    - BankGlobal
    - Bank Loan
      - LoanAccount
      - LoanAccount.Branch
      - LoanAccount.LoanAccountBalance
      - LoanAccount.LoanProduct
    - Bundle
    - Content
    - CS
    - ELCM
    - Interest
    - International Payment
    - Limits And Collateral
    - Origination domain
    - Payments
    - Pricing
    - Product Manufacturing
    - Taxation
    - TD
  - Collection
  - Global
  - Restriction Framework

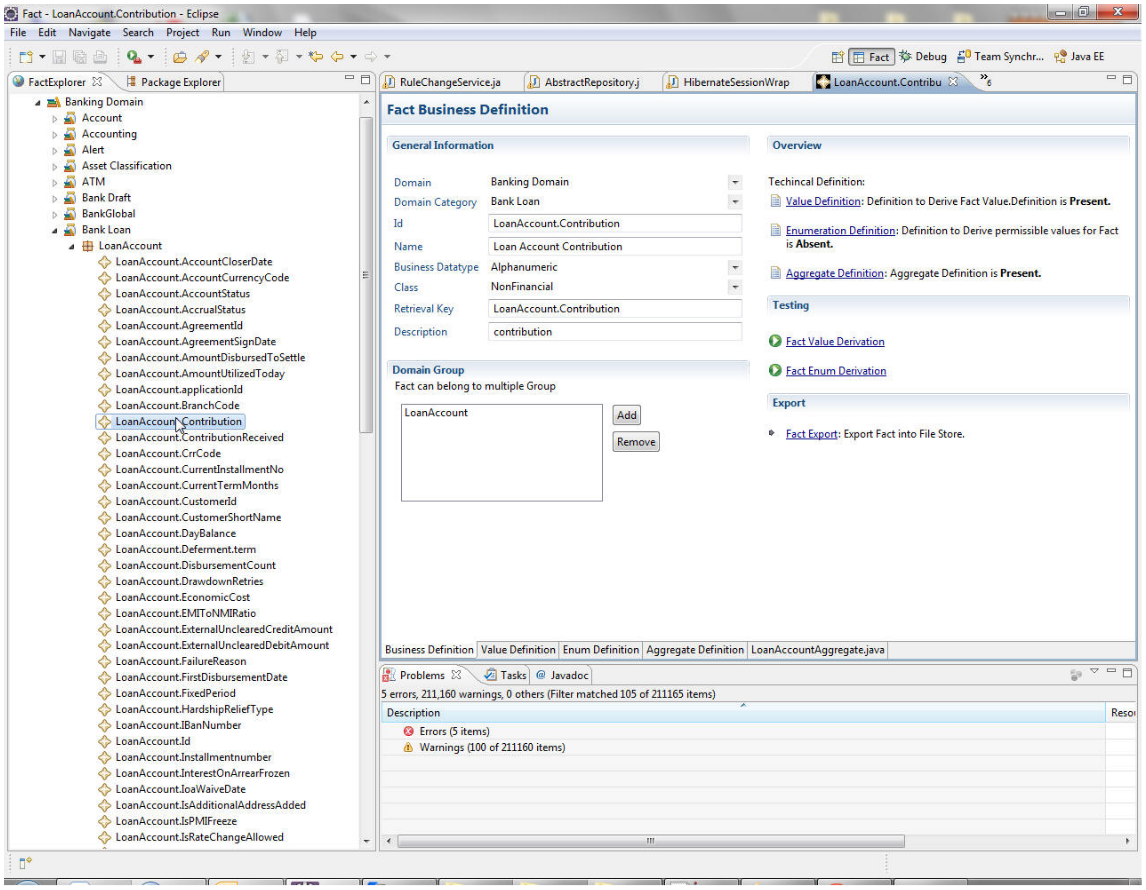
```

    /**
     * @see Session#saveOrUpdate(java.lang.Object)
     * @modelguid {737E074B-DD39-4C79-80C3-7E9D6EA0C461}
     */
    public void saveOrUpdate(IHibernateEntity object) throws PersistenceException {

        try {
            fetchSessionInterceptor().markEntityDirty(object);
            if (isBatchMode()) {
                fetchActionQueue().merge(this, object, object.getClass().getName());
            } else {
                wrappee.saveOrUpdate(object);
            }
            if (StaticCacheRegistry.isRegistered(object.getClass())) {
                StaticCacheRegistry.save(this, object, null);
            }
        } catch (HibernateException e) {
            // flush();
            throw HibernateExceptionTranslator.translate(e);
        }
    }

    /**
     * Either <tt>save()</tt> or <tt>update()</tt> the given instance, depending upon the value of its identi
     * property. By default the instance is always saved. This behaviour may be adjusted by specifying an
     * <tt>unsaved-value</tt> attribute of the identifier property mapping.
     *
     * @see Session#save(java.lang.Object)
     * @see Session#update(Object object, Serializable id)
     * @param entityName
     *            Alias name used in the class mapping
     * @param object
     *            a transient instance containing new or updated state
     * @throws PersistenceException
     */
    @Override
    public void saveOrUpdate(String entityName, IHibernateEntity object) throws PersistenceException {

        try {
            fetchSessionInterceptor().markEntityDirty(object);
```

5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)

Description	Reso
Errors (5 items)	
Warnings (100 of 211160 items)	

Each Fact Groups contains its **Facts**.

Figure 15–12 Facts



Banking Domain
- Account
- Accounting
- Alert
- Asset Classification
- ATM
- Bank Draft
- BankGlobal
- Bank Loan
  - LoanAccount
    - LoanAccount.AccountCloserDate
    - LoanAccount.AccountCurrencyCode
    - LoanAccount.AccountStatus
    - LoanAccount.AccrualStatus
    - LoanAccount.AgreementId
    - LoanAccount.AgreementSignDate
    - LoanAccount.AmountDisbursedToSettle
    - LoanAccount.AmountUtilizedToday
    - LoanAccount.applicationId
    - LoanAccount.BranchCode
    - LoanAccount.Contribution
    - LoanAccount.ContributionReceived
    - LoanAccount.CrrCode
    - LoanAccount.CurrentInstallmentNo
    - LoanAccount.CurrentTermMonths
    - LoanAccount.CustomerId
    - LoanAccount.CustomerShortName
    - LoanAccount.DayBalance
    - LoanAccount.Deferment.term
    - LoanAccount.DisbursementCount
    - LoanAccount.DrawdownRetries
    - LoanAccount.EconomicCost
    - LoanAccount.EMIToNMIRatio
    - LoanAccount.ExternalUnclearedCreditAmount
    - LoanAccount.ExternalUnclearedDebitAmount
    - LoanAccount.FailureReason
    - LoanAccount.FirstDisbursementDate
    - LoanAccount.FixedPeriod
    - LoanAccount.HardshipReliefType
    - LoanAccount.IBanNumber
    - LoanAccount.Id
    - LoanAccount.Installmentnumber
    - LoanAccount.InterestOnArrearFrozen
    - LoanAccount.IoaWaiveDate
    - LoanAccount.IsAdditionalAddressAdded
    - LoanAccount.IsPMIFreeze
    - LoanAccount.IsRateChangeAllowed

Fact Business Definition

General Information

Domain	Banking Domain
Domain Category	Bank Loan
Id	LoanAccount.Contribution
Name	Loan Account Contribution
Business Datatype	Alphanumeric
Class	NonFinancial
Retrieval Key	LoanAccount.Contribution
Description	contribution

Domain Group

Fact can belong to multiple Group

LoanAccount

Overview

Techincal Definition:

Value Definition: Definition to Derive Fact Value.Definition is **Present.**

Enumeration Definition: Definition to Derive permissible values for Fact is **Absent.**

Aggregate Definition: Aggregate Definition is **Present.**

Testing

Fact Value Derivation

Fact Enum Derivation

Export

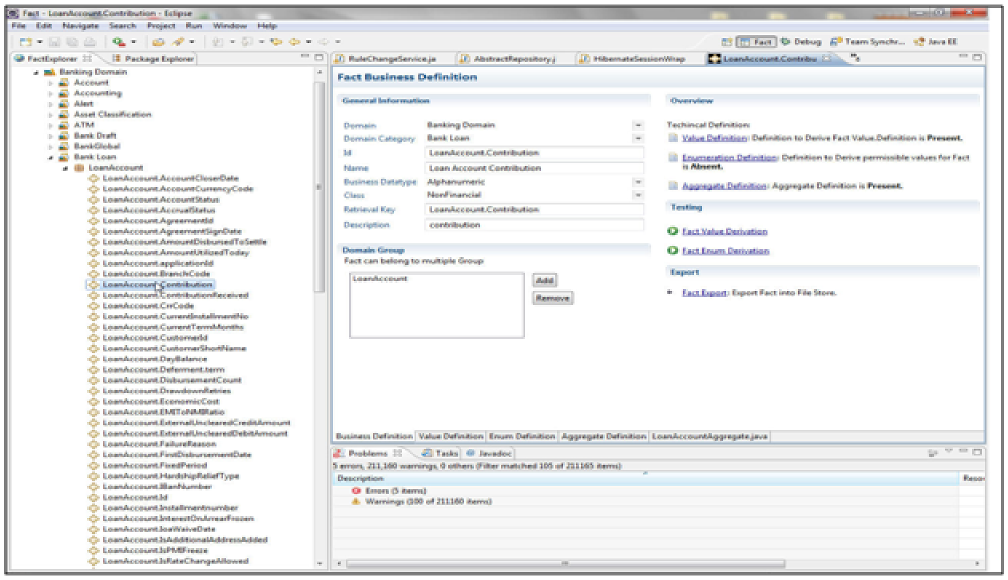
Fact Export: Export Fact into File Store.

5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)

Description	Reso
Errors (5 items)	
Warnings (100 of 211160 items)	

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 15–13 Business Definition Tab



Fact Business Definition

General Information

Domain	Banking Domain
Domain Category	Bank Loan
Id	LoanAccount.Contribution
Name	Loan Account Contribution
Business Datatype	Alphanumeric
Class	NonFinancial
Retrieval Key	LoanAccount.Contribution
Description	contribution

Domain Group

Fact can belong to multiple Group

LoanAccount

Overview

Techincal Definition:

Value Definition: Definition to Derive Fact Value.Definition is **Present.**

Enumeration Definition: Definition to Derive permissible values for Fact is **Absent.**

Aggregate Definition: Aggregate Definition is **Present.**

Testing

Fact Value Derivation

Fact Enum Derivation

Export

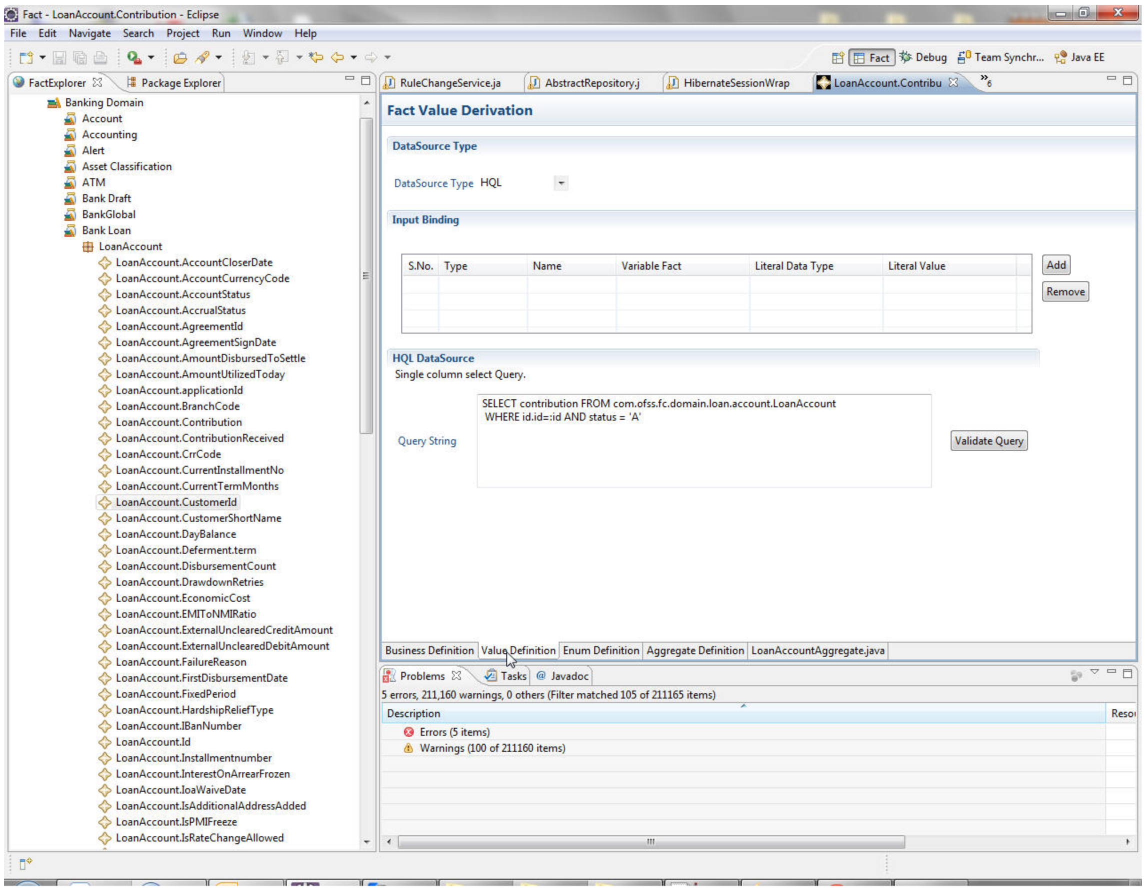
Fact Export: Export Fact into File Store.

5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)

Errors (5 items)

Warnings (100 of 211160 items)

Figure 15–14 Value Definition Tab



Fact - LoanAccount.Contribution - Eclipse

Banking Domain
- Account
- Accounting
- Alert
- Asset Classification
- ATM
- Bank Draft
- BankGlobal
- Bank Loan
  - LoanAccount
    - LoanAccount.AccountCloserDate
    - LoanAccount.AccountCurrencyCode
    - LoanAccount.AccountStatus
    - LoanAccount.AccrualStatus
    - LoanAccount.AgreementId
    - LoanAccount.AgreementSignDate
    - LoanAccount.AmountDisbursedToSettle
    - LoanAccount.AmountUtilizedToday
    - LoanAccount.applicationId
    - LoanAccount.BranchCode
    - LoanAccount.Contribution
    - LoanAccount.ContributionReceived
    - LoanAccount.CrrCode
    - LoanAccount.CurrentInstallmentNo
    - LoanAccount.CurrentTermMonths
    - LoanAccount.CustomerId
    - LoanAccount.CustomerShortName
    - LoanAccount.DayBalance
    - LoanAccount.Deferment.term
    - LoanAccount.DisbursementCount
    - LoanAccount.DrawdownRetries
    - LoanAccount.EconomicCost
    - LoanAccount.EMIToNMIRatio
    - LoanAccount.ExternalUnclearedCreditAmount
    - LoanAccount.ExternalUnclearedDebitAmount
    - LoanAccount.FailureReason
    - LoanAccount.FirstDisbursementDate
    - LoanAccount.FixedPeriod
    - LoanAccount.HardshipReliefType
    - LoanAccount.IBanNumber
    - LoanAccount.Id
    - LoanAccount.Installmentnumber
    - LoanAccount.InterestOnArrearFrozen
    - LoanAccount.IoaWaiveDate
    - LoanAccount.IsAdditionalAddressAdded
    - LoanAccount.IsPMIFreeze
    - LoanAccount.IsRateChangeAllowed

Fact Value Derivation

DataSource Type

DataSource Type HQL

Input Binding

S.No.	Type	Name	Variable Fact	Literal Data Type	Literal Value

HQL DataSource

Single column select Query.

Query String

```
SELECT contribution FROM com.ofss.fc.domain.loan.account.LoanAccount
 WHERE id.id=:id AND status = 'A'
```

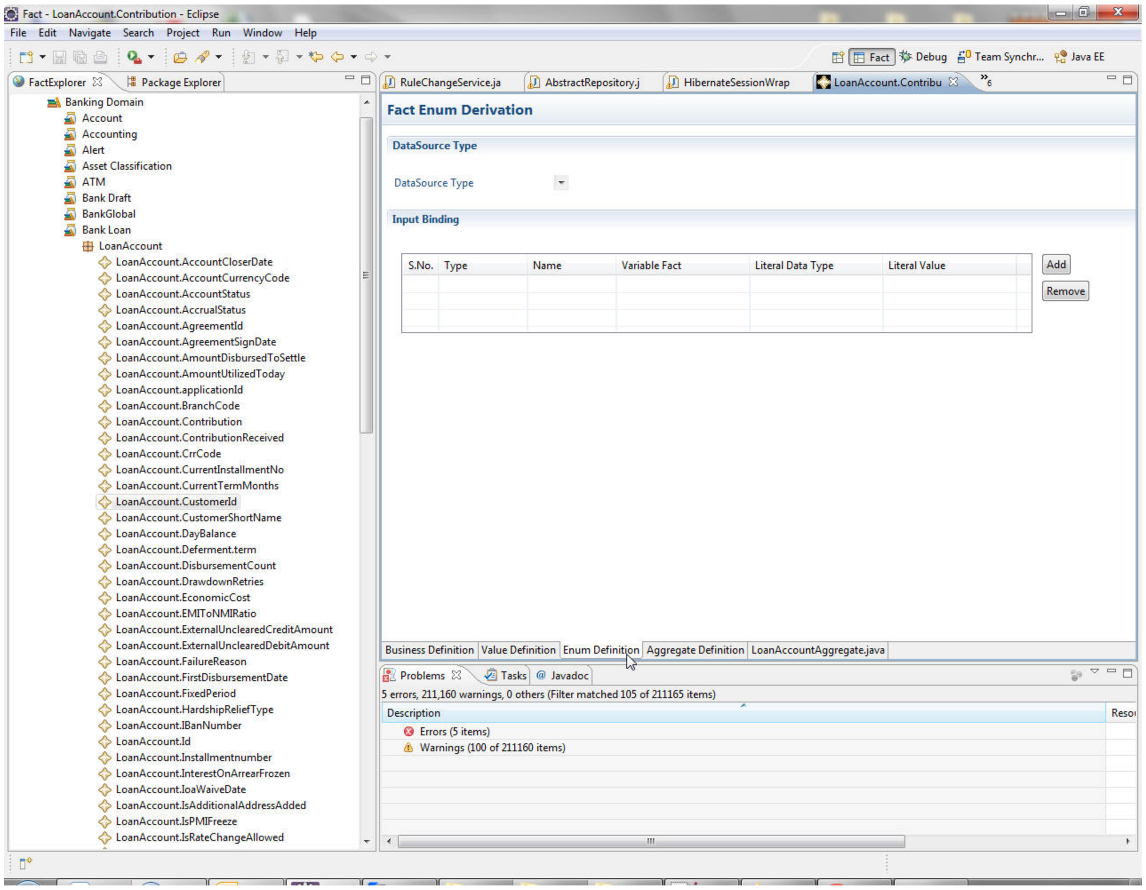
5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)

Errors (5 items)

Warnings (100 of 211160 items)

File Edit Navigate Search Project Run Window Help
Fact Debug Team Synchr... Java EE
FactExplorer Package Explorer
RuleChangeService.ja AbstractRepository.j HibernateSessionWrap LoanAccount.Contribu
Add
Remove
Validate Query
Business Definition Value Definition Enum Definition Aggregate Definition LoanAccountAggregate.java
Problems Tasks Javadoc
Description

Figure 15–15 Enum Definition Tab


Fact - LoanAccount.Contribution - Eclipse
File Edit Navigate Search Project Run Window Help
Fact Debug Team Synchr... Java EE
FactExplorer
Package Explorer
RuleChangeService.ja
AbstractRepository.j
HibernateSessionWrap
LoanAccount.Contribu

Banking Domain
- Account
- Accounting
- Alert
- Asset Classification
- ATM
- Bank Draft
- BankGlobal
- Bank Loan
  - LoanAccount
    - LoanAccount.AccountCloserDate
    - LoanAccount.AccountCurrencyCode
    - LoanAccount.AccountStatus
    - LoanAccount.AccrualStatus
    - LoanAccount.AgreementId
    - LoanAccount.AgreementSignDate
    - LoanAccount.AmountDisbursedToSettle
    - LoanAccount.AmountUtilizedToday
    - LoanAccount.applicationId
    - LoanAccount.BranchCode
    - LoanAccount.Contribution
    - LoanAccount.ContributionReceived
    - LoanAccount.CrrCode
    - LoanAccount.CurrentInstallmentNo
    - LoanAccount.CurrentTermMonths
    - LoanAccount.CustomerId
    - LoanAccount.CustomerShortName
    - LoanAccount.DayBalance
    - LoanAccount.Deferment.term
    - LoanAccount.DisbursementCount
    - LoanAccount.DrawdownRetries
    - LoanAccount.EconomicCost
    - LoanAccount.EMIToNMIRatio
    - LoanAccount.ExternalUnclearedCreditAmount
    - LoanAccount.ExternalUnclearedDebitAmount
    - LoanAccount.FailureReason
    - LoanAccount.FirstDisbursementDate
    - LoanAccount.FixedPeriod
    - LoanAccount.HardshipReliefType
    - LoanAccount.IBanNumber
    - LoanAccount.Id
    - LoanAccount.Installmentnumber
    - LoanAccount.InterestOnArrearFrozen
    - LoanAccount.IoaWaiveDate
    - LoanAccount.IsAdditionalAddressAdded
    - LoanAccount.IsPMIFreeze
    - LoanAccount.IsRateChangeAllowed

Fact Enum Derivation

DataSource Type

DataSource Type

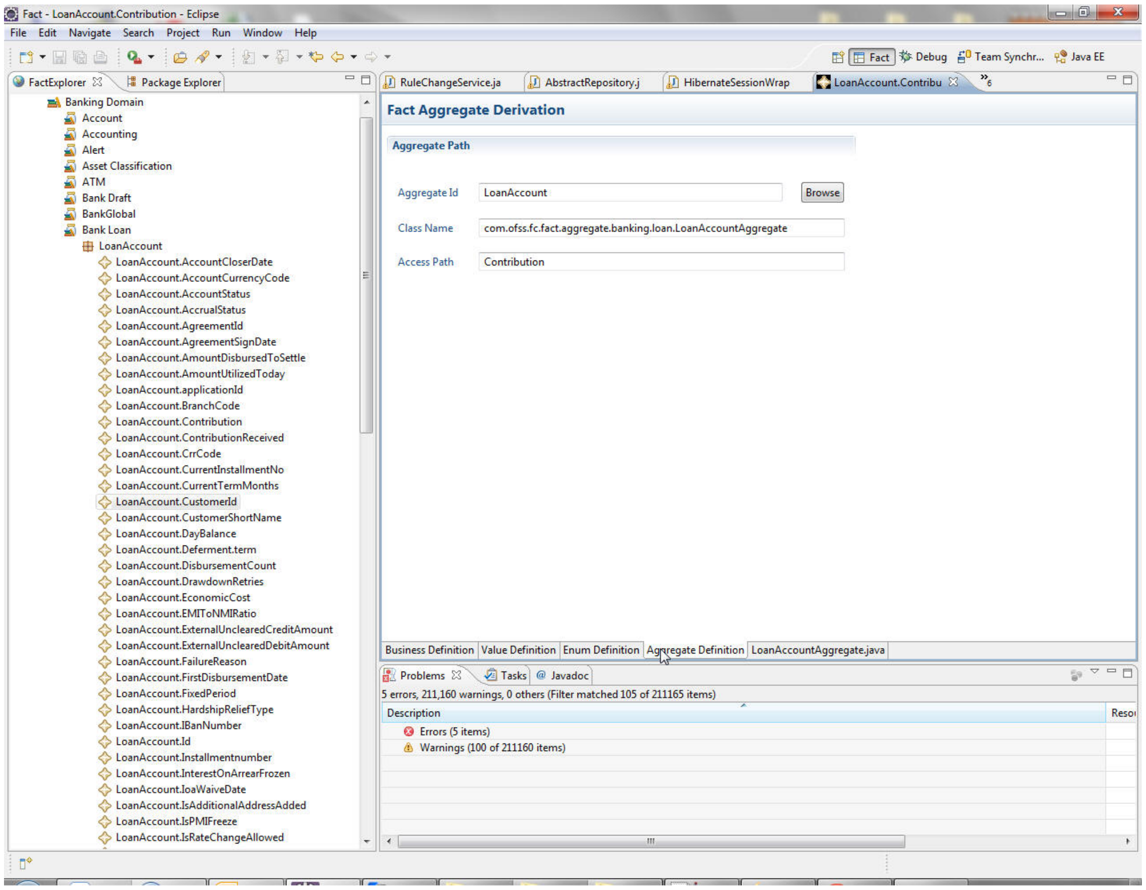
Input Binding

S.No.	Type	Name	Variable Fact	Literal Data Type	Literal Value

5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)

Description	Resou
Errors (5 items)	
Warnings (100 of 211160 items)	

Figure 15–16 Aggregrate Definition Tab


Fact - LoanAccount.Contribution - Eclipse
File Edit Navigate Search Project Run Window Help
Fact Debug Team Synchr... Java EE
FactExplorer
Package Explorer
RuleChangeService.ja
AbstractRepository.j
HibernateSessionWrap
LoanAccount.Contribu
Banking Domain
Account
Accounting
Alert
Asset Classification
ATM
Bank Draft
BankGlobal
Bank Loan
LoanAccount
LoanAccount.AccountCloserDate
LoanAccount.AccountCurrencyCode
LoanAccount.AccountStatus
LoanAccount.AccrualStatus
LoanAccount.AgreementId
LoanAccount.AgreementSignDate
LoanAccount.AmountDisbursedToSettle
LoanAccount.AmountUtilizedToday
LoanAccount.applicationId
LoanAccount.BranchCode
LoanAccount.Contribution
LoanAccount.ContributionReceived
LoanAccount.CrrCode
LoanAccount.CurrentInstallmentNo
LoanAccount.CurrentTermMonths
LoanAccount.CustomerId
LoanAccount.CustomerShortName
LoanAccount.DayBalance
LoanAccount.Deferment.term
LoanAccount.DisbursementCount
LoanAccount.DrawdownRetries
LoanAccount.EconomicCost
LoanAccount.EMIToNMIRatio
LoanAccount.ExternalUnclearedCreditAmount
LoanAccount.ExternalUnclearedDebitAmount
LoanAccount.FailureReason
LoanAccount.FirstDisbursementDate
LoanAccount.FixedPeriod
LoanAccount.HardshipReliefType
LoanAccount.IBanNumber
LoanAccount.Id
LoanAccount.Installmentnumber
LoanAccount.InterestOnArrearFrozen
LoanAccount.IoaWaiveDate
LoanAccount.IsAdditionalAddressAdded
LoanAccount.IsPMIFreeze
LoanAccount.IsRateChangeAllowed
Browse
Business Definition
Value Definition
Enum Definition
Aggregate Definition
LoanAccountAggregate.java
Problems
Tasks
Javadoc
Description
Resou

Fact Aggregate Derivation

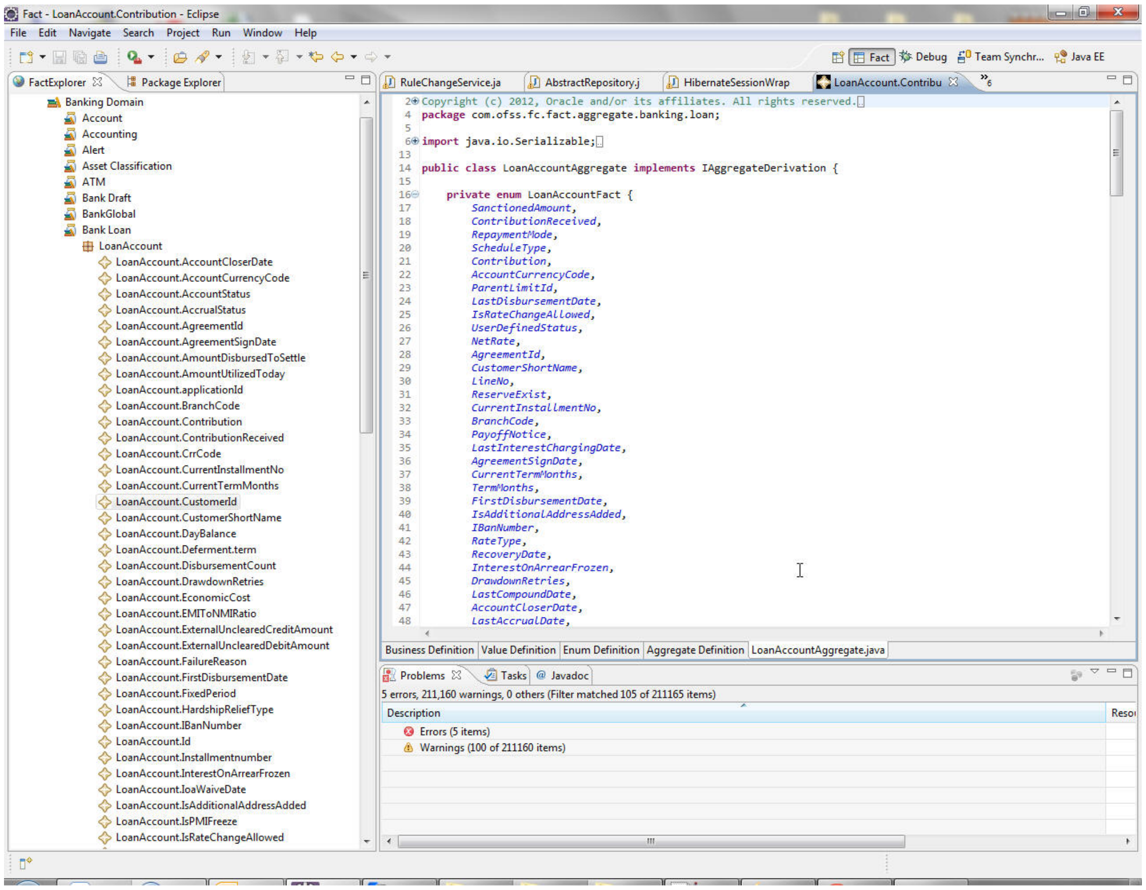
Aggregate Path

Aggregate Id	LoanAccount
Class Name	com.ofss.fc.fact.aggregate.banking.loan.LoanAccountAggregate
Access Path	Contribution

5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)

Errors (5 items)
Warnings (100 of 211160 items)

Figure 15–17 Aggregate File Tab

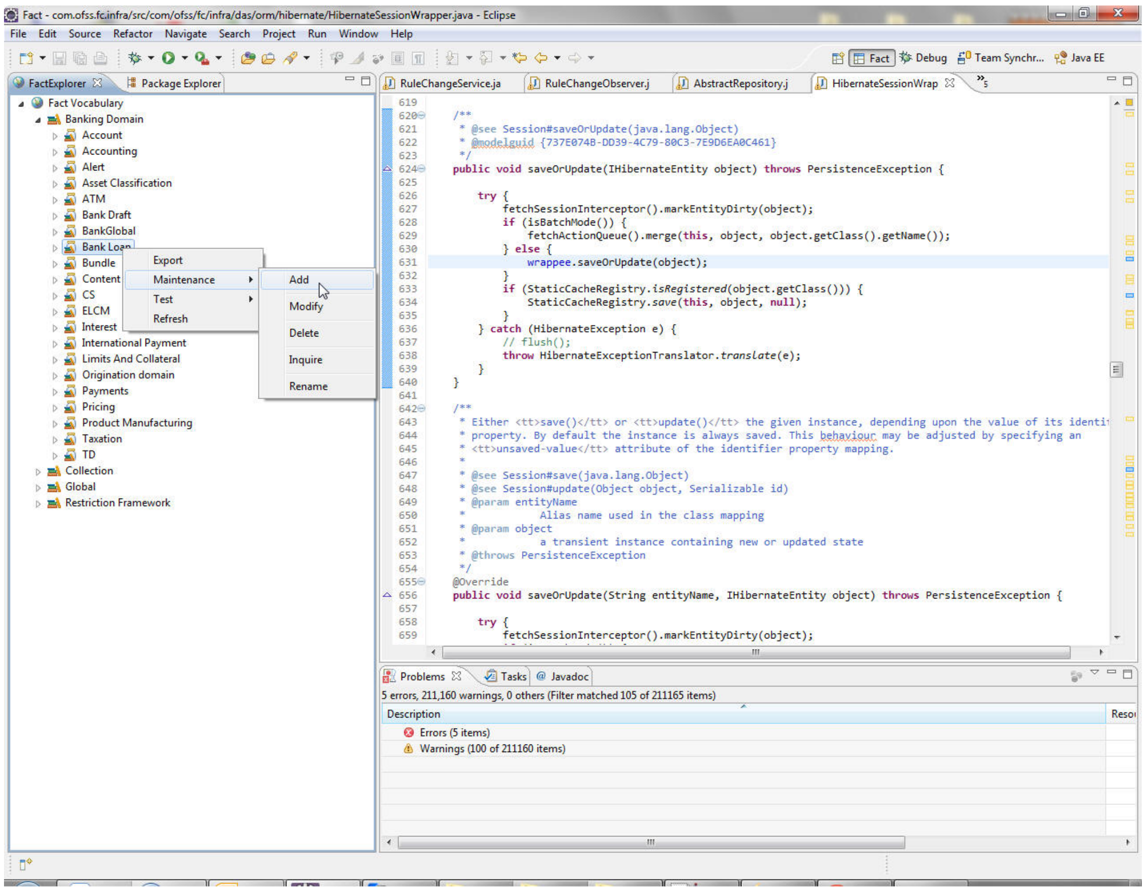

Fact - LoanAccount.Contribution - Eclipse
File Edit Navigate Search Project Run Window Help
Fact Debug Team Synchr... Java EE
FactExplorer
Package Explorer
RuleChangeService.ja
AbstractRepository.j
HibernateSessionWrap
LoanAccount.Contribu
Banking Domain
Account
Accounting
Alert
Asset Classification
ATM
Bank Draft
BankGlobal
Bank Loan
LoanAccount
LoanAccount.AccountCloserDate
LoanAccount.AccountCurrencyCode
LoanAccount.AccountStatus
LoanAccount.AccrualStatus
LoanAccount.AgreementId
LoanAccount.AgreementSignDate
LoanAccount.AmountDisbursedToSettle
LoanAccount.AmountUtilizedToday
LoanAccount.applicationId
LoanAccount.BranchCode
LoanAccount.Contribution
LoanAccount.ContributionReceived
LoanAccount.CrrCode
LoanAccount.CurrentInstallmentNo
LoanAccount.CurrentTermMonths
LoanAccount.CustomerId
LoanAccount.CustomerShortName
LoanAccount.DayBalance
LoanAccount.Deferment.term
LoanAccount.DisbursementCount
LoanAccount.DrawdownRetries
LoanAccount.EconomicCost
LoanAccount.EMIToNMIRatio
LoanAccount.ExternalUnclearedCreditAmount
LoanAccount.ExternalUnclearedDebitAmount
LoanAccount.FailureReason
LoanAccount.FirstDisbursementDate
LoanAccount.FixedPeriod
LoanAccount.HardshipReliefType
LoanAccount.IBanNumber
LoanAccount.Id
LoanAccount.Installmentnumber
LoanAccount.InterestOnArrearFrozen
LoanAccount.IoaWaiveDate
LoanAccount.IsAdditionalAddressAdded
LoanAccount.IsPMIFreeze
LoanAccount.IsRateChangeAllowed
Business Definition
Value Definition
Enum Definition
Aggregate Definition
LoanAccountAggregate.java
Problems
Tasks
Javadoc
Description
Resou

```
Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.fact.aggregate.banking.loan;

import java.io.Serializable;

public class LoanAccountAggregate implements IAggregateDerivation {

    private enum LoanAccountFact {
        SanctionedAmount,
        ContributionReceived,
        RepaymentMode,
        ScheduleType,
        Contribution,
        AccountCurrencyCode,
        ParentLimitId,
        LastDisbursementDate,
        IsRateChangeAllowed,
        UserDefinedStatus,
        NetRate,
        AgreementId,
        CustomerShortName,
        LineNo,
        ReserveExist,
        CurrentInstallmentNo,
        BranchCode,
        PayoffNotice,
        LastInterestChargingDate,
        AgreementSignDate,
        CurrentTermMonths,
        TermMonths,
        FirstDisbursementDate,
        IsAdditionalAddressAdded,
        IBanNumber,
        RateType,
        RecoveryDate,
        InterestOnArrearFrozen,
        DrawdownRetries,
        LastCompoundDate,
        AccountCloserDate,
        LastAccrualDate,
```

```
5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)
Errors (5 items)
Warnings (100 of 211160 items)
```

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

Figure 15–18 Creating New Fact - Add



```
    /**
     * @see Session#saveOrUpdate(java.lang.Object)
     * @modelguid {737E074B-DD39-4C79-80C3-7E9D6EA0C461}
     */
    public void saveOrUpdate(IHibernateEntity object) throws PersistenceException {

        try {
            fetchSessionInterceptor().markEntityDirty(object);
            if (isBatchMode()) {
                fetchActionQueue().merge(this, object, object.getClass().getName());
            } else {
                wrappee.saveOrUpdate(object);
            }
            if (StaticCacheRegistry.isRegistered(object.getClass())) {
                StaticCacheRegistry.save(this, object, null);
            }
        } catch (HibernateException e) {
            // flush();
            throw HibernateExceptionTranslator.translate(e);
        }
    }

    /**
     * Either <tt>save()</tt> or <tt>update()</tt> the given instance, depending upon the value of its identi
     * property. By default the instance is always saved. This behaviour may be adjusted by specifying an
     * <tt>unsaved-value</tt> attribute of the identifier property mapping.
     *
     * @see Session#save(java.lang.Object)
     * @see Session#update(Object object, Serializable id)
     * @param entityName
     *            Alias name used in the class mapping
     * @param object
     *            a transient instance containing new or updated state
     * @throws PersistenceException
     */
    @Override
    public void saveOrUpdate(String entityName, IHibernateEntity object) throws PersistenceException {

        try {
            fetchSessionInterceptor().markEntityDirty(object);
```

5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)

Errors (5 items)

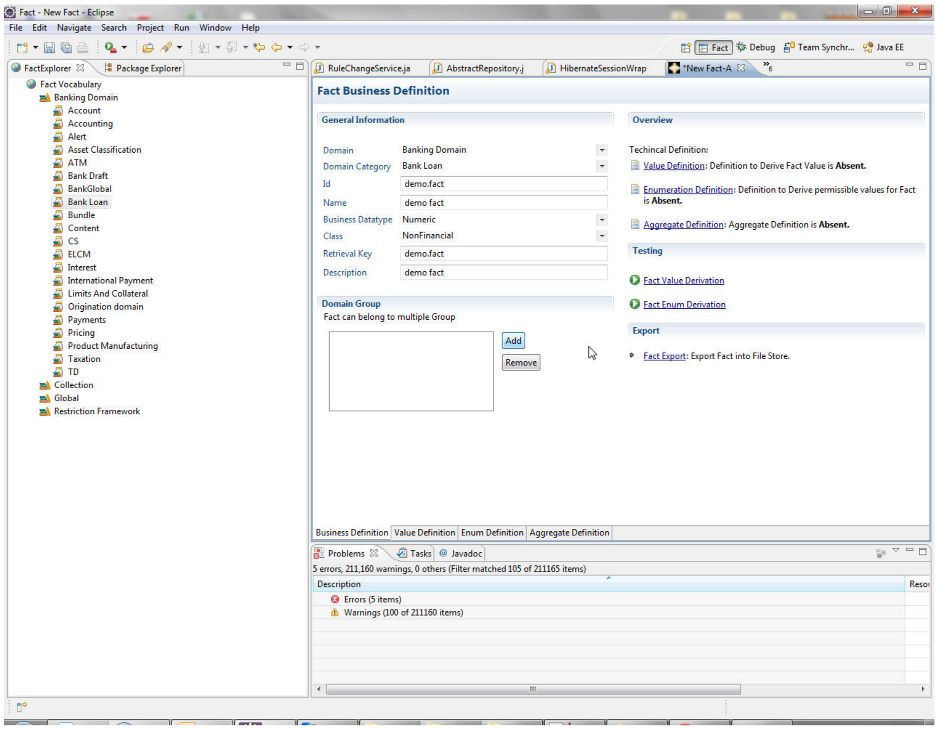
Warnings (100 of 211160 items)

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 15–19 Creating New Fact - Fact Business Definition



Fact Business Definition

General Information

Domain	Banking Domain
Domain Category	Bank Loan
Id	demo.fact
Name	demo fact
Business Datatype	Numeric
Class	NonFinancial
Retrieval Key	demo.fact
Description	demo fact

Domain Group

Fact can belong to multiple Group

Overview

Techincal Definition:

Value Definition: Definition to Derive Fact Value is **Absent.**

Enumeration Definition: Definition to Derive permissible values for Fact is **Absent.**

Aggregate Definition: Aggregate Definition is **Absent.**

Testing

Fact Value Derivation

Fact Enum Derivation

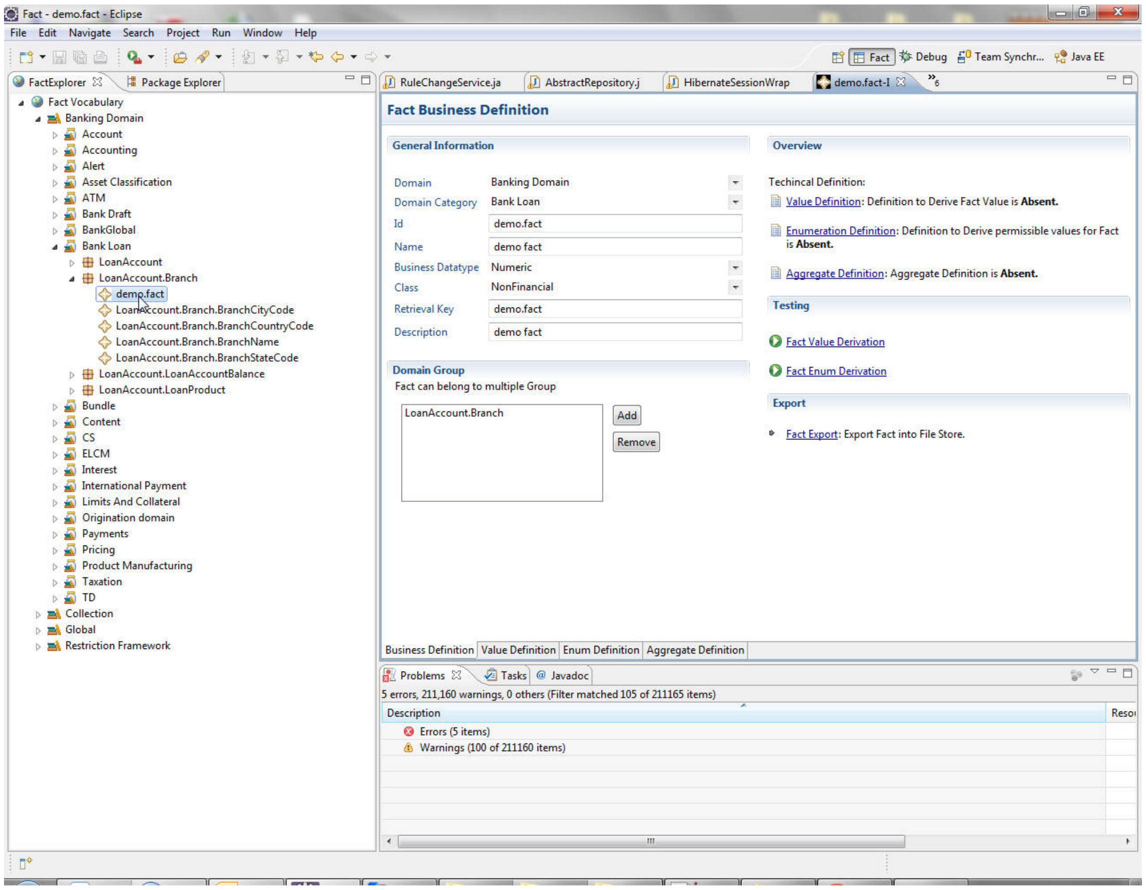
Export

Fact Export: Export Fact into File Store.

5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)

Description	Reso
Errors (5 items)	
Warnings (100 of 211160 items)	

Figure 15–20 Creating New Fact - Domain Group


Fact - demo.fact - Eclipse
File Edit Navigate Search Project Run Window Help
Fact Debug Team Synchr... Java EE
FactExplorer Package Explorer
RuleChangeService.ja AbstractRepository.j HibernateSessionWrap demo.fact-I
Add
Remove
Business Definition Value Definition Enum Definition Aggregate Definition
Problems Tasks Javadoc

Fact Vocabulary
- Banking Domain
  - Account
  - Accounting
  - Alert
  - Asset Classification
  - ATM
  - Bank Draft
  - BankGlobal
  - Bank Loan
    - LoanAccount
    - LoanAccount.Branch
      - demo.fact
      - LoanAccount.Branch.BranchCityCode
      - LoanAccount.Branch.BranchCountryCode
      - LoanAccount.Branch.BranchName
      - LoanAccount.Branch.BranchStateCode
    - LoanAccount.LoanAccountBalance
    - LoanAccount.LoanProduct
  - Bundle
  - Content
  - CS
  - ELCM
  - Interest
  - International Payment
  - Limits And Collateral
  - Origination domain
  - Payments
  - Pricing
  - Product Manufacturing
  - Taxation
  - TD
- Collection
- Global
- Restriction Framework

Fact Business Definition

General Information

Domain	Banking Domain
Domain Category	Bank Loan
Id	demo.fact
Name	demo fact
Business Datatype	Numeric
Class	NonFinancial
Retrieval Key	demo.fact
Description	demo fact

Domain Group

Fact can belong to multiple Group

LoanAccount.Branch

Overview

Techincal Definition:

Value Definition: Definition to Derive Fact Value is **Absent.**

Enumeration Definition: Definition to Derive permissible values for Fact is **Absent.**

Aggregate Definition: Aggregate Definition is **Absent.**

Testing

Fact Value Derivation

Fact Enum Derivation

Export

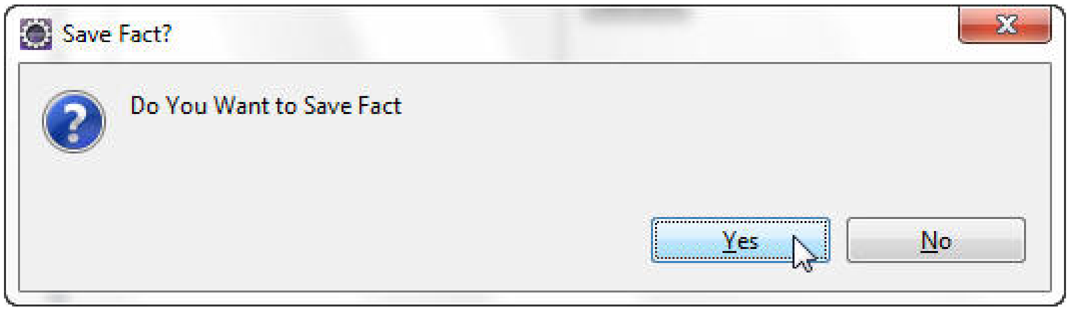
Fact Export: Export Fact into File Store.

5 errors, 211,160 warnings, 0 others (Filter matched 105 of 211165 items)

Description	Resou
Errors (5 items)	
Warnings (100 of 211160 items)	

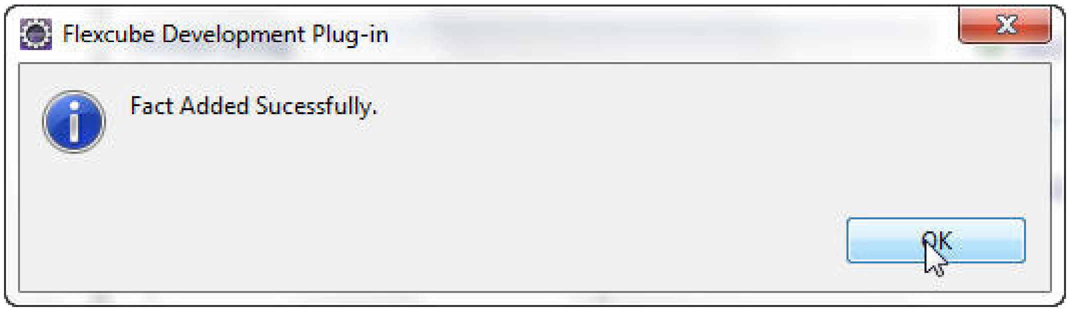
14. Enter the values in the fields and press CTRL+S, click **Yes** to save and fact will be created.

Figure 15–21 Saving New Fact



Do You Want to Save Fact

Figure 15–22 Saving New Fact - Fact Added


Flexcube Development Plug-in

Fact Added Sucessfully.

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.

15.1.4 Object Facts

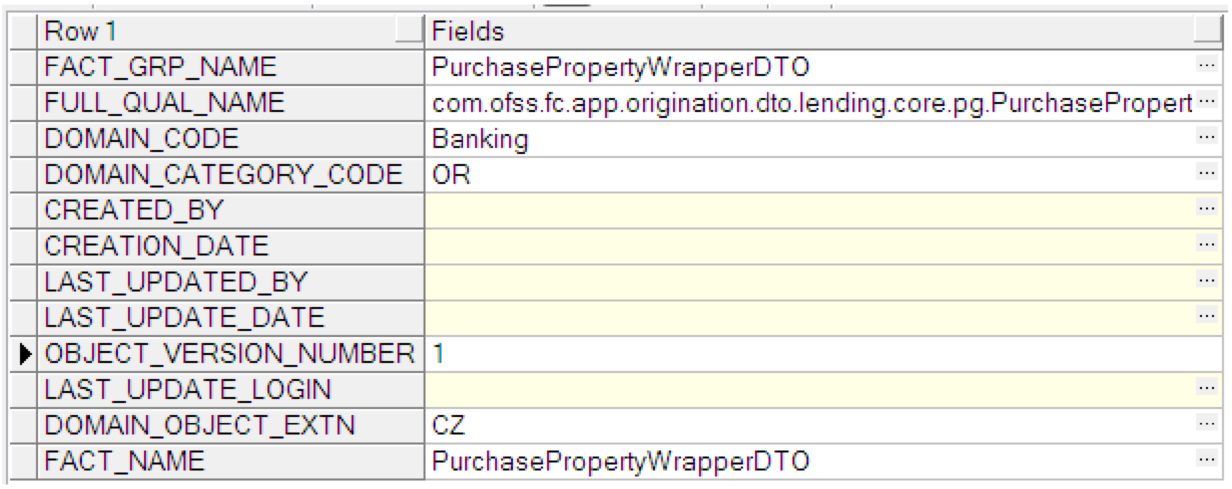
Apart from the normal facts that have to be maintained explicitly, there is a way to define an object as a fact. The idea behind having object fact is to ease the fact definition phase when a particular class holds maximum attributes that are likely to be used in a given rule along the execution path. 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 15–23 Designate Class as Object Fact

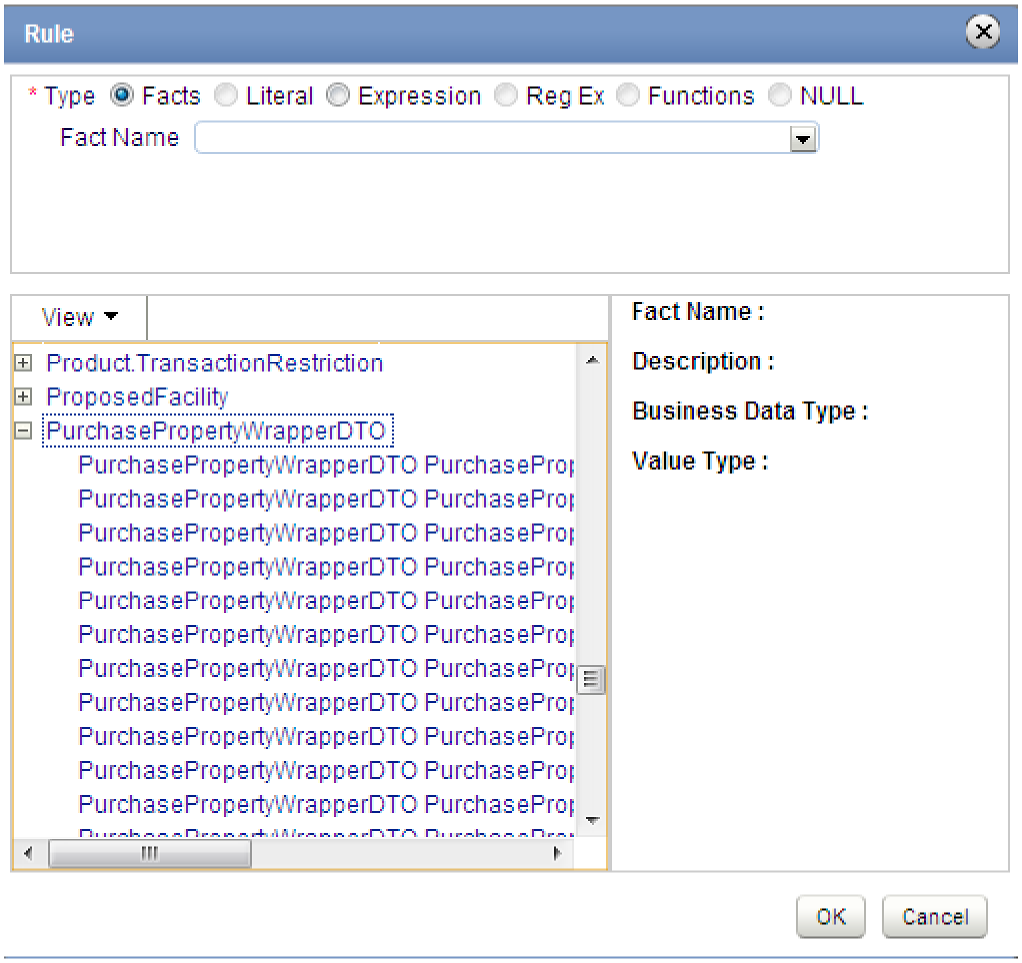


Row 1	Fields
FACT_GRP_NAME	PurchasePropertyWrapperDTO
FULL_QUAL_NAME	com.ofss.fc.app.origination.dto.lending.core.pg.PurchasePropert
DOMAIN_CODE	Banking
DOMAIN_CATEGORY_CODE	OR
CREATED_BY	
CREATION_DATE	
LAST_UPDATED_BY	
LAST_UPDATE_DATE	
OBJECT_VERSION_NUMBER	1
LAST_UPDATE_LOGIN	
DOMAIN_OBJECT_EXTN	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 15–24 Object Fact in UI


Rule
* Type [✓ Facts] Literal Expression Reg Ex Functions NULL
Fact Name
View
Product.TransactionRestriction
ProposedFacility
PurchasePropertyWrapperDTO
PurchasePropertyWrapperDTO PurchaseProp
Fact Name :
Description :
Business Data Type :
Value Type :
OK
Cancel

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)

15.2 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.

15.2.1 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.

15.2.2 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

15.2.3 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 15–1 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.

15.2.4 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 15–2 Actions

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

Table 15–3 Conditions

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

ConditionID	LeftExpression	RelationalOperator	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

15.2.5 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.

15.2.6 Rules Versioning

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

Table 15–4 Rules Versioning

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

15.3 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");
```

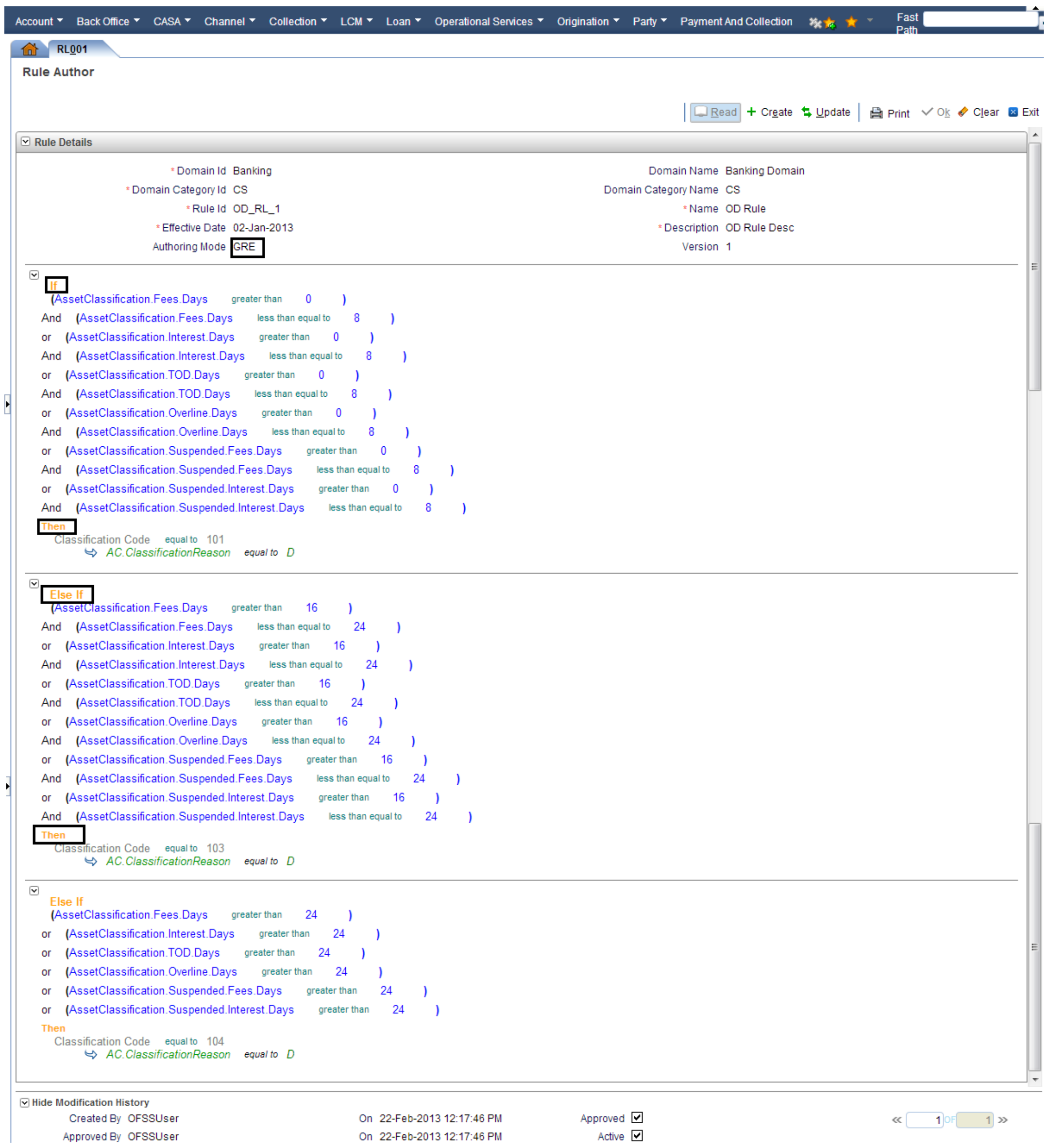
  At the time of CashTransaction Event, code will look like:

```
// 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");
```

15.3.1 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 15–25 Generic Rule Configuration



Rule Author

Rule Details

* Domain Id	Banking	Domain Name	Banking Domain
* Domain Category Id	CS	Domain Category Name	CS
* Rule Id	OD_RL_1	* Name	OD Rule
* Effective Date	02-Jan-2013	* Description	OD Rule Desc
Authoring Mode	GRE	Version	1

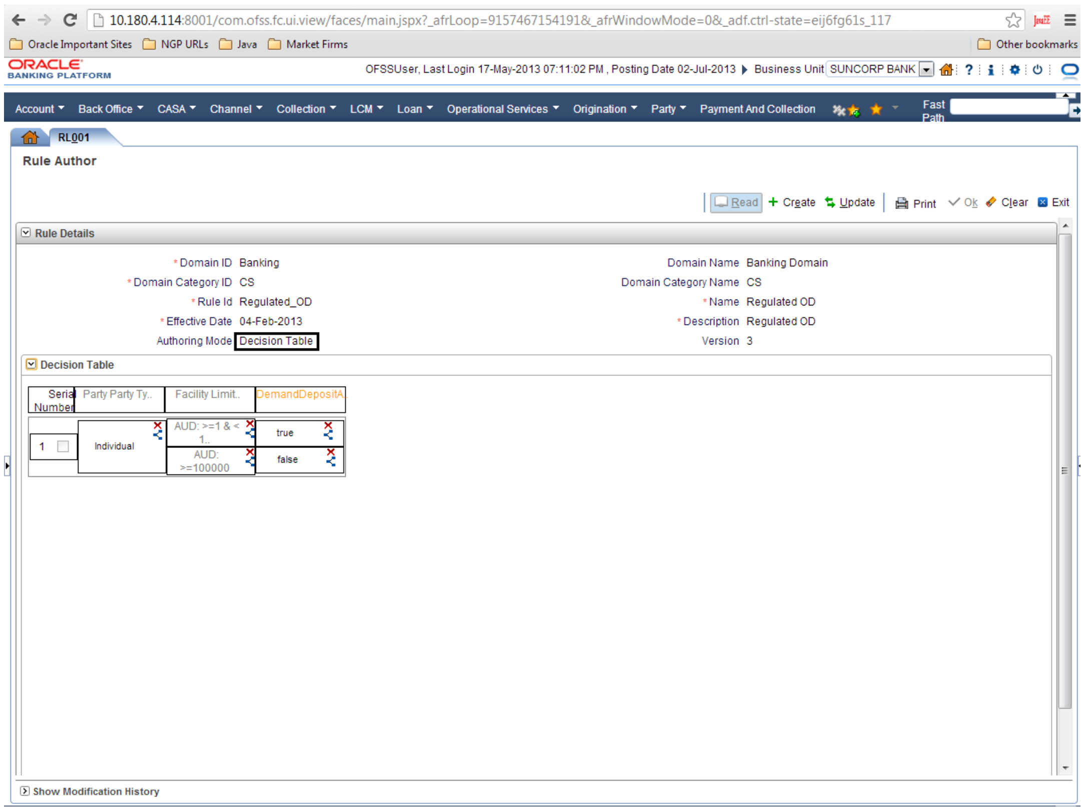
```
If
    (AssetClassification.Fees.Days               greater than            0    )
And (AssetClassification.Fees.Days               less than equal to      8    )
or  (AssetClassification.Interest.Days           greater than            0    )
And (AssetClassification.Interest.Days           less than equal to      8    )
or  (AssetClassification.TOD.Days                greater than            0    )
And (AssetClassification.TOD.Days                less than equal to      8    )
or  (AssetClassification.Overline.Days           greater than            0    )
And (AssetClassification.Overline.Days           less than equal to      8    )
or  (AssetClassification.Suspended.Fees.Days     greater than            0    )
And (AssetClassification.Suspended.Fees.Days     less than equal to      8    )
or  (AssetClassification.Suspended.Interest.Days greater than            0    )
And (AssetClassification.Suspended.Interest.Days less than equal to      8    )
Then
    Classification Code   equal to   101
        AC.ClassificationReason   equal to   D

Else If
    (AssetClassification.Fees.Days               greater than            16   )
And (AssetClassification.Fees.Days               less than equal to      24   )
or  (AssetClassification.Interest.Days           greater than            16   )
And (AssetClassification.Interest.Days           less than equal to      24   )
or  (AssetClassification.TOD.Days                greater than            16   )
And (AssetClassification.TOD.Days                less than equal to      24   )
or  (AssetClassification.Overline.Days           greater than            16   )
And (AssetClassification.Overline.Days           less than equal to      24   )
or  (AssetClassification.Suspended.Fees.Days     greater than            16   )
And (AssetClassification.Suspended.Fees.Days     less than equal to      24   )
or  (AssetClassification.Suspended.Interest.Days greater than            16   )
And (AssetClassification.Suspended.Interest.Days less than equal to      24   )
Then
    Classification Code   equal to   103
        AC.ClassificationReason   equal to   D

Else If
    (AssetClassification.Fees.Days               greater than            24   )
or  (AssetClassification.Interest.Days           greater than            24   )
or  (AssetClassification.TOD.Days                greater than            24   )
or  (AssetClassification.Overline.Days           greater than            24   )
or  (AssetClassification.Suspended.Fees.Days     greater than            24   )
or  (AssetClassification.Suspended.Interest.Days greater than            24   )
Then
    Classification Code   equal to   104
        AC.ClassificationReason   equal to   D
```

Hide Modification History

Created By	OFSSUser	On	22-Feb-2013 12:17:46 PM	Approved	✓
Approved By	OFSSUser	On	22-Feb-2013 12:17:46 PM	Active	✓

1 OF 1

Figure 15–26 Rule Author - Decision Table



Rule Author

Rule Details

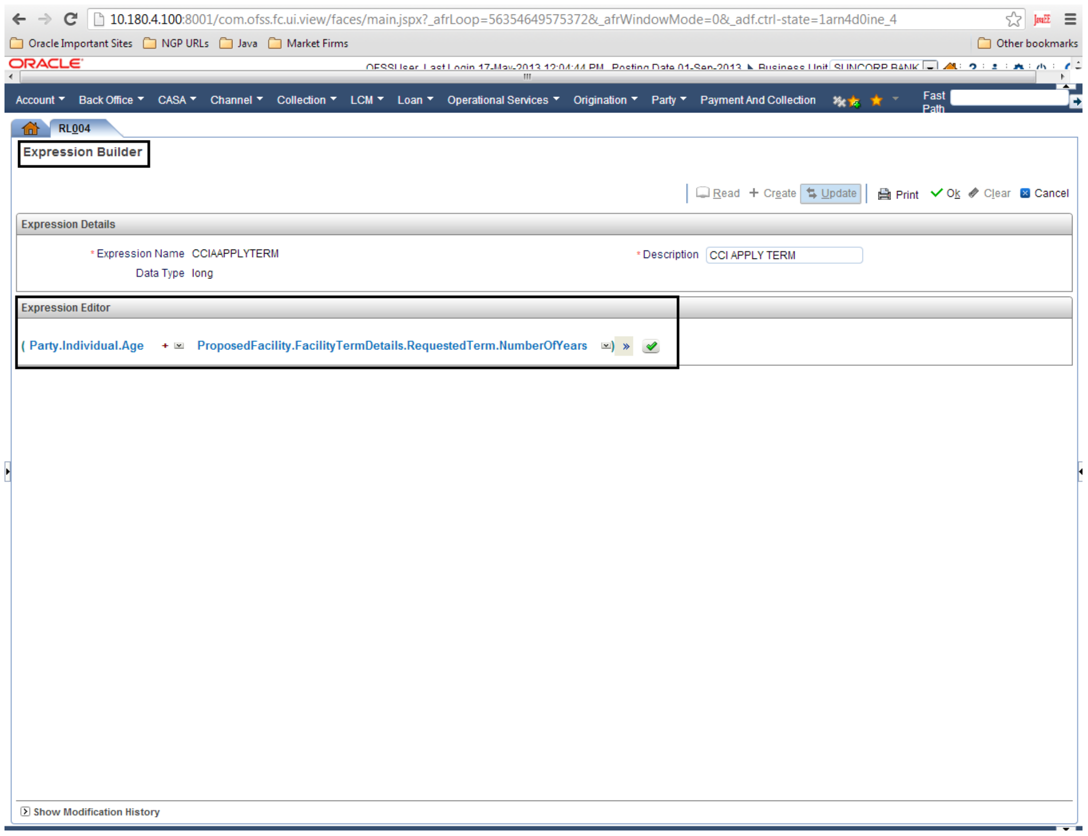
* Domain ID	Banking	Domain Name	Banking Domain
* Domain Category ID	CS	Domain Category Name	CS
* Rule Id	Regulated_OD	* Name	Regulated OD
* Effective Date	04-Feb-2013	* Description	Regulated OD
Authoring Mode	Decision Table	Version	3

Decision Table

Serial Number	Party Party Ty..	Facility Limit..	DemandDepositA..
1	Individual	AUD: >=1 & < 1..	true
		AUD: >=100000	false

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 15–27 Rule Author - Expression Builder



Expression Builder

Expression Details

*Expression Name CCIAAPPLYTERM

Data Type long

*Description CCI APPLY TERM

Expression Editor

(Party.Individual.Age + ProposedFacility.FacilityTermDetails.RequestedTerm.NumberOfYears)

15.4 Rules Migration

This section describes the rules migration.

15.4.1 Rules Configured for Modules

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

Table 15–5 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.

16 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.

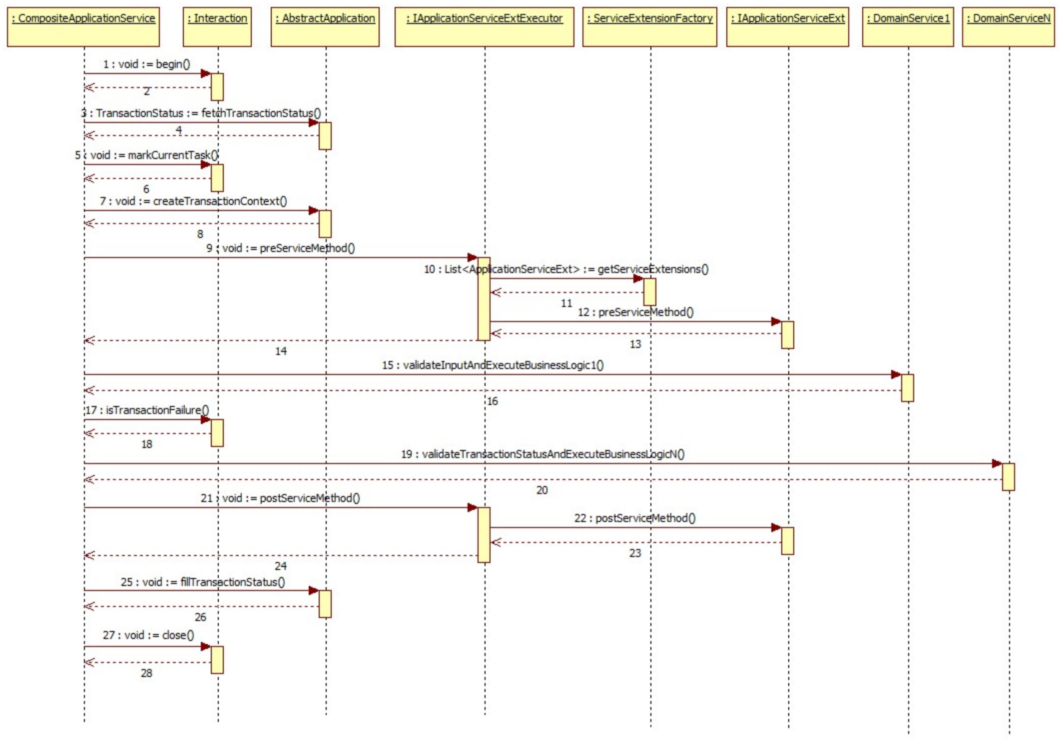
For the transaction control of the transaction manager, there are two defined patterns:

- **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

16.1 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 16–1 Composite Application Service Architecture


: CompositeApplicationService
: Interaction
: AbstractApplication
: IApplicationServiceExtExecutor
: ServiceExtensionFactory
: IApplicationServiceExt
: DomainService1
: DomainServiceN
1 : void := begin()
2
3 : TransactionStatus := fetchTransactionStatus()
4
5 : void := markCurrentTask()
6
7 : void := createTransactionContext()
8
9 : void := preServiceMethod()
10 : List<ApplicationServiceExt> := getServiceExtensions()
11
12 : preServiceMethod()
13
14
15 : validateInputAndExecuteBusinessLogic1()
16
17 : isTransactionFailure()
18
19 : validateTransactionStatusAndExecuteBusinessLogicN()
20
21 : void := postServiceMethod()
22 : postServiceMethod()
23
24
25 : void := fillTransactionStatus()
26
27 : void := close()
28

16.2 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 16–1 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;
}
}
```

17 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:

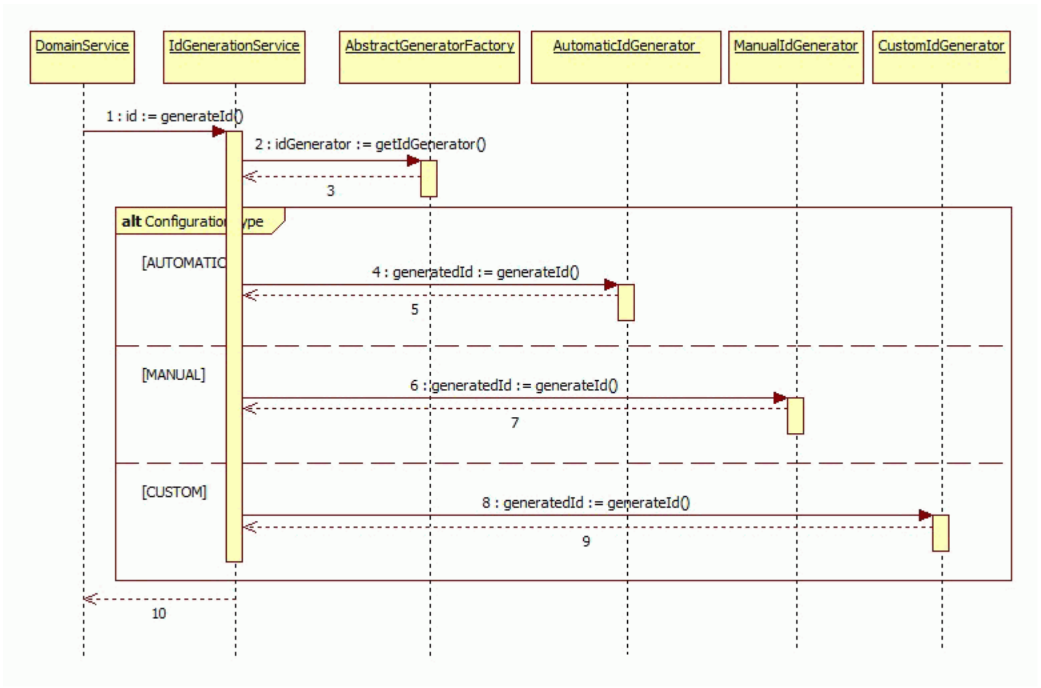
If the configuration type for the ID generation is set to automatic, the ID is generated as per the defined generation logic for the automated ID generation. You can set the pattern, sequence, weights and check digit modulo and modify the automatic generation logic.

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 17–1 Configuration of ID Generation Process



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

17.1 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 17–1 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, NNNBBBYYYYSSSSSSS
CONFIG_TYP	Represents Generation type values are AUT for Automatic, MAN for Manual, CUS for Custom
GENERATOR_CLASS_NAME	Fully Qualified classname of ID generator for config type Custom
SEQ_VALUE	Running Serial Number
WEIGHT	Comma separated Weight for each character defined in the pattern text Example: '0,0,7,6,5,4,3,2', '3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1'
CHK_DIGIT_MODULO	Check digit modulo
CREATED_BY	Indicates the User who created the row
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 17–2 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 17–3 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

17.1.1 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.

17.2 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

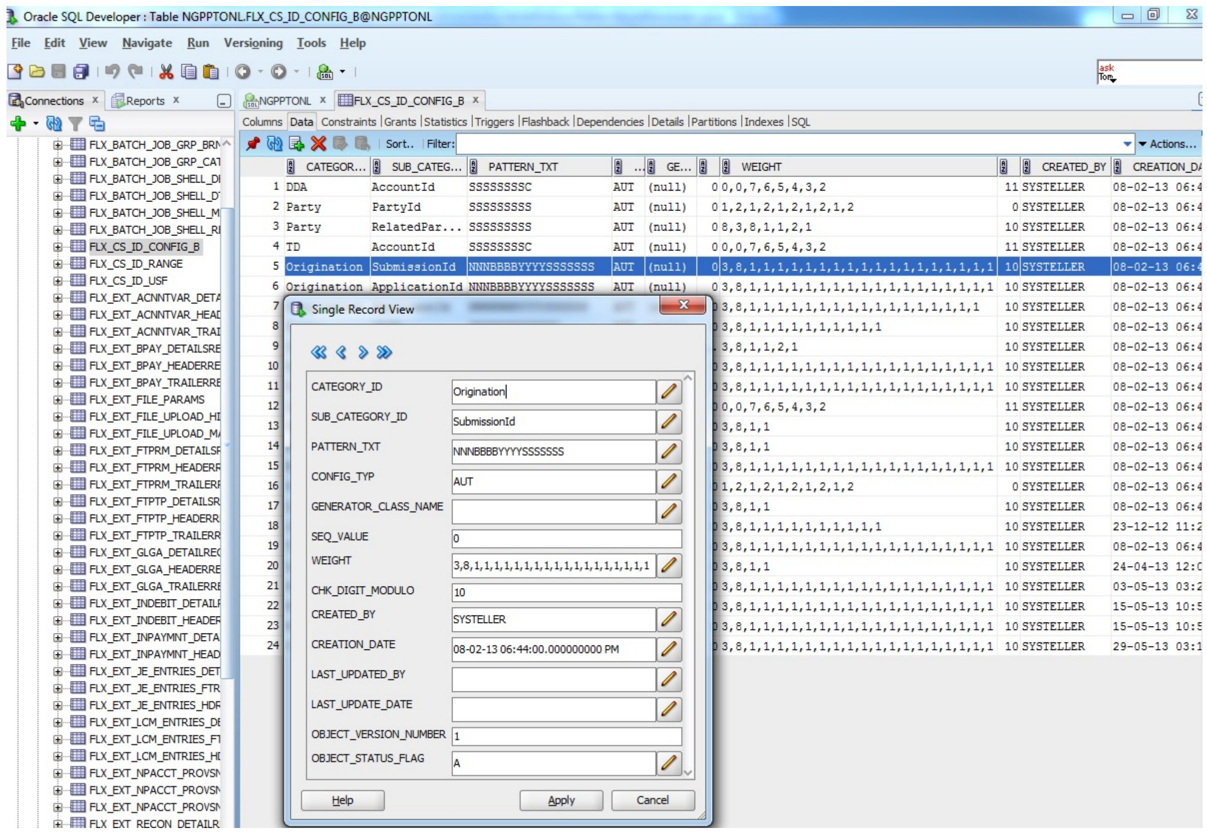
current position value of range until it's size becomes pattern's length. For example, the pattern is 'SSSSSS' and the generated range gives the value as '2345' then the actual value will become '002345'.

- 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 17–2 Automated ID Generation - Single Record View


Oracle SQL Developer : Table NGPPTONL.FLX_CS_ID_CONFIG_B@NGPPTONL
File Edit View Navigate Run Versioning Tools Help
Connections
Reports
NGPPTONL
FLX_CS_ID_CONFIG_B
Columns Data Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL
Sort..
Filter:
Actions...
Single Record View
Help
Apply
Cancel

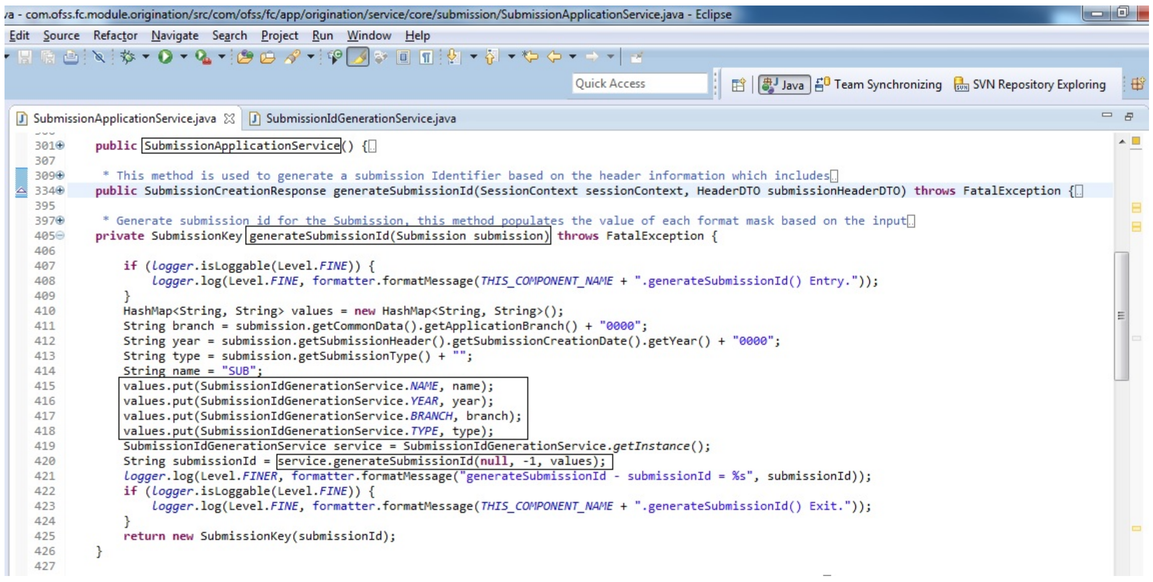
FLX_BATCH_JOB_GRP_BRN
FLX_BATCH_JOB_GRP_CAT
FLX_BATCH_JOB_SHELL_DI
FLX_BATCH_JOB_SHELL_DT
FLX_BATCH_JOB_SHELL_M
FLX_BATCH_JOB_SHELL_RI
FLX_CS_ID_CONFIG_B
FLX_CS_ID_RANGE
FLX_CS_ID_USF
FLX_EXT_ACNNTVAR_DETA
FLX_EXT_ACNNTVAR_HEAD
FLX_EXT_ACNNTVAR_TRAI
FLX_EXT_BPAY_DETAILSRE
FLX_EXT_BPAY_HEADERRE
FLX_EXT_BPAY_TRAILERRE
FLX_EXT_FILE_PARAMS
FLX_EXT_FILE_UPLOAD_HI
FLX_EXT_FILE_UPLOAD_MA
FLX_EXT_FTPRM_DETAILSF
FLX_EXT_FTPRM_HEADERR
FLX_EXT_FTPRM_TRAILERF
FLX_EXT_FTPTP_DETAILSR
FLX_EXT_FTPTP_HEADERR
FLX_EXT_FTPTP_TRAILERR
FLX_EXT_GLGA_DETAILREC
FLX_EXT_GLGA_HEADERRE
FLX_EXT_GLGA_TRAILERRE
FLX_EXT_INDEBIT_DETAILF
FLX_EXT_INDEBIT_HEADER
FLX_EXT_INPAYMNT_DETA
FLX_EXT_INPAYMNT_HEAD
FLX_EXT_JE_ENTRIES_DET
FLX_EXT_JE_ENTRIES_FTR
FLX_EXT_JE_ENTRIES_HDR
FLX_EXT_LCM_ENTRIES_DE
FLX_EXT_LCM_ENTRIES_FT
FLX_EXT_LCM_ENTRIES_HI
FLX_EXT_NPACCT_PROVSN
FLX_EXT_NPACCT_PROVSN
FLX_EXT_NPACCT_PROVSN
FLX_EXT_RECON_DETAILR

	CATEGOR...	SUB_CATEG...	PATTERN_TXT	...	GE...		WEIGHT		CREATED_BY	CREATION_DA
1	DDA	AccountId	SSSSSSSSC	AUT	(null)	0	0,0,7,6,5,4,3,2	11	SYSTELLER	08-02-13 06:4
2	Party	PartyId	SSSSSSSSS	AUT	(null)	0	1,2,1,2,1,2,1,2,1,2	0	SYSTELLER	08-02-13 06:4
3	Party	RelatedPar...	SSSSSSSSS	AUT	(null)	0	8,3,8,1,1,2,1	10	SYSTELLER	08-02-13 06:4
4	TD	AccountId	SSSSSSSSC	AUT	(null)	0	0,0,7,6,5,4,3,2	11	SYSTELLER	08-02-13 06:4
5	Origination	SubmissionId	NNNBBBBYYYYSSSSSSS	AUT	(null)	0	3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	08-02-13 06:4
6	Origination	ApplicationId	NNNBBBBYYYYSSSSSSS	AUT	(null)	0	3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	08-02-13 06:4
7							3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	08-02-13 06:4
8							3,8,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	08-02-13 06:4
9							3,8,1,1,2,1	10	SYSTELLER	08-02-13 06:4
10							3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	08-02-13 06:4
11							3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	08-02-13 06:4
12							0,0,7,6,5,4,3,2	11	SYSTELLER	08-02-13 06:4
13							3,8,1,1	10	SYSTELLER	08-02-13 06:4
14							3,8,1,1	10	SYSTELLER	08-02-13 06:4
15							3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	08-02-13 06:4
16							1,2,1,2,1,2,1,2,1,2	0	SYSTELLER	08-02-13 06:4
17							3,8,1,1	10	SYSTELLER	08-02-13 06:4
18							3,8,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	23-12-12 11:2
19							3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	08-02-13 06:4
20							3,8,1,1	10	SYSTELLER	24-04-13 12:0
21							3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	03-05-13 03:2
22							3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	15-05-13 10:5
23							3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	15-05-13 10:5
24							3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10	SYSTELLER	29-05-13 03:1

CATEGORY_ID	Origination
SUB_CATEGORY_ID	SubmissionId
PATTERN_TXT	NNNBBBBYYYYSSSSSSS
CONFIG_TYP	AUT
GENERATOR_CLASS_NAME	
SEQ_VALUE	0
WEIGHT	3,8,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
CHK_DIGIT_MODULO	10
CREATED_BY	SYSTELLER
CREATION_DATE	08-02-13 06:44:00.000000000 PM
LAST_UPDATED_BY	
LAST_UPDATE_DATE	
OBJECT_VERSION_NUMBER	1
OBJECT_STATUS_FLAG	A

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.

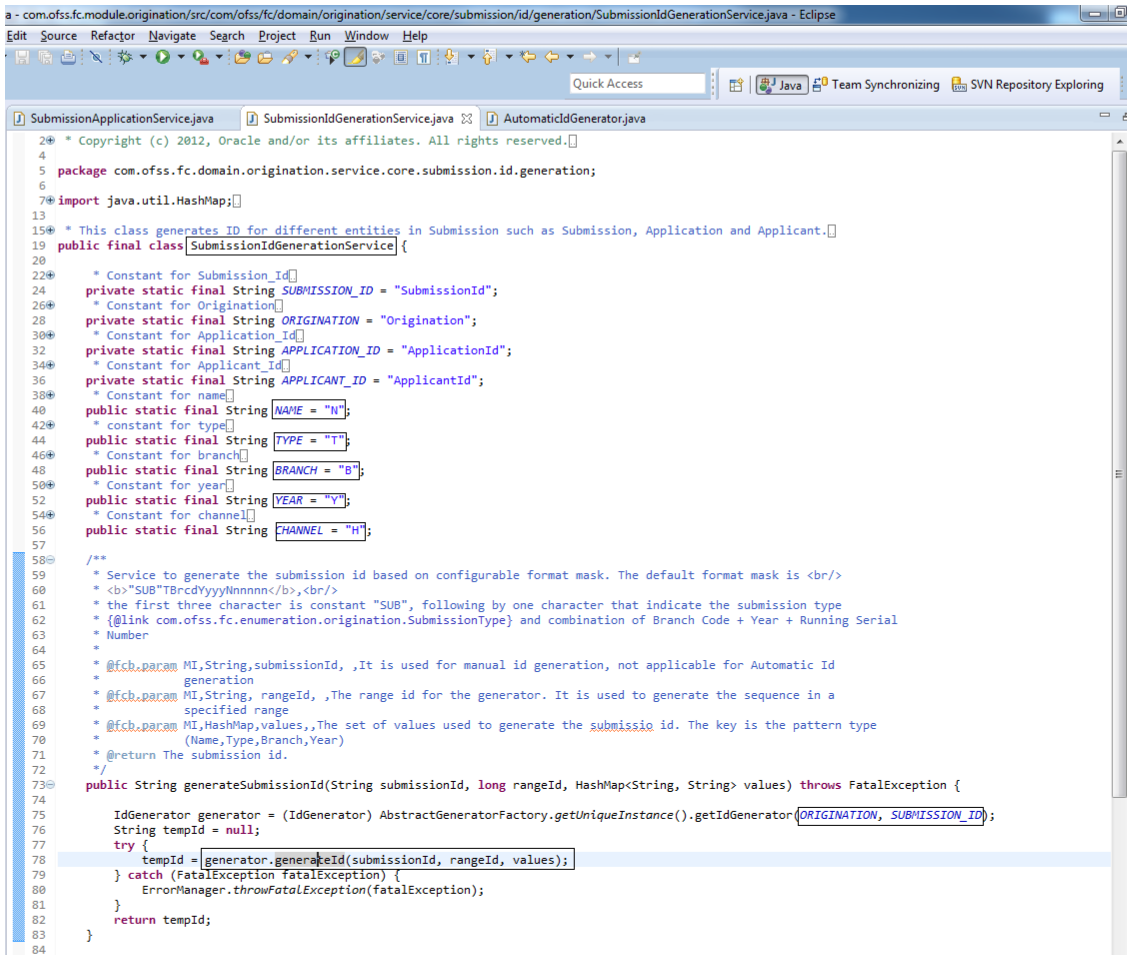
Figure 17–3 Automated ID Generation - Generate Submission ID


Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Team Synchronizing
SVN Repository Exploring
SubmissionApplicationService.java
SubmissionIdGenerationService.java

```
a - com.ofss.fc.module.origination/src/com/ofss/fc/app/origination/service/core/submission/SubmissionApplicationService.java - Eclipse
```

```
    public SubmissionApplicationService() {

     * This method is used to generate a submission Identifier based on the header information which includes
    public SubmissionCreationResponse generateSubmissionId(SessionContext sessionContext, HeaderDTO submissionHeaderDTO) throws FatalException {

     * Generate submission id for the Submission, this method populates the value of each format mask based on the input
    private SubmissionKey generateSubmissionId(Submission submission) throws FatalException {

        if (logger.isLoggable(Level.FINE)) {
            logger.log(Level.FINE, formatter.formatMessage(THIS_COMPONENT_NAME + ".generateSubmissionId() Entry."));
        }
        HashMap<String, String> values = new HashMap<String, String>();
        String branch = submission.getCommonData().getApplicationBranch() + "0000";
        String year = submission.getSubmissionHeader().getSubmissionCreationDate().getYear() + "0000";
        String type = submission.getSubmissionType() + "";
        String name = "SUB";
        values.put(SubmissionIdGenerationService.NAME, name);
        values.put(SubmissionIdGenerationService.YEAR, year);
        values.put(SubmissionIdGenerationService.BRANCH, branch);
        values.put(SubmissionIdGenerationService.TYPE, type);
        SubmissionIdGenerationService service = SubmissionIdGenerationService.getInstance();
        String submissionId = service.generateSubmissionId(null, -1, values);
        logger.log(Level.FINER, formatter.formatMessage("generateSubmissionId - submissionId = %s", submissionId));
        if (logger.isLoggable(Level.FINE)) {
            logger.log(Level.FINE, formatter.formatMessage(THIS_COMPONENT_NAME + ".generateSubmissionId() Exit."));
        }
        return new SubmissionKey(submissionId);
    }

```

Figure 17–4 Automated ID Generation - Submission ID Generation Service


Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Team Synchronizing
SVN Repository Exploring
SubmissionApplicationService.java
SubmissionIdGenerationService.java
AutomaticIdGenerator.java

```
a - com.ofss.fc.module.origination/src/com/ofss/fc/domain/origination/service/core/submission/id/generation/SubmissionIdGenerationService.java - Eclipse
```

```
 * Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.

package com.ofss.fc.domain.origination.service.core.submission.id.generation;

import java.util.HashMap;

 * This class generates ID for different entities in Submission such as Submission, Application and Applicant.
public final class SubmissionIdGenerationService {

     * Constant for Submission_Id
    private static final String SUBMISSION_ID = "SubmissionId";
     * Constant for Origination
    private static final String ORIGINATION = "Origination";
     * Constant for Application_Id
    private static final String APPLICATION_ID = "ApplicationId";
     * Constant for Applicant_Id
    private static final String APPLICANT_ID = "ApplicantId";
     * Constant for name
    public static final String NAME = "N";
     * constant for type
    public static final String TYPE = "T";
     * Constant for branch
    public static final String BRANCH = "B";
     * Constant for year
    public static final String YEAR = "Y";
     * Constant for channel
    public static final String CHANNEL = "H";

    /**
     * Service to generate the submission id based on configurable format mask. The default format mask is <br/>
     * <b>"SUB"TBrcdYyyyNnnnnn</b>,<br/>
     * the first three character is constant "SUB", following by one character that indicate the submission type
     * {@link com.ofss.fc.enumeration.origination.SubmissionType} and combination of Branch Code + Year + Running Serial
     * Number
     *
     * @fcb.param MI,String,submissionId, ,It is used for manual id generation, not applicable for Automatic Id
     *            generation
     * @fcb.param MI,String, rangeId, ,The range id for the generator. It is used to generate the sequence in a
     *            specified range
     * @fcb.param MI,HashMap,values,,The set of values used to generate the submissio id. The key is the pattern type
     *            (Name,Type,Branch,Year)
     * @return The submission id.
     */
    public String generateSubmissionId(String submissionId, long rangeId, HashMap<String, String> values) throws FatalException {

        IdGenerator generator = (IdGenerator) AbstractGeneratorFactory.getUniqueInstance().getIdGenerator(ORIGINATION, SUBMISSION_ID);
        String tempId = null;
        try {
            tempId = generator.generateId(submissionId, rangeId, values);
        } catch (FatalException fatalException) {
            ErrorManager.throwFatalException(fatalException);
        }
        return tempId;
    }

```

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.

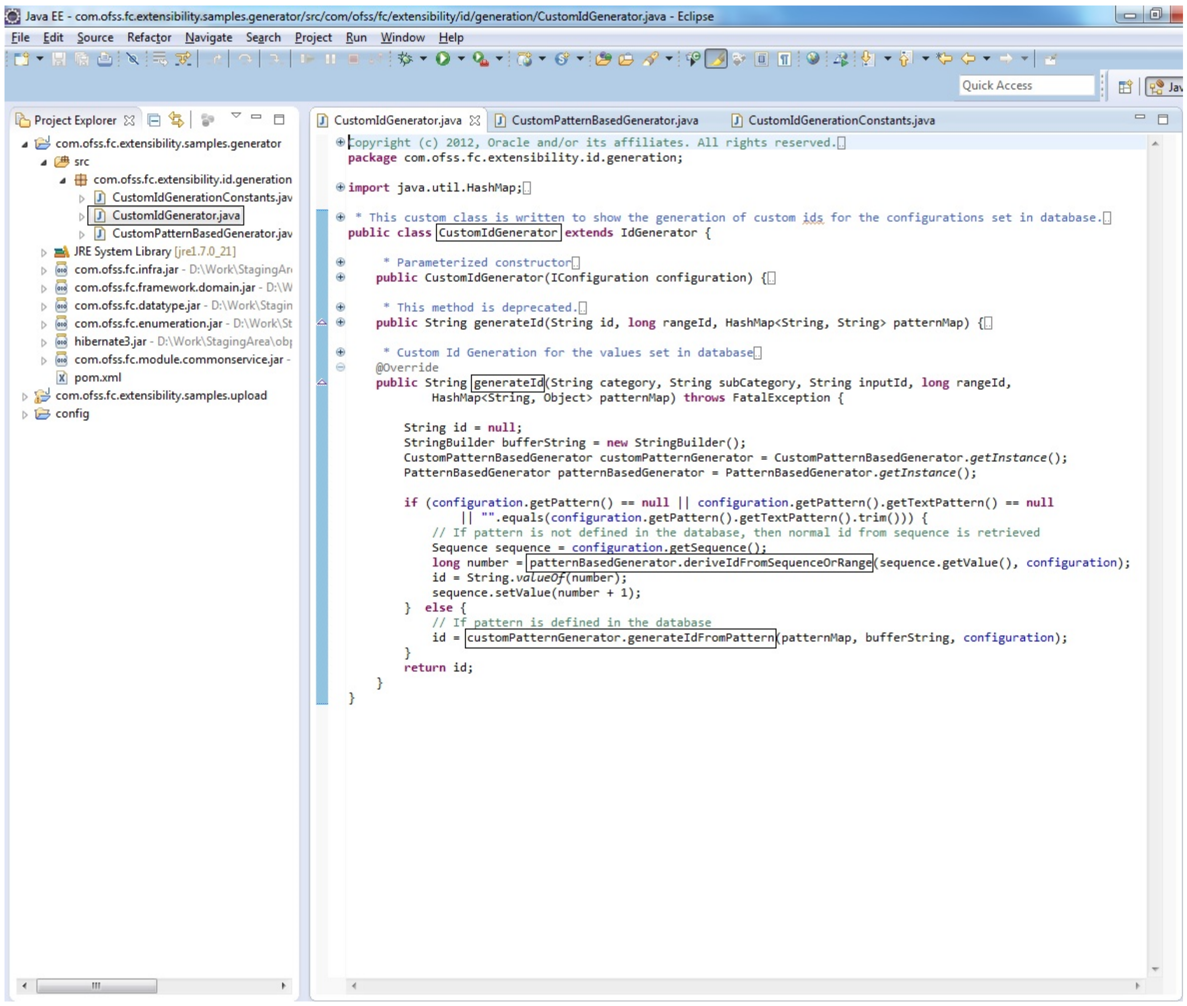
In case of without mask configuration's pattern. If range ID is -1, it means that there is no range defined for the mask configuration, it then picks up the range details with range ID based on the category and sub-category. The generated ID will become the current position of range. If range is not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

17.3 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.

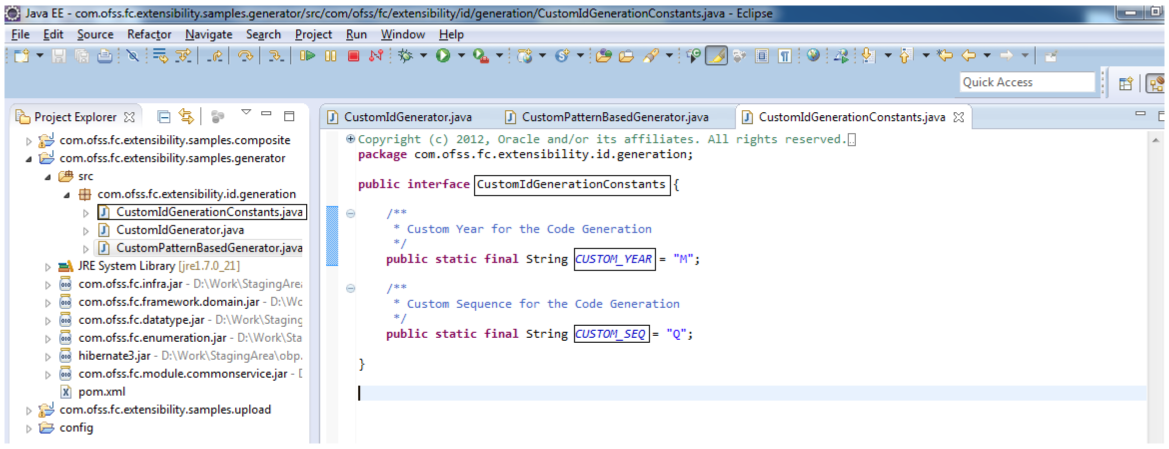
Figure 17–5 Custom ID Generation - Custom ID Generator


Java EE - com.ofss.fc.extensibility.samples.generator/src/com/ofss/fc/extensibility/id/generation/CustomIdGenerator.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
com.ofss.fc.extensibility.samples.generator
src
com.ofss.fc.extensibility.id.generation
CustomIdGenerationConstants.jav
CustomIdGenerator.java
CustomPatternBasedGenerator.jav
JRE System Library [jre1.7.0_21]
com.ofss.fc.infra.jar - D:\Work\StagingAr
com.ofss.fc.framework.domain.jar - D:\W
com.ofss.fc.datatype.jar - D:\Work\Stagin
com.ofss.fc.enumeration.jar - D:\Work\St
hibernate3.jar - D:\Work\StagingArea\ob
com.ofss.fc.module.commonservice.jar -
pom.xml
com.ofss.fc.extensibility.samples.upload
config
CustomIdGenerator.java
CustomPatternBasedGenerator.java
CustomIdGenerationConstants.java

```
Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.extensibility.id.generation;

import java.util.HashMap;

 * This custom class is written to show the generation of custom ids for the configurations set in database.
public class CustomIdGenerator extends IdGenerator {

     * Parameterized constructor
    public CustomIdGenerator(IConfiguration configuration) {

     * This method is deprecated.
    public String generateId(String id, long rangeId, HashMap<String, String> patternMap) {

     * Custom Id Generation for the values set in database
    @Override
    public String generateId(String category, String subCategory, String inputId, long rangeId,
            HashMap<String, Object> patternMap) throws FatalException {

        String id = null;
        StringBuilder bufferString = new StringBuilder();
        CustomPatternBasedGenerator customPatternGenerator = CustomPatternBasedGenerator.getInstance();
        PatternBasedGenerator patternBasedGenerator = PatternBasedGenerator.getInstance();

        if (configuration.getPattern() == null || configuration.getPattern().getTextPattern() == null
                || "".equals(configuration.getPattern().getTextPattern().trim())) {
            // If pattern is not defined in the database, then normal id from sequence is retrieved
            Sequence sequence = configuration.getSequence();
            long number = patternBasedGenerator.deriveIdFromSequenceOrRange(sequence.getValue(), configuration);
            id = String.valueOf(number);
            sequence.setValue(number + 1);
        }  else {
            // If pattern is defined in the database
            id = customPatternGenerator.generateIdFromPattern(patternMap, bufferString, configuration);
        }
        return id;
    }
}
```

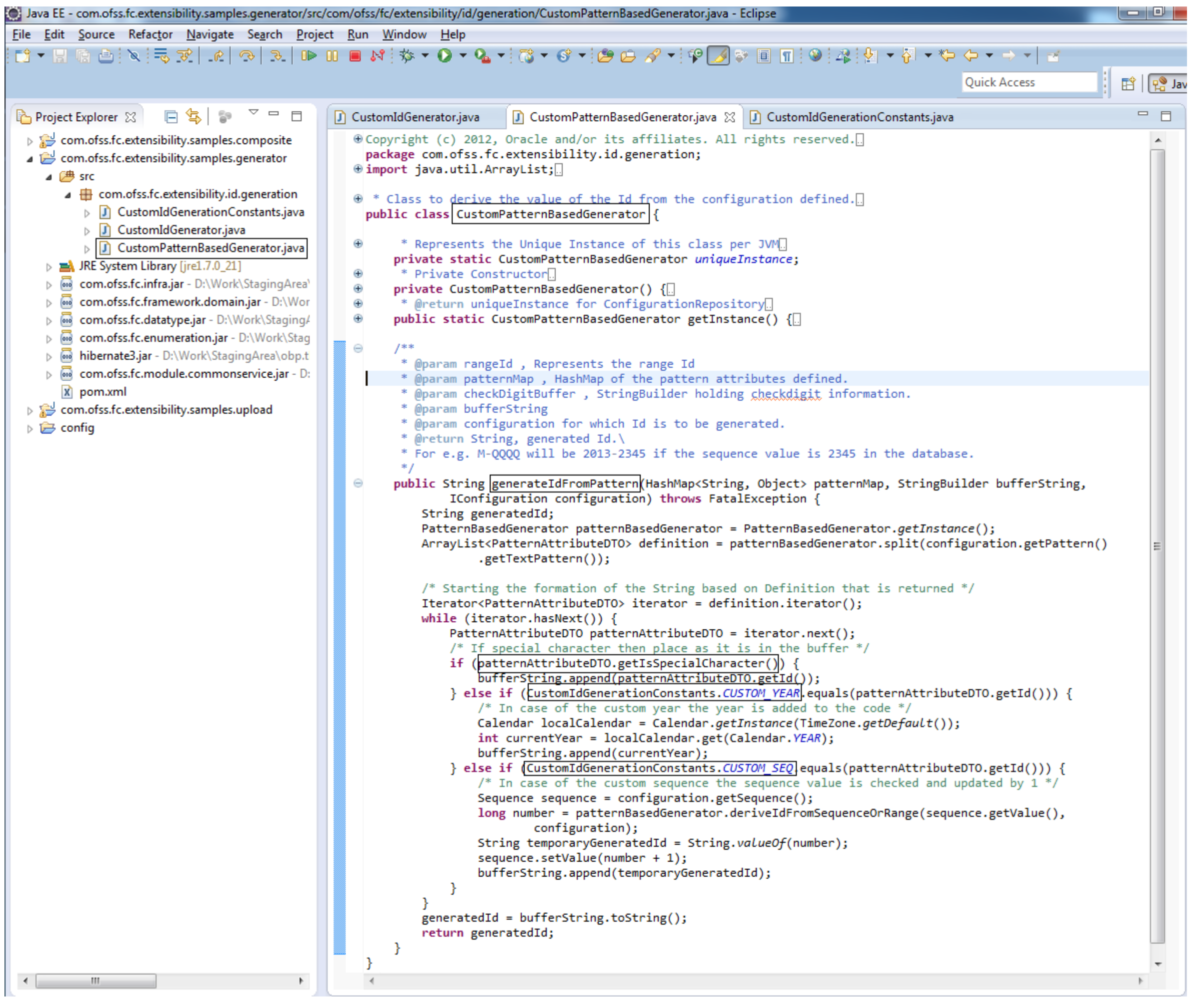
In case the user want to write the custom generation logic in a specific customized pattern definition, then user can do that by writing the custom constant class and the custom pattern class which can pick the defined pattern from the configuration object set in the PATTERN_TXT column of the FLX_CS_ID_CONFIG_B 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 17–6 Custom ID Generation - Custom ID Generation Constants


Java EE - com.ofss.fc.extensibility.samples.generator/src/com/ofss/fc/extensibility/id/generation/CustomIdGenerationConstants.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
com.ofss.fc.extensibility.samples.composite
com.ofss.fc.extensibility.samples.generator
src
com.ofss.fc.extensibility.id.generation
CustomIdGenerationConstants.java
CustomIdGenerator.java
CustomPatternBasedGenerator.java
JRE System Library [jre1.7.0_21]
com.ofss.fc.infra.jar - D:\Work\StagingAre
com.ofss.fc.framework.domain.jar - D:\Wc
com.ofss.fc.datatype.jar - D:\Work\Staging
com.ofss.fc.enumeration.jar - D:\Work\Sta
hibernate3.jar - D:\Work\StagingArea\obp.
com.ofss.fc.module.commonservice.jar - [
pom.xml
com.ofss.fc.extensibility.samples.upload
config
CustomIdGenerator.java
CustomPatternBasedGenerator.java
CustomIdGenerationConstants.java

```
Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.extensibility.id.generation;

public interface CustomIdGenerationConstants {

    /**
     * Custom Year for the Code Generation
     */
    public static final String CUSTOM_YEAR = "M";

    /**
     * Custom Sequence for the Code Generation
     */
    public static final String CUSTOM_SEQ = "Q";

}
```

Figure 17–7 Custom ID Generation - Custom Pattern Based Generator


Java EE - com.ofss.fc.extensibility.samples.generator/src/com/ofss/fc/extensibility/id/generation/CustomPatternBasedGenerator.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
com.ofss.fc.extensibility.samples.composite
com.ofss.fc.extensibility.samples.generator
src
com.ofss.fc.extensibility.id.generation
CustomIdGenerationConstants.java
CustomIdGenerator.java
CustomPatternBasedGenerator.java
JRE System Library [jre1.7.0_21]
com.ofss.fc.infra.jar - D:\Work\StagingArea\
com.ofss.fc.framework.domain.jar - D:\Wor
com.ofss.fc.datatype.jar - D:\Work\Staging/
com.ofss.fc.enumeration.jar - D:\Work\Stag
hibernate3.jar - D:\Work\StagingArea\obp.t
com.ofss.fc.module.commonservice.jar - D:
pom.xml
com.ofss.fc.extensibility.samples.upload
config
CustomIdGenerator.java
CustomPatternBasedGenerator.java
CustomIdGenerationConstants.java

```java
Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
package com.ofss.fc.extensibility.id.generation;
import java.util.ArrayList;

 * Class to derive the value of the Id from the configuration defined.
public class CustomPatternBasedGenerator {

     * Represents the Unique Instance of this class per JVM
    private static CustomPatternBasedGenerator uniqueInstance;
     * Private Constructor
    private CustomPatternBasedGenerator() {
     * @return uniqueInstance for ConfigurationRepository
    public static CustomPatternBasedGenerator getInstance() {

    /**
     * @param rangeId , Represents the range Id
     * @param patternMap , HashMap of the pattern attributes defined.
     * @param checkDigitBuffer , StringBuilder holding checkdigit information.
     * @param bufferString
     * @param configuration for which Id is to be generated.
     * @return String, generated Id.\
     * For e.g. M-QQQQ will be 2013-2345 if the sequence value is 2345 in the database.
     */
    public String generateIdFromPattern(HashMap<String, Object> patternMap, StringBuilder bufferString,
            IConfiguration configuration) throws FatalException {
        String generatedId;
        PatternBasedGenerator patternBasedGenerator = PatternBasedGenerator.getInstance();
        ArrayList<PatternAttributeDTO> definition = patternBasedGenerator.split(configuration.getPattern()
                .getTextPattern());

        /* Starting the formation of the String based on Definition that is returned */
        Iterator<PatternAttributeDTO> iterator = definition.iterator();
        while (iterator.hasNext()) {
            PatternAttributeDTO patternAttributeDTO = iterator.next();
            /* If special character then place as it is in the buffer */
            if (patternAttributeDTO.getIsSpecialCharacter()) {
                bufferString.append(patternAttributeDTO.getId());
            } else if (CustomIdGenerationConstants.CUSTOM_YEAR.equals(patternAttributeDTO.getId())) {
                /* In case of the custom year the year is added to the code */
                Calendar localCalendar = Calendar.getInstance(TimeZone.getDefault());
                int currentYear = localCalendar.get(Calendar.YEAR);
                bufferString.append(currentYear);
            } else if (CustomIdGenerationConstants.CUSTOM_SEQ.equals(patternAttributeDTO.getId())) {
                /* In case of the custom sequence the sequence value is checked and updated by 1 */
                Sequence sequence = configuration.getSequence();
                long number = patternBasedGenerator.deriveIdFromSequenceOrRange(sequence.getValue(),
                        configuration);
                String temporaryGeneratedId = String.valueOf(number);
                sequence.setValue(number + 1);
                bufferString.append(temporaryGeneratedId);
            }
        }
        generatedId = bufferString.toString();
        return generatedId;
    }
}
```

18 Extensibility of Domain Objects using Flex Fields

This chapter describes about the Flex Field provisioning by the product at the service layer. Flexfields are additional attributes provisioned to the consultant upfront or through configuration, with basic validation. By the use of flex field, consultant or client can add additional data elements as part of the entity, without adding custom codes.

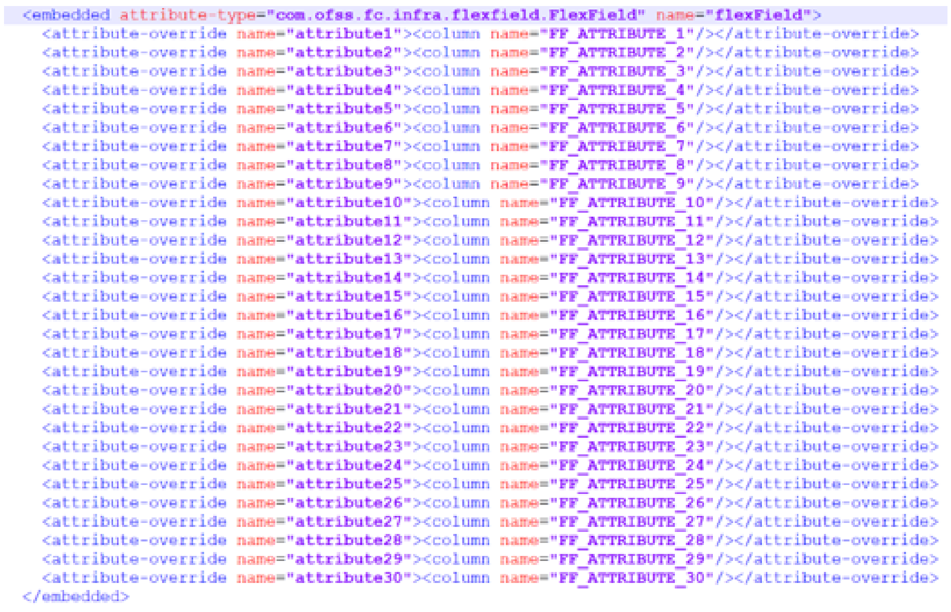
18.1 Flex Field - Provisioning details

- Maximum 30 attributes per entity is provisioned at each entity level. Attributes data type declared as String, for flexibility. This has been added as part of AbstractDomainObject (will be available for all OBP entities). For Over and above fields, consultant is expected to go via customized entity extension approach.
    - **ORM level**: Provisioned as FlexField embedded attribute. As performance effectiveness, if flexfield is not used for specific entity, this can be removed by replacing the ORM.

      Note

      Dynamic way of enabling this only for the entities required is in the future scope of product.

Figure 18–1 Example - ORM Level

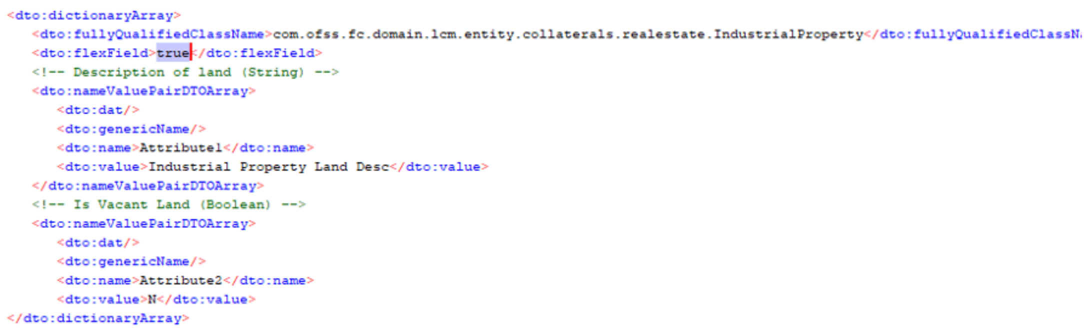


```
<embedded attribute-type="com.ofss.fc.infra.flexfield.FlexField" name="flexField">
  <attribute-override name="attribute1"><column name="FF_ATTRIBUTE_1"/></attribute-override>
  <attribute-override name="attribute2"><column name="FF_ATTRIBUTE_2"/></attribute-override>
  <attribute-override name="attribute3"><column name="FF_ATTRIBUTE_3"/></attribute-override>
  <attribute-override name="attribute4"><column name="FF_ATTRIBUTE_4"/></attribute-override>
  <attribute-override name="attribute5"><column name="FF_ATTRIBUTE_5"/></attribute-override>
  <attribute-override name="attribute6"><column name="FF_ATTRIBUTE_6"/></attribute-override>
  <attribute-override name="attribute7"><column name="FF_ATTRIBUTE_7"/></attribute-override>
  <attribute-override name="attribute8"><column name="FF_ATTRIBUTE_8"/></attribute-override>
  <attribute-override name="attribute9"><column name="FF_ATTRIBUTE_9"/></attribute-override>
  <attribute-override name="attribute10"><column name="FF_ATTRIBUTE_10"/></attribute-override>
  <attribute-override name="attribute11"><column name="FF_ATTRIBUTE_11"/></attribute-override>
  <attribute-override name="attribute12"><column name="FF_ATTRIBUTE_12"/></attribute-override>
  <attribute-override name="attribute13"><column name="FF_ATTRIBUTE_13"/></attribute-override>
  <attribute-override name="attribute14"><column name="FF_ATTRIBUTE_14"/></attribute-override>
  <attribute-override name="attribute15"><column name="FF_ATTRIBUTE_15"/></attribute-override>
  <attribute-override name="attribute16"><column name="FF_ATTRIBUTE_16"/></attribute-override>
  <attribute-override name="attribute17"><column name="FF_ATTRIBUTE_17"/></attribute-override>
  <attribute-override name="attribute18"><column name="FF_ATTRIBUTE_18"/></attribute-override>
  <attribute-override name="attribute19"><column name="FF_ATTRIBUTE_19"/></attribute-override>
  <attribute-override name="attribute20"><column name="FF_ATTRIBUTE_20"/></attribute-override>
  <attribute-override name="attribute21"><column name="FF_ATTRIBUTE_21"/></attribute-override>
  <attribute-override name="attribute22"><column name="FF_ATTRIBUTE_22"/></attribute-override>
  <attribute-override name="attribute23"><column name="FF_ATTRIBUTE_23"/></attribute-override>
  <attribute-override name="attribute24"><column name="FF_ATTRIBUTE_24"/></attribute-override>
  <attribute-override name="attribute25"><column name="FF_ATTRIBUTE_25"/></attribute-override>
  <attribute-override name="attribute26"><column name="FF_ATTRIBUTE_26"/></attribute-override>
  <attribute-override name="attribute27"><column name="FF_ATTRIBUTE_27"/></attribute-override>
  <attribute-override name="attribute28"><column name="FF_ATTRIBUTE_28"/></attribute-override>
  <attribute-override name="attribute29"><column name="FF_ATTRIBUTE_29"/></attribute-override>
  <attribute-override name="attribute30"><column name="FF_ATTRIBUTE_30"/></attribute-override>
</embedded>
```

- **DB level:** The above columns will be part of table, with datatype as varchar.

- Service input / data transfer is supported through Dictionary Object (Separate indicator is provided to distinguish flex field dictionary object). The attributes which is passed as part of the Dictionary object with the indicator flex field, will be persisted as flex field in the respective element. The attribute name follow the name convention as “Attribute<<attribute number>>” [‘A’ caps, like Attribute1, Attribute2, Attribute3, …. and Attribute30].

Figure 18–2 Example of Service Input / Data Transfer through Dictionary Object



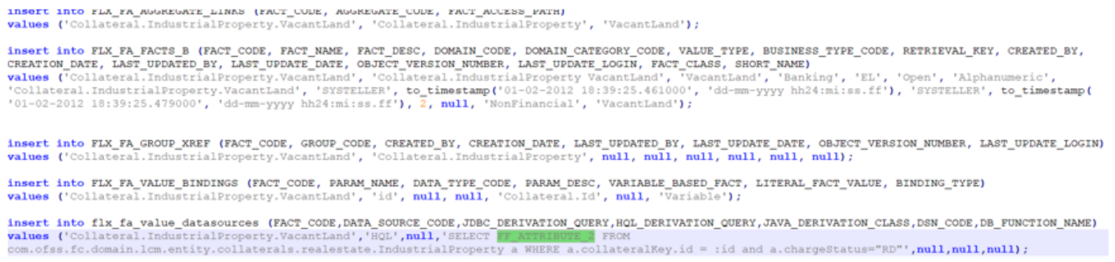
```
<dto:dictionaryArray>
   <dto:fullyQualifiedClassName>com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty</dto:fullyQualifiedClassN
   <dto:flexField>true</dto:flexField>
   <!-- Description of land (String) -->
   <dto:nameValuePairDTOArray>
      <dto:dat/>
      <dto:genericName/>
      <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:dictionaryArray>
```

18.2 Flex Field - Fact support

Flex fields provisioned can be consumed as facts as below,

- Object entity based facts can directly use, since available as part of AbstractDomainObject.
- Derived facts can be created, using custom code / value data sources (HQL) based on the embedded object (#FlexField).

Figure 18–3 Example



```
insert into FLX_FA_AGGREGATE_LINKS (FACT_CODE, AGGREGATE_CODE, FACT_ACCESS_PATH)
values ('Collateral.IndustrialProperty.VacantLand', 'Collateral.IndustrialProperty', 'VacantLand');

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,DATA_SOURCE_CODE,JDBC_DERIVATION_QUERY,HQL_DERIVATION_QUERY,JAVA_DERIVATION_CLASS,DSN_CODE,DB_FUNCTION_NAME)
values ('Collateral.IndustrialProperty.VacantLand','HQL',null,'SELECT FF_ATTRIBUTE_2 FROM
com.ofss.fc.domain.lcm.entity.collaterals.realestate.IndustrialProperty a WHERE a.collateralKey.id = :id and a.chargeStatus="RD"',null,null,null);
```

18.3 Flex Field – Validation Support

Basic validation is supported for flex fields using configuration. Flex field has metadata where at each attribute level for the entity, supported validation can be configured. Below are the details on the metadata configuration for supported validations. This needs be seeded, requires restart to reflect.

Note

Configuration screen is in future scope of product.

Table 18–1 Metadata Table - flx_fw_ff_metadata

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 / Attribute2 / so on...	Attribute1
LABEL	Label or description of the attribute. When validation error message is thrown, this is used to throw exception. If not maintained, then the attribute name will be used for the val-	Description of land

Column	Description	Example
	idation message.	
ATTRIBUTE_DATA_TYPE	Attribute data type. Enumeration (String / BigDecimal / Enum / Date ..). Used when validating the field based on the datatype.	STRING
IS_MANDATORY	Validator field: Indicates whether attribute value is mandatory. Check for not null / empty values.	Y
MIN_LENGTH	Validator field: Indicates the minimum length required for the	5

Column	Descrip-tion	Example
	attribute. Validates, if main-tained some value.	
MAX_ LENGTH	Validator field: Indic-ates the maximum length required for the attribute. Validates, if main-tained some value.	250
PATTER-N_REGEX	Validator field: Indic-ates the regular expres-sion sup-ported by the attrib-ute. Val-idates, if main-tained some value.	^[a-zA-Z0-9]*$
ENUM_ TYPE_ NAME	Validator field: Indic-	<<Applicable only for Enum type, e.g., com.ofss.fc.e-numeration.lcm.collaterals.CollateralType>>

Column	Descrip-tion	Example
	ates the enu-meration type sup-ported by the attrib-ute. Fully qualified name. Checks with the enu-meration value sent. Val-idates, if main-tained some value and data type is Enum.	
MAX_ DATE_ VALIDATO-R_TYPE	Validator field: Indic-ates the maximum date val-idator type. (Posting date / Sys-tem date / Value date). Val-idates whether	<<Applicable only for Date type, e.g., POSTING_DATE>>

Column	Description	Example
	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.	

18.4 Flex Field – Usage Instructions

Perform the following steps for usage:

- Identify the entity for which flex field support is required. Verify with the product team whether flex field provisioning is already available. If not, you can add similar to Section 18.1 Flex Field - Provisioning details. [Post dynamic provisioning, this will be enabled via configuration].
- Pass / Retrieve the attributes via Dictionary object, as per Section 18.1 Flex Field - Provisioning details.
- If validation required for any of the attributes in the flex field, configure / seed the metadata as per Section 18.3 Flex Field – Validation Support [Post dynamic provisioning & configuration screen, this will be enabled via configuration screen].
- If fact required, follow Section 18.2 Flex Field - Fact support for details.

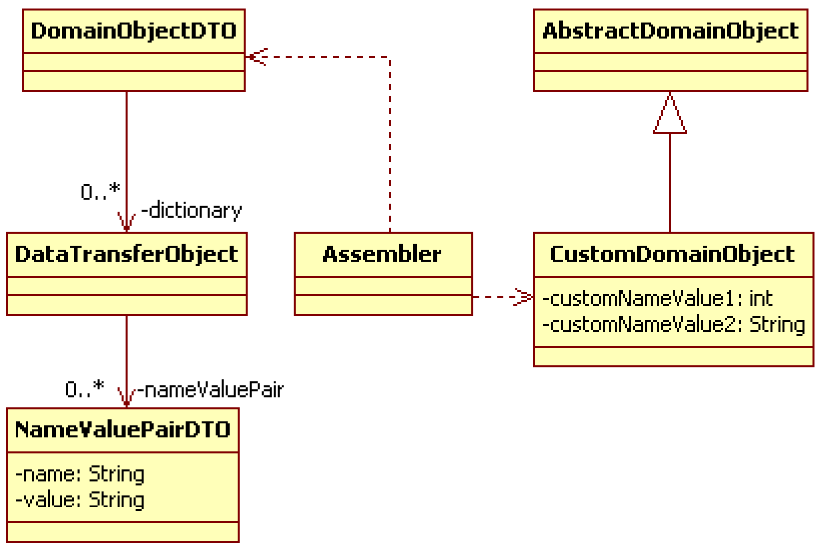
19 Extensibility of Domain Objects - Dictionary Pattern

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 19–1 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.

19.1 Customized Domain Object Attribute Placeholders

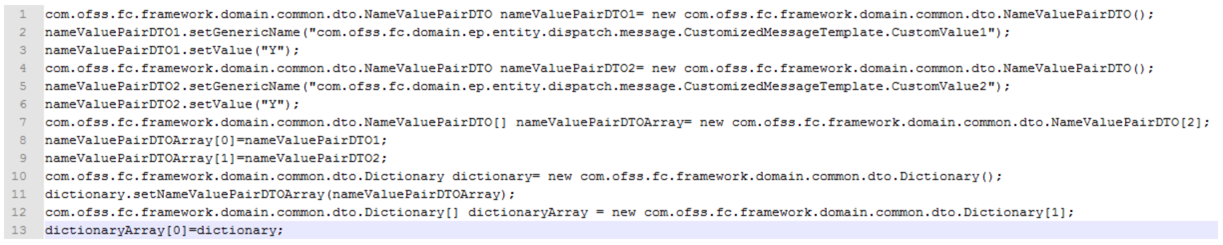
Data transfer object (DTO) is a design pattern used to transfer data between an external system and the application service. All the information may be wrapped in a single DTO containing all the details and passed as input request as well as returned as an output response. The client can then invoke accessor (or getter) methods on the DTO to get the individual attribute values from the Transfer Object. All request response classes in OBP application services are modelled as data transfer objects. These objects extend a base class DataTransferObject which holds an array of Dictionary object. The Dictionary encapsulates an array of NameValuePairDTO which is used to pass data of custom data fields or attributes from the UI layer to the host middleware. The following is mentioned as points below:

- All DTO classes should extend DomainObjectDTO class.
- The DomainObjectDTO class has been made to extend DataTransferObject class.
- This class has a single attribute which is an array of Dictionary class.
- Dictionary class has a single attribute which is an array of NameValuePairDTO

Using an array of name value pairs inside an array of dictionary allows for supporting two dimensional grid structures in the UI layer.

At present whenever any third party requires support for additional attributes in a Domain Object, the information regarding the corresponding Customized Domain Object name and attribute name-value pair is required to be populated as an array of NameValuePairDTO which in turn is set in the Dictionary class as the first and only element of the 'dictionaryArray' attribute of the DataTransferObject. This is shown in the following code extract.

Figure 19–2 Code Extract



```
com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePairDTO1= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO();
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;
```

19.2 Customized Domain Object DTO Interceptor in UI Layer

All DTO classes should extend DomainObjectDTO in case maintenance fields are required.

For example, ’MessageDataAttributeDTO’ Class which extends ’DomainObjectDTO’ is used to transfer data between an external system and the application service and persist data for Domain Object ’MessageDataAttribute’.

’CustomizedMessageDataAttribute’ is a subclass of this Customizable Maintenance Domain Object called ’MessageDataAttribute’ which is extended by the partners or consulting teams to include and subsequently persist extra attributes along with those of ’MessageDataAttribute’.

This information can be mapped as input and output to the application services with the help of dictionaryArray attribute of MessageDataAttributeDTO inherited from DataTransferObject.

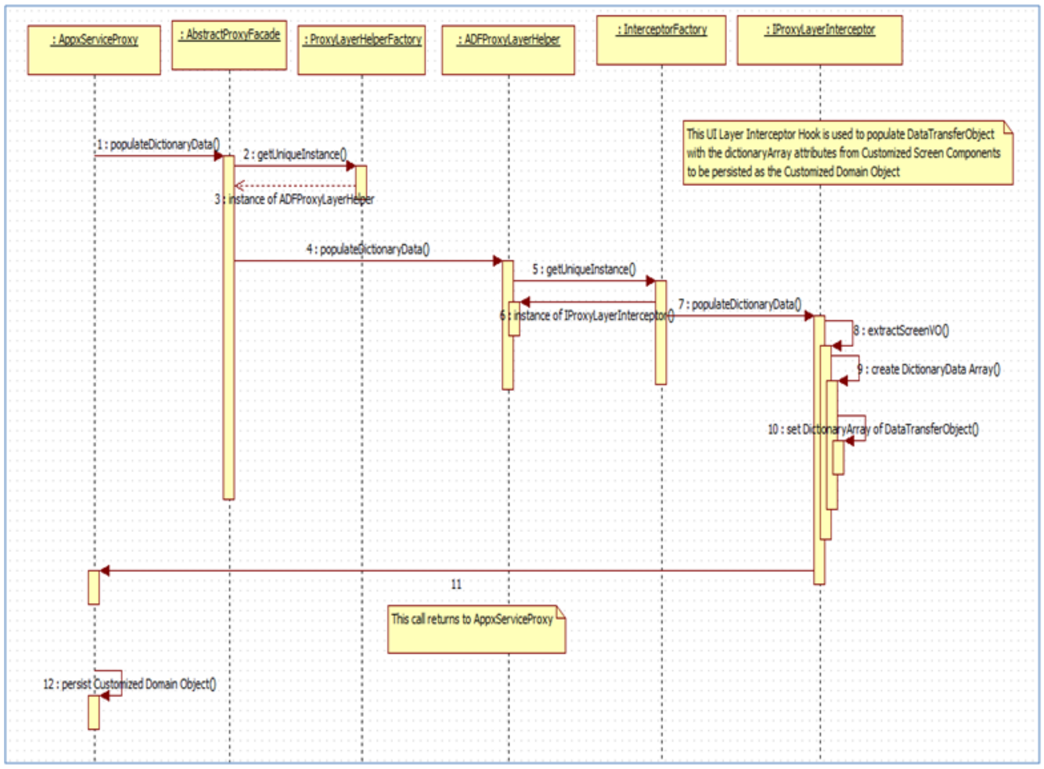
19.2.1 Interceptor Hook to Persist Customized Domain Object Attributes

This UI Layer Interceptor Hook is used during Create or Update mode to populate DataTransferObject with the dictionaryArray attributes from customized Screen Components to be persisted as the Customized Domain Object.

In the UI Layer, the ApplicationServiceProxyFacade is used to send the DataTransferObject on to the Host to be persisted. Before it does so, it uses the InterceptorFactory to instantiate the appropriate IProxyLayerInterceptor defined in the DictionaryInterceptor.properties corresponding to the key for this application service or task code. Thereafter it invokes the ’populateDictionaryArray’ method of this IProxyLayerInterceptor to populate DataTransferObject with the dictionaryArray attributes from customized Screen Components. Thereafter, it sends the entire DataTransferObject on to the Host for persistence as the Customized Domain Object.

The following figure provides the details of Interceptor Hook to populate and persist Customized Domain Object.

Figure 19–3 Interceptor Hook to Persist Customized Domain Object


: AppxServiceProxy
: AbstractProxyFacade
: ProxyLayerHelperFactory
: ADFProxyLayerHelper
: InterceptorFactory
: IProxyLayerInterceptor
This UI Layer Interceptor Hook is used to populate DataTransferObject with the dictionaryArray attributes from Customized Screen Components to be persisted as the Customized Domain Object
1 : populateDictionaryData()
2 : getUniqueInstance()
3 : instance of ADFProxyLayerHelper
4 : populateDictionaryData()
5 : getUniqueInstance()
6 : instance of IProxyLayerInterceptor()
7 : populateDictionaryData()
8 : extractScreenVO()
9 : create DictionaryData Array()
10 : set DictionaryArray of DataTransferObject()
11
This call returns to AppxServiceProxy
12 : persist Customized Domain Object()

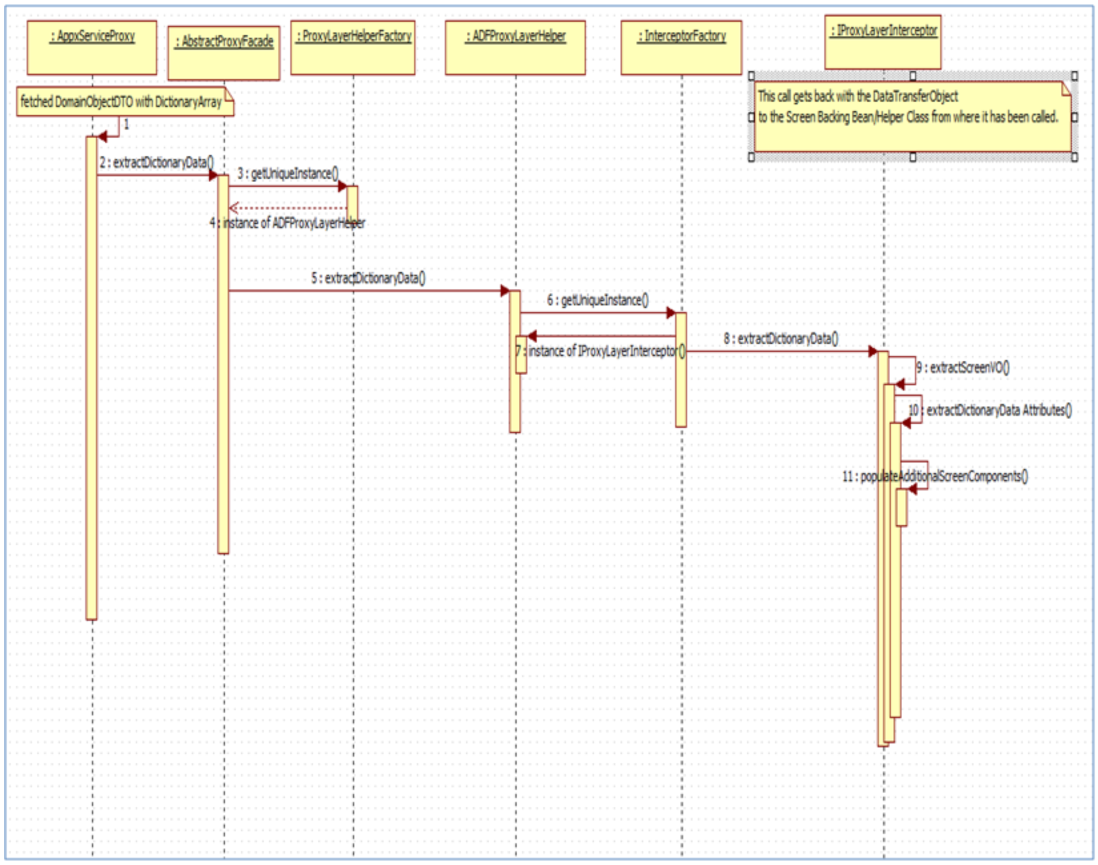
19.2.2 Interceptor Hook to Fetch Customized Domain Object Attributes

This UI Layer Interceptor Hook is used during read mode to extract the dictionaryArray attributes from the DataTransferObject and populate the customized Screen Components with the help of the screen view object.

In the UI Layer, the ApplicationServiceProxyFacade is used to receive the DataTransferObject from the Host. After it does so, it uses the InterceptorFactory to instantiate the appropriate IProxyLayerInterceptor defined in the DictionaryInterceptor.properties corresponding to the key for this application service or task code. Thereafter, it invokes the ’extractDictionaryArray’ method of this IProxyLayerInterceptor to extract the dictionaryArray attributes from the DataTransferObject and populate the customized Screen Components with the help of the screen view object. Thereafter, it returns the entire DataTransferObject on to the Screen Backing Bean or Helper Class from where the proxy fetch call was invoked.

The following figure provides the details of Interceptor Hook to fetch Customized Domain Object and populate extra Screen Components.

Figure 19–4 Interceptor Hook to Fetch Customized Domain Object


: AppxServiceProxy
: AbstractProxyFacade
: ProxyLayerHelperFactory
: ADFProxyLayerHelper
: InterceptorFactory
: IProxyLayerInterceptor
fetched DomainObjectDTO with DictionaryArray
This call gets back with the DataTransferObject to the Screen Backing Bean/Helper Class from where it has been called.
1
2 : extractDictionaryData()
3 : getUniqueInstance()
4 : instance of ADFProxyLayerHelper
5 : extractDictionaryData()
6 : getUniqueInstance()
7 : instance of IProxyLayerInterceptor()
8 : extractDictionaryData()
9 : extractScreenVO()
10 : extractDictionaryData Attributes()
11 : populateAdditionalScreenComponents()

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

19.3 Dictionary Data Transfer from UI to Host

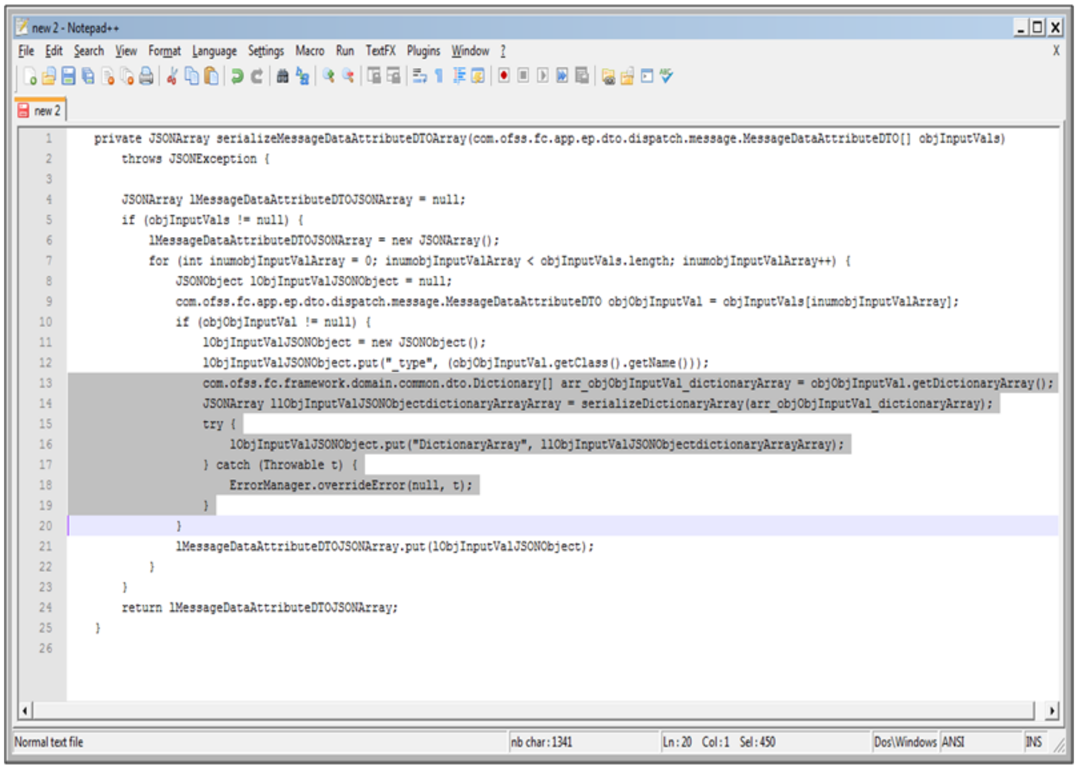
The section describes the dictionary data transfer from UI to Host.

19.3.1 Customized Domain Object DTO Transfer from UI to Host

In UI server <ApplicationService>JSONClient constructs the JSON Object for <DomainObjectDTO> which includes the dictionaryArray of the DataTransferObject.

For example, in UI server MessageTemplateApplicationServiceJSONClient constructs the JSON Object for MessageTemplateDTO which includes MessageTemplateAttributeDTO and the dictionaryArray of DataTransferObject as shown below.

Figure 19–5 JSONClient constructs the JSON Object

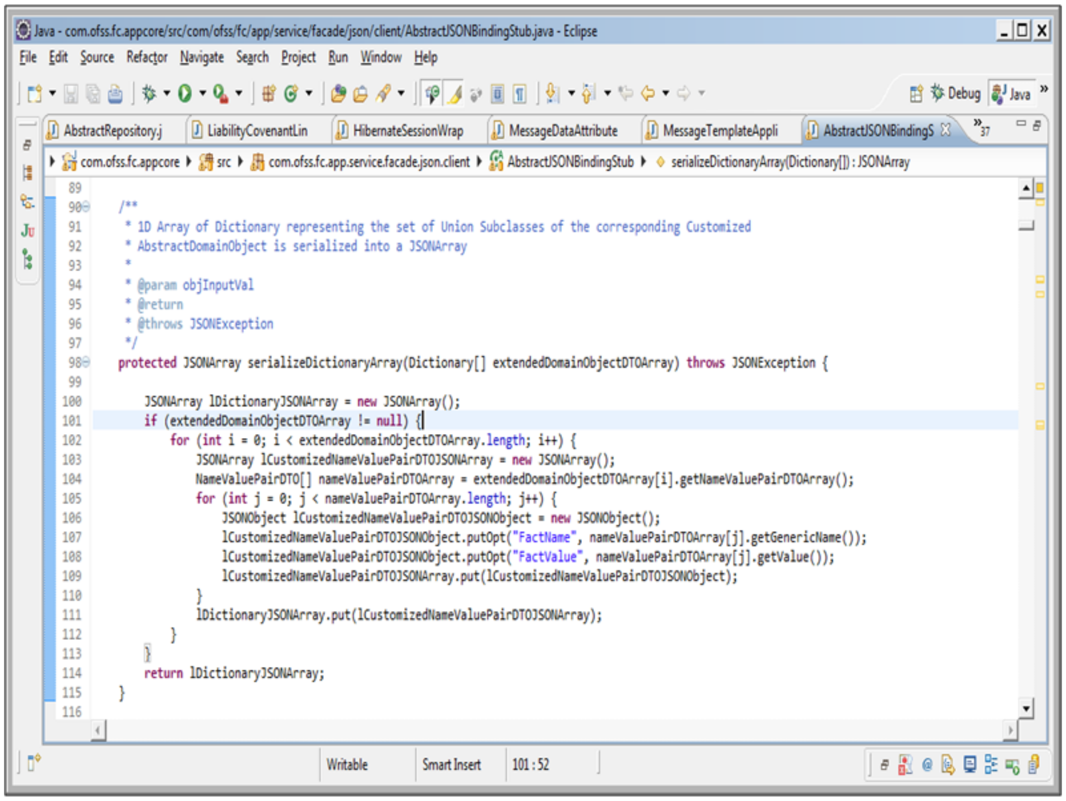

new 2 - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
new 2
Normal text file
nb char : 1341
Ln : 20 Col : 1 Sel : 450
Dos\Windows ANSI
INS

```
private JSONArray serializeMessageDataAttributeDTOArray(com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO[] objInputVals)
    throws JSONException {

    JSONArray lMessageDataAttributeDTOJSONArray = null;
    if (objInputVals != null) {
        lMessageDataAttributeDTOJSONArray = new JSONArray();
        for (int inumobjInputValArray = 0; inumobjInputValArray < objInputVals.length; inumobjInputValArray++) {
            JSONObject lObjInputValJSONObject = null;
            com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO objObjInputVal = objInputVals[inumobjInputValArray];
            if (objObjInputVal != null) {
                lObjInputValJSONObject = new JSONObject();
                lObjInputValJSONObject.put("_type", (objObjInputVal.getClass().getName()));
                com.ofss.fc.framework.domain.common.dto.Dictionary[] arr_objObjInputVal_dictionaryArray = objObjInputVal.getDictionaryArray();
                JSONArray llObjInputValJSONObjectdictionaryArrayArray = serializeDictionaryArray(arr_objObjInputVal_dictionaryArray);
                try {
                    lObjInputValJSONObject.put("DictionaryArray", llObjInputValJSONObjectdictionaryArrayArray);
                } catch (Throwable t) {
                    ErrorManager.overrideError(null, t);
                }
            }
            lMessageDataAttributeDTOJSONArray.put(lObjInputValJSONObject);
        }
    }
    return lMessageDataAttributeDTOJSONArray;
}
```

<ApplicationService>JSONClient constructs the JSON Object for <DomainObjectDTO> which includes the dictionaryArray of the DataTransferObject

The above process uses AbstractJSONBindingStub class' serializeDictionaryArray to include 'genericName' and 'value' attributes of NameValuePairDTOArray which was inside dictionaryArray attribute of MessageTemplateAttributeDTO.

Figure 19–6 SerializeDictionaryArray to include GenericName and Value attributes


Java - com.ofss.fc.appcore/src/com/ofss/fc/app/service/facade/json/client/AbstractJSONBindingStub.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug Java
AbstractRepository.j LiabilityCovenantLin HibernateSessionWrap MessageDataAttribute MessageTemplateAppli AbstractJSONBindingS
com.ofss.fc.appcore src com.ofss.fc.app.service.facade.json.client AbstractJSONBindingStub serializeDictionaryArray(Dictionary[]) : JSONArray
Writable Smart Insert 101 : 52

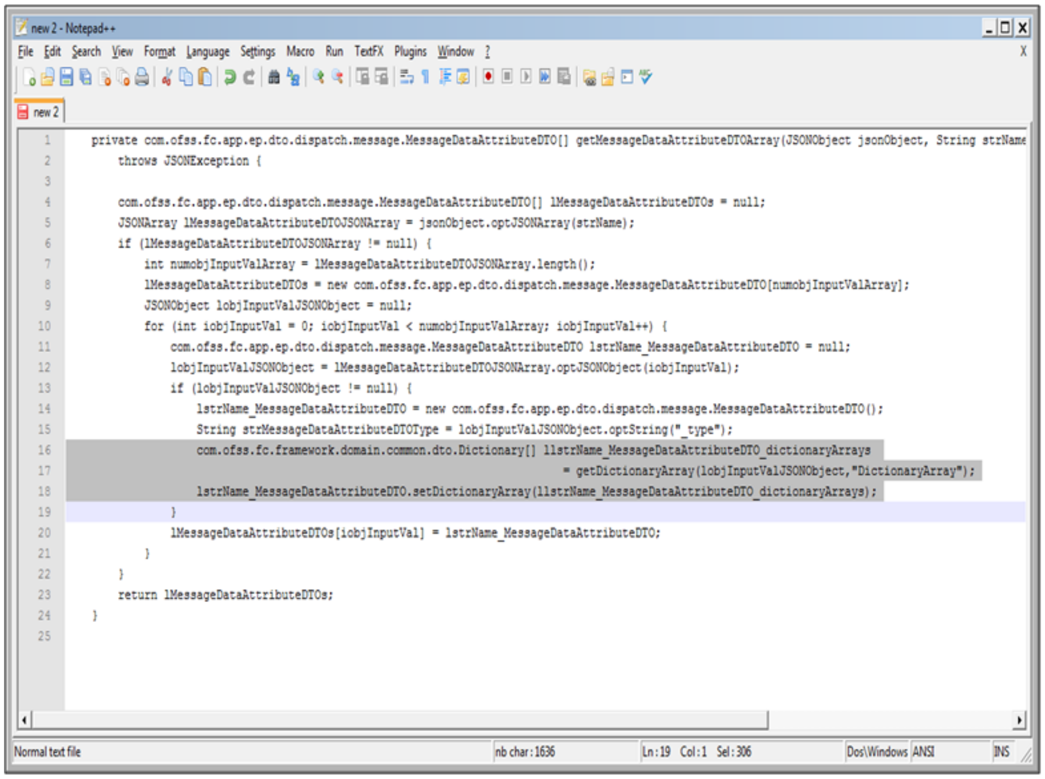
```
    /**
     * 1D Array of Dictionary representing the set of Union Subclasses of the corresponding Customized
     * AbstractDomainObject is serialized into a JSONArray
     *
     * @param objInputVal
     * @return
     * @throws JSONException
     */
    protected JSONArray serializeDictionaryArray(Dictionary[] extendedDomainObjectDTOArray) throws JSONException {

        JSONArray lDictionaryJSONArray = new JSONArray();
        if (extendedDomainObjectDTOArray != null) {
            for (int i = 0; i < extendedDomainObjectDTOArray.length; i++) {
                JSONArray lCustomizedNameValuePairDTOJSONArray = new JSONArray();
                NameValuePairDTO[] nameValuePairDTOArray = extendedDomainObjectDTOArray[i].getNameValuePairDTOArray();
                for (int j = 0; j < nameValuePairDTOArray.length; j++) {
                    JSONObject lCustomizedNameValuePairDTOJSONObject = new JSONObject();
                    lCustomizedNameValuePairDTOJSONObject.putOpt("FactName", nameValuePairDTOArray[j].getGenericName());
                    lCustomizedNameValuePairDTOJSONObject.putOpt("FactValue", nameValuePairDTOArray[j].getValue());
                    lCustomizedNameValuePairDTOJSONArray.put(lCustomizedNameValuePairDTOJSONObject);
                }
                lDictionaryJSONArray.put(lCustomizedNameValuePairDTOJSONArray);
            }
        }
        return lDictionaryJSONArray;
    }
```

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

In the Host Server <ApplicationService>JSONFacade extracts the 'DictionaryArray' attribute of JSON Object and sets it as <DomainObjectDTO>'s dictionaryArray attribute.

For example, in the Host Server, MessageTemplateApplicationServiceJSONFacade extracts the 'DictionaryArray' attribute of JSON Object and sets it as MessageDataAttributeDTO's dictionaryArray attribute.

Figure 19–7 Host Server JSONFacade extracts the attribute of JSON Object



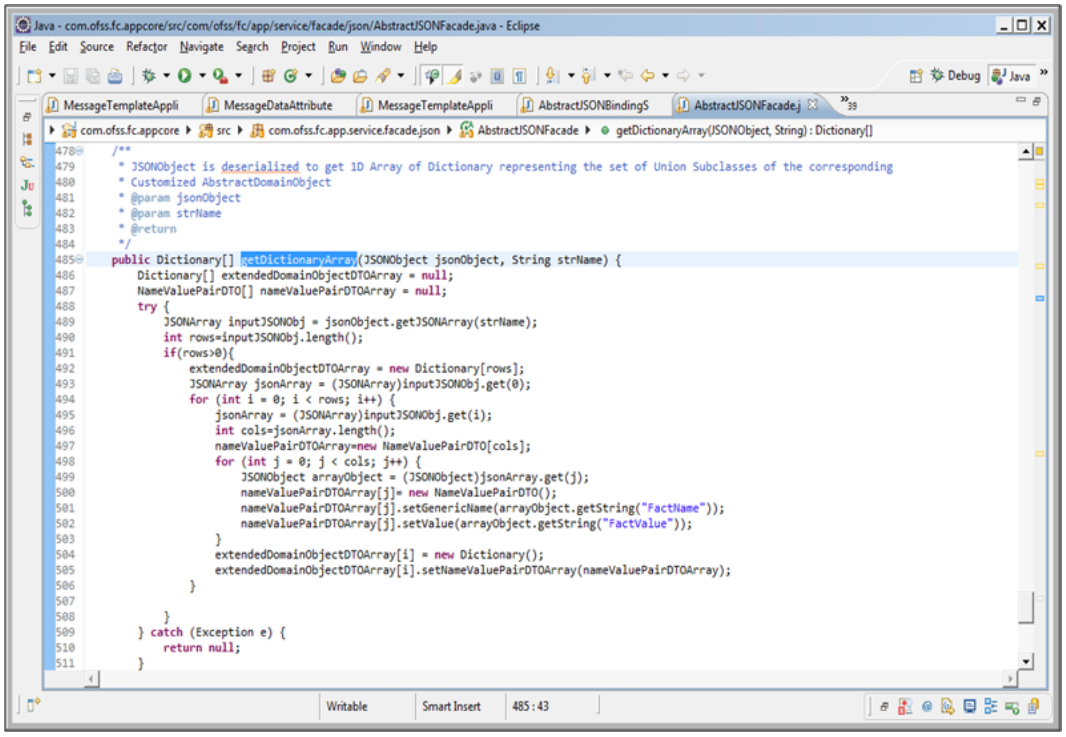
```
private com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO[] getMessageDataAttributeDTOArray(JSONObject jsonObject, String strName
    throws JSONException {

    com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO[] lMessageDataAttributeDTOs = null;
    JSONArray lMessageDataAttributeDTOJSONArray = jsonObject.optJSONArray(strName);
    if (lMessageDataAttributeDTOJSONArray != null) {
        int numobjInputValArray = lMessageDataAttributeDTOJSONArray.length();
        lMessageDataAttributeDTOs = new com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO[numobjInputValArray];
        JSONObject lobjInputValJSONObject = null;
        for (int iobjInputVal = 0; iobjInputVal < numobjInputValArray; iobjInputVal++) {
            com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO lstrName_MessageDataAttributeDTO = null;
            lobjInputValJSONObject = lMessageDataAttributeDTOJSONArray.optJSONObject(iobjInputVal);
            if (lobjInputValJSONObject != null) {
                lstrName_MessageDataAttributeDTO = new com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO();
                String strMessageDataAttributeDTOType = lobjInputValJSONObject.optString("_type");
                com.ofss.fc.framework.domain.common.dto.Dictionary[] llstrName_MessageDataAttributeDTO_dictionaryArrays
                                                          = getDictionaryArray(lobjInputValJSONObject,"DictionaryArray");
                lstrName_MessageDataAttributeDTO.setDictionaryArray(llstrName_MessageDataAttributeDTO_dictionaryArrays);
            }
            lMessageDataAttributeDTOs[iobjInputVal] = lstrName_MessageDataAttributeDTO;
        }
    }
    return lMessageDataAttributeDTOs;
}
```

Normal text file   nb char : 1636   Ln : 19   Col : 1   Sel : 306   Dos\Windows   ANSI   INS

In the Host Server <ApplicationService>JSONFacade extracts the "DictionaryArray" attribute of JSON Object and sets it as <DomainObjectDTO>'s dictionaryArray attribute.

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

Figure 19–8 AbstractJSONFacade's getDictionaryArray method


Java - com.ofss.fc.appcore/src/com/ofss/fc/app/service/facade/json/AbstractJSONFacade.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Java
MessageTemplateAppli
MessageDataAttribute
MessageTemplateAppli
AbstractJSONBindingS
AbstractJSONFacade.j
com.ofss.fc.appcore
src
com.ofss.fc.app.service.facade.json
AbstractJSONFacade
getDictionaryArray(JSONObject, String) : Dictionary[]
Writable
Smart Insert
485 : 43

```
    /**
     * JSONObject is deserialized to get 1D Array of Dictionary representing the set of Union Subclasses of the corresponding
     * Customized AbstractDomainObject
     * @param jsonObject
     * @param strName
     * @return
     */
    public Dictionary[] getDictionaryArray(JSONObject jsonObject, String strName) {
        Dictionary[] extendedDomainObjectDTOArray = null;
        NameValuePairDTO[] nameValuePairDTOArray = null;
        try {
            JSONArray inputJSONObj = jsonObject.getJSONArray(strName);
            int rows=inputJSONObj.length();
            if(rows>0){
                extendedDomainObjectDTOArray = new Dictionary[rows];
                JSONArray jsonArray = (JSONArray)inputJSONObj.get(0);
                for (int i = 0; i < rows; i++) {
                    jsonArray = (JSONArray)inputJSONObj.get(i);
                    int cols=jsonArray.length();
                    nameValuePairDTOArray=new NameValuePairDTO[cols];
                    for (int j = 0; j < cols; j++) {
                        JSONObject arrayObject = (JSONObject)jsonArray.get(j);
                        nameValuePairDTOArray[j]= new NameValuePairDTO();
                        nameValuePairDTOArray[j].setGenericName(arrayObject.getString("FactName"));
                        nameValuePairDTOArray[j].setValue(arrayObject.getString("FactValue"));
                    }
                    extendedDomainObjectDTOArray[i] = new Dictionary();
                    extendedDomainObjectDTOArray[i].setNameValuePairDTOArray(nameValuePairDTOArray);
                }

            }
        } catch (Exception e) {
            return null;
        }
```

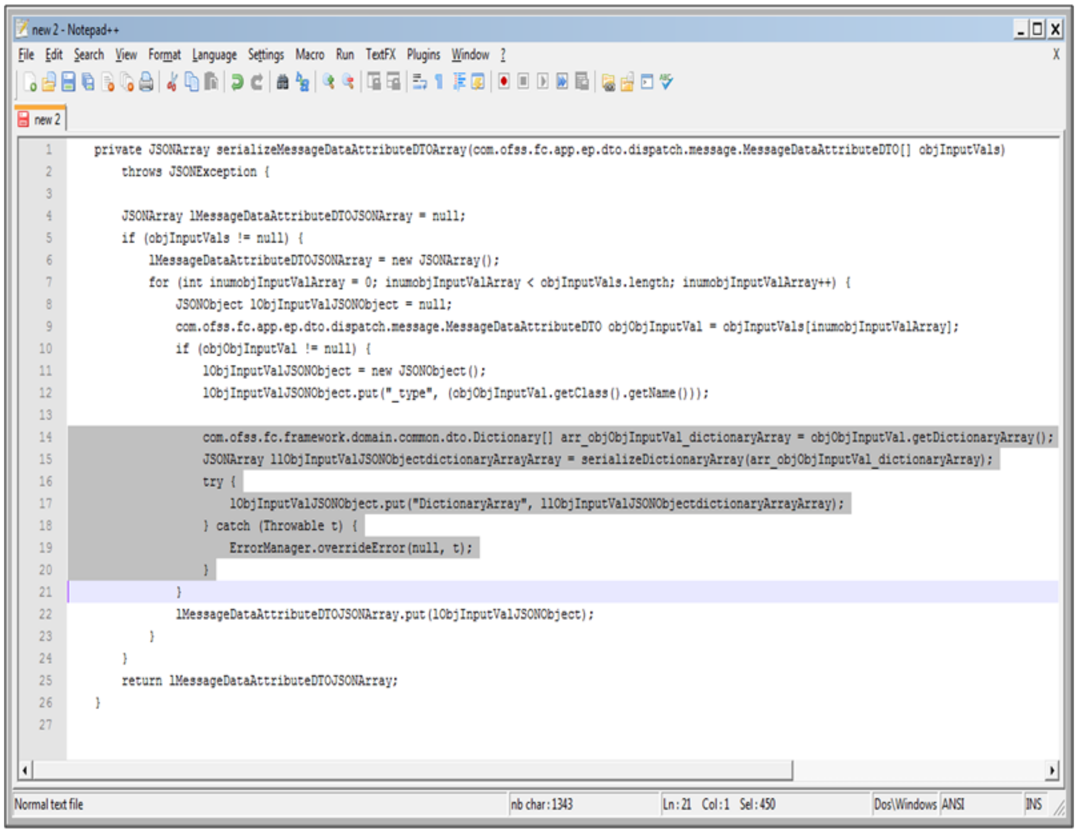
AbstractJSONFacade's getDictionaryArray method that unmarshalls the "genericName" and "value" from JSON Object to get the dictionaryArray attribute

19.3.2 Customized Domain Object DTO transfer from Host to UI

In the Host Server <ApplicationService>JSONFacade constructs the JSON Object for <DomainObjectDTO> and the dictionaryArray of DataTransferObject

MessageTemplateApplicationServiceJSONFacade's method serializeMessageDataAttributeDTOArray in Host Server constructs the JSON Object for MessageTemplateDTO which includes MessageTemplateAttributeDTO and the dictionaryArray of DataTransferObject as shown below:

Figure 19–9 Host Server JSONFacade constructs the JSON Object

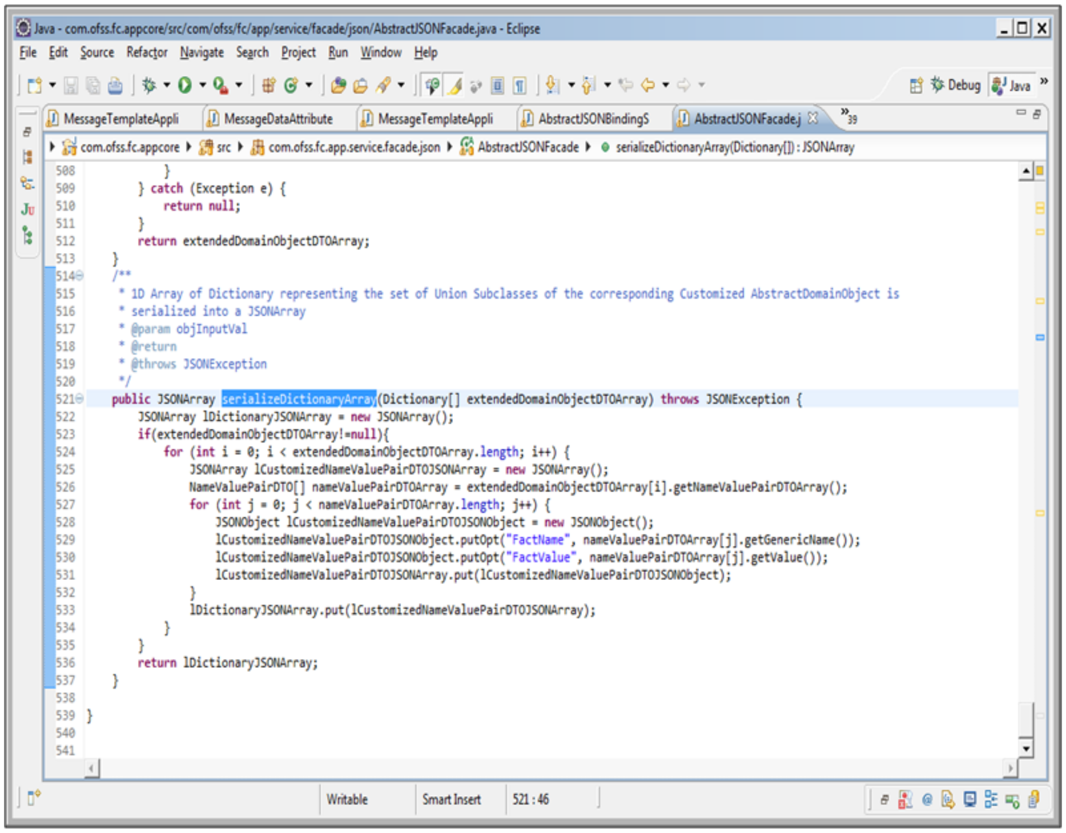


```
private JSONArray serializeMessageDataAttributeDTOArray(com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO[] objInputVals)
    throws JSONException {

    JSONArray lMessageDataAttributeDTOJSONArray = null;
    if (objInputVals != null) {
        lMessageDataAttributeDTOJSONArray = new JSONArray();
        for (int inumobjInputValArray = 0; inumobjInputValArray < objInputVals.length; inumobjInputValArray++) {
            JSONObject lObjInputValJSONObject = null;
            com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO objObjInputVal = objInputVals[inumobjInputValArray];
            if (objObjInputVal != null) {
                lObjInputValJSONObject = new JSONObject();
                lObjInputValJSONObject.put("_type", (objObjInputVal.getClass().getName()));

                com.ofss.fc.framework.domain.common.dto.Dictionary[] arr_objObjInputVal_dictionaryArray = objObjInputVal.getDictionaryArray();
                JSONArray llObjInputValJSONObjectdictionaryArrayArray = serializeDictionaryArray(arr_objObjInputVal_dictionaryArray);
                try {
                    lObjInputValJSONObject.put("DictionaryArray", llObjInputValJSONObjectdictionaryArrayArray);
                } catch (Throwable t) {
                    ErrorManager.overrideError(null, t);
                }
            }
            lMessageDataAttributeDTOJSONArray.put(lObjInputValJSONObject);
        }
    }
    return lMessageDataAttributeDTOJSONArray;
}
```

In the Host Server <ApplicationService>JSONFacade constructs the JSON Object for <DomainObjectDTO> and the dictionaryArray of DataTransferObject

The above process uses AbstractJSONFacade's serializeDictionaryArray to include ’genericName’ and ’value’ attributes of NameValuePairDTOArray which was inside dictionaryArray attribute of MessageTemplateAttributeDTO.

Figure 19–10 AbstractJSONFacade's serializeDictionaryArray to include Generic Name and Value attributes


Java - com.ofss.fc.appcore/src/com/ofss/fc/app/service/facade/json/AbstractJSONFacade.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug Java
MessageTemplateAppli MessageDataAttribute MessageTemplateAppli AbstractJSONBindingS AbstractJSONFacade.j
com.ofss.fc.appcore src com.ofss.fc.app.service.facade.json AbstractJSONFacade serializeDictionaryArray(Dictionary[]) : JSONArray
Writable Smart Insert 521 : 46

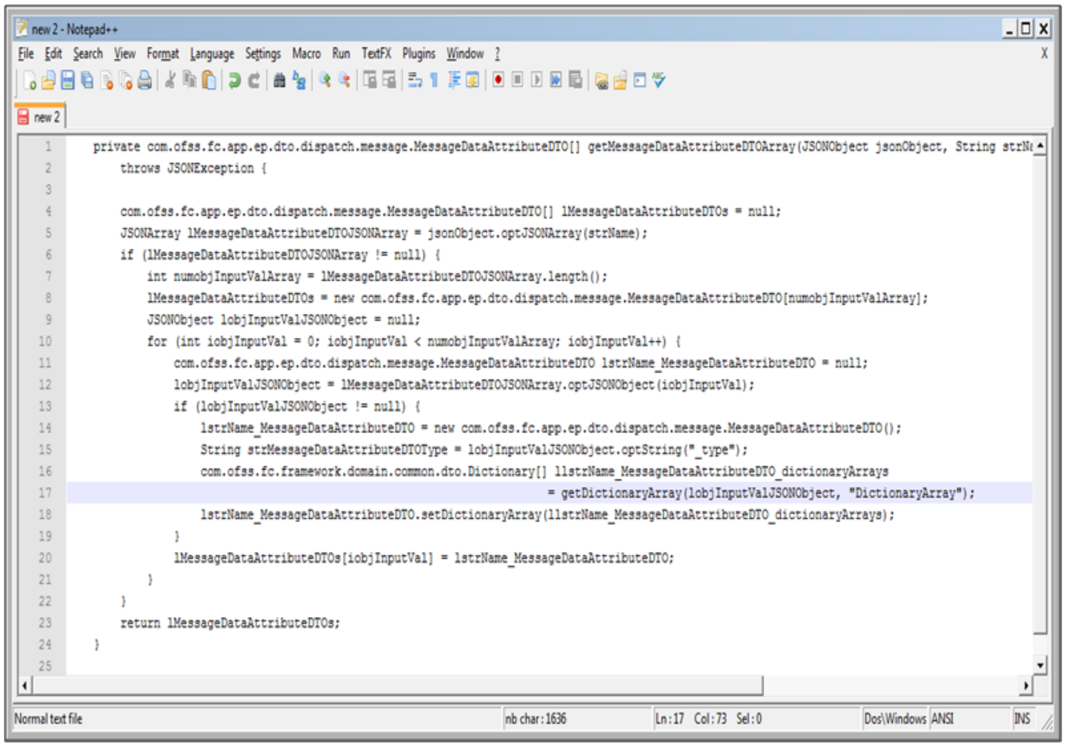
```
            }
        } catch (Exception e) {
            return null;
        }
        return extendedDomainObjectDTOArray;
    }
    /**
     * 1D Array of Dictionary representing the set of Union Subclasses of the corresponding Customized AbstractDomainObject is
     * serialized into a JSONArray
     * @param objInputVal
     * @return
     * @throws JSONException
     */
    public JSONArray serializeDictionaryArray(Dictionary[] extendedDomainObjectDTOArray) throws JSONException {
        JSONArray lDictionaryJSONArray = new JSONArray();
        if(extendedDomainObjectDTOArray!=null){
            for (int i = 0; i < extendedDomainObjectDTOArray.length; i++) {
                JSONArray lCustomizedNameValuePairDTOJSONArray = new JSONArray();
                NameValuePairDTO[] nameValuePairDTOArray = extendedDomainObjectDTOArray[i].getNameValuePairDTOArray();
                for (int j = 0; j < nameValuePairDTOArray.length; j++) {
                    JSONObject lCustomizedNameValuePairDTOJSONObject = new JSONObject();
                    lCustomizedNameValuePairDTOJSONObject.putOpt("FactName", nameValuePairDTOArray[j].getGenericName());
                    lCustomizedNameValuePairDTOJSONObject.putOpt("FactValue", nameValuePairDTOArray[j].getValue());
                    lCustomizedNameValuePairDTOJSONArray.put(lCustomizedNameValuePairDTOJSONObject);
                }
                lDictionaryJSONArray.put(lCustomizedNameValuePairDTOJSONArray);
            }
        }
        return lDictionaryJSONArray;
    }

}
```

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

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

In the UI Server, MessageTemplateApplicationServiceJSONClient extracts the ’DictionaryArray’ attribute of JSON Object and sets it as MessageDataAttributeDTO's dictionaryArray attribute.

Figure 19–11 UI Server JSONClient extracts the DictionaryArray attribute

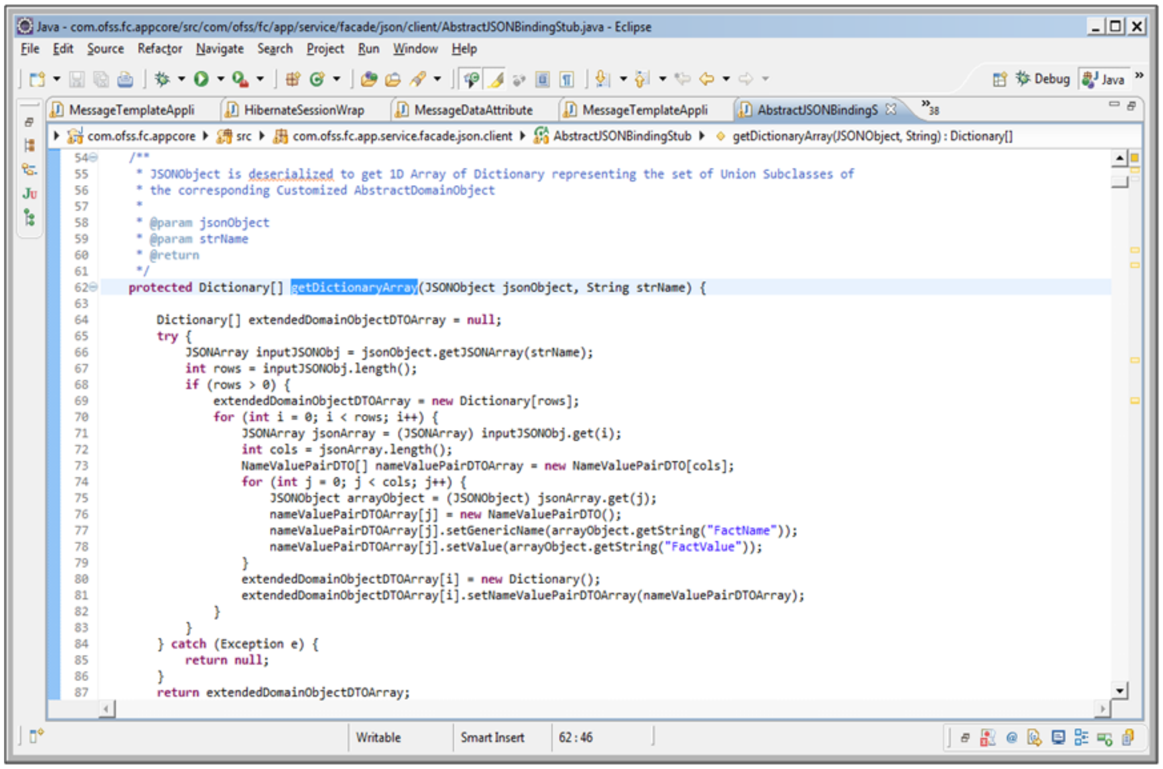


```
    private com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO[] getMessageDataAttributeDTOArray(JSONObject jsonObject, String strNa
        throws JSONException {

        com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO[] lMessageDataAttributeDTOs = null;
        JSONArray lMessageDataAttributeDTOJSONArray = jsonObject.optJSONArray(strName);
        if (lMessageDataAttributeDTOJSONArray != null) {
            int numobjInputValArray = lMessageDataAttributeDTOJSONArray.length();
            lMessageDataAttributeDTOs = new com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO[numobjInputValArray];
            JSONObject lobjInputValJSONObject = null;
            for (int iobjInputVal = 0; iobjInputVal < numobjInputValArray; iobjInputVal++) {
                com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO lstrName_MessageDataAttributeDTO = null;
                lobjInputValJSONObject = lMessageDataAttributeDTOJSONArray.optJSONObject(iobjInputVal);
                if (lobjInputValJSONObject != null) {
                    lstrName_MessageDataAttributeDTO = new com.ofss.fc.app.ep.dto.dispatch.message.MessageDataAttributeDTO();
                    String strMessageDataAttributeDTOType = lobjInputValJSONObject.optString("_type");
                    com.ofss.fc.framework.domain.common.dto.Dictionary[] llstrName_MessageDataAttributeDTO_dictionaryArrays
                                                                            = getDictionaryArray(lobjInputValJSONObject, "DictionaryArray");
                    lstrName_MessageDataAttributeDTO.setDictionaryArray(llstrName_MessageDataAttributeDTO_dictionaryArrays);
                }
                lMessageDataAttributeDTOs[iobjInputVal] = lstrName_MessageDataAttributeDTO;
            }
        }
        return lMessageDataAttributeDTOs;
    }

```

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 19–12 AbstractJSONBindingStub's getDictionaryArray method


Java - com.ofss.fc.appcore/src/com/ofss/fc/app/service/facade/json/client/AbstractJSONBindingStub.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Java
MessageTemplateAppli
HibernateSessionWrap
MessageDataAttribute
MessageTemplateAppli
AbstractJSONBindingS
com.ofss.fc.appcore
src
com.ofss.fc.app.service.facade.json.client
AbstractJSONBindingStub
getDictionaryArray(JSONObject, String) : Dictionary[]
Writable
Smart Insert
62 : 46

```java
    /**
     * JSONObject is deserialized to get 1D Array of Dictionary representing the set of Union Subclasses of
     * the corresponding Customized AbstractDomainObject
     *
     * @param jsonObject
     * @param strName
     * @return
     */
    protected Dictionary[] getDictionaryArray(JSONObject jsonObject, String strName) {

        Dictionary[] extendedDomainObjectDTOArray = null;
        try {
            JSONArray inputJSONObj = jsonObject.getJSONArray(strName);
            int rows = inputJSONObj.length();
            if (rows > 0) {
                extendedDomainObjectDTOArray = new Dictionary[rows];
                for (int i = 0; i < rows; i++) {
                    JSONArray jsonArray = (JSONArray) inputJSONObj.get(i);
                    int cols = jsonArray.length();
                    NameValuePairDTO[] nameValuePairDTOArray = new NameValuePairDTO[cols];
                    for (int j = 0; j < cols; j++) {
                        JSONObject arrayObject = (JSONObject) jsonArray.get(j);
                        nameValuePairDTOArray[j] = new NameValuePairDTO();
                        nameValuePairDTOArray[j].setGenericName(arrayObject.getString("FactName"));
                        nameValuePairDTOArray[j].setValue(arrayObject.getString("FactValue"));
                    }
                    extendedDomainObjectDTOArray[i] = new Dictionary();
                    extendedDomainObjectDTOArray[i].setNameValuePairDTOArray(nameValuePairDTOArray);
                }
            }
        } catch (Exception e) {
            return null;
        }
        return extendedDomainObjectDTOArray;
```

AbstractJSONBindingStub's getDictionaryArray method that unmarshalls the "genericName" and "value" from JSON Object

The provision of marshalling and un-marshalling of 'dictionaryArray' attribute of all DataTransferObjects has been included in the JSON layer for all application services.

19.4 Translating Dictionary Data into Custom Domain Object

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

19.4.1 Instantiation and Persistence of Custom Domain Objects

If a method has an input parameter that is a DataTransferObject, the first line of the method in the assembler will be of the form:

(populateDataTransferObjectDTOMap('Fully Qualified Name of this DataTransferObject>', dataTransferObject);

This method is defined in AbstractAssembler.java which newly instantiates referenceDataTransferObjectDTOMap if required and populates the map with the above entry.

This map is used as a set of globally available DataTransferObject's which can be retrieved by invoking another method defined in AbstractAssembler.java which is of the form:

```
retrieveDataTransferObjectDTOMapElement('<Fully Qualified Name of
this DataTransferObject >');
```

Whenever any AbstractDomainObject is instantiated, the Customized AbstractDomainObject should be instantiated instead of the original AbstractDomainObject wherever applicable.

The AbstractDomainObject is instantiated with the help of the below code fragment:

```
IAbstractDomainObject domainObject=null;
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>;
}
```

19.4.2 Fetching of Customized Domain Objects

If a method has an input parameter that is an IAbstractDomainObject, the first line of the method in the assembler will be of the form:

```
populateAbstractDomainObjectMap("<Fully_Qualified_Name_
IAbstractDomainObject>", abstractDomainObject);
```

This method is defined in AbstractAssembler.java which newly instantiates referenceAbstractDomainObjectMap if required and populates the map with the above entry.

This map is used as a set of globally available IAbstractDomainObject's which can be retrieved by invoking another method defined in AbstractAssembler.java which is of the form:

```
retrieveDataTransferObjectDTOMapElement("<Fully_Qualified_Name_
IAbstractDomainObject>");
```

Whenever any DataTransferObject is instantiated, we populate its ’dictionaryArray’ attribute immediately after it's instantiation.

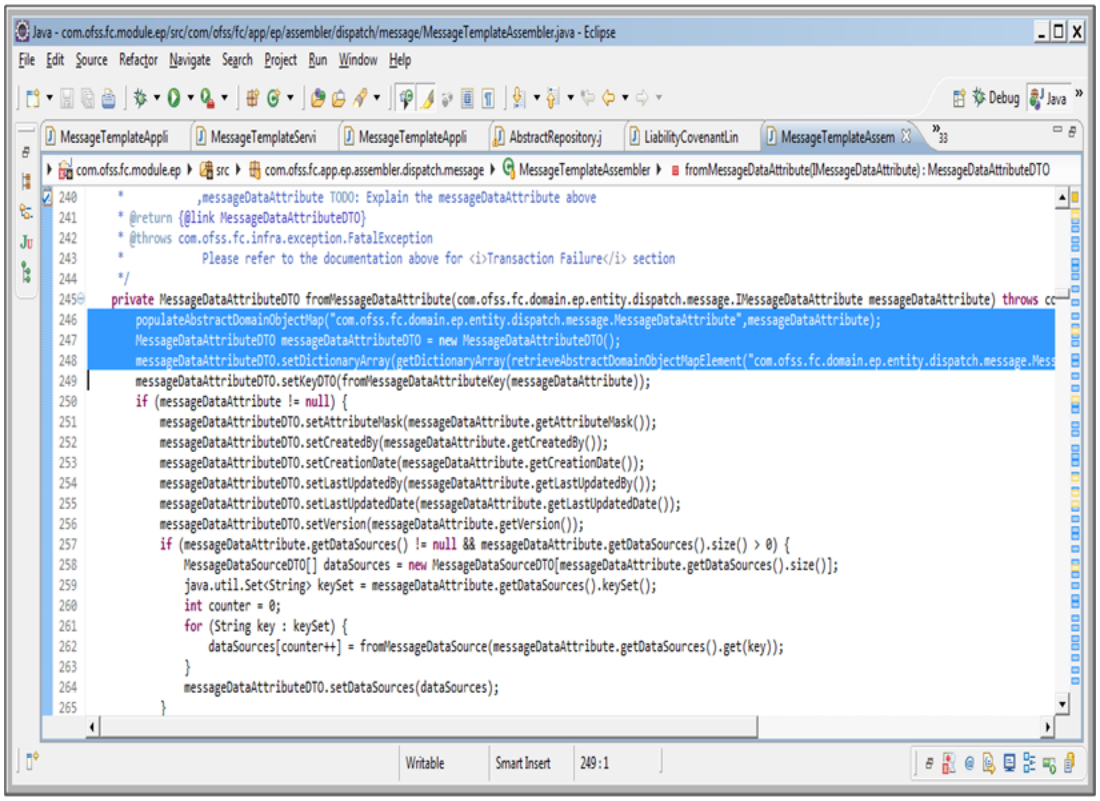
In AbstractAssembler.java, we have defined the method à

```
public Dictionary[] getDictionaryArray(IAbstractDomainObject obj)
```

This method creates and returns a dictionary array from the IAbstractDomainObject passed to it as input parameter.

Example of final piece of code:

Figure 19–13 Instantiation of DataTransferObjects


Java - com.ofss.fc.module.ep/src/com/ofss/fc/app/ep/assembler/dispatch/message/MessageTemplateAssembler.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Java
MessageTemplateAppli
MessageTemplateServi
MessageTemplateAppli
AbstractRepository.j
LiabilityCovenantLin
MessageTemplateAssem
com.ofss.fc.module.ep ▸ src ▸ com.ofss.fc.app.ep.assembler.dispatch.message ▸ MessageTemplateAssembler ▸ fromMessageDataAttribute(IMessageDataAttribute) : MessageDataAttributeDTO
Writable
Smart Insert
249 : 1

```
     *          ,messageDataAttribute TODO: Explain the messageDataAttribute above
     * @return {@link MessageDataAttributeDTO}
     * @throws com.ofss.fc.infra.exception.FatalException
     *           Please refer to the documentation above for <i>Transaction Failure</i> section
     */
    private MessageDataAttributeDTO fromMessageDataAttribute(com.ofss.fc.domain.ep.entity.dispatch.message.IMessageDataAttribute messageDataAttribute) throws cc
        populateAbstractDomainObjectMap("com.ofss.fc.domain.ep.entity.dispatch.message.MessageDataAttribute",messageDataAttribute);
        MessageDataAttributeDTO messageDataAttributeDTO = new MessageDataAttributeDTO();
        messageDataAttributeDTO.setDictionaryArray(getDictionaryArray(retrieveAbstractDomainObjectMapElement("com.ofss.fc.domain.ep.entity.dispatch.message.Mess
        messageDataAttributeDTO.setKeyDTO(fromMessageDataAttributeKey(messageDataAttribute));
        if (messageDataAttribute != null) {
            messageDataAttributeDTO.setAttributeMask(messageDataAttribute.getAttributeMask());
            messageDataAttributeDTO.setCreatedBy(messageDataAttribute.getCreatedBy());
            messageDataAttributeDTO.setCreationDate(messageDataAttribute.getCreationDate());
            messageDataAttributeDTO.setLastUpdatedBy(messageDataAttribute.getLastUpdatedBy());
            messageDataAttributeDTO.setLastUpdatedDate(messageDataAttribute.getLastUpdatedDate());
            messageDataAttributeDTO.setVersion(messageDataAttribute.getVersion());
            if (messageDataAttribute.getDataSources() != null && messageDataAttribute.getDataSources().size() > 0) {
                MessageDataSourceDTO[] dataSources = new MessageDataSourceDTO[messageDataAttribute.getDataSources().size()];
                java.util.Set<String> keySet = messageDataAttribute.getDataSources().keySet();
                int counter = 0;
                for (String key : keySet) {
                    dataSources[counter++] = fromMessageDataSource(messageDataAttribute.getDataSources().get(key));
                }
                messageDataAttributeDTO.setDataSources(dataSources);
            }
```

19.4.3 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.

19.5 Customized Domain Object ORM Configuration

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

19.5.1 Case 1 - Non-Inheritance based mapping

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

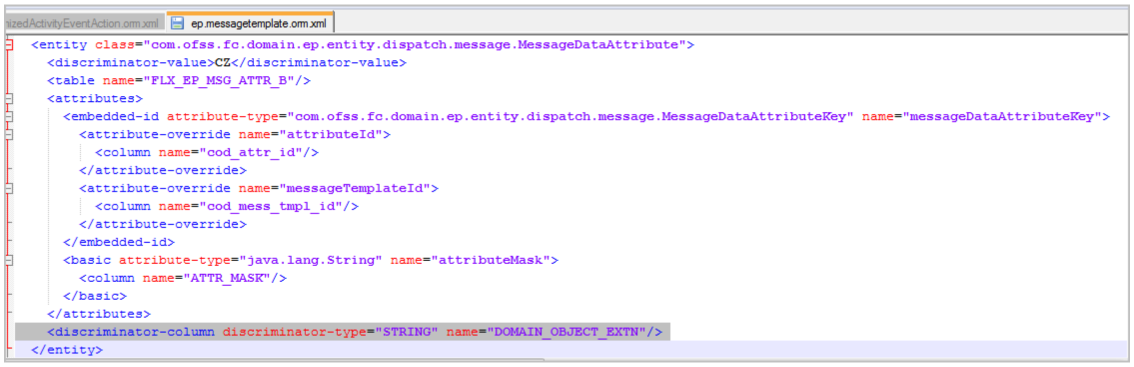
Adding Discriminator column mapping in existing ORM file

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 19–14 Adding Discriminator Column Mapping in Existing ORM file

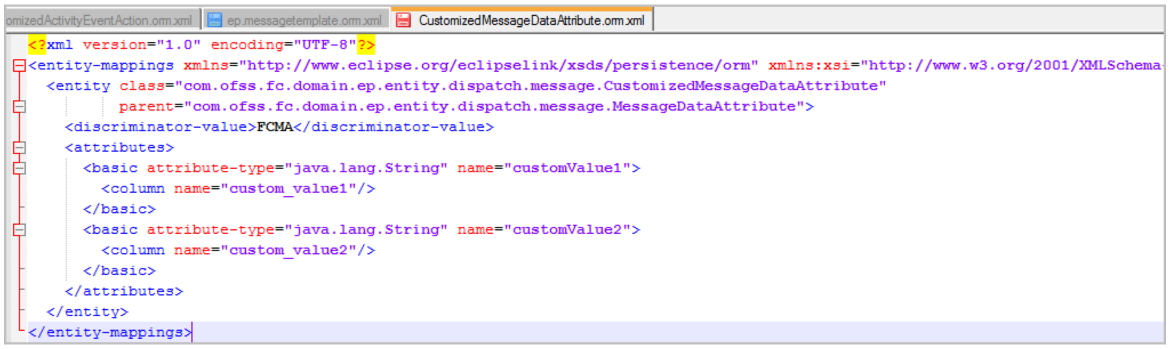


```
<entity class="com.ofss.fc.domain.ep.entity.dispatch.message.MessageDataAttribute">
  <discriminator-value>CZ</discriminator-value>
  <table name="FLX_EP_MSG_ATTR_B"/>
  <attributes>
    <embedded-id attribute-type="com.ofss.fc.domain.ep.entity.dispatch.message.MessageDataAttributeKey" name="messageDataAttributeKey">
      <attribute-override name="attributeId">
        <column name="cod_attr_id"/>
      </attribute-override>
      <attribute-override name="messageTemplateId">
        <column name="cod_mess_tmpl_id"/>
      </attribute-override>
    </embedded-id>
    <basic attribute-type="java.lang.String" name="attributeMask">
      <column name="ATTR_MASK"/>
    </basic>
  </attributes>
  <discriminator-column discriminator-type="STRING" name="DOMAIN_OBJECT_EXTN"/>
</entity>
```

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 19–15 ORM File Mapping to Customized Domain Object



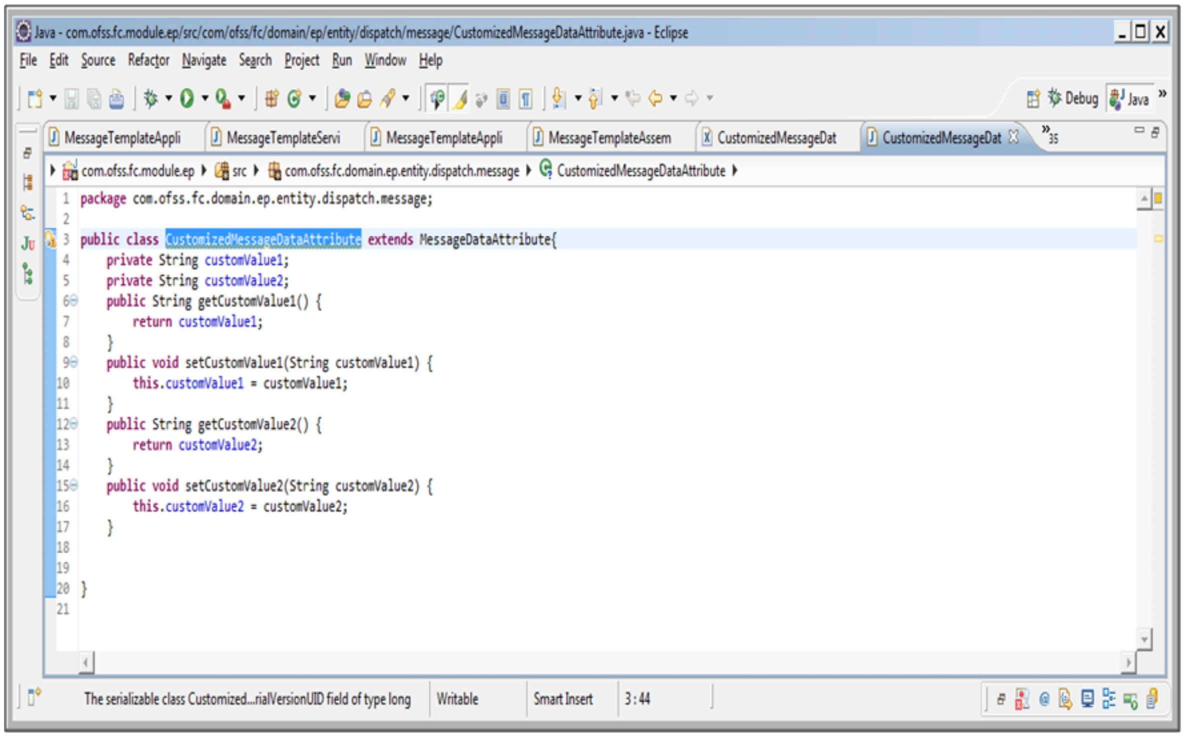
```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings xmlns="http://www.eclipse.org/eclipselink/xsds/persistence/orm" xmlns:xsi="http://www.w3.org/2001/XMLSchema
  <entity class="com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageDataAttribute"
          parent="com.ofss.fc.domain.ep.entity.dispatch.message.MessageDataAttribute">
    <discriminator-value>FCMA</discriminator-value>
    <attributes>
      <basic attribute-type="java.lang.String" name="customValue1">
        <column name="custom_value1"/>
      </basic>
      <basic attribute-type="java.lang.String" name="customValue2">
        <column name="custom_value2"/>
      </basic>
    </attributes>
  </entity>
</entity-mappings>
```

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 19–16 Adding New Java File to the Customized Domain Object


Java - com.ofss.fc.module.ep/src/com/ofss/fc/domain/ep/entity/dispatch/message/CustomizedMessageDataAttribute.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Java
MessageTemplateAppli
MessageTemplateServi
MessageTemplateAppli
MessageTemplateAssem
CustomizedMessageDat
CustomizedMessageDat
com.ofss.fc.module.ep
src
com.ofss.fc.domain.ep.entity.dispatch.message
CustomizedMessageDataAttribute

```java
package com.ofss.fc.domain.ep.entity.dispatch.message;

public class CustomizedMessageDataAttribute extends MessageDataAttribute{
    private String customValue1;
    private String customValue2;
    public String getCustomValue1() {
        return customValue1;
    }
    public void setCustomValue1(String customValue1) {
        this.customValue1 = customValue1;
    }
    public String getCustomValue2() {
        return customValue2;
    }
    public void setCustomValue2(String customValue2) {
        this.customValue2 = customValue2;
    }

}

```

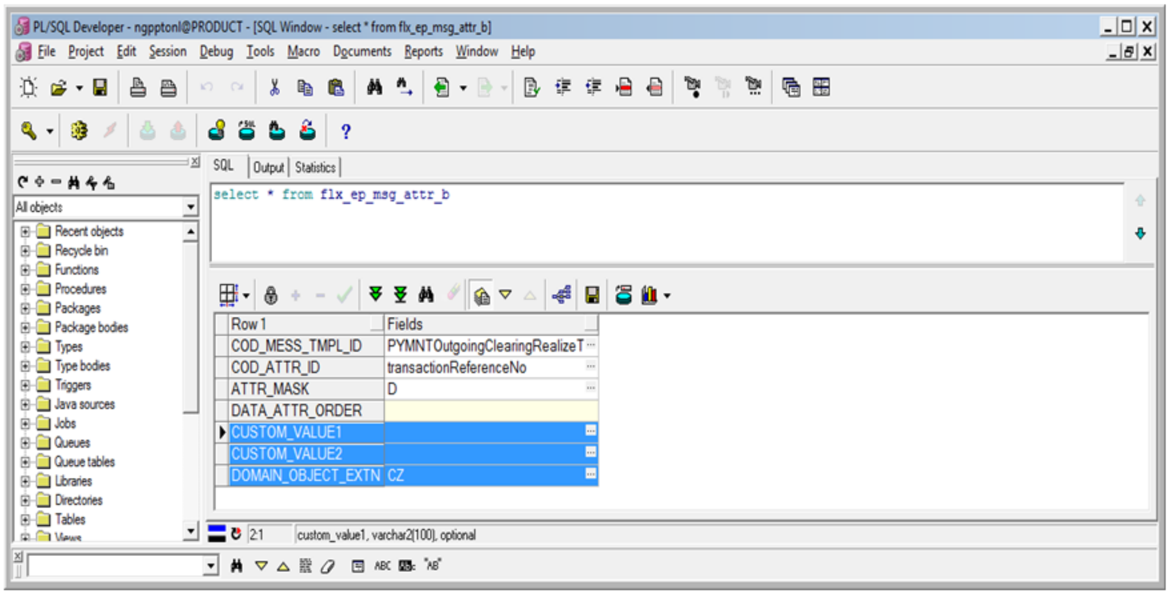
The serializable class Customized...rialVersionUID field of type long | Writable | Smart Insert | 3 : 44

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

Adding extra columns along with the discriminator column to the domain object table

The following figure explains adding a new Java file corresponding to the Customized Domain Object.

Figure 19–17 Adding Extra Columns along with the Discriminator Column



```sql
select * from flx_ep_msg_attr_b
```

Row 1	Fields
COD_MESS_TMPL_ID	PYMNTOutgoingClearingRealizeT
COD_ATTR_ID	transactionReferenceNo
ATTR_MASK	D
DATA_ATTR_ORDER	
CUSTOM_VALUE1	
CUSTOM_VALUE2	
DOMAIN_OBJECT_EXTN	CZ

2:1 custom_value1, varchar2(100), optional

The extra columns along with the discriminator column have to be added to the domain object table of this domain object.

In case of Creation or Updation of ’CustomizedMessageDataAttribute’ instead of ’MessageDataAttribute’ the new discriminator column ’DOMAIN_OBJECT_EXTN’ has the value of ’FCMA’ instead of ’CZ’ and an additional value in columns ’CUSTOM_VALUE1’ and ’CUSTOM_VALUE2’ in table FLX_EP_MSG_ATTR_B.

In case of Creation or Updation of ’MessageDataAttribute’ the new discriminator column ’DOMAIN_OBJECT_EXTN’ has the value of ’CZ’ and NULL values in columns ’CUSTOM_VALUE1’ and ’CUSTOM_VALUE2’ in table FLX_EP_MSG_ATTR_B.

19.5.2 Case 2 - Mapped as ORM Subclass

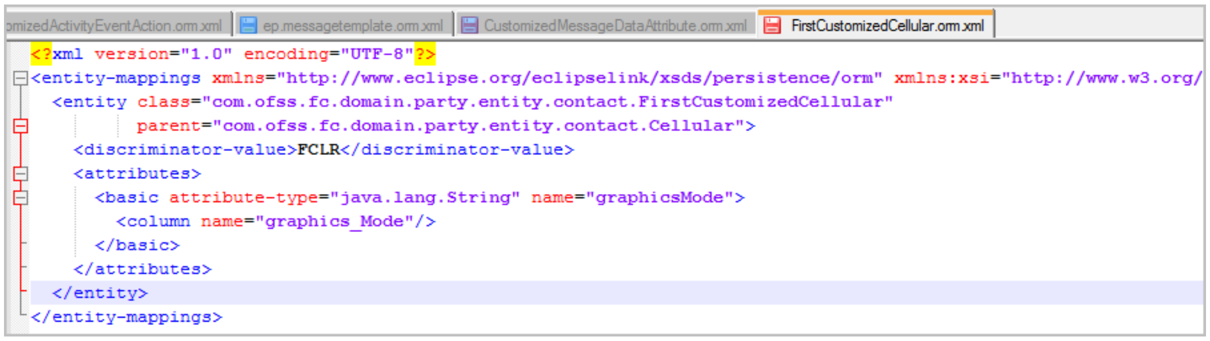
The maintenance domain objects which are mapped as a Subclass already have an existing discriminator. For the purpose of identifying the extended domain object in the same table, we shall be using the existing discriminator.

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 19–18 Adding a New ORM File Mapping to Customized Domain Object


omizedActivityEventAction.orm.xml
ep.messagetemplate.orm.xml
CustomizedMessageDataAttribute.orm.xml
FirstCustomizedCellular.orm.xml

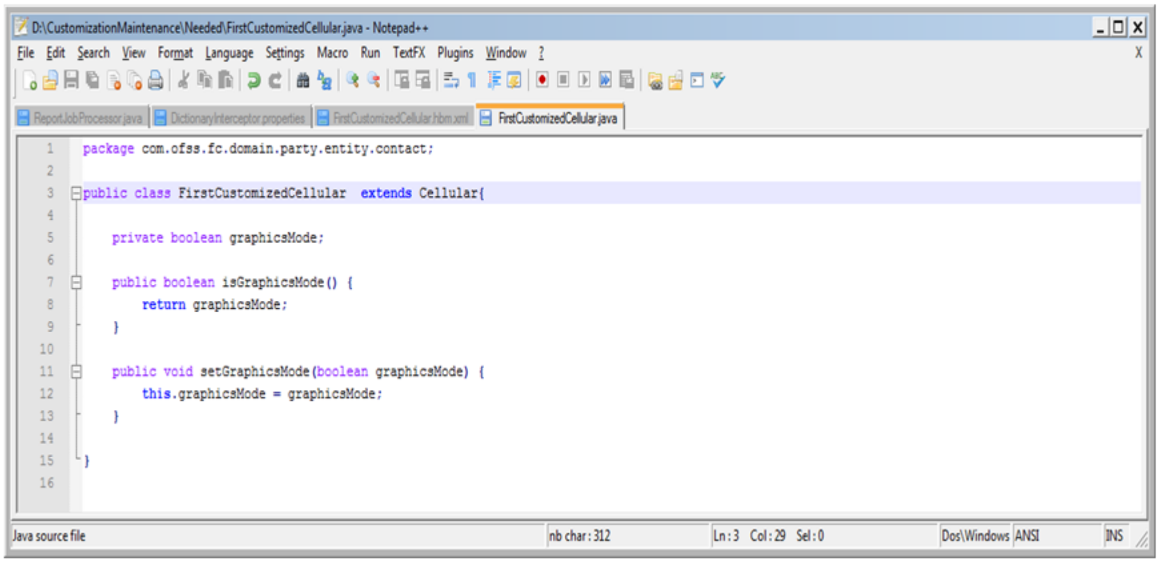
```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings xmlns="http://www.eclipse.org/eclipselink/xsds/persistence/orm" xmlns:xsi="http://www.w3.org/
  <entity class="com.ofss.fc.domain.party.entity.contact.FirstCustomizedCellular"
          parent="com.ofss.fc.domain.party.entity.contact.Cellular">
    <discriminator-value>FCLR</discriminator-value>
    <attributes>
      <basic attribute-type="java.lang.String" name="graphicsMode">
        <column name="graphics_Mode"/>
      </basic>
    </attributes>
  </entity>
</entity-mappings>
```

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 19–19 Adding New Java File to Customized Domain Object


D:\CustomizationMaintenance\Needed\FirstCustomizedCellular.java - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
ReportJobProcessor.java
DictionaryInterceptor.properties
FirstCustomizedCellular.hbm.xml
FirstCustomizedCellular.java
Java source file
nb char : 312
Ln : 3 Col : 29 Sel : 0
Dos\Windows ANSI
INS

```java
package com.ofss.fc.domain.party.entity.contact;

public class FirstCustomizedCellular  extends Cellular{

    private boolean graphicsMode;

    public boolean isGraphicsMode() {
        return graphicsMode;
    }

    public void setGraphicsMode(boolean graphicsMode) {
        this.graphicsMode = graphicsMode;
    }

}
```

A Java File 'com.ofss.fc.domain.party.entity.contact.FirstCustomizedCellular' is added corresponding to the existing Abstract Domain Object. This will be extending the Abstract Domain Object that we are extending.

Adding Extra Columns to the Domain Object Table

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

In this case GRAPHICS_MODE column is added to FLX_PI_CONTACT_POINT table.

So in case of Creation or Updation of 'FirstCustomizedCellular' instead of 'Cellular' the existing discriminator column 'CONTACT_POINT_TYPE' has the value of 'FCLR' instead of 'CLR' and an additional value in column 'GRAPHICS_MODE' in table FLX_PI_CONTACT_POINT.

And in case of Creation or Updation of 'Cellular' the existing discriminator column 'CONTACT_POINT_TYPE' has the value of 'CLR' and NULL values in column 'GRAPHICS_MODE' in table FLX_PI_CONTACT_POINT.

19.5.3 Case 3 - Mapped as 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.

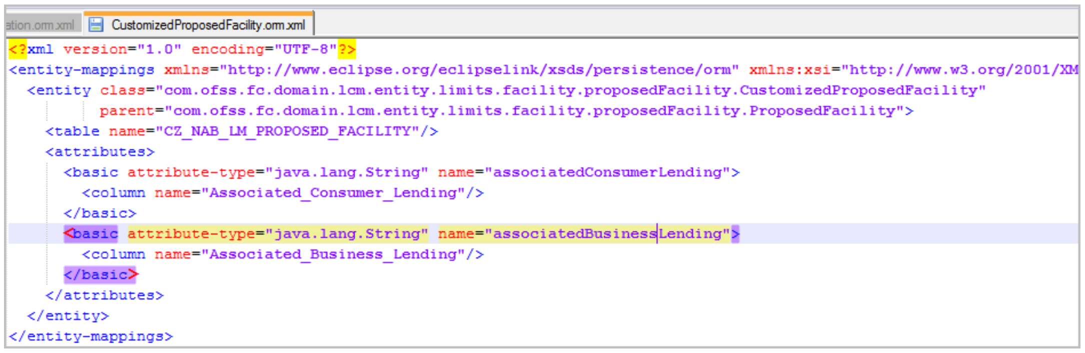
Adding Discriminator in ORM file where base class has been mapped is not required

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 19–20 New ORM File Mapping



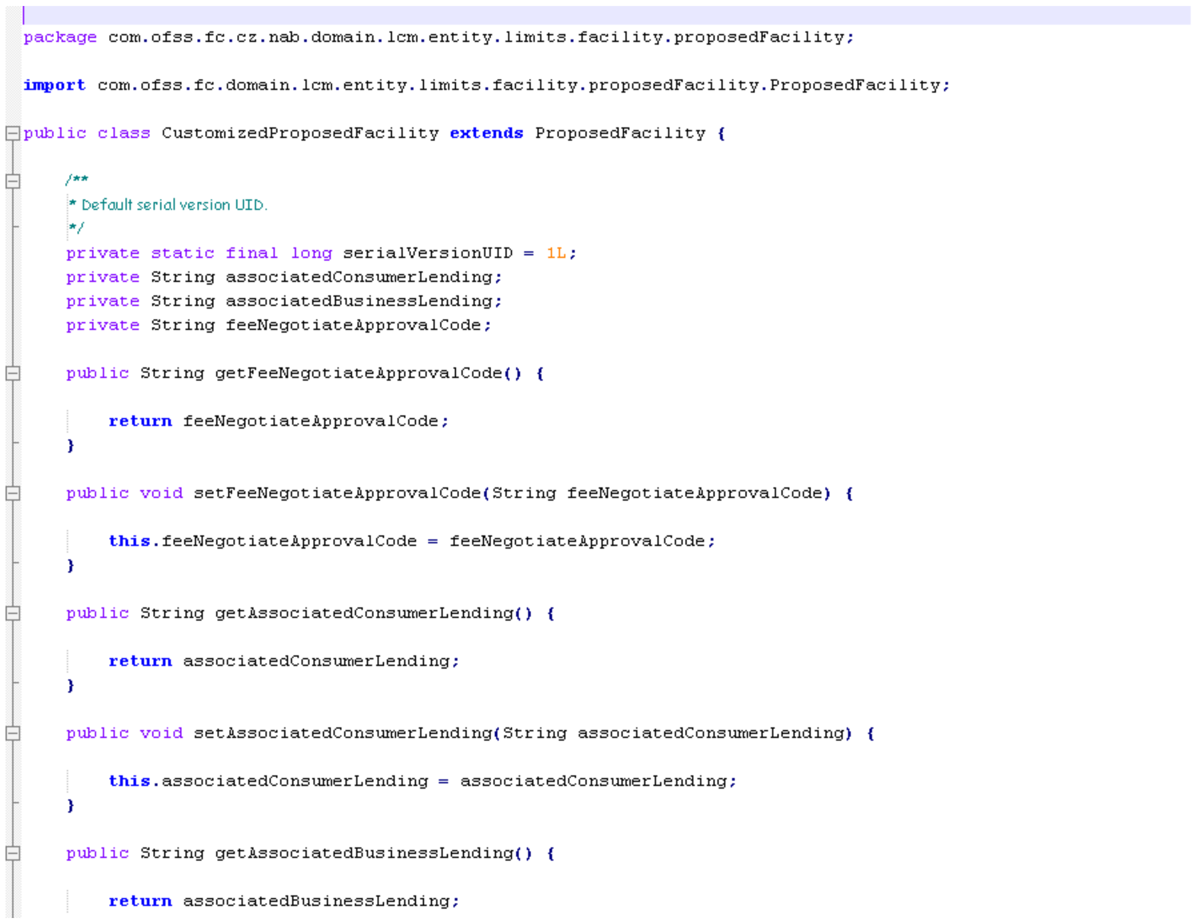
ation.orm.xml | CustomizedProposedFacility.orm.xml

```xml
<?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/XM
  <entity class="com.ofss.fc.domain.lcm.entity.limits.facility.proposedFacility.CustomizedProposedFacility"
          parent="com.ofss.fc.domain.lcm.entity.limits.facility.proposedFacility.ProposedFacility">
    <table name="CZ_NAB_LM_PROPOSED_FACILITY"/>
    <attributes>
      <basic attribute-type="java.lang.String" name="associatedConsumerLending">
        <column name="Associated_Consumer_Lending"/>
      </basic>
      <basic attribute-type="java.lang.String" name="associatedBusinessLending">
        <column name="Associated_Business_Lending"/>
      </basic>
    </attributes>
  </entity>
</entity-mappings>
```

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 19–21 Adding New Java File



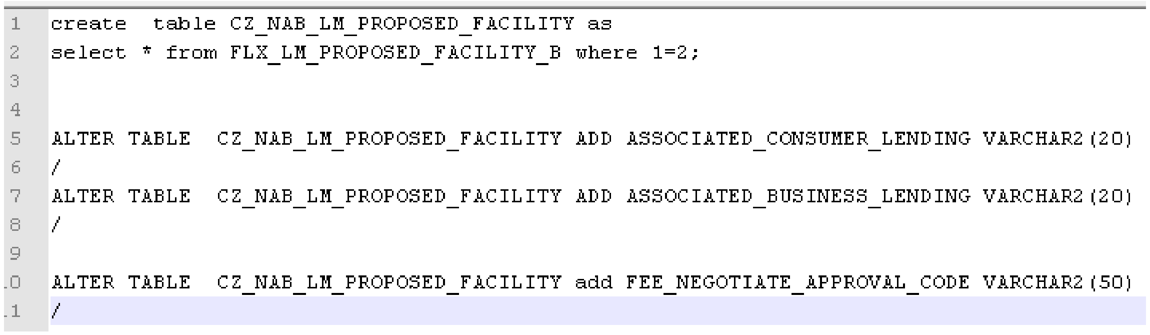
```java
package com.ofss.fc.cz.nab.domain.lcm.entity.limits.facility.proposedFacility;

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 19–22 Create a New Table CZ_NAB_LM_PROPOSED_FACILITY



```
create  table CZ_NAB_LM_PROPOSED_FACILITY as
select * from FLX_LM_PROPOSED_FACILITY_B where 1=2;


ALTER TABLE  CZ_NAB_LM_PROPOSED_FACILITY ADD ASSOCIATED_CONSUMER_LENDING VARCHAR2(20)
/
ALTER TABLE  CZ_NAB_LM_PROPOSED_FACILITY ADD ASSOCIATED_BUSINESS_LENDING VARCHAR2(20)
/

ALTER TABLE  CZ_NAB_LM_PROPOSED_FACILITY add FEE_NEGOTIATE_APPROVAL_CODE VARCHAR2(50)
/
```

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.

19.5.4 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.

19.6 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.

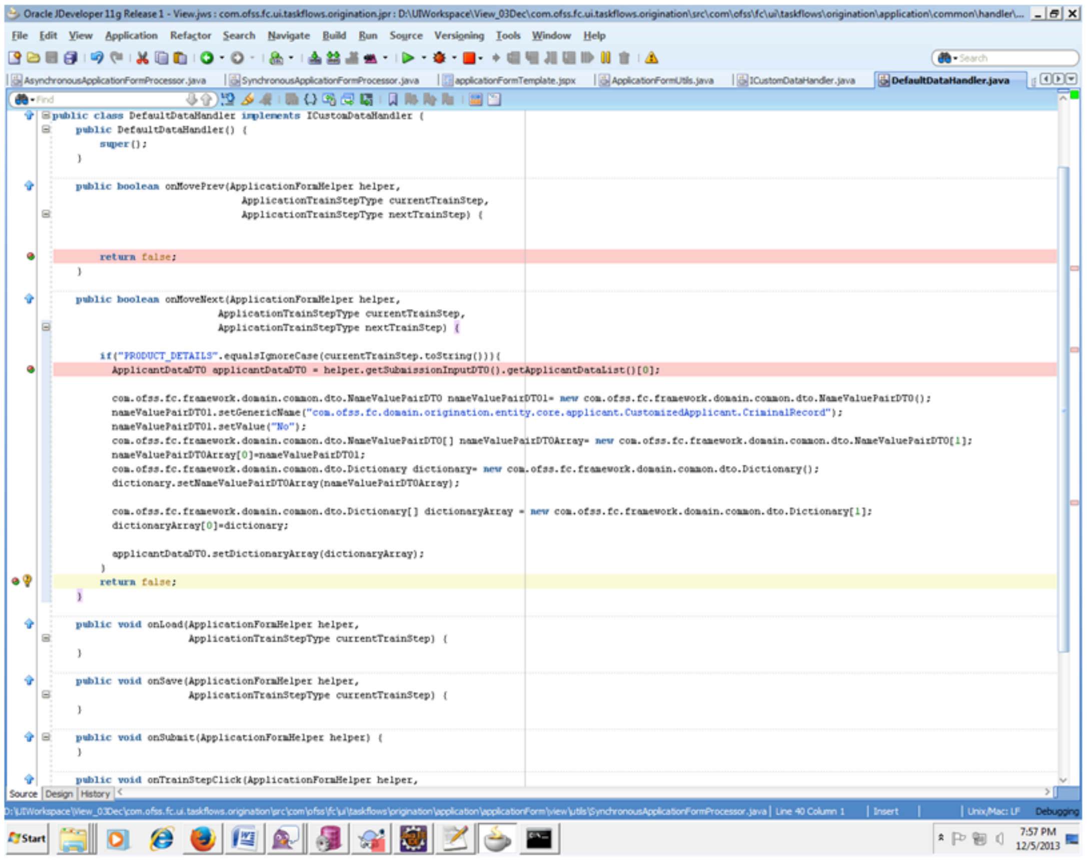
19.6.1 ICustomDataHandler's as DictionaryArray Interceptor

The backing bean method of OR097 - Application Form 'com.ofss.fc.ui.taskflows.origination.application.applicationForm.view.backing.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 19–23 CustomDataHandler's as DictionaryArray Interceptor



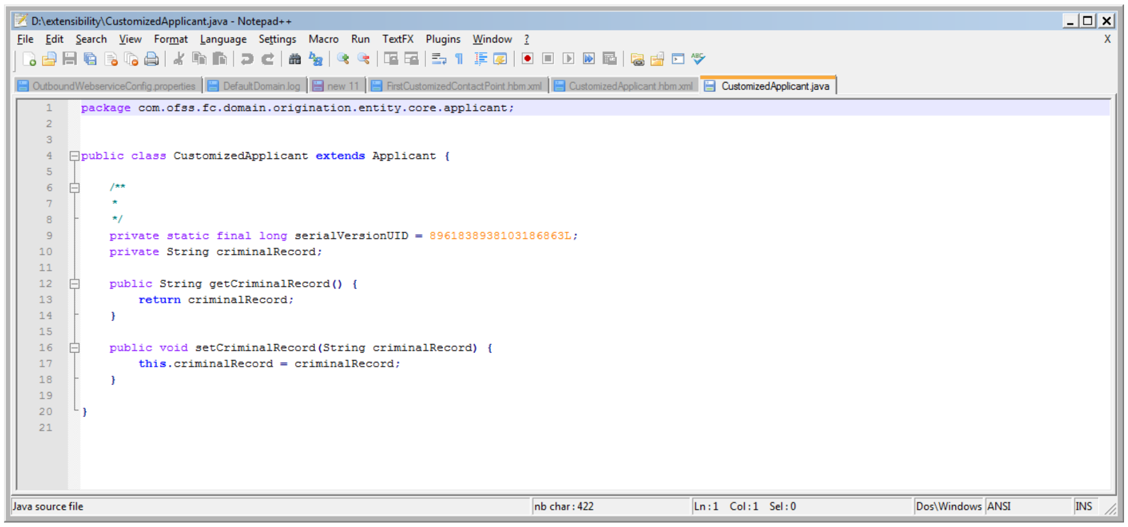
```java
public class DefaultDataHandler implements ICustomDataHandler {
    public DefaultDataHandler() {
        super();
    }

    public boolean onMovePrev(ApplicationFormHelper helper,
                              ApplicationTrainStepType currentTrainStep,
                              ApplicationTrainStepType nextTrainStep) {


        return false;
    }

    public boolean onMoveNext(ApplicationFormHelper helper,
                            ApplicationTrainStepType currentTrainStep,
                            ApplicationTrainStepType nextTrainStep) {

        if("PRODUCT_DETAILS".equalsIgnoreCase(currentTrainStep.toString())){
          ApplicantDataDTO applicantDataDTO = helper.getSubmissionInputDTO().getApplicantDataList()[0];

          com.ofss.fc.framework.domain.common.dto.NameValuePairDTO nameValuePairDTO1= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO();
          nameValuePairDTO1.setGenericName("com.ofss.fc.domain.origination.entity.core.applicant.CustomizedApplicant.CriminalRecord");
          nameValuePairDTO1.setValue("No");
          com.ofss.fc.framework.domain.common.dto.NameValuePairDTO[] nameValuePairDTOArray= new com.ofss.fc.framework.domain.common.dto.NameValuePairDTO[1];
          nameValuePairDTOArray[0]=nameValuePairDTO1;
          com.ofss.fc.framework.domain.common.dto.Dictionary dictionary= new com.ofss.fc.framework.domain.common.dto.Dictionary();
          dictionary.setNameValuePairDTOArray(nameValuePairDTOArray);

          com.ofss.fc.framework.domain.common.dto.Dictionary[] dictionaryArray = new com.ofss.fc.framework.domain.common.dto.Dictionary[1];
          dictionaryArray[0]=dictionary;

          applicantDataDTO.setDictionaryArray(dictionaryArray);
        }
        return false;
    }

    public void onLoad(ApplicationFormHelper helper,
                       ApplicationTrainStepType currentTrainStep) {
    }

    public void onSave(ApplicationFormHelper helper,
                       ApplicationTrainStepType currentTrainStep) {
    }

    public void onSubmit(ApplicationFormHelper helper) {
    }

    public void onTrainStepClick(ApplicationFormHelper helper,
```

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

19.6.2 Create Customized Abstract Domain Object Class

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

Figure 19–24 Create Customized Abstract Domain Object Class

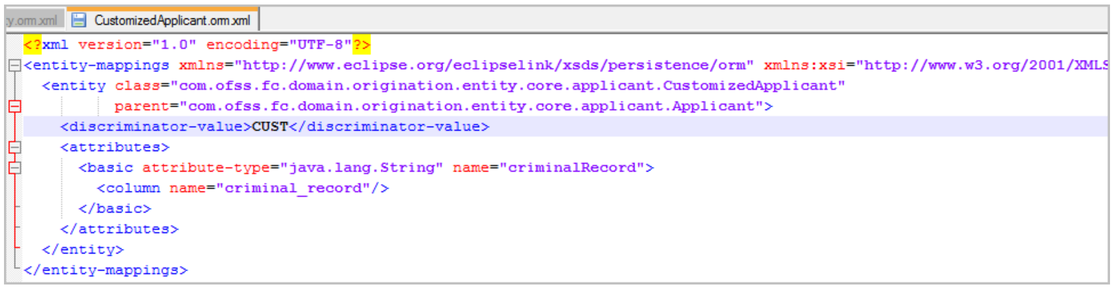


```
package com.ofss.fc.domain.origination.entity.core.applicant;


public class CustomizedApplicant extends Applicant {

    /**
     *
     */
    private static final long serialVersionUID = 8961838938103186863L;
    private String criminalRecord;

    public String getCriminalRecord() {
        return criminalRecord;
    }

    public void setCriminalRecord(String criminalRecord) {
        this.criminalRecord = criminalRecord;
    }

}
```

19.6.3 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 19–25 Create Customized Abstract Domain Object ORM Mapping File

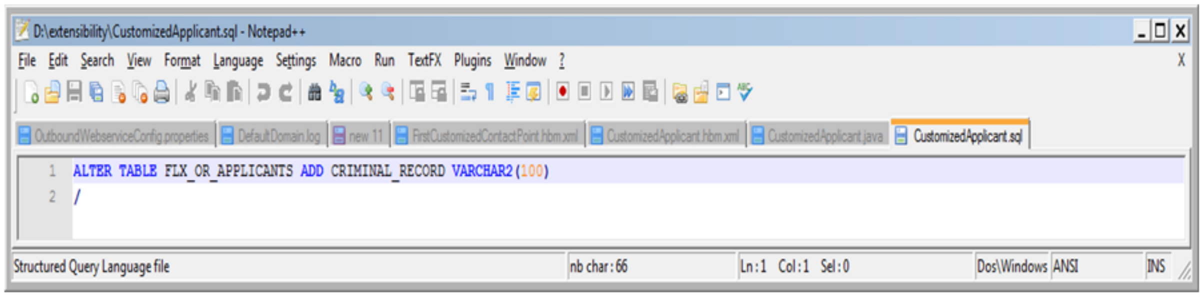


```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings xmlns="http://www.eclipse.org/eclipselink/xsds/persistence/orm" xmlns:xsi="http://www.w3.org/2001/XMLS
  <entity class="com.ofss.fc.domain.origination.entity.core.applicant.CustomizedApplicant"
          parent="com.ofss.fc.domain.origination.entity.core.applicant.Applicant">
    <discriminator-value>CUST</discriminator-value>
    <attributes>
      <basic attribute-type="java.lang.String" name="criminalRecord">
        <column name="criminal_record"/>
      </basic>
    </attributes>
  </entity>
</entity-mappings>
```

19.6.4 Create Customized Abstract Domain Object Attribute Columns

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

Figure 19–26 Create Customized Abstract Domain Object Attribute Columns



```
ALTER TABLE FLX_OR_APPLICANTS ADD CRIMINAL_RECORD VARCHAR2(100)
/
```

In case of Creation or Updation of ’CustomizedApplicant’ instead of ’Applicant’ the existing discriminator column ’DOMAIN_OBJECT_EXTN’ has the value of ’CUST’ instead of ’CZ’ and an additional value in column ’CRIMINAL_RECORD’ in table FLX_OR_APPLICANTS.

In case of Creation or Updation of ’Applicant’ the existing discriminator column ’DOMAIN_OBJECT_EXTN’ has the value of ’CZ’ and NULL values in column ’CRIMINAL_RECORD’ in table FLX_OR_APPLICANTS.

Similarly, other DomainObjectDTO's can have their dictionary arrays populated in the ICustomDataHandler class being used and the corresponding customized domain object will get persisted instead of the usual domain object.

19.7 Extensibility using Attributes of Various Supported Datatypes

Extensibility of maintenance domain objects now supports extended attributes with all data types that have a public constructor with a single argument of data-type "String".

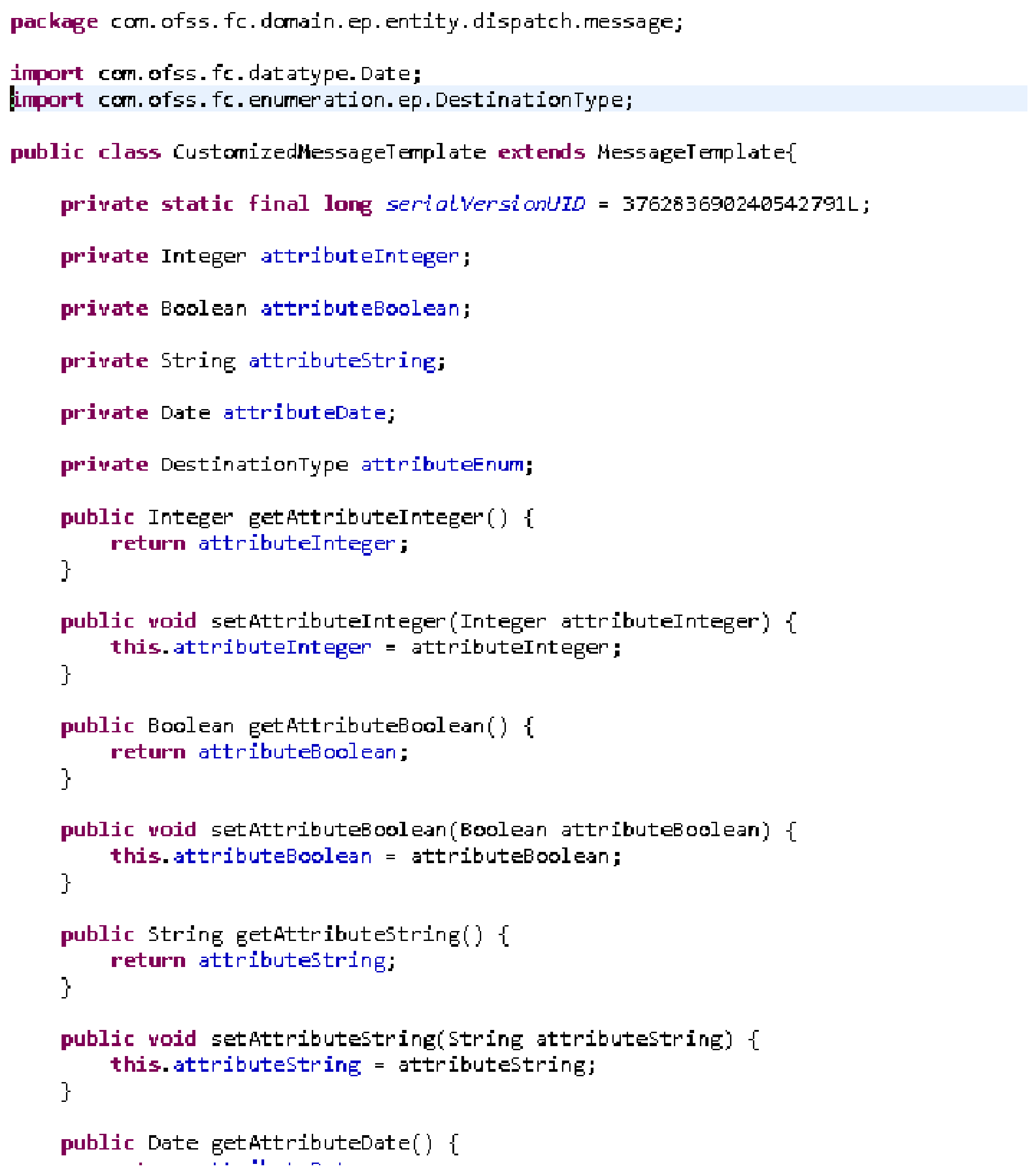
This includes attributes of data-type "com.ofss.fc.datatype.Date" whose "toString()" method should be invoked to set its value in NameValuePairDTO array element of Dictionary array. The value set is of the format given in root.properties file.

Additionally extensibility of maintenance domain objects is now also supporting extended attributes with enumeration data types defined in "com.ofss.fc.enumeration" project.

Here is an example of extensibility of "com.ofss.fc.domain.ep.entity.dispatch.message.MessageTemplate" using attributes of different supported datatypes.

The following customized class is created that contains the additional attributes.

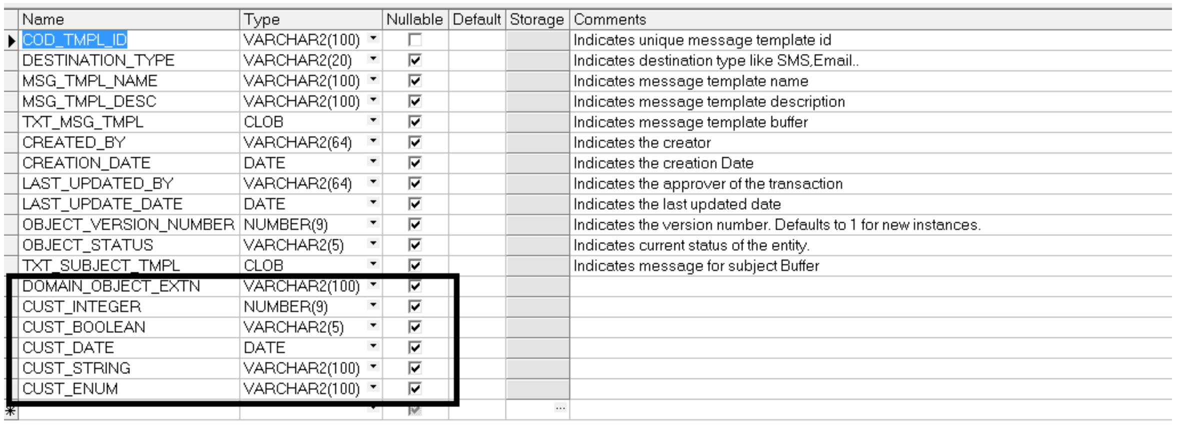
Figure 19–27 Customized Message Template Class



```
package com.ofss.fc.domain.ep.entity.dispatch.message;

import com.ofss.fc.datatype.Date;
import com.ofss.fc.enumeration.ep.DestinationType;

public class CustomizedMessageTemplate extends MessageTemplate{

    private static final long serialVersionUID = 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".

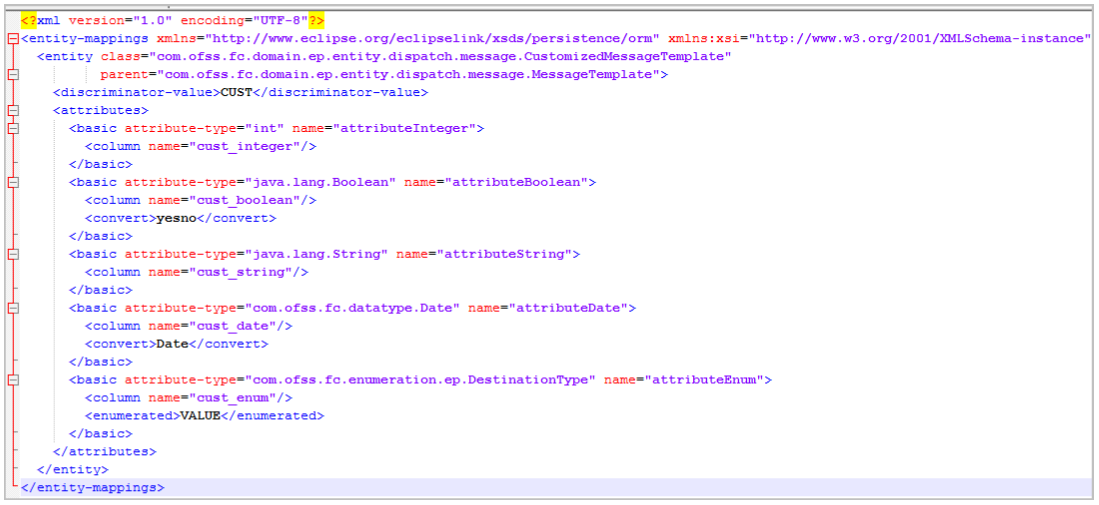
Figure 19–28 Domain Object Table



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.

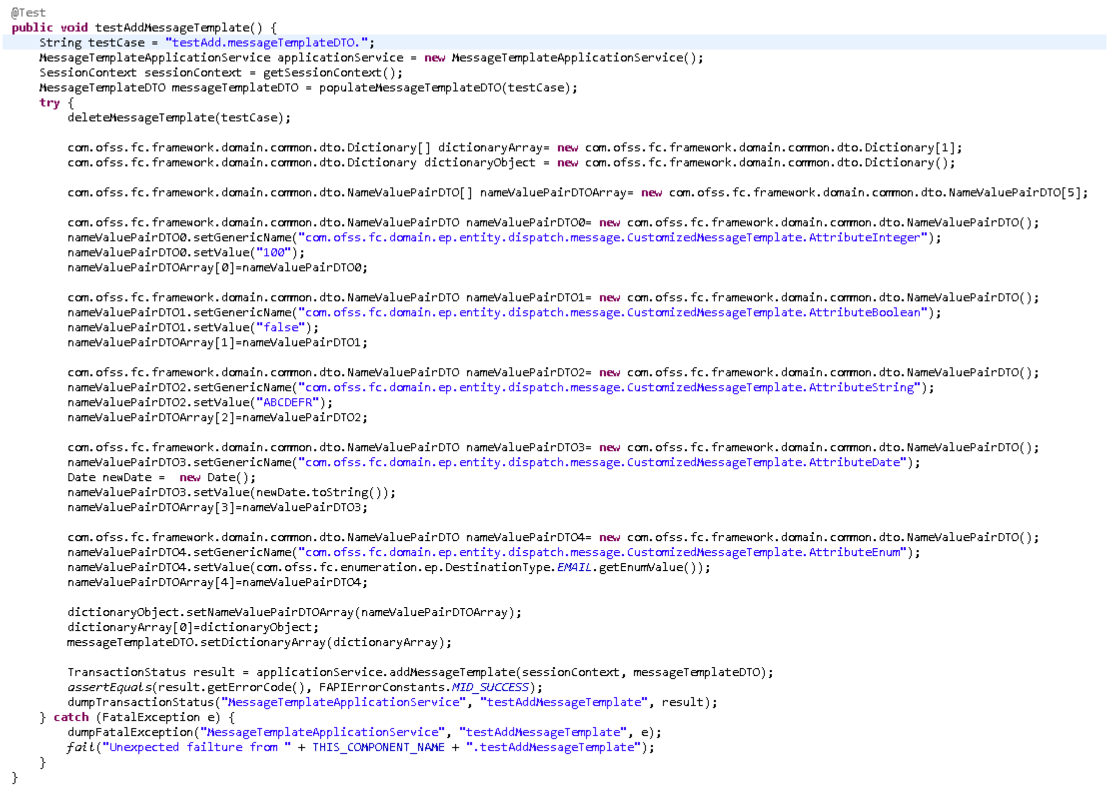
Figure 19–29 ORM File



```xml
<?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.

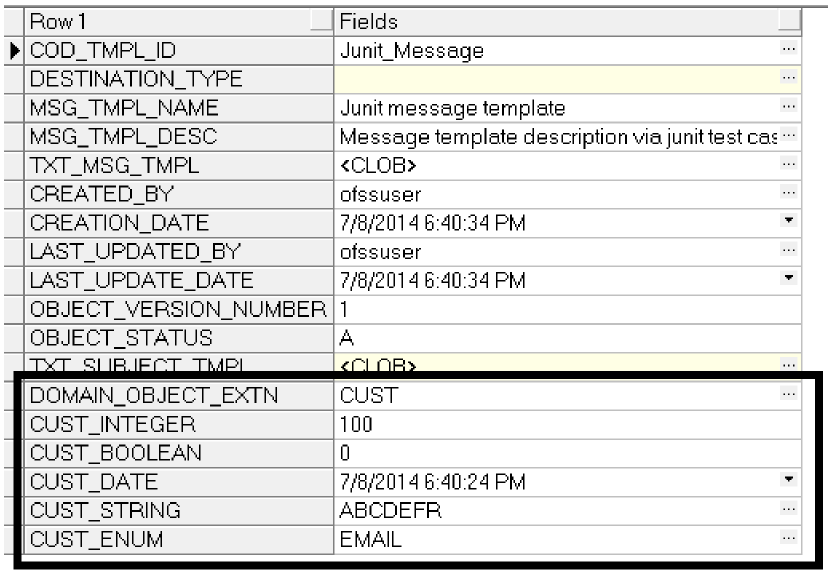
Figure 19–30 JUnit Test Case



```
@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.

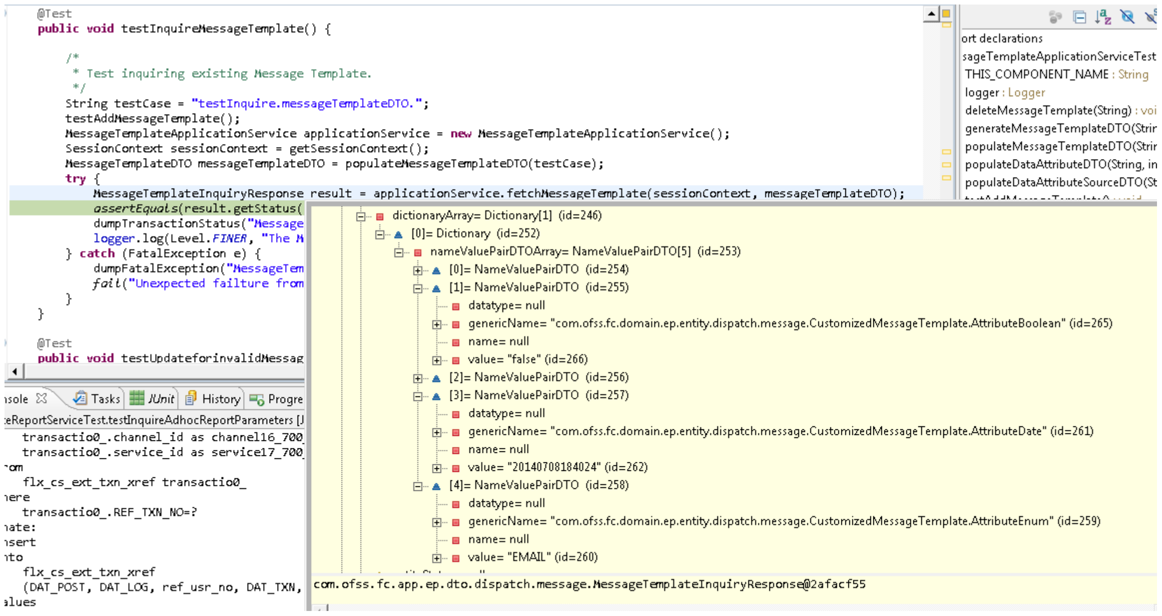
Figure 19–31 JUnit Adds Table Record



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 19–32 Dictionary Array Values



```
    @Test
    public void testInquireMessageTemplate() {

        /*
         * Test inquiring existing Message Template.
         */
        String testCase = "testInquire.messageTemplateDTO.";
        testAddMessageTemplate();
        MessageTemplateApplicationService applicationService = new MessageTemplateApplicationService();
        SessionContext sessionContext = getSessionContext();
        MessageTemplateDTO messageTemplateDTO = populateMessageTemplateDTO(testCase);
        try {
            MessageTemplateInquiryResponse result = applicationService.fetchMessageTemplate(sessionContext, messageTemplateDTO);
            assertEquals(result.getStatus(
            dumpTransactionStatus("Message
            logger.log(Level.FINER, "The M
        } catch (FatalException e) {
            dumpFatalException("MessageTem
            fail("Unexpected failture from
        }
    }

    @Test
    public void testUpdateforinvalidMessag
```

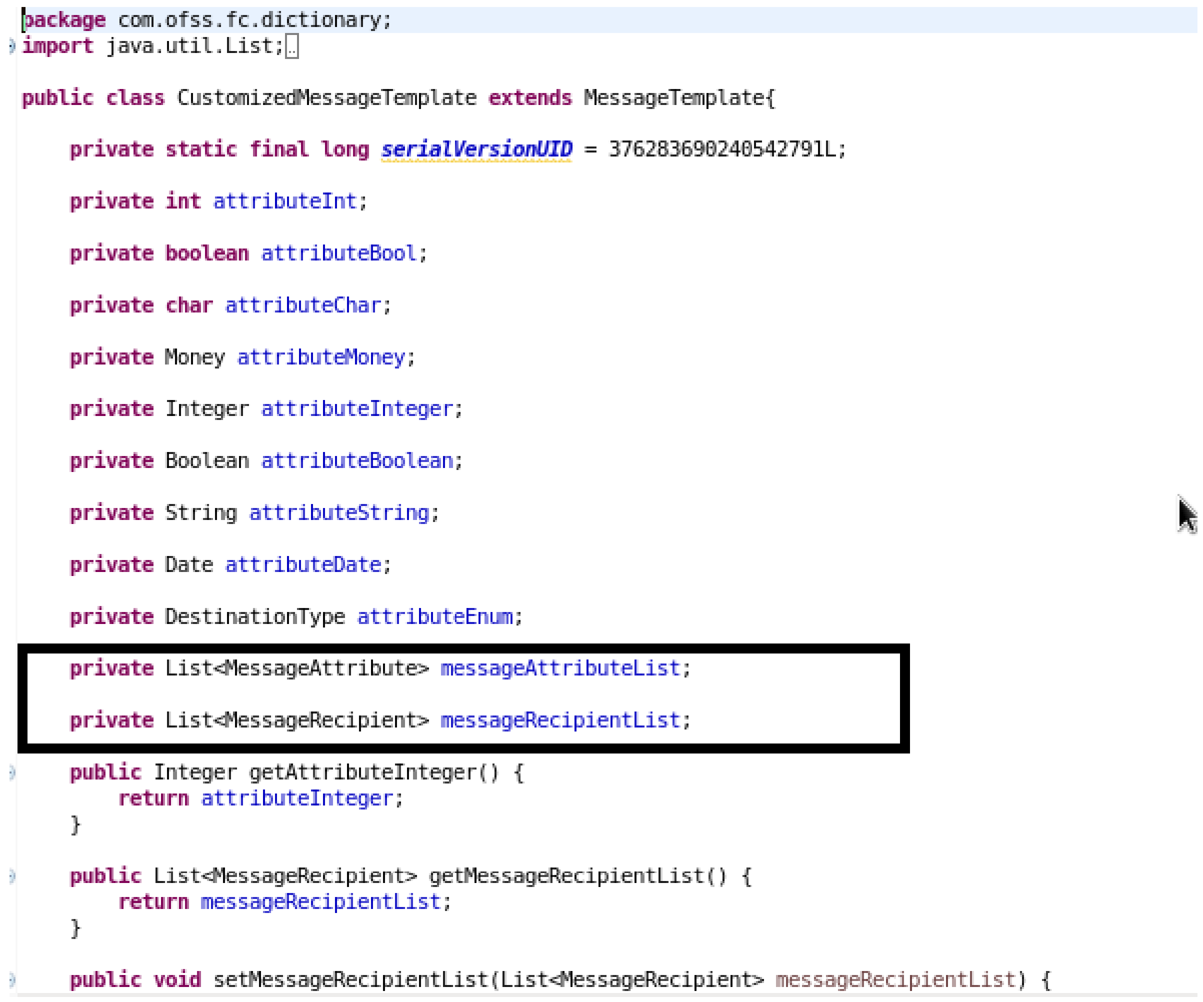
```
ort declarations
sageTemplateApplicationServiceTest
 THIS_COMPONENT_NAME : String
 logger : Logger
 deleteMessageTemplate(String) : voi
 generateMessageTemplateDTO(Strir
 populateMessageTemplateDTO(Strir
 populateDataAttributeDTO(String, in
 populateDataAttributeSourceDTO(St
```

```
eReportServiceTest.testInquireAdhocReportParameters [J
   transactio0_.channel_id as channel16_700
   transactio0_.service_id as service17_700
rom
   flx_cs_ext_txn_xref transactio0_
here
   transactio0_.REF_TXN_NO=?
nate:
nsert
nto
   flx_cs_ext_txn_xref
   (DAT_POST, DAT_LOG, ref_usr_no, DAT_TXN,
alues
```

```
dictionaryArray= Dictionary[1]  (id=246)
  [0]= Dictionary  (id=252)
    nameValuePairDTOArray= NameValuePairDTO[5]  (id=253)
      [0]= NameValuePairDTO  (id=254)
      [1]= NameValuePairDTO  (id=255)
        datatype= null
        genericName= "com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate.AttributeBoolean" (id=265)
        name= null
        value= "false" (id=266)
      [2]= NameValuePairDTO  (id=256)
      [3]= NameValuePairDTO  (id=257)
        datatype= null
        genericName= "com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate.AttributeDate" (id=261)
        name= null
        value= "20140708184024" (id=262)
      [4]= NameValuePairDTO  (id=258)
        datatype= null
        genericName= "com.ofss.fc.domain.ep.entity.dispatch.message.CustomizedMessageTemplate.AttributeEnum" (id=259)
        name= null
        value= "EMAIL" (id=260)
com.ofss.fc.app.ep.dto.dispatch.message.MessageTemplateInquiryResponse@2afacf55
```

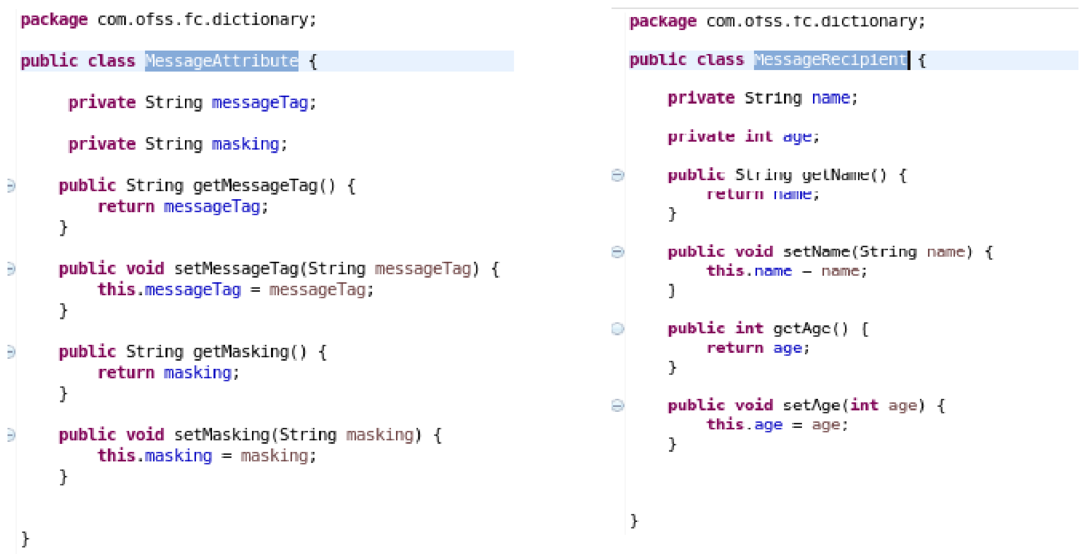
19.8 Customized Domain Object having Collection of Objects as Attributes

Figure 19–33 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 = 3762836902405427911L;

    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) {
```

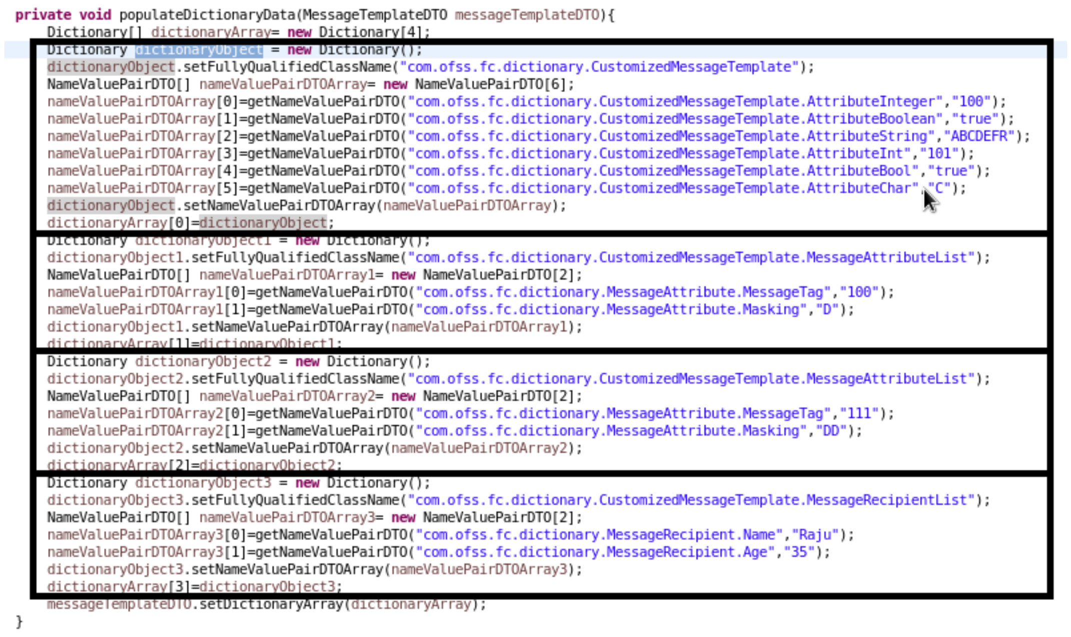
Figure 19–34 Member Attributes of Customized Domain Object



```
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;
    }

}
```

```
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 19–35 Dictionary Array Elements



```
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);
}
```

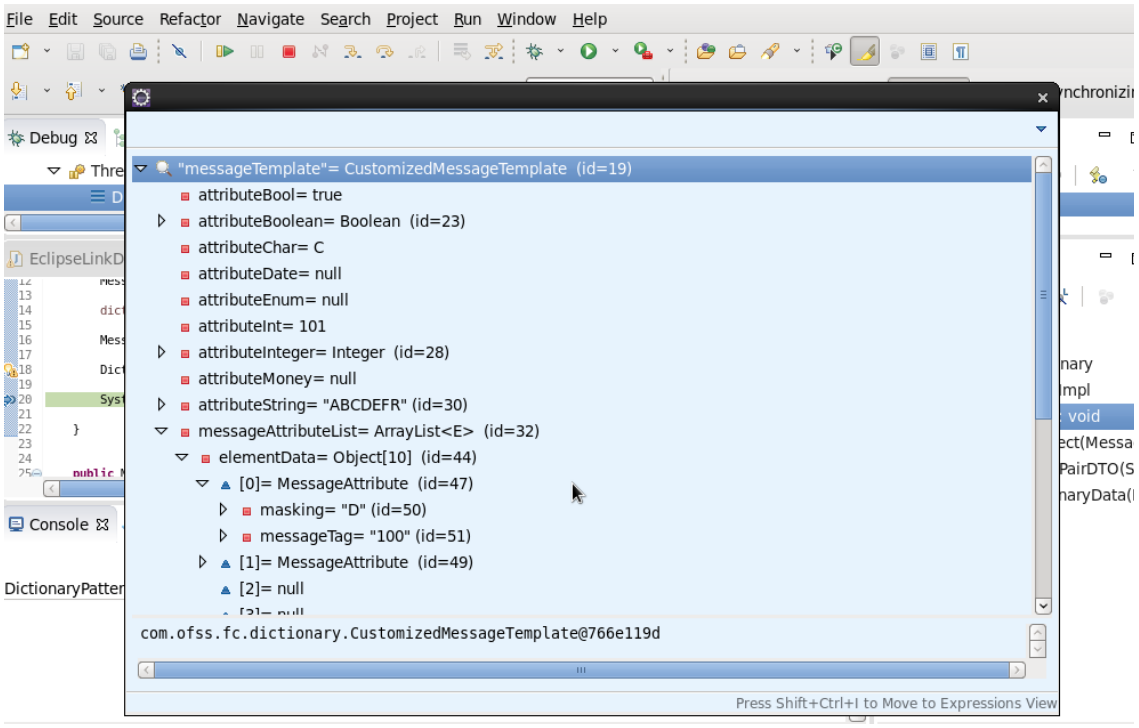
To construct a CustomizedMessageTemplate having 2 elements in messageAttributeList and 1 element in messageRecipientList, set the dictionaryArray of MessageTemplateDTO as follows:

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.

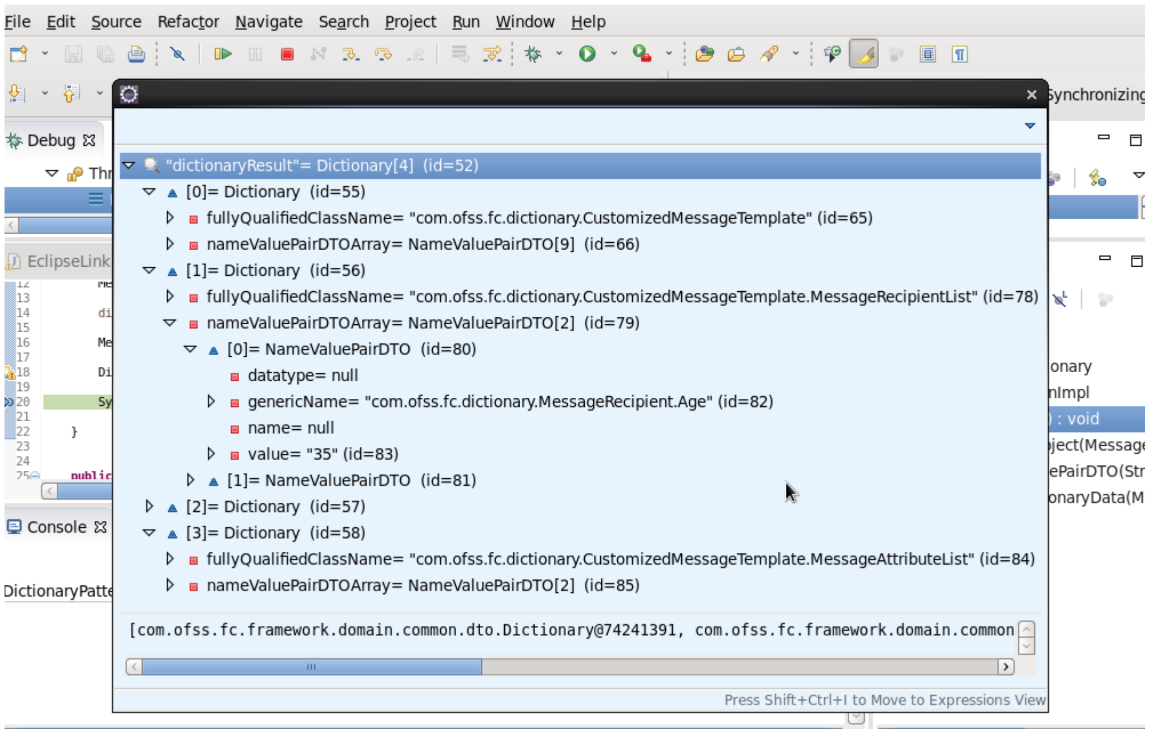
Figure 19–36 Customized Domain Object constructed by AbstractAssembler


File Edit Source Refactor Navigate Search Project Run Window Help
Debug
EclipseLinkD
Console
DictionaryPatter
Press Shift+Ctrl+I to Move to Expressions View

```
"messageTemplate"= CustomizedMessageTemplate  (id=19)
    attributeBool= true
    attributeBoolean= Boolean  (id=23)
    attributeChar= C
    attributeDate= null
    attributeEnum= null
    attributeInt= 101
    attributeInteger= Integer  (id=28)
    attributeMoney= null
    attributeString= "ABCDEFR" (id=30)
    messageAttributeList= ArrayList<E>  (id=32)
        elementData= Object[10]  (id=44)
            [0]= MessageAttribute  (id=47)
                masking= "D" (id=50)
                messageTag= "100" (id=51)
            [1]= MessageAttribute  (id=49)
            [2]= null
```

```
com.ofss.fc.dictionary.CustomizedMessageTemplate@766e119d
```

Figure 19–37 Dictionary Array returned by AbstractAssembler


File Edit Source Refactor Navigate Search Project Run Window Help
Debug
EclipseLink
Console
Synchronizing

```
"dictionaryResult"= Dictionary[4]  (id=52)
  [0]= Dictionary  (id=55)
    fullyQualifiedClassName= "com.ofss.fc.dictionary.CustomizedMessageTemplate" (id=65)
    nameValuePairDTOArray= NameValuePairDTO[9]  (id=66)
  [1]= Dictionary  (id=56)
    fullyQualifiedClassName= "com.ofss.fc.dictionary.CustomizedMessageTemplate.MessageRecipientList" (id=78)
    nameValuePairDTOArray= NameValuePairDTO[2]  (id=79)
      [0]= NameValuePairDTO  (id=80)
        datatype= null
        genericName= "com.ofss.fc.dictionary.MessageRecipient.Age" (id=82)
        name= null
        value= "35" (id=83)
      [1]= NameValuePairDTO  (id=81)
  [2]= Dictionary  (id=57)
  [3]= Dictionary  (id=58)
    fullyQualifiedClassName= "com.ofss.fc.dictionary.CustomizedMessageTemplate.MessageAttributeList" (id=84)
    nameValuePairDTOArray= NameValuePairDTO[2]  (id=85)

[com.ofss.fc.framework.domain.common.dto.Dictionary@74241391, com.ofss.fc.framework.domain.common

Press Shift+Ctrl+I to Move to Expressions View
```

19.9 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

20 Deployment Guideline

This chapter explains the deployment guidelines.

20.1 Customized Project Jars

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

20.2 Database Objects

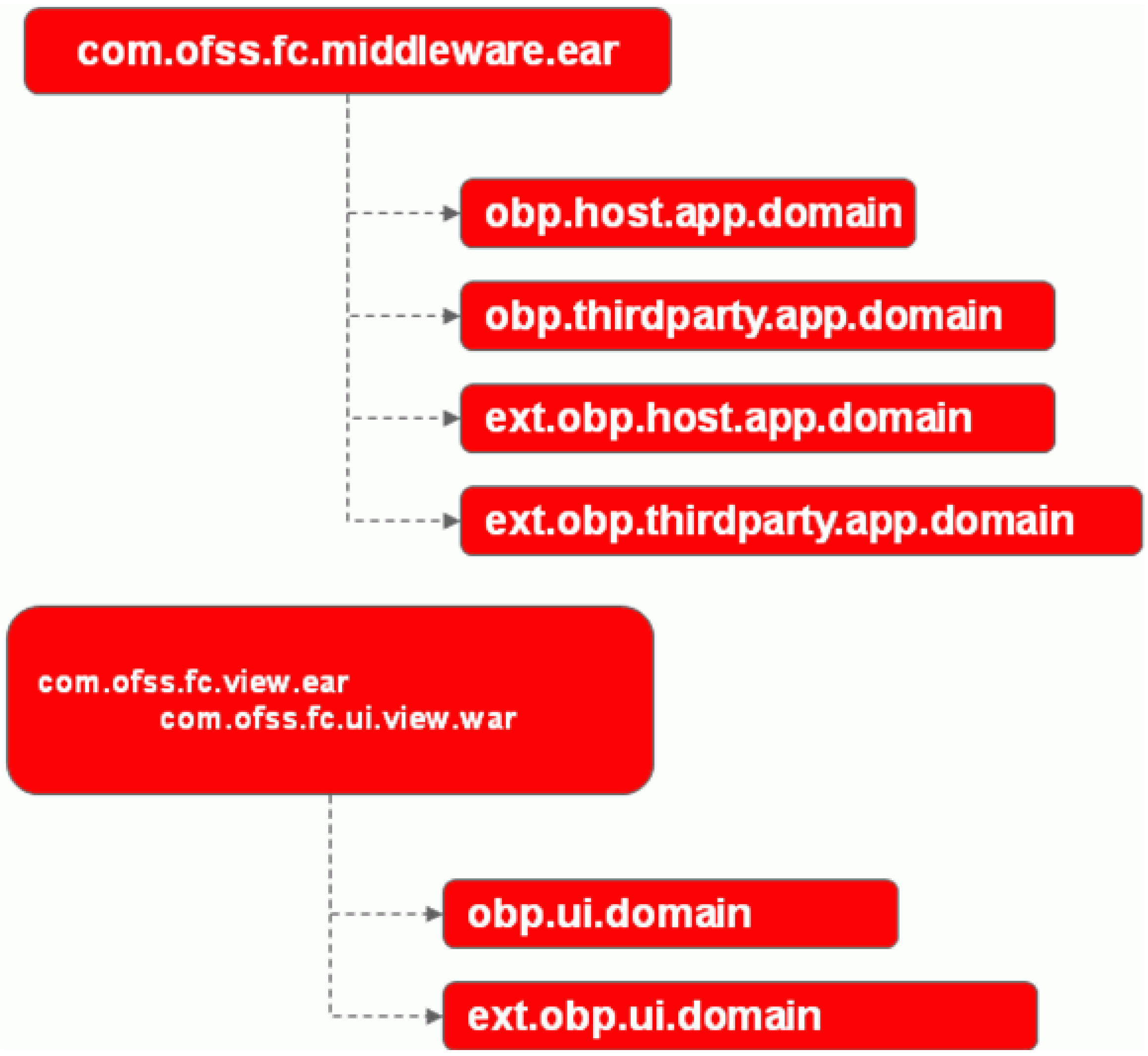
User has to update the corresponding seed data for the implementation of different extensibility features.

20.3 Extensibility Deployment

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

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

Figure 20–1 Extensibility Deployment



21 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.

21.1 Integration Adapter Interface

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

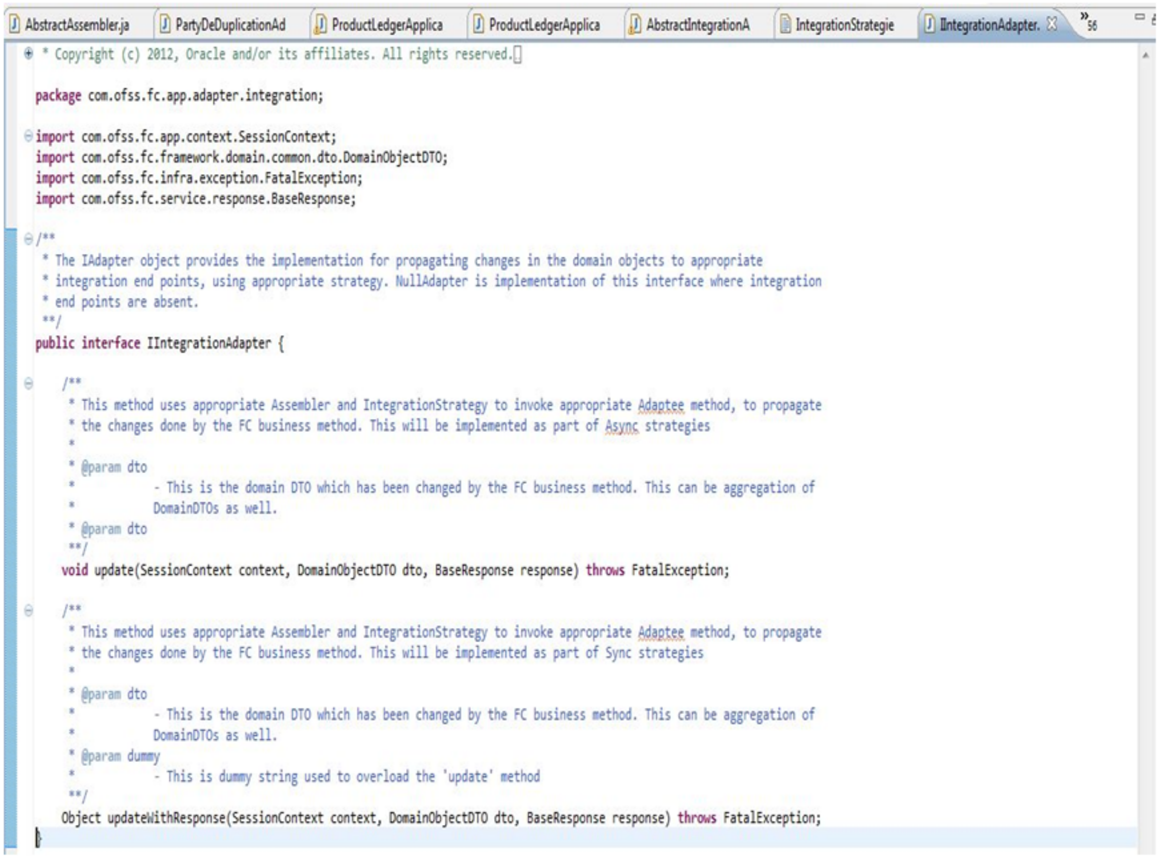
These two methods must be implemented by any adapter implementing the interface and use them for publishing data to OCH. Signature of these two methods are:

```
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 21–1 Integration Adapter Interface



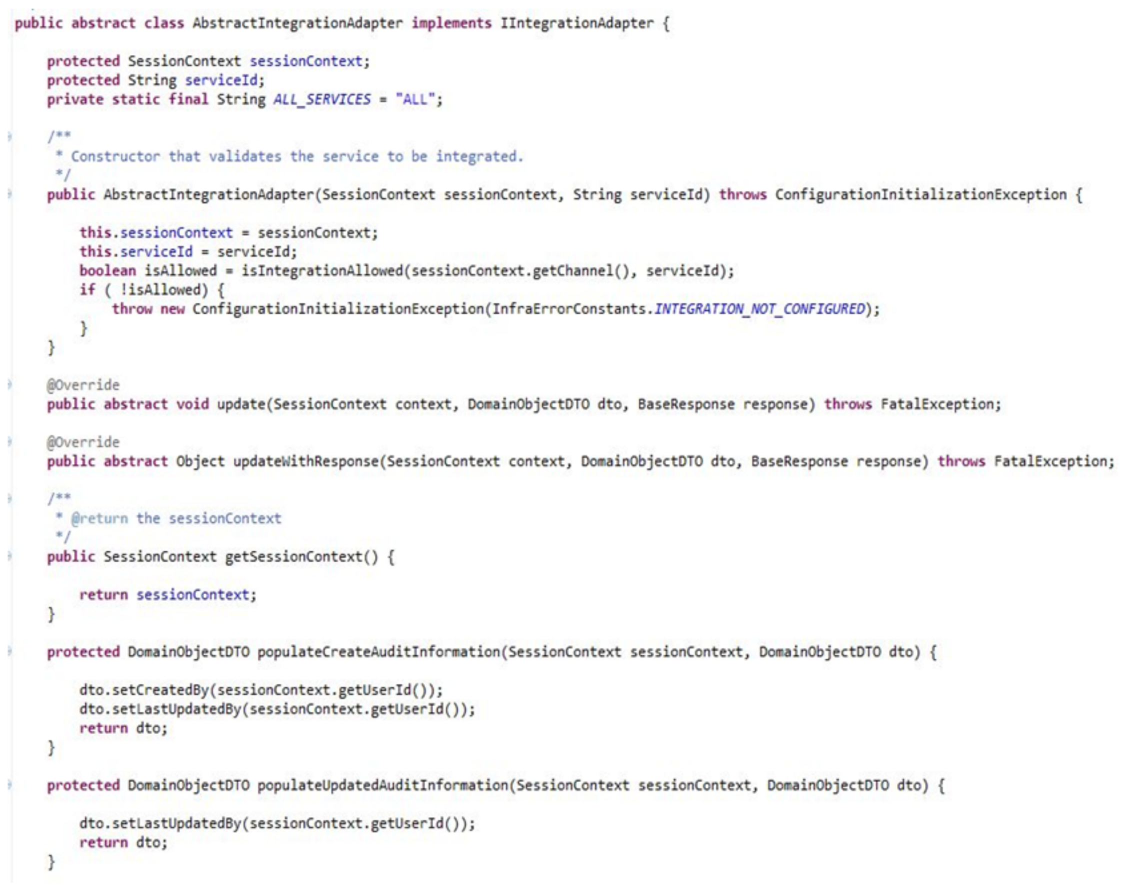
```
 * Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.

package com.ofss.fc.app.adapter.integration;

import com.ofss.fc.app.context.SessionContext;
import com.ofss.fc.framework.domain.common.dto.DomainObjectDTO;
import com.ofss.fc.infra.exception.FatalException;
import com.ofss.fc.service.response.BaseResponse;

/**
 * The IAdapter object provides the implementation for propagating changes in the domain objects to appropriate
 * integration end points, using appropriate strategy. NullAdapter is implementation of this interface where integration
 * end points are absent.
 **/
public interface IIntegrationAdapter {

    /**
     * This method uses appropriate Assembler and IntegrationStrategy to invoke appropriate Adaptee method, to propagate
     * the changes done by the FC business method. This will be implemented as part of Async strategies
     *
     * @param dto
     *            - This is the domain DTO which has been changed by the FC business method. This can be aggregation of
     *            DomainDTOs as well.
     * @param dto
     **/
    void update(SessionContext context, DomainObjectDTO dto, BaseResponse response) throws FatalException;

    /**
     * This method uses appropriate Assembler and IntegrationStrategy to invoke appropriate Adaptee method, to propagate
     * the changes done by the FC business method. This will be implemented as part of Sync strategies
     *
     * @param dto
     *            - This is the domain DTO which has been changed by the FC business method. This can be aggregation of
     *            DomainDTOs as well.
     * @param dummy
     *            - This is dummy string used to overload the 'update' method
     **/
    Object updateWithResponse(SessionContext context, DomainObjectDTO dto, BaseResponse response) throws FatalException;
}
```

21.2 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 21–2 Abstract Integration Adapter Class

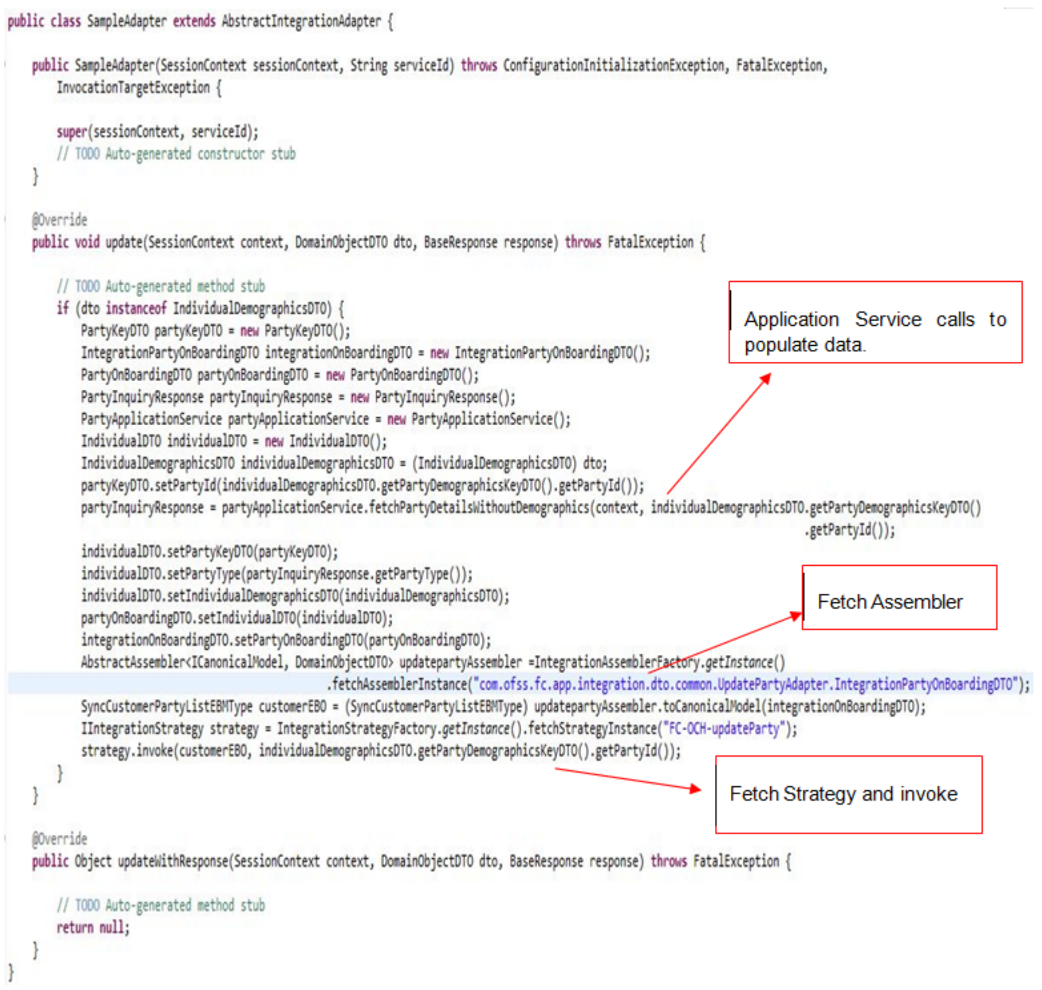


```java
public abstract class AbstractIntegrationAdapter implements IIntegrationAdapter {

    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;
    }
```

21.3 Sample Integration Adapter

The following figure is a sample adapter for customer information:

Figure 21–3 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;
    }
}
```

Application Service calls to populate data.
Fetch Assembler
Fetch Strategy and invoke

21.4 Integration Abstract Assembler

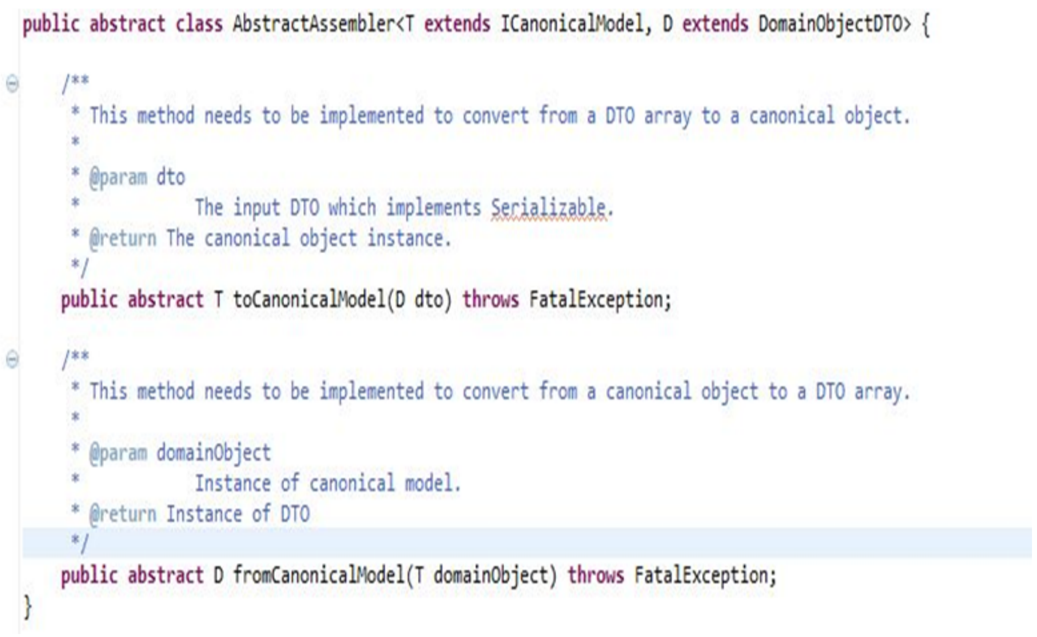
OBP framework has as abstract class AbstractAssembler which provides design for DTO to EBO conversion. These methods are used while mapping DTO to EBO and vice versa.

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 21–4 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;
}
```

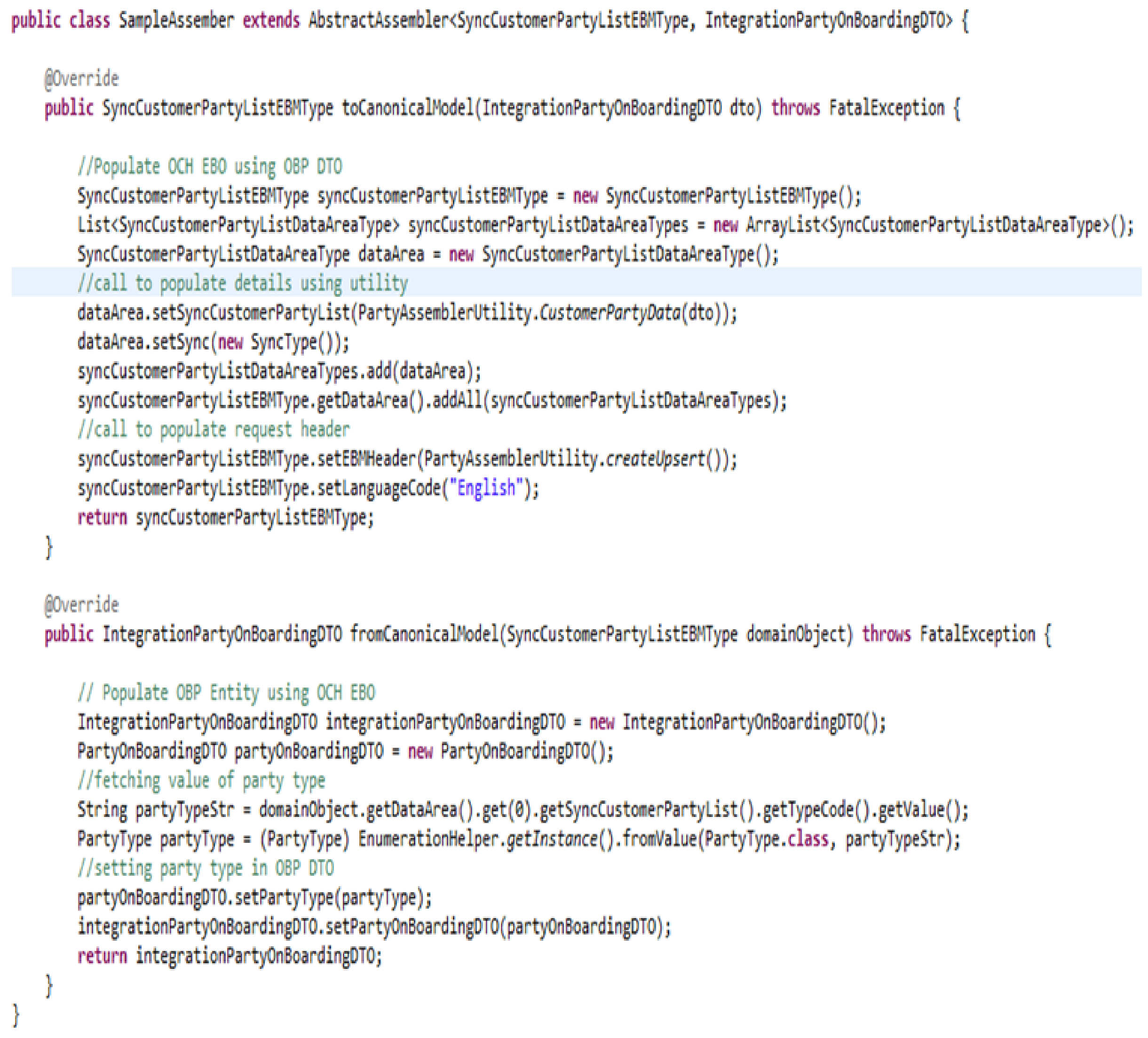
All the assemblers must implement these two methods for conversion of DTO to EBO and vice versa.

Assemblers also populate the header of the request which is posted to the queue.

21.5 Sample Assembler

A sample assembler which extends AbstractAssembler should be like:

Figure 21–5 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.

22 Algorithm Extensions

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

22.1 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.

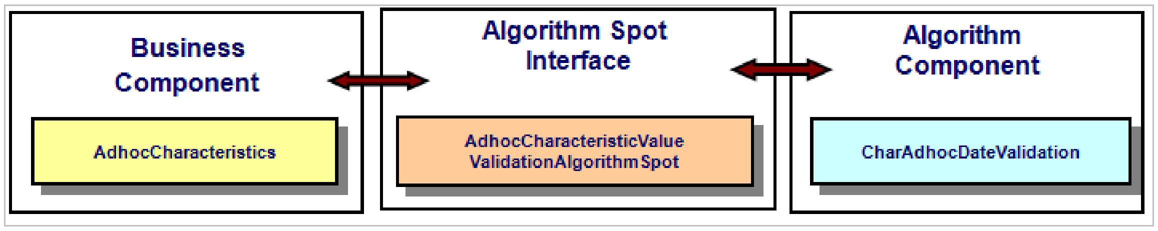
Algorithms are defined in 2 places:

- 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.

22.2 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 22–1 Algorithm Spot Interface

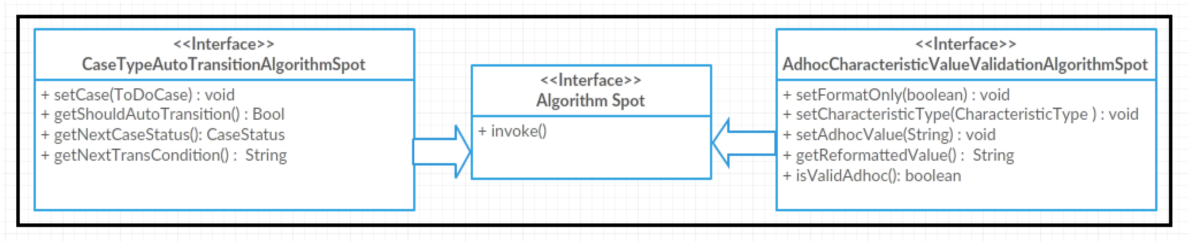


Attributes of an algorithm spot interface class:

- 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 22–2 Example: Algorithm Spot Interface

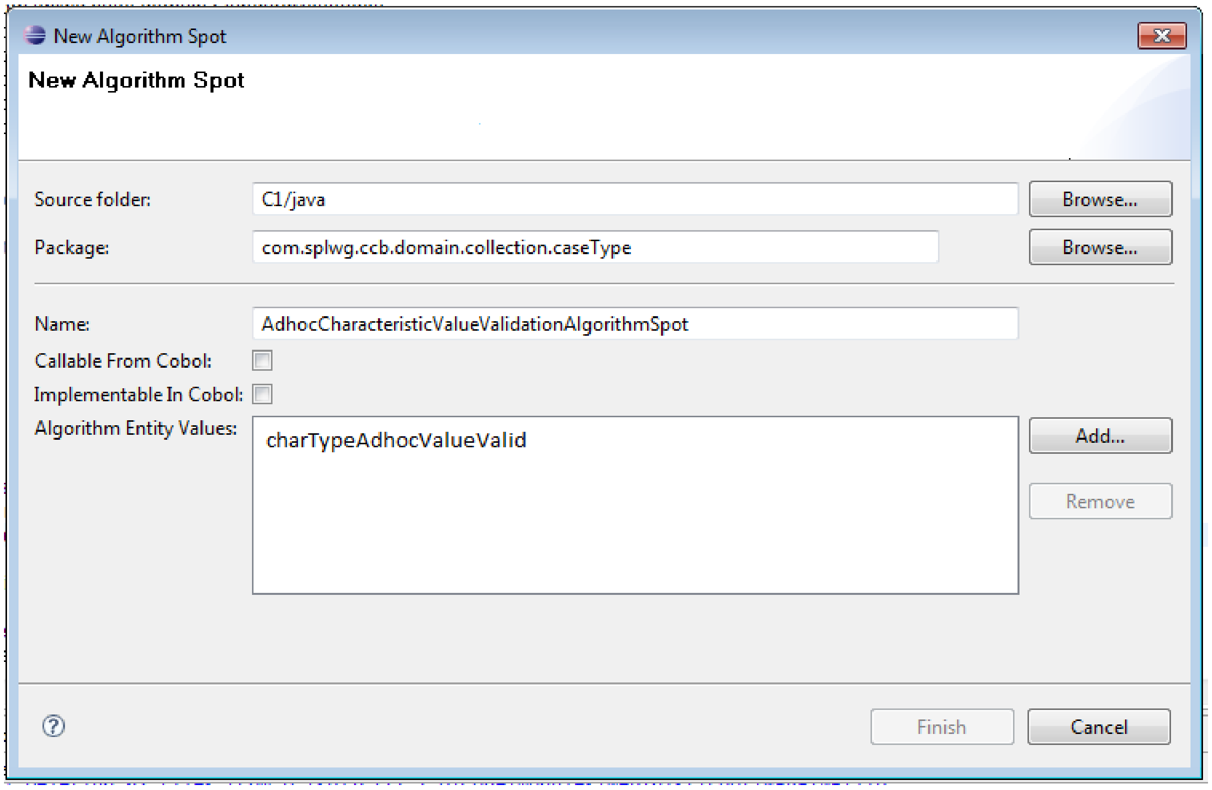

<<Interface>>
CaseTypeAutoTransitionAlgorithmSpot
+ setCase(ToDoCase) : void
+ getShouldAutoTransition() : Bool
+ getNextCaseStatus(): CaseStatus
+ getNextTransCondition() : String
<<Interface>>
Algorithm Spot
+ invoke()
<<Interface>>
AdhocCharacteristicValueValidationAlgorithmSpot
+ setFormatOnly(boolean) : void
+ setCharacteristicType(CharacteristicType) : void
+ setAdhocValue(String) : void
+ getReformattedValue() : String
+ isValidAdhoc(): boolean

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 22–3 Example: Adding New Algorithm Spot



New Algorithm Spot

New Algorithm Spot

Source folder: C1/java

Package: com.splwg.ccb.domain.collection.caseType

Name: AdhocCharacteristicValueValidationAlgorithmSpot

Callable From Cobol: ☐

Implementable In Cobol: ☐

Algorithm Entity Values: charTypeAdhocValueValid

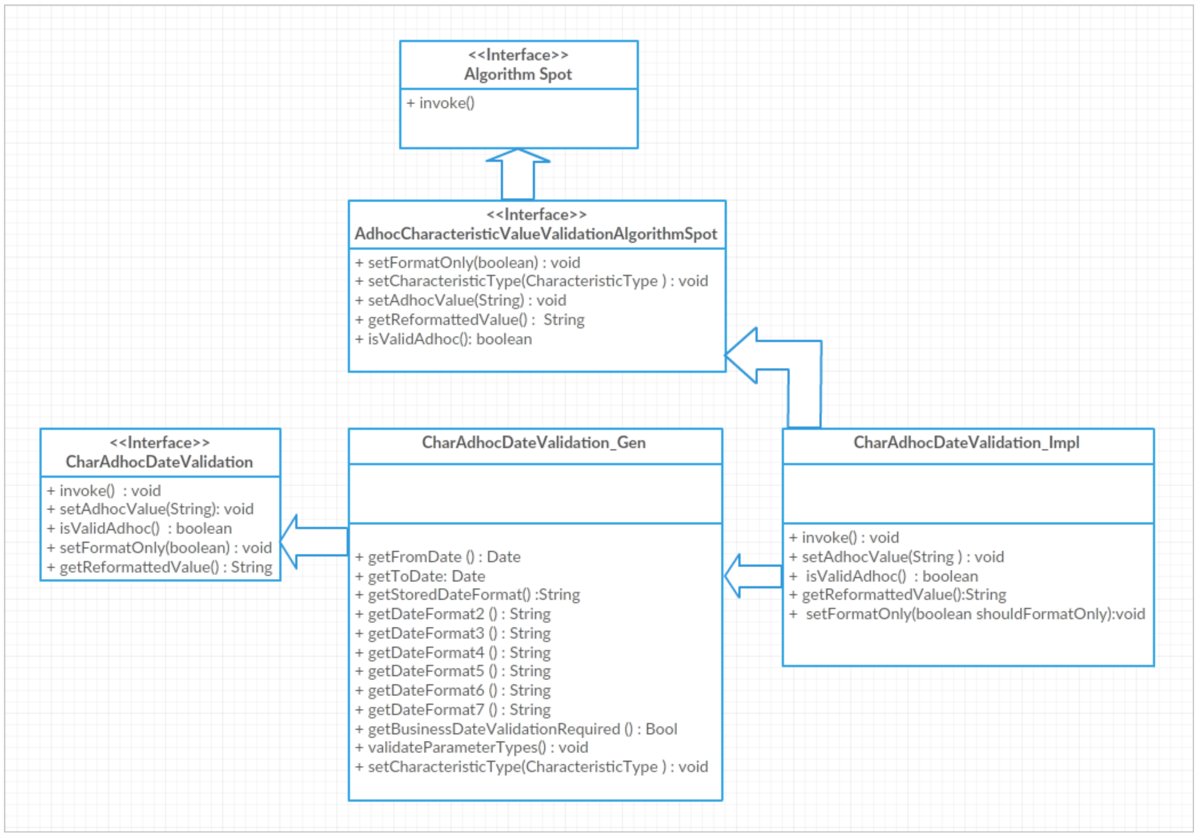
22.3 Algorithm Components

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 22–4 Example: Data Validation Algorithm Component

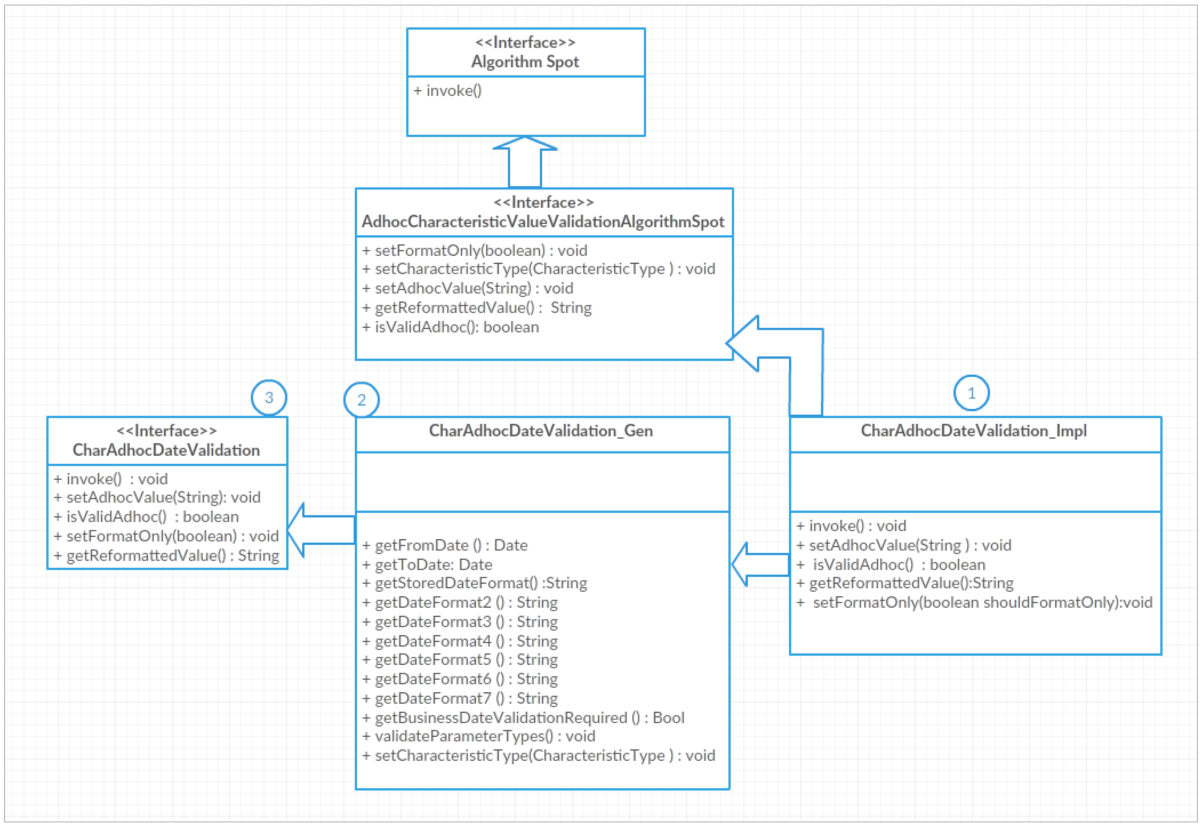


<<Interface>>
Algorithm Spot
+ invoke()
<<Interface>>
AdhocCharacteristicValueValidationAlgorithmSpot
+ setFormatOnly(boolean) : void
+ setCharacteristicType(CharacteristicType) : void
+ setAdhocValue(String) : void
+ getReformattedValue() : String
+ isValidAdhoc(): boolean
<<Interface>>
CharAdhocDateValidation
+ invoke() : void
+ setAdhocValue(String): void
+ isValidAdhoc() : boolean
+ setFormatOnly(boolean) : void
+ getReformattedValue() : String
CharAdhocDateValidation_Gen
+ getFromDate () : Date
+ getToDate: Date
+ getStoredDateFormat() :String
+ getDateFormat2 () : String
+ getDateFormat3 () : String
+ getDateFormat4 () : String
+ getDateFormat5 () : String
+ getDateFormat6 () : String
+ getDateFormat7 () : String
+ getBusinessDateValidationRequired () : Bool
+ validateParameterTypes() : void
+ setCharacteristicType(CharacteristicType) : void
CharAdhocDateValidation_Impl
+ invoke() : void
+ setAdhocValue(String) : void
+ isValidAdhoc() : boolean
+ getReformattedValue():String
+ setFormatOnly(boolean shouldFormatOnly):void

Algorithm Components Example - CharAdhocDateValidation

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

Figure 22–5 Example: CharAdhocDateValidation



<<Interface>>
Algorithm Spot
+ invoke()
<<Interface>>
AdhocCharacteristicValueValidationAlgorithmSpot
+ setFormatOnly(boolean) : void
+ setCharacteristicType(CharacteristicType) : void
+ setAdhocValue(String) : void
+ getReformattedValue() : String
+ isValidAdhoc(): boolean
3
<<Interface>>
CharAdhocDateValidation
+ invoke() : void
+ setAdhocValue(String): void
+ isValidAdhoc() : boolean
+ setFormatOnly(boolean) : void
+ getReformattedValue() : String
2
CharAdhocDateValidation_Gen
+ getFromDate () : Date
+ getToDate: Date
+ getStoredDateFormat() :String
+ getDateFormat2 () : String
+ getDateFormat3 () : String
+ getDateFormat4 () : String
+ getDateFormat5 () : String
+ getDateFormat6 () : String
+ getDateFormat7 () : String
+ getBusinessDateValidationRequired () : Bool
+ validateParameterTypes() : void
+ setCharacteristicType(CharacteristicType) : void
1
CharAdhocDateValidation_Impl
+ invoke() : void
+ setAdhocValue(String) : void
+ isValidAdhoc() : boolean
+ getReformattedValue():String
+ setFormatOnly(boolean shouldFormatOnly):void

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).

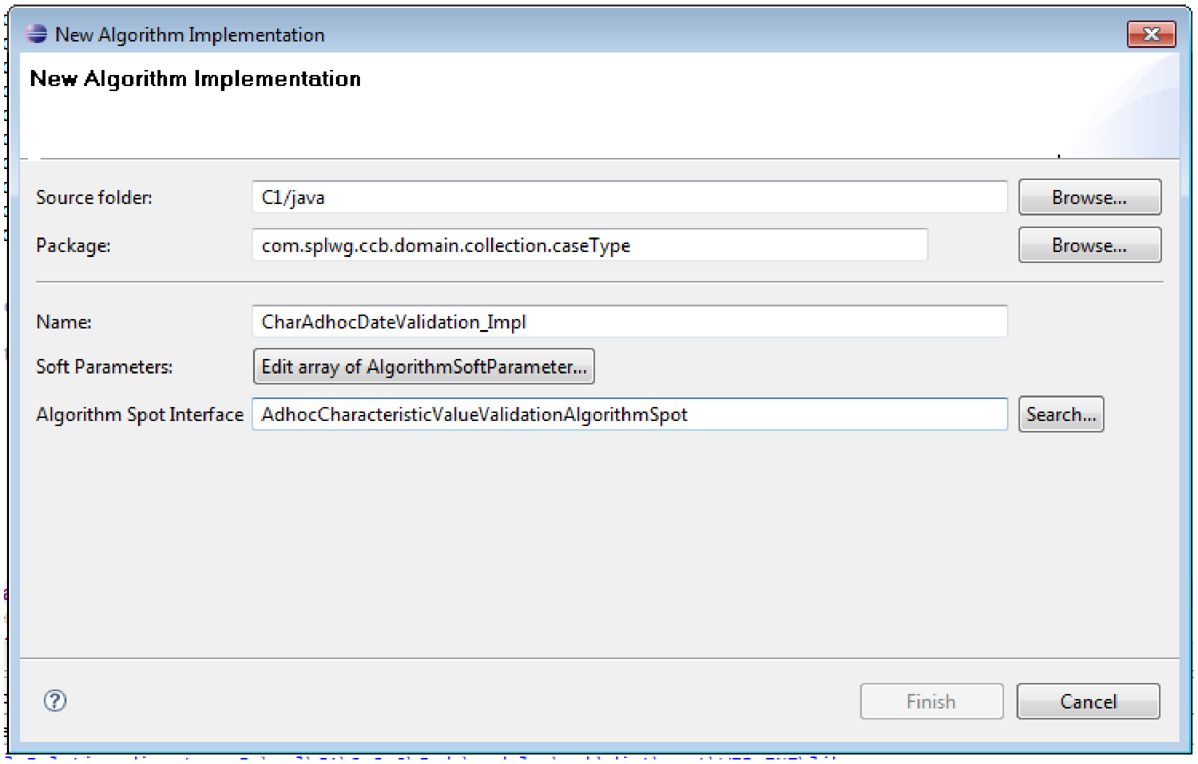
Algorithm Implementation Class:

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 22–6 New Algorithm Implementation



New Algorithm Implementation

Source folder: C1/java

Package: com.splwg.ccb.domain.collection.caseType

Name: CharAdhocDateValidation_Impl

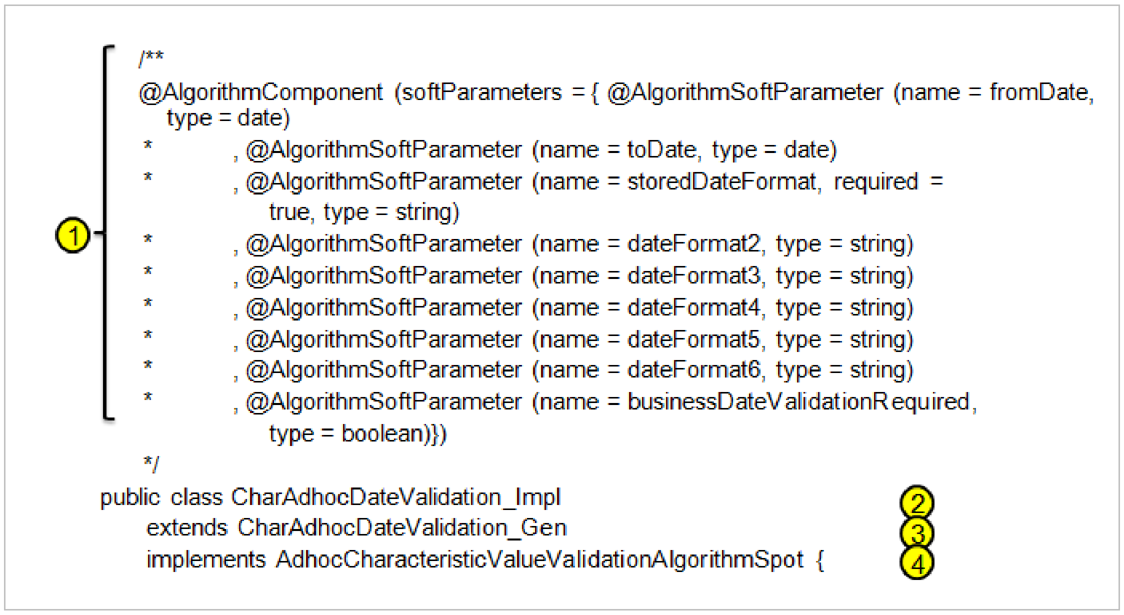
Soft Parameters: Edit array of AlgorithmSoftParameter...

Algorithm Spot Interface AdhocCharacteristicValueValidationAlgorithmSpot

Adhoc Characteristic Date Validation Example:

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

Figure 22–7 Adhoc Characteristic Date Validation



```
/**
@AlgorithmComponent (softParameters = { @AlgorithmSoftParameter (name = fromDate,
    type = date)
 *          , @AlgorithmSoftParameter (name = toDate, type = date)
 *          , @AlgorithmSoftParameter (name = storedDateFormat, required =
                true, type = string)
 *          , @AlgorithmSoftParameter (name = dateFormat2, type = string)
 *          , @AlgorithmSoftParameter (name = dateFormat3, type = string)
 *          , @AlgorithmSoftParameter (name = dateFormat4, type = string)
 *          , @AlgorithmSoftParameter (name = dateFormat5, type = string)
 *          , @AlgorithmSoftParameter (name = dateFormat6, type = string)
 *          , @AlgorithmSoftParameter (name = businessDateValidationRequired,
                type = boolean)})
 */
public class CharAdhocDateValidation_Impl
    extends CharAdhocDateValidation_Gen
    implements AdhocCharacteristicValueValidationAlgorithmSpot {
```

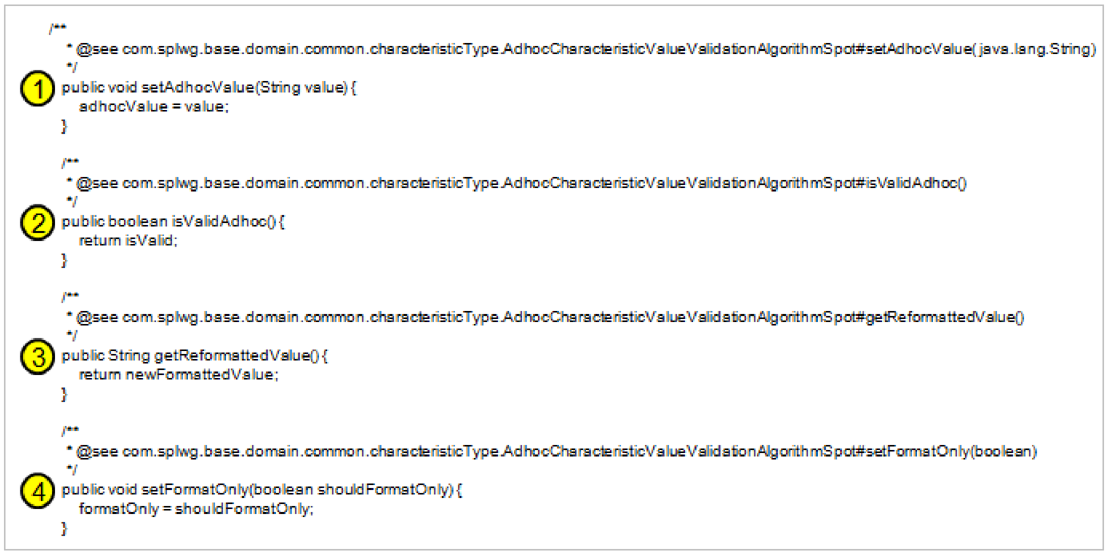
The annotations marked in the above diagram are explained as follows:

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:

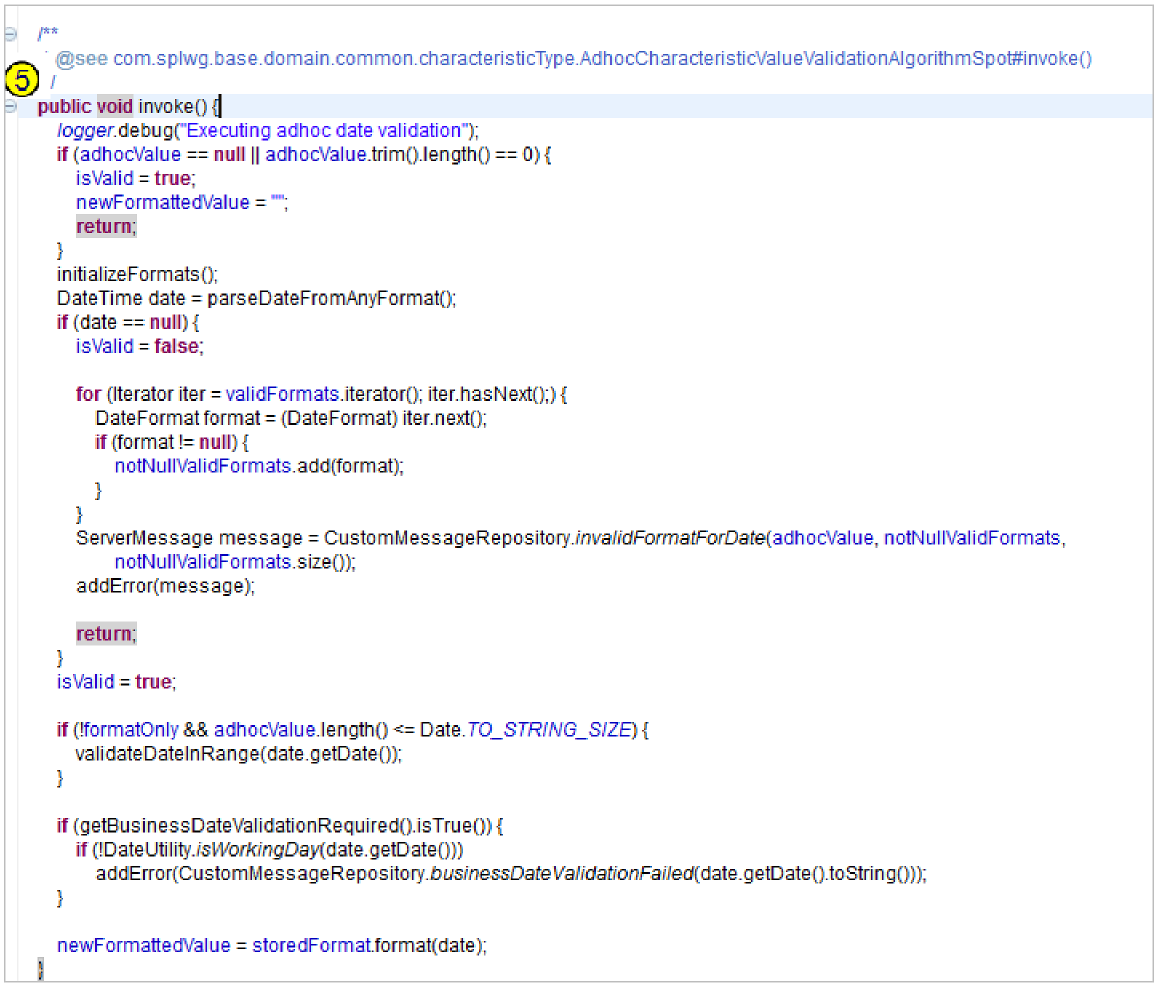
The following diagrams present the Algorithm Spot interface methods that are implemented in _Impl class.

Figure 22–8 Algorithm Spot Interface Methods



```
/**
 * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#setAdhocValue( java.lang.String)
 */
public void setAdhocValue(String value) {
    adhocValue = value;
}

/**
 * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#isValidAdhoc()
 */
public boolean isValidAdhoc() {
    return isValid;
}

/**
 * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#getReformattedValue()
 */
public String getReformattedValue() {
    return newFormattedValue;
}

/**
 * @see com.splwg.base.domain.common.characteristicType.AdhocCharacteristicValueValidationAlgorithmSpot#setFormatOnly(boolean)
 */
public void setFormatOnly(boolean shouldFormatOnly) {
    formatOnly = shouldFormatOnly;
}
```

Figure 22–9 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);
}
```

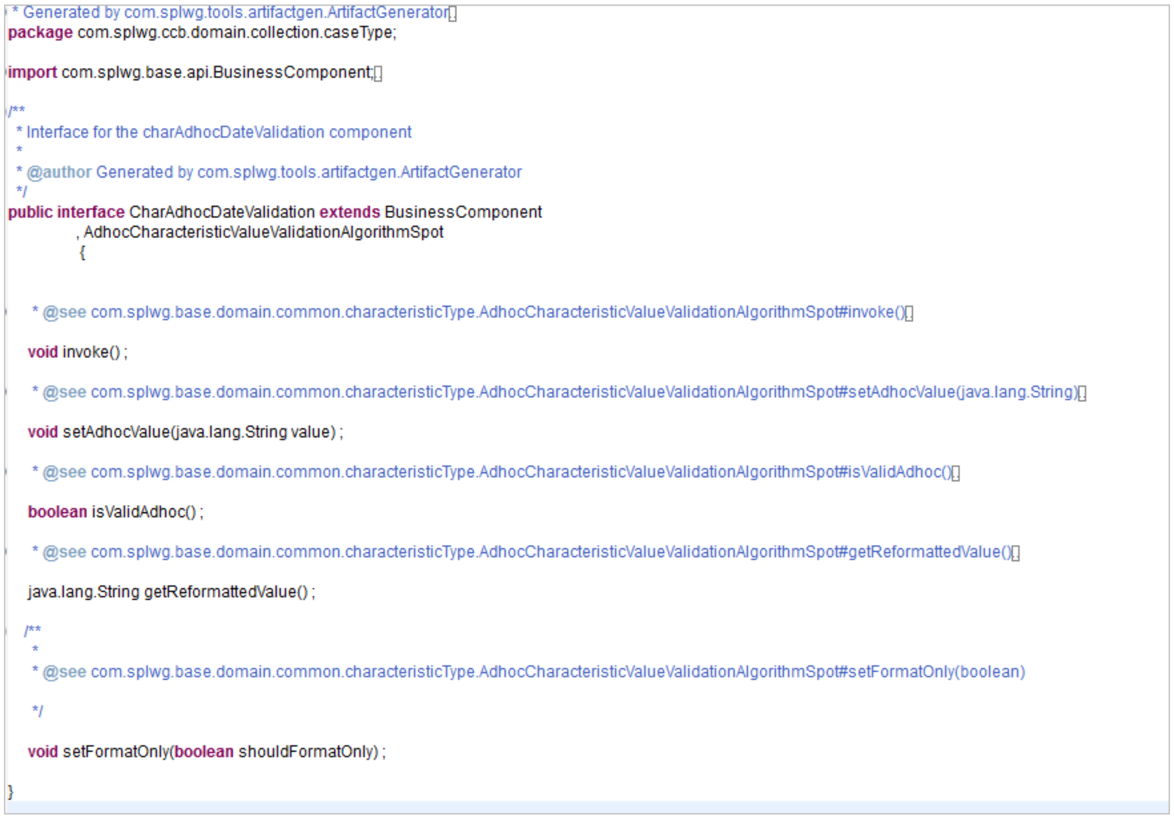
The annotations marked in the above diagrams are explained as follows:

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

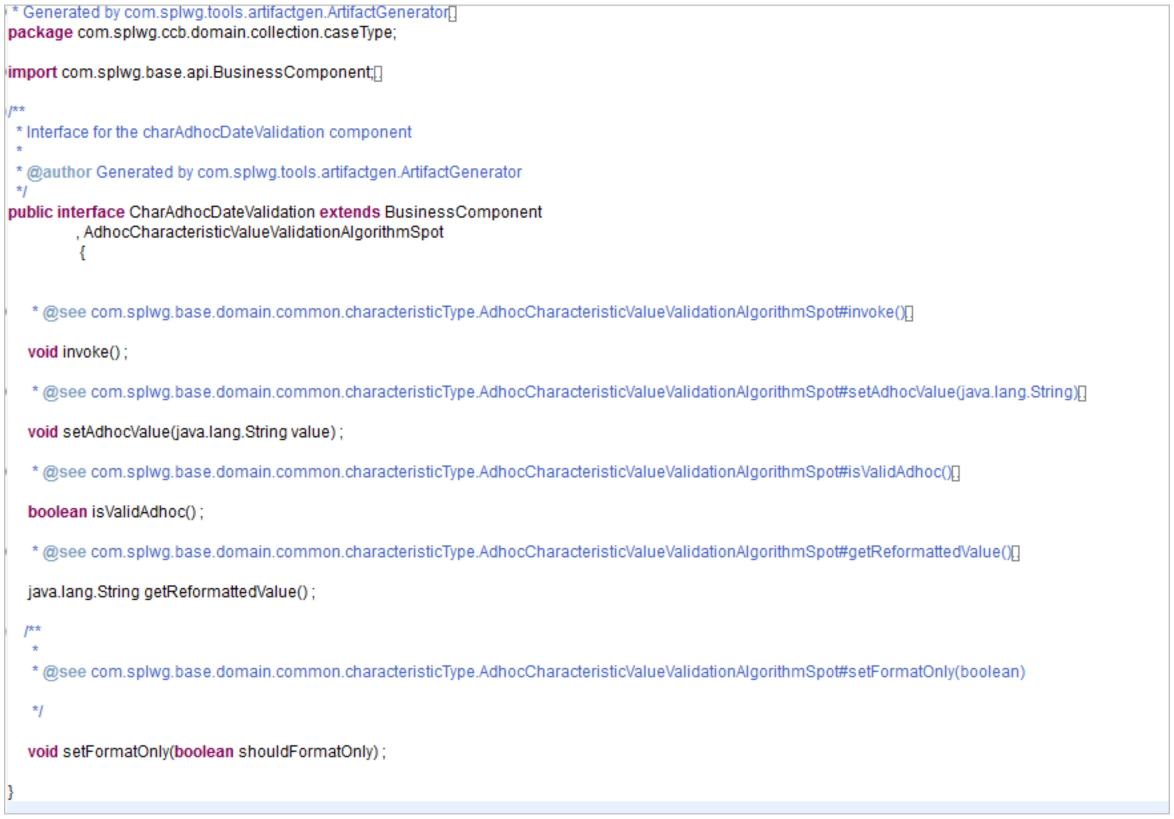
Figure 22–10 Generated Artifacts



```
 * 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 22–11 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:

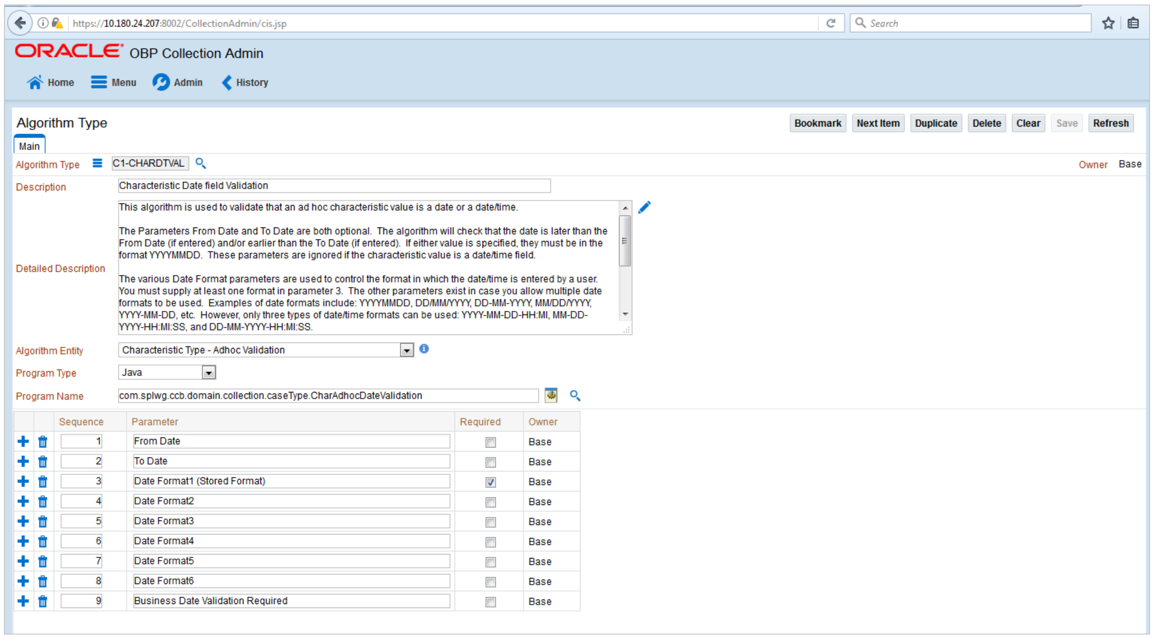
1. Determine the Algorithm Spot interface name. The Javadocs can be used for this.
2. Create the "_Impl" class, implementing the appropriate Algorithm Spot interface.
3. Add default implementations for all the Algorithm Spot methods (For example, using the Eclipse *Source, Override/implement Methods*… 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 22–12 Create Algorithm Type



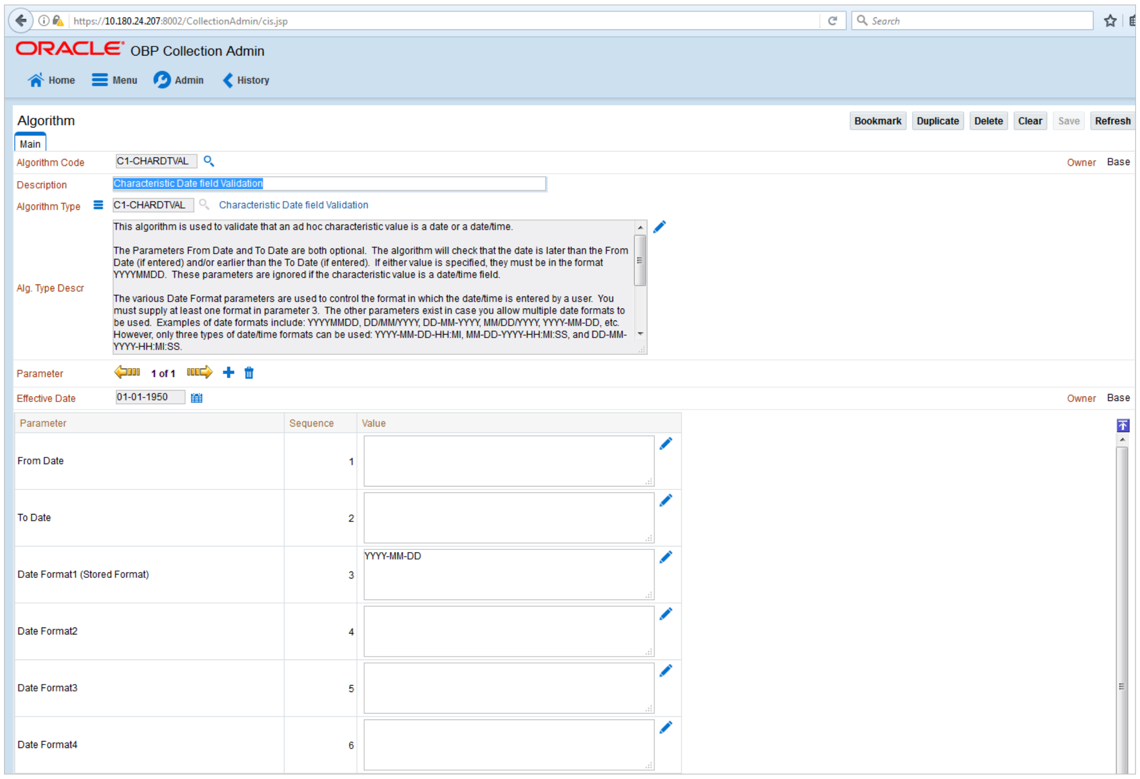
Algorithm Type

Algorithm Type	C1-CHARDTVAL
Owner	Base
Description	Characteristic Date field Validation
Detailed Description	This algorithm is used to validate that an ad hoc characteristic value is a date or a date/time. The Parameters 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 entered). If either 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 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 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, etc. However, only three types of date/time formats can be used: YYYY-MM-DD-HH:MI, MM-DD-YYYY-HH:MI:SS, and DD-MM-YYYY-HH:MI:SS.
Algorithm Entity	Characteristic Type - Adhoc Validation
Program Type	Java
Program Name	com.splwg.ccb.domain.collection.caseType.CharAdhocDateValidation

Sequence	Parameter	Required	Owner
1	From Date	☐	Base
2	To Date	☐	Base
3	Date Format1 (Stored Format)	✓	Base
4	Date Format2	☐	Base
5	Date Format3	☐	Base
6	Date Format4	☐	Base
7	Date Format5	☐	Base
8	Date Format6	☐	Base
9	Business Date Validation Required	☐	Base

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

Figure 22–13 Attach Algorithm


https://10.180.24.207:8002/CollectionAdmin/cis.jsp
Search
ORACLE OBP Collection Admin
Home
Menu
Admin
History
Bookmark
Duplicate
Delete
Clear
Save
Refresh
Main

Algorithm

Algorithm Code	C1-CHARDTVAL
Owner	Base
Description	Characteristic Date field Validation
Algorithm Type	C1-CHARDTVAL Characteristic Date field Validation
Alg. Type Descr	This algorithm is used to validate that an ad hoc characteristic value is a date or a date/time. The Parameters 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 entered). If either 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 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 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, etc. However, only three types of date/time formats can be used: YYYY-MM-DD-HH:MI, MM-DD-YYYY-HH:MI:SS, and DD-MM-YYYY-HH:MI:SS.
Parameter	1 of 1
Effective Date	01-01-1950
Owner	Base

Parameter	Sequence	Value
From Date	1	
To Date	2	
Date Format1 (Stored Format)	3	YYYY-MM-DD
Date Format2	4	
Date Format3	5	
Date Format4	6	

22.4 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.

Table 22–1 List of Algorithm Spots

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
AdhocCharacteristicValueValidation Algorithm Spot	This algorithm spot is i	void setFormatOnly (boolean formatOnly); void setCharacteristicType (CharacteristicType type); void setAdhocValue (String value); String getReformattedValue();	com.splwg.ccb.domain.collection.caseType.CharAdhocDateValidation	com.splwg.ccb.domain.collection.caseType.CharAdhocDateValidation_Impl	Characteristic Date field Validation: C1-CHARDTVAL	This algorithm is used to validate that an ad hoc characteristic value is a date or a date/time. The Parameters From

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	nvokedoncharacteristicadh	boolean isValidAdhoc();				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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	) validate that the value is corr					if the characteristic value is a date/time field. The various Date Format parameters are used to control the format in which the date/time is entered

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e					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 YYYY-MM-DD-HH:MI:SS for date/tim

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CureEntityAlgorithmSpot	This algorithm spot is used to cur	void setAccountId (Account_Id acctId);	com.splwg.ccb.domain.collection.batch.algorithm.CureEntityAlgorithm	com.splwg.ccb.domain.collection.batch.algorithm.CureEntityAlgorithm_Impl	Cure Account: C1-FINCOLL	This algorithm 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 CI_PARTY_

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e the account.					COLLECT as posting date. - Update number of times account is self cured (used for statistics). - Remove strategy review date. Parameter: contact Methods: This soft parameter accept the comma separate

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
SaTypeSaStopAlgorithmSpot	This algorithm spot is used to sto	void setServiceAgreement (ServiceAgreement serviceAgreement);	com.splwg.ccb.domain.collection.batch.algorithm.FinalizeCollectionContractStopAlgoComp	com.splwg.ccb.domain.collection.batch.algorithm.FinalizeCollectionContractStopAlgoComp_Impl	Stop Contract: C1-CURENTITY	This algorithm 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
	pthecontarct.					
AllocationGroupQueueAlgorithmSpot	This algorithm sp	void setAllocationGroup (String allocationGroup); void setBusinessDate (Date	Com.splwg.ccb.domain.collection.batch.algorithm.AllocationGroupQueueAlgoComp	com.splwg.ccb.domain.collection.batch.algorithm.AllocationGroupQueueAlgoComp_Impl	Queue Allocation: C1-ALLOCQU	This Algorithm type is used to allocate the entities such as cases to queues. ci_allocation_monitor_vw view

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	otis used to allocate the entities.	businessDate); void seToQueueBean(AllocationGroupCasesToQueueBean caseAllocToQueue); AllocationGroupCasesToQueueBean getCaseToQueueBean(); AllocationGr			EU	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
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algorithm	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.batch.algorithm.CustomerLevelSwitchUpdateAlgorithm	com.splwg.ccb.domain.collection.batch.algorithm.CustomerLevelSwitchUpdateAlgorithm_Impl	UpdateCustomerSwitch: C1-CUSTSW	Update customer level case status on case enter processing. Customer Level Switch Name: Please enter the customer level case status switch which needs to update. eg. BANKRUPT_SW,

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	spotistoexecutethebusinesslo	getNextTransCondition()				HARDSHIP_SW, IMPRISONED_SW, DECEASED_SW, ABSCONDING_SW etc Switch Value: Please enter the switch value as Y or N

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	pecific status.					
CaseTypeEnterStatusAlgorithmSpot	Thepu	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseS	com.splwg.ccb.domain.collection.batch.algorithm.RepoAndLegalCaseUpdateAlgorithm	com.splwg.ccb.domain.collection.batch.algorithm.RepoAndLegalCaseUpdateAlgorithm_Impl	Update Legal/Repo Switch: C1-	Algorithm Type to update Legal and Repo case status on enter process Legal Repo Switch Name:

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	rpose of the algorithm sp	tatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTransCondition()			LEREPOCT	Please enter the Legal or Repo case switch 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	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	toexecutethebusinesslo					

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	status.Thespecificsamplealg					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
UserAllocationAlgorithmSpot	This spot being used for allocatio	void setUserToQueueBean (UserAllocationToQueueBean userAllocToQueue); UserAllocationToQueueBean getUserToQueueBean(); UserAllocationToQueueBean	com.splwg.ccb.domain.collection.batch.algorithm.UserAllocationRoundRobinAlgorithm	com.splwg.ccb.domain.collection.batch.algorithm.UserAllocationRoundRobinAlgorithm_Impl	User Allocation - Round Robin: C1-USRALCRR	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 User.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	nofuserusingvariousalgorithms.	getUserAllocationMap();				

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
UserAllocationAlgorithmSpot	This spot being used for allocation	void setUserToQueueBean (UserAllocationToQueueBean userAllocToQueue); UserAllocationToQueueBean getUserToQueueBean(); UserAllocationToQueueBean getUserAllo	com.splwg.ccb.domain.collection.batch.algorithm.UserAllocationPercentageBaseAlgorithm	com.splwg.ccb.domain.collection.batch.algorithm.UserAllocationPercentageBaseAlgorithm_Impl	User Allocation - % Based: C1-USRALCPR	User Allocation Percentage based algorithm type allocates 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	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	ofuserusingvariousalgorithms.	cation Map();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
VendorAllocationAlgorithmSpot	This spot being used for allocation	void setVendorToQueueBean (VendorAllocationToQueueBean vendorAllocToQueue); VendorAllocationToQueueBean getVendorToQueueBean (); VendorAllocationToQueueBean	com.splwg.ccb.domain.collection.batch.algorithm.VendorAllocationRoundRobinAlgorithm	com.splwg.ccb.domain.collection.batch.algorithm.VendorAllocationRoundRobinAlgorithm_Impl	Vendor Allocation - Round Robin: C1-VENALCRR	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 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
	of vendor using various algorithms.	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 used for allocation	void setVendorToQueueBean(VendorAllocationToQueueBean vendorAllocToQueue); VendorAllocationToQueueBean getVendorToQueueBean(); VendorAllocationToQueueBean	com.splwg.ccb.domain.collection.batch.algorithm.VendorAllocationPercentageBaseAlgorithm	com.splwg.ccb.domain.collection.batch.algorithm.VendorAllocationPercentageBaseAlgorithm_Impl	Vendor Allocation - % Based: C1-VENALCPR	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 User of the queue.

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
	of vendor using various algorithms.	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 is used for cr	void setAccountId (String accountId); void setContactClass (String contactClass); void setContactTypeCode (String contactTypeCode); void setMo	com.splwg.ccb.domain.collection.batch.algorithm.BulkContactCreationAlgoComp	com.splwg.ccb.domain.collection.batch.algorithm.BulkContactCreationAlgoComp_Impl	BulkContactCreation: C1-BLKCNTCRE	This algorithm type is called from Bulk Contact Creation Batch. It invokes business service 'C1-GenMultipleCorrespondence' which creates a customer contact for the accounts filtered by the condition builder attached

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	eation of contact for accounts in bul	de (String mode); void setCharacteristicType (String characteristicType); void setCharacteristicValue (String characteristicValue); void setJointNominationFor				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
CrossStrategyActionMatrix Algorithm Spot	This algorithm spot is u	void setCase (ToDoCase toDoCase); void setCaseOriginalStatus (CaseStatus caseStatus); String getNextCaseStatus();	com.splwg.ccb.domain.collection.batch.algorithm.CrossStrategyActionMatrixAlgorithm	com.splwg.ccb.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 on

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	sed to execute the businessl					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
	into a particular state and all the					

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

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	m spot is to call an algorithm d	saction nType); void setRegularFinancialTransaction (FinancialTransaction regularFinancialTransaction); Bool getFinancialTransactionProcessAdded();				

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	orithm forFT FreezeSystem Even					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the alg	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.CheckAssociationReview	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.CheckAssociationReview_Impl	Association Review Check: C1-ASORVCHK	This is to decide if the system association of entities should be reviewed by the user or not. Soft Parameters: Next Status: Values Possible for Next Status {ASSNEWLSP}. This is

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithm spot is to execute t	getNextTransCondition()				applicable if Association Review Required is set N. Association Review Required= Possible Values {Y,N} If Association Review is Required {Y}- Stay in current status for user review. Set display

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

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	asetoitasFK Characteristic					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algori	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.DefaultNoticeExpiryCheck	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.DefaultNoticeExpiryCheck_Impl	Validate Expired Default Notice: C1-DEFNOEXP	System should check that for associated accounts default notice has expired, This check can be for primary account or for all associated delinquent account based on parameter. 1.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	thm spot is to execute the	sCondition()				Association Type={P,A}.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 N. 3. To Do Role= To Do Role for

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	ismovedintospecificstatus.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	FK Characteristic					
CaseTypeEnterStatusAlgorithmSpot	Th	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.CheckLegalCase	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.CheckLegalCase_Impl	AssociateRelatedEntity:	The algorithm checks the associated

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	epurpose of the algorith	(Case Status caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTransCondition()			C1-ASSOENTY	accounts of the primary account. The association of the primary account is done on the basis of the persons attached to the account and their financially responsible status.if the account

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spotistoexecutethebusi					nsible persons attached as in the case for the primary account, the accound is associated. The algorithm parameter are as follows: 1)To Do Role:Specifies the role for the To Do Type created in case of any

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	intospecificstatus.Thespe					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	tic					
CaseTypeEnterStatusAlgorithmSpot	The purpose of the al	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.CheckLegalCase	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.CheckLegalCase_Impl	Validate Legal Case Exists: C1-CHKLGL	The Algorithm checks if there is already open legal case for the primary account/ Associated accounts linked to the case.The algorithm takes the paramet

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	gorithm spot is to execut	String getNextTransCondition()				ers as follows: 1)To Do Role:Specifies the Role for the To Do Type. 2)To Do Type:Specifies the todo type created when the legal case has been created from batch mode and there is open legal case for

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

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
	nks the Case to its FK Characteristic					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algori	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition () String getNextCaseStatus() String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.AssignNewLSP	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.AssignNewLSP_Impl	Assign New LSP: C1-ASGNLSP	This algorithm will assign a new LSP to the current case. LSP is a external vendor which is mapped LEGAL service

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	thm spot is to execute the	sCondition()				Type. If manual review is not required then case will automatically transition to next status metntioned in softparameter.Below are the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into specific status.					on Check: Possible values {Y, N}. If this switch is Y system will check if a legal case was created for any of the accounts associat

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	FK Characteristic					allocation without review option. MANUAL=Manual allocation. System will not allocate LSP. 5. New Allocation And Review Option= Possible values {AUTO_WITH_REVIEW,AUT

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the halg	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.CreateApprovalRequest	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.CreateApprovalRequest_Impl	CheckApprovalRequirement:C1-APPRCHK	This algorithm creates approval request if required based on certain conditions.This process will check if LSP assignment needs to be approve

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
	orithm spot is to execute t	getNextTransCondition()				d if LSP assignment status = "Pending Approval" Approval would be required if either of below is true: - System allocation override by user i.e. user has changed the LSP assigned by the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	Case is moved into specific st					a specified threshold. 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

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	asetoitasFK Characteristic					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algori	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.ResumeStatusLSP	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.ResumeStatusLSP_Impl	ResumeStatusfromPreviousLSP:C1-RESSTATUS	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
	thm spot is to execute the	sCondition()				

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	ismovedintospecificstatus.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	FK Characteristic					
CaseTypeExitStatusAlgorithmSpot	The pur	void setCase (ToDoCase toDoCase); void setNextCaseStatus (Case	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.CheckSubmissionDateExitProcessing	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.CheckSubmissionDateExitProcessing_Impl	CheckSubmissionDate:C	Check Submission Date

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	poseofthealgorithm spotist	Status caseStatus);			1-CHKSUBDT2	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	rm additional logic when a Caset					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	extstatus.					
CaseTypeEnterStatusAlgorithmSpot	The purpos	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldA	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.UpdateLSPAssignment	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal.UpdateLSPAssignment_Impl	Update LSP (CLOS): C1-LSPST	Set LSP assignment status to value provided in the parameter. This should be done only for Latest LSP assignment and if it was

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e of the algorithm spot is	utoTransition() String getNextCaseStatus() String getNextTransCondition()			ATUS	done by current legal case. If Status = Closed or Cancelled set Assignment End date = Business Date Status possible values {CLOS, REJ,CAN,PNAP} CLOS= Closed REJ=Rejected PNAP= Pending for Approva

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	thebusinesslogicwhenCa					

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	samplealgorithm createsToDoen					

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
CaseTypeEnterStatusAlgorithmSpot	Thepurposeofthealg	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.legal		ValidateExpiredDefaultNotice:C1-DEFNOEXP	System should check that for associated accounts default notice has expired, This check can be for primary account or for all associated delinquent account based on

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithm spot is to execute t	getNextTransCondition()				parameter. 1. Association Type={P,A}.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 N. 3. To Do Role=

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	Caseismovedintospecificst					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ase to it as FK Characteristic					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algori	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.CollateralVerification	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.CollateralVerification_Impl	Collateral Verification: C1-VRFYCOLS	This will perform following validations for the collateral with the case: - If soft parameter Collateral type to this algorithm type is "PROPERTY" then, Only one collateral is associated with

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	thm spotistoexecutethe	sCondition()				the case also that Collateral is associated with Facility for the primary account associated with the case. - If collateral type soft parameter is blank, then above validation should be ignored and Collater

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into specific status.					should be terminated

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	FK Characteristic					
CaseTypeEnterStatusAlgorithmSpot	Th	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.AccountAssociationForAssetRepossessionCase	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.AccountAssociationForAssetRepossessionCase_Impl	Account Association for	This algorithm will perform following actions: - It will get all facilities to which

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	epurpose of the algorith	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTransCondition()			AssetRepossessionCase: C1-ARSACCTS	this collateral is associated also it will get all accounts for these facilities. - It will Associate these accounts with the case. Scope of this association is limited to accounts already in collectio

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	spotistoexecutethebusi					oesn't have any soft parameter.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	intospecificstatus.Thespe					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	tic					
CaseTypeEnterStatusAlgorithmSpot	The purpose of the al	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.CustomerAssociationForAssetRepossessionCase	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.CustomerAssociationForAssetRepossessionCase_Impl	CustomerAssociationforAssetRepossessionCase:C	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 Scope of this association is limited

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	gorithm spot is to execut	String getNextTransCondition()			1-ARSCUSTS	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	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	whenCaseismovedintospecif					

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	nks the Case to it as FK Characteristic					

Algorithm Spot	**Spot Detail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**Collections Algorithm Description and Code**	**Algorithm Summary**
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algori	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralProperty	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralProperty_Impl	Update Collateral Property: C1-UPCOLPROP	This algorithm will perform foolowing operations: 1)if the value of updateCollateral Property soft parameter is "SET" and type of possession is "Warrant" then Fetch the collateral for

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	thm spot is to execute the	sCondition()				which case is created and update the IS_LEGAL_SW= "Y" and populate the case_id on this collateral. 2)if the value of updateCollateral Property soft parameter is "RESET" then Fetch the collateral for which

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into specific status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	FK Characteristic					
CaseTypeExitStatusAlgorithmSpot		void setCase (ToDoCase toDoCase); void setNextCaseStatus (Case	com.splwg.ccb.domain.collection.caseType.specialisedCollections.CloseTodo	com.splwg.ccb.domain.collection.caseType.specialisedCollections.CloseTodo_Impl	Close Todo's Algorithm:	This process will close all To-Do's of specific To-do types associated with the case.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
		Status caseStatus);			C1-CLSTODO	Up to 5 To-DO types can be given to this algorithm to close.
CaseTypeEnterStatusAlgorithmSpot	The purpose	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTra	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.MandatoryCharacteristics	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.MandatoryCharacteristics_Impl	Validations for Mandatory Characteristics:	Subjective Validations for Mandatory Characteristics: This process will validate specified characteristics to

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	of the algorithm spot is t	nsition() String getNextCaseStatus() String getNextTransCondition()			C1-CHARVALS	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 and up to 5 validation

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	ebusinesslogicwhenCasei					ic, system should validate that mandatory characteristic types have some value captured. If the parameter specifying mandatory characteristic type is blank, it should be ignored

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
	algorithm createsToDoentryand					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the alg	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.Mandatory Characteristics	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.MandatoryCharacteristics_Impl	Validations for MandatoryCharacteristics:CI_CHARVAL	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
	orithm spot is to execute t	getNextTransCondition()				selected by the user for one of the characteristics. This algorithm will have reference characteristic type and up to 5 validation characteristic type as parameters,So based on the reference

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	Caseismovedintospecificst					pecifying mandatory characteristic type is blank, it should be ignored

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	asetoitasFK Characteristic					

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algori	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralStatusInTheHost	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralStatusInTheHost_Impl	Update Collateral Status in the Host: C1-CHARVALZ	Subjective Validations for Mandatory Characteristics: This process will validate specified characteristics to be present on the case with reference to value selected by the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	thm spot is to execute the	sCondition()				user for one of the characteristics. This algorithm will have reference characteristic type and up to 5 validation characteristic type as parameters,So based on the reference characteristic

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	ismovedintospecificstatus.					tic 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
	FK Characteristic					
CaseTypeEnterStatusAlgorithmSpot	Th	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralStatusInTheHost	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralStatusInTheHost_Impl	Initiate Collateral Valuation:	this alogrithm will work as follows: System

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	epurpose of the algorith	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTransCondition()			C1-COLLVALX	should check if "X" days have elapsed since the last assessment was done for the collateral. That is check if (Assessment date + X) <= Current business date. Number of days, X, will

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spot is to execute the busi					or this process. If yes - Create a To Do to alert the user that collateral valuation is required. This To Do should be associated with the case. To Do Type is passed as a parameter to 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. The spe					To Do should be assigned to the default role.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	tic					
CaseTypeExitStatusAlgorithmSpot	Thepurposeofthealgorit	void setCase (ToDoCase toDoCase); void setNextCaseStatus (CaseStatus caseStatus);	com.splwg.ccb.domain.collection.caseType.specialisedCollections.CloseTodo	com.splwg.ccb.domain.collection.caseType.specialisedCollections.CloseTodo_Impl	Close Todo's Algorithm: C1-CLSTODO	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	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	hm spot is to perform addition					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	itionsoutofthecurrentstatusto t					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the hea l g	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.Mandatory Characteristics	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.MandatoryCharacteristics_Impl	Validations for MandatoryCharacteristics:C1-CHARVALS	Subjective Validations for Mandatory Characteristics: This process will validate specified characteristics to be present on the case with reference to value

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	orithm spot is to execute t	getNextTransCondition()				selected by the user for one of the characteristics. This algorithm will have reference characteristic type and up to 5 validation characteristic type as parameters,So based on the reference

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	Case is moved into specifics t					pecifying mandatory characteristic type is blank, it should be ignored

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	asetoitasFK Characteristic					

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	Thepurposeofthealgori	void setCase (ToDoCase toDoCase); void setCaseOriginalStatus (CaseStatus caseStatus); Bool getShouldAutoTransition(); String getNextCaseStatus(); String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralStatusInTheHost	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralStatusInTheHost_Impl	Update Collateral Status in the Host: C1-CHARVALZ	Subjective Validations for Mandatory Characteristics: This process will validate specified characteristics to be present on the case with reference to value selected by the

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	thm spotistoexecutethe	sCondition();				user for one of the characteristics. This algorithm will have reference characteristic type and up to 5 validation characteristic type as parameters,So based on the reference characteristic

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into specific status.					tic 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
	FK Characteristic					
CaseTypeExitStatusAlgorithmSpot	The pur	void setCase (ToDoCase toDoCase); void setNextCaseStatus (Case	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.ValidateCollateralSettlementStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.ValidateCollateralSettlementStatus_Impl	Validation Settlement: C1-	This algorithm will perform following actions: Before completing the asset

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pose of the algorithm spot is t	Status caseStatus);			VALSET	repossession case below validations should be done for the case 1. Collateral should have a settlement date 2. Realization status for the collateral should be "REALIZATION_ COMPLETE" Possible

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	rm additional logic when a Case t					COMPLETE. 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
	extstatus.					
CaseTypeExitStatusAlgorithmSpot	The purpose of t	void setCase (ToDoCase toDoCase); void setNextCaseStatus (CaseStatus caseStatus);	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.InitiateLMIP	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.InitiateLMIP_Impl	Initiate LMI Proces s: C1-INITLMI	Parameters to the algorithm must be as follows: - For Initiate LMI Options: 1) "Initiate LMI with highest insured amount"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	healgorithm spot is to perfor					use HIGH_ INSUR_ AMT 2) "Initiate LMI from a specific insurer first" use SPEC_ INSUR ER. - For No LMI Option 1)"Mark primary account for strategy review" use PRIMA RY 2)"Mark all account s for strategy

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	icwhenaCasetransitionsoutofth					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusAlgorithmSpot	The purpose of the algorith	void setCase (ToDoCase toDoCase); void setNextCaseStatus (CaseStatus caseStatus);	com.splwg.ccb.domain.collection.caseType.specialisedCollections.CloseTodo	com.splwg.ccb.domain.collection.caseType.specialisedCollections.CloseTodo_Impl	Close To do's Algorithm: C1-CLSTODO	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
	m spot is to perform additiona					

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	ionsoutofthecurrentstatustothe					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the alg	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralProperty	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralProperty_Impl	Update Collateral Property: C1-UPCOLPROP	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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithm spot is to execute t	getNextTransCondition()				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 updateCollateral Property soft parameter is "RESET" then Fetch the collatera

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	Caseismovedintospecificst					

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	ase to it as FK Characteristic					

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algori	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition () String getNextCaseStatus() String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralStatusInTheHost	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.UpdateCollateralStatusInTheHost_Impl	Update Collateral Status in the Host: C1-UPCOLLSTS	Update Collateral Status in the host

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	thm spot is to execute the	sCondition()				

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	ismovedintospecificstatus.					

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**Collections Algorithm Description and Code**	**Algorithm Summary**
	FK Characteristic					
PtpActiveForNgp Algorithm Spot	This alg	void setPromiseToPay (PromiseToPay promiseToPay); PaymentPlanStatu	com.splwg.ccb.domain.customerinfo.paymentPlan.CollectionPTPActiveForNgpAlgorithm	com.splwg.ccb.domain.customerinfo.paymentPlan.CollectionPTPActiveForNgpAlgorithm_Impl	PTP Active Algorithm : C	This algorithm is used to perform additional

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithm spot is used for perform	sLookup getPaymentPlanStatus();			1-PTPACTIVE	processing when the status of a PTP becomes Active. Customer Contacts can be generated via this algorithm.Contact Class, method and type have to be specified. Following paramet

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	tional processing like generation o					ypeForLetter -- Contact Type for Letter. 2)contactClassForLetter -- Contact Class for Letter. 3)contactMethodForLetter -- Contact Method for Letter. (Value should be -- OTBL (Outbound Letter))

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
PtpKeptForNgpAlgorithmSpot	This algorithm spot is used fo	void setPromiseToPay (PromiseToPay promiseToPay); PaymentPlanStatusLookup getPaymentPlanStatus ();	com.splwg.ccb.domain.customerinfo.paymentPlan.CollectionPTPKeptForNgpAlgorithm	com.splwg.ccb.domain.customerinfo.paymentPlan.CollectionPTPKeptForNgpAlgorithm_Impl	PTPActiveAlgorithm: C1-PTPKEPT	This algorithm is used to perform additional processing when the status of a PTP becomes Kept. Customer Contacts can be generated via this algorithm. Contact

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	rperformingadditionalprocessin					Class, method and type have to be specified. Following parameters used to perform processing--- 1) contact TypeForLetter -- Contact Type for Letter. 2)contactClassForLetter -- Contact Class

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						ntactClassForSMS---Contact 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
PtpBrokenForNgp Algorithm Spot	This Algorithm spot is used for p	void setPromiseToPay (PromiseToPay promiseToPay); PaymentPlanStatusLookup getPaymentPlanStatus();	com.splwg.ccb.domain.customerinfo.paymentPlan.CollectionPTPBrokenForNgpAlgorithm	com.splwg.ccb.domain.customerinfo.paymentPlan.CollectionPTPBrokenForNgpAlgorithm_Impl	PTP Broken Algorithm: C1-BRKPTPNGP	This algorithm is used to perform additional processing when the status of a PTP is set to Broken. Customer Contacts can be generated via this algorithm. Following parameters used

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	erformingadditionalprocessingw					to perform processing--- 1) contact TypeForLetter -- Contact Type for Letter. 2)contactClassForLetter -- Contact Class for Letter. 3)contactMethodForLetter -- Contact Method for Letter. (Value should

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						ld be--OTBS (Outbound Short Message Service))
RuleFactsPopulationAlgorithmSpot	ThisAlgorithm spoti	void setInputKeyValue1 (String inputKayValue1); void setInputKeyValue2 (String inputK	com.splwg.ccb.domain.collection.RuleFactsPopulation	com.splwg.ccb.domain.collection.RuleFactsPopulation_Impl	Rule facts populating algorithm: C1-BR	This algorithm is used to populate the facts required for rule engine. Input Key Input Key 1 to 5 represen

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	s used for populating facts which char	ayValue2); void setInputKeyValue 3 (String inputKayValue3); void setInputKeyValue 4 (String inputKayValue4); void setInputKeyValue 5 (String inputK			LSR	t primary key of BO(used in Input BO name 1 - 5) Note: Currently you can use only Input key 1,2 and 3 and Input BO 1,2 and 3 Valid values in Input key and Input BO Input key 1 (Mandatory)

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ngine.	void setRuleEffectiveDate (String ruleEffectiveDate); void setFactDetails (CollectionsFactDetailsLoader collectionsFactDetailsLoader); RuleFactParameters getRuleFactParam				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 Input BO name 2 BO having primary key as input

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the alg	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition () String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssociateDelinquentAccount	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssociateDelinquentAccount_Impl	BorrowerCentricCaseLifecycle-C1-ASSODELAC	Associate new delinquent account of the customer to the case.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	orithm spot is to execute t	getNextTransCondition()				

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
	Caseismovedintospecificst					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	aseto it as FK Characteristic					
PreprocessBusinessObject Request Algorithm Spot			com.splwg.ccb.domain.collection.address.PersonCollectionAddressPreProcess	com.splwg.ccb.domain.collection.address.PersonCollectionAddressPostProcess_Impl	UpdateC	This is a reference implementation of Post

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					ollection Address on Borrower Panel-C1-PERADDPP	processing Algo. Customization team can utilize this hook.
PreprocessBusine			com.splwg.ccb.domain.collection.address.ContactPreferencePreProcess	com.splwg.ccb.domain.collection.address.ContactPreferencePreProcess_Impl	U	Contact

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ssObject Request Algorithm Spot					pdateCollectionContactPoint-C1-PCONTPRE	Point 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
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algori	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.CheckBankruptcyCaseExist	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.CheckBankruptcyCaseExist_Impl	Check if Special Case Already Exist on the Customer-Ent	Check if any active case is present of a given case category or case type on the customer Processing steps are as below 1. If only Case Category is specified check if any

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	thm spot is to execute the	sCondition()			er Processing: C1-CKSPLCASE	active case is running on the customer whose a. Case category is same as the 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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into specific status.					. 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
	FK Characteristic					
CaseTypeEnterStatusAlgorithmSpot	Th	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyPullNonDelinquentAcc	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyPullNonDelinquentAcc_Impl	Pull all the non delinquen	Processing steps are as below: - Pull all

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	epurpose of the algorith	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTransCondition()			t accounts of the customer into collections - Enter Processing: C	Not in Collections accounts into OB Collections (from OBP) where the associated customer is one of the borrower. - 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 the busi					er. If Account Relationships = FO consider all accounts where the customer is a financial owner. If Account Relationship = All consider all accounts where the customer is a financial or non

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	into specific status. The spe					, ALL 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
	tic					
CaseTypeEnterStatusAlgorithmSpot	The purpose of the al	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus()	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyAssociateAcc	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyAssociateAcc_Impl	Associate all accounts to the case where customer is a	Associate all accounts to the case where customer is a primary borrower For the primary customer associated with the case: - Get all accounts where this customer is

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	gorithm spot is to execut	String getNextTransCondition()			primary borrower-Enter Processing: C1-ASSCTEACC	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 case. - Associa

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

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	nks the Case to it as FK Characteristic					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algori	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyExcludeAccDlr	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyExcludeAccDlr_Impl	Exclude all the associated accounts from Dialer-Enter Processi	For all the accounts associated with the case: - Call the Dialer Exclusion Service to exclude the accounts from feed to Dialer

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	thm spot is to execute the	sCondition()			ng: C1-ExcAccDIr	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	ismovedintospecificstatus.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	FK Characteristic					
CaseTypeEnterStatusAlgorithmSpot	Th	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyInitiateCollateralValuation	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyInitiateCollateralValuation_Impl	InitiateCollateralVal	For each collateral on the associated account if last valuation was done 'X'

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e purpose of the algorith	(CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTransCondition()			uation for all collaterals whose last valuation was done 'X' days be	days before than create a Collateral Valuation Task. Enter the Collateral Code; Collateral Type and Collateral Description as Remarks Exclude Collaterals with Collateral Types specified in paramet

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spot is to execute the busi				C1-IniCltVal	es of Validation Date: POSTING DATE, SYSTEM DATE

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	intospecificstatus.Thespe					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	tic					
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used to	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitorChargeOffDelinquency	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitorChargeOffDelinquency_Impl	Monitor if any of the associated account need to be charge	If any of the associated account has delinquency Start Date = Today's posting date Create Bankruptcy Notification as: 'Account <Account Numbe

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	perform autotransitionprocessin				d of f and monitor delinquency-Monitoring: C1-MTRCRGDQ	r> has become Delinquent' Set Display Date of the case to current business date. Monitor Charge Off: If any of the associated account has DPD= Charge Off Threshold Create Bankrup

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
						= Yes than associated accounts with Secured Switch = Y should also be considered. Monitor Delinquency = "Y" or "N" ,Monitor Charge Off = "Y" or "N" ,Secured Accounts = "Y" or "N" Values of Validation Date:

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used to p	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitor341Hearing	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitor341Hearing_Impl	Notify the Bankruptcy Specialist on Hearing Dates-Monito	If 341 Hearing 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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	erform autotransition processing				ring: C1-MTR341HRG	Business Date = 341 Hearing 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 and is in

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						Notification: "Capture details of Objection Hearing for Debtors Proposed Plan" Set Display Date of the case to current Business Date Values of Validation Date: POSTING DATE, SYSTEM DATE

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used to per	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitorPaymentPlan	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitorPaymentPlan_Impl	Monitor if the payment plan on any of the associated accounts	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 specified in the parameter should

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	form autotransition processing fo				is Broken for more than x days-Monitoring: C1-MTRPYMPL	be considered Notification: <PTP Type> broken for account <Account Number>. Days since plan broken <Days Since PTP Broken>. Set Display Date of the case to current business date. Values

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the alg	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitorAssetLiquidation	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitorAssetLiquidation_Impl	Notify the Bankruptcy Specialist if the Liquidation reache	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 create a

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	orithm spot is to execute t	getNextTransCondition()			s a specific status-Monitoring: C1-MNTRASLQD	notification for the Bankruptcy Specialist. Notification: "Liquidation for Account <Account Number>; Collateral <Collateral Code> has reached status <Case Status> Set Display Date of the

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	Case is moved into specific st					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ase to it as FK Characteristic					
CaseTypeAutoTransitionAlgorithmSpot	Th	void setCase (ToDoCase	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitorHearingDate	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitorHearingDate_Impl	Notify th	If for any of the associated

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is algorithm type is used to perfo	toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();			eBankruptcy Specialist on RFS Hearing Date-Monitoring: C1-	account on the case if the RFS Hearing Date is reached Create Notification: "Capture details for RFS Hearing for Account <Account Number> When Business date = Hearing Date + 1 Set Display Date of the case to current

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	autotransitionprocessingforaCa					ation Date: POSTING DATE, SYSTEM DATE

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
ResultTypePostProcessing Algorithm Spot	This Algorithm spot decides	void setActionEntity (String actionEntity); void setActionSourceId (String actionSourceId); void setActionSourceStatusCode (String actionSourceStatu	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.DetermineBankruptcyTreatment	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.DetermineBankruptcyTreatment_Impl	Determine in which status the case should proceed for B	Bankruptcy Chapter Field should be passed as a Filing Information Chapter (FC) or Converted to Chapter (CC) as an input parameter If Bankruptcy Chapter = Chapter

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	in which status transition has to b	sCd); void setActionId(String actionId); void setActionType(ActionType actionType); void setResultType(ResultType resultType); boolean getIsProcessingComplete();			ankruptcy Treatment-Post Processing C1-DTMBKTRTM	7 Then Transition to Manage Chapter 7 Bankruptcy Status If Bankruptcy Chapter = Chapter 13 Then Transition to Manage Chapter 13 Bankruptcy Status If Bankruptcy Chapter = Chapter other

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	f result.					
ResultTypePostProcessing Algorithm Spot	This Algorithm spot	void setActionEntity (String actionEntity); void setActionSourceId (String actionSourceId); void setActionSo	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.ValidateBankruptcyCaseData	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.ValidateBankruptcyCaseData_Impl	Validate if appropriate Case Details have	Validate if the Dynamic Panel

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	decides in which status transitio	urceStatusCode (String actionSourceStatusCd); void setActionId (String actionId); void setActionType (ActionType actionType); void setResultType (ResultType resultType);			been entered by the user-Post Processing C1-VLDBCDAT	Fields mentioned for the corresponding Dynamic panels have some values for the case.If yes the

Algorithm Spot	**Spot Detail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**Collections Algorithm Description and Code**	**Algorithm Summary**
	edon processing of result.					y. If no system should throw an error message for the first blank field that it will encounter. Error Messag e:"<Fiel

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
						esponding Panel Fields: ENTITY_ NAME, PHONE ,EMAIL, FAX_ NUMBER,CONTACT_ POINT_ NAME, CONTACT_ POINT_ PHON

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						rresponding Panel Fields: DATE_OF_BNKPT_CASE_FILE,BNKPT_CASE_NUM,COURT,CHAPTER Panel Name:bankruptcyConfirmPlanInformationPanel Corresponding Panel

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
Business ObjectEnterStatus Algorithm Spot			com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyNotifyPaymentPlanKept	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyNotifyPaymentPlanKept_Impl	NotifyBankruptcySpecialistwhenaPaymentPlanstat	Create Notification Notification: <PTP Type> Kept for account <Account Number>. Set Display Date of the case to current business date.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					usbecomesKept-PostProcessingC1-NTPYMPLNK	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
ToDoTypeToDoPostProcessAlgorithmSpot	ThisAlgorithm spot is used forn		com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyNotifyTaskCompletion	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyNotifyTaskCompletion_Impl	NotifyBankruptcySpecialist of TaskCompletion-Post Pr	Create Notification Notification: <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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	otifying task completion and also fo				ocessing C1-NTFTSKCMP	associated on the Account

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ToDoTypeToDoPostProcessAlgorithmSpot	This Algorithm spot is used fo		com.splwg.ccb.domain.collection.vendor.VendorManagementAutomaticTaskAllocation	com.splwg.ccb.domain.collection.vendor.VendorManagementAutomaticTaskAllocation_Impl	Vendor Management - Automatic Allocation of tasks	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 the queue.

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	r notifying task completion and also				to Vendors - TODOType - Post Processing C1-TSKVNDR	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the alg	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.financialHardship.HardshipAssociation	com.splwg.ccb.domain.collection.caseType.specialisedCollections.financialHardship.HardshipAssociation_Impl	Hardship - Associate Accounts of Main Customer - Ent	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
	orithm spot is to execute t	getNextTransCondition()			erProcessingC1-HARASOPND	

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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	aseto it as FK Characteristic					
CaseTypeAutoTransitionAlgorithmSpot	Th	void setCase (ToDo Case	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitorHearingDate	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.BankruptcyMonitorHearingDate_Impl		

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	isalgorithm type is used to perfo	toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();				

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	autotransitionprocessingforaCa					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used to p	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.scra.algorithm.ActiveServiceAlgorithm	com.splwg.ccb.domain.collection.scra.algorithm.ActiveServiceAlgorithm_Impl	C1-ACTMEMCHK	This algorithm will Transit the case to 'Suspend Status' if the customer is in Active Service or dependent of a person in Active Service. Validate against all Financial Owners parameter will decide if

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	erform autotransition processing					check has to be done for main customer or all financial owners. If Validate against all Financial Owners parameter value is Y, algorithm will check active service member against all financial owners.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
GenericEventHostUpdateAlgorithmSpot	This is generic algorithm spo	void setPerson(Person person); void setToDoCase(ToDoCase toDoCase); void setAccount(Account account);	com.splwg.ccb.domain.collection.loan.UpdateDisputeFlagAlgorithm	com.splwg.ccb.domain.collection.loan.UpdateDisputeFlagAlgorithm_Impl	Generic Algorithm to update host flag through event manager	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
	twhichcanbeusedtogenerateGen				Code-C1-EVTHSTUPD	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	thmSpot					
ResultTypePostProcessing Algorithm Spot	This Algorithm spotd	void setActionEntity (String actionEntity); void setActionSourceId (String actionSourceId); void setActionSourceSt	com.splwg.ccb.domain.collection.algorithms.ScheduleCallPostProcessingAlgorithm	com.splwg.ccb.domain.collection.algorithms.ScheduleCallPostProcessingAlgorithm_Impl	Code - C1-SCHCALL	This algorithm is used to fulfil request by customer to collector for calling at specific time. - The Call Back Time will get saved

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ecidesinwhichstatustransition	atusCode (String actionSourceStatusCd); void setActionId (String actionId); void setActionType (ActionType actionType); void setResultType (ResultType resultType); boolea				as the Next Action Time on the 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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	on processing of result.					
CaseTypeEnterStatusValidationAlgorithmSpot	Th	void setCase (ToDoCase toDoCase); void	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.ValidateDemandLetterandAccelerationLetter	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.ValidateDemandLetterandAccelerationLetter_Impl	Code - C1-VALI	If DL Template Code has

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e purpose of the algorithm spot	setOriginalCaseStatus(CaseStatus caseOriginalStatus);			DDLAL	been mentioned validate if Demand Letter has been sent in last X days. If AL Template Code has been mentioned validate if Acceleration Letter has been sent in last X

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	toexecutethevalidationlogicbe					pecified just check if the letters have been sent on the account. Checks will be done for all associated accounts unless 'Only Primary Account = Yes' in which case the check will be only on primary associated

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	tus.					
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algorit	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCas	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.ActiveMilitaryServiceCheckonAssociatedCustomers	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.ActiveMilitaryServiceCheckonAssociatedCustomers_Impl	Code - C1-BLOCKREPO	If any of the customers associated with the repossession case satisfy below criteria block repossession initiation. The customer is a Service Member and The

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	hm spot is to execute the busines	eStatus() String getNextTransCondition()				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 missing in

Algorithm Spot	**Spot Detail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**Collections Algorithm Description and Code**	**Algorithm Summary**
	into specific status.					
CaseTypeAutoTransitionAlgorithmSpot	This al	void setCase (ToDoCase toDoCase); Bool getShouldA	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.MonitorDemandLetterandAccelerationLetterExpiry	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.MonitorDemandLetterandAccelerationLetterExpiry_Impl	Code - C1-MNTRD	If DL Template Code has been mention

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	gorithm type is used to perform a	utoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();			LAL	ed validate if Demand letter has been sent and current date > Demand Letter Expiry Date. If AL Template Code has been mentioned validate if Acceleration letter has been

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	nsition processing for a Case.					tion letter Expiry Date. If 'Only Primary Account' = Yes then the above checks need to be done only on Primary account else the checks should be done on all associated accounts. If both are true transition the case to 'Reposs

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algorithm	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition () String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.AutoApprovalCheckforRepossession	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.AutoApprovalCheckforRepossession_Impl	Code - C1-REPOAPRV	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 Repossession

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	spotistoexecutethebusinesslo	getNextTransCondition()				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 Repossession Reason Outstan

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	pecific status.					eral
ResultTypePostProcessing Algorithm Spot	This Algorith	void setActionEntity (String actionEntity); void setActionSourceId (String action	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.RepossessionApprovalResultPostProcessingAlgorithm	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.RepossessionApprovalResultPostProcessingAlgorithm_Impl	Code - C1-RAPRVRSLT	Transition the case to given Case Status if Case Status is configured. 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
	m spot decides in which status t	SourceId); void setActionSourceStatusCode (String actionSourceStatusCd); void setActionId (String actionId); void setActionType (ActionType actionType); void setRe				present on the case if approval task type is configured. Copy the comments in the result to the Approver remarks field

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
	tobemadebasedonprocessingofre					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algorithm	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.AutomaticSendingofRedemptionLetters	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.AutomaticSendingofRedemptionLetters_Impl	Code - C1-REDEMPLTR	For each of the accounts associated to the repossession case send the Redemption letter (create customer contact of given template code) If 'Only Primary Account = Yes' then send letter only on the

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	spotistoexecutethebusinesslo	getNextTransCondition()				primary account.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	pecificstatus.					
CaseTypeEnterStatusAlgorithmSpot	Thepurposeo	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseS	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.RepossessionAssignmentAlert	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.RepossessionAssignmentAlert_Impl	Code-C1-REPOASAL	Generate and send the email to the email id of the contact person associated to the service type mentioned in the

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	f the algorithm spot is to execut	tatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTransCondition()				parameter. Email of specified template code will be sent. The algorithm will generate the contact as well as initiate contact processing

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
	slogicwhenCaseismovedintospecif					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
LetterTemplateLetterExtractCollectionAlgorithmSpot	Extract all the Collater	void setCustomerContact (CustomerContact customerContact); LetterTemplateInfoBean getLetterTemplateInfo(); ReportDefinition getReportDefinition();	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.ExtractRepossessionAssignmentAlgorithm	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.ExtractRepossessionAssignmentAlgorithm_Impl	Code - C1-REPEMTEMP	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	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	al,AccountandCustomerIn					

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	AlertModule.Thecontactpe					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e the alert.					
GenericAlgorithmSpot	This is generical g	void setPerson (Person person); void setToDoCase (ToDoCase toDoCase); void setAccount (Acco	com.splwg.ccb.domain.collection.dmdc.VerifyDMDCDetailsAlgorithm	com.splwg.ccb.domain.collection.dmdc.VerifyDMDCDetailsAlgorithm_Impl	Code - C1-DMDCREQ	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	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	orithm spotwhichcanbeusedtoge	unt account); Bool getDMDCVerificationRequired();				

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	rithm oftypeAlgorithmSpot					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algorithm sp	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTran	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.ChkBkpcyOnAssociateCust	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.ChkBkpcyOnAssociateCust_Impl	Code - C1-CHKBKPTCY	If Repossession Reason <> Bankruptcy For each customer associated with the case Check if the Bankruptcy_ Switch = Y. If yes Case Creation will be rolled back and below error message will be

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	otistoexecutethebusinesslogi	sCondition()				displayed. "One or more of the collateral owners have claimed Bankruptcy. Repossession process should be initiated from Bankruptcy process"

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	fic status.					
CaseTypeEnterStatusAlgorithmSpot	The purpose of the	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTra	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.AssociateCustAssRepo	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.AssociateCustAssRepo_Impl	Code - C1-ASSOCUST	Associate all financial owners on the associated accounts to the Repossession case.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	algorithm spot is to execute the	nsition() String getNextCaseStatus() String getNextTransCondition()				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	Caseis moved into specific status.					

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 algorithm sp	void setCase (ToDoCase toDoCase); void setOriginalCaseStatus (CaseStatus caseOriginalStatus);	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.ValidateCollateral	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.ValidateCollateral_Impl	Code - C1-VALDCOLL	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 category specified in the paramet

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	o t i s t o e x e c u t e t h e v a li d a ti o n l o g i					ers. 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. There is no repossession case active on the collateral (IS_

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	cific status.					
ResultTypePostProcessing Algorithm Spot	ThisAlgorithm s	void setActionEntity (String actionEntity); void setActionSourceId (String actionSourceId);	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.RepossessionTransition	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.RepossessionTransition_Impl	Code - C1-RSTUPCMPL	If Repossession Reason = "Voluntary Repossession" transition to 'Repossession In Progress - Voluntary

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	potdecidesinwhichstatustran	void setActionSourceStatusCode (String actionSourceStatusCd); void setActionId (String actionId); void setActionType (ActionType actionType); void setResultType (Resul				Surrender' else transition to 'Repossession in Progress"

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	madebasedonprocessingofresult					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ToDoTypeToDoPostProcessAlgorithmSpot		void setOldToDoEntryDTO (ToDoEntry_DTO oldDTO); void setNewToDoEntry (ToDoEntry newToDoEntry);	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.NotifyOnTaskCompletion	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.NotifyOnTaskCompletion_Impl	Code - C1-NOTRSTSK	Create Notification Notification: <Task Id> - <Task Name> complete for <Collateral Code> <Collateral Description>. Set Display Date of the case to current business date. Notificat

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						ion should be created on the case associated to the task. This algorithm can be attached to any case level task on the Repossession case to alert the repossession specialist.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used to per	void setCase(ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.MonitorForRedemptionProc	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.MonitorForRedemptionProc_Impl	When the outstanding amount of all the associated accounts	When the outstanding amount of all the associated accounts becomes zero 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
	form autotransition processing fo				becomes zero move the case to Closed Status. Code - C1-REDE	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used to p	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.MonitorForLiquidationSetUpComplete	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.MonitorForLiquidationSetUpComplete_Impl	When Repo Title Received Date and Vehicle at Sale	When Repo Title Received Date and Vehicle at Sale Location Date is available the case is moved to the next status.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	erform autotransitionprocessing				LocationDate is available the case is moved to the next status.	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algorithm	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.AutoTaskCreationForVendor	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.AutoTaskCreationForVendor_Impl	Code - C1-AUTOTASKC	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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spotistoexecutethebusinesslo	getNextTransCondition()				case. If the vendor is 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
	pecific status.					
ResultTypePostProcessing Algorithm Spot	This Algorith	void setActionEntity (String actionEntity); void setActionSourceId (String action	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.ValidateRepoCaseData	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.ValidateRepoCaseData_Impl	Code - C1-VALDATAPR	Validate if the Dynamic Panel Data Elements and Case Characteristics mentioned in the parameters have

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	m spot decides in which status t	SourceId); void setActionSourceStatusCode (String actionSourceStatusCd); void setActionId (String actionId); void setActionType (ActionType actionType); void setRe				some values for the case. If yes the Follow Up is saved successfully 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

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	tobemadebasedonprocessingofre					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used to p	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.MonitorRedemptionClearDate	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.MonitorRedemptionClearDate_Impl	When the redemption clear date is reached transition th	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
	erform autotransitionprocessing				e case to the LiquidationSetupStatus. Code-C1-REDCLRD	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algorithm	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.Metro2ConsumerInformationIndicator	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.Metro2ConsumerInformationIndicator_Impl	Set CII = X based on Chapter entered in Filing Inform	Set CII = X based on Chapter entered in Filing Information for all customers associated to the case.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	spotistoexecutethebusinesslo	getNextTransCondition()			ationforallcustomersassociatedtothecase.Code-C1-CONI	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	pecific status.					
CaseTypeEnterStatusAlgorithmSpot	The purpose o	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseS	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.Metro2ConsumerInfoIndiChap13PostDis	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.Metro2ConsumerInfoIndiChap13PostDis_Impl	If any associated secured	If any associated secured account without confirmed plan on it report CII = Q Else Report CII = G for Chapter 12

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	f the algorithm spot is to execut	tatus) Bool getShouldAutoTransition() String getNextCaseStatus() String getNextTransCondition()			account without confirmed plan on it report CII = Q Else R	Report CII = H for Chapter 13.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	slogicwhenCaseismovedintospecif				12Report CII=H for Chapter13. Code - C1-CIPSTDIS	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the alg	void setCase (ToDoCase toDoCase); void setCaseOriginalStatus (CaseStatus caseStatus); Bool getShouldAutoTransition(); String getNextCaseStatus(); String	com.splwg.ccb.domain.collection.tasks.algo.AutomaticTaskCreatiomn	com.splwg.ccb.domain.collection.tasks.algo.AutomaticTaskCreatiomn_Impl	If case level task create a task on the case id. If	If case level task create a task on the case id. If account level task create a task each on all the accounts associated on the case. If customer level task create a task each on all the custome

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	orithm spot is to execute t	getNextTransCondition();			account level task create a task each on all the accounts	rs associated on the case.

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
	Caseismovedintospecificst				eataskeachonallthecustomersassociatedonthecase.Cas	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ase to it as FK Characteristic					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeExitStatusValidationAlgorithmSpot	The purpose of the algorithm sp	void setCase (ToDoCase toDoCase); void setPreviousCaseStatus (CaseStatus caseStatus);	com.splwg.ccb.domain.collection.tasks.algo.ValidateTaskCompletion	com.splwg.ccb.domain.collection.tasks.algo.ValidateTaskCompletion_Impl	Validate if given tasks have been completed	Validate if given tasks have been completed before exiting the status. For case level tasks check if any open tasks on the case id. For account level tasks check if any open tasks on the account

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	otistoexecutebusinesslogicwh				before exiting the status. For case level task	s 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
	specific status.				r account level tasks check if any open tasks on th	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
					Type-ExitValidationC1-VALTASKEX	

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 algorithm sp	void setCase(ToDoCase toDoCase); void setOriginalCaseStatus(CaseStatus caseOriginalStatus);	com.splwg.ccb.domain.collection.tasks.algo.ValidateTaskCompletionClosure	com.splwg.ccb.domain.collection.tasks.algo.ValidateTaskCompletionClosure_Impl	Validate if given tasks have been completed before	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 if any open tasks on the accounts

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	otistoexecutethevalidationlogi				entering the status For case level tasks check if any	associated with the case.

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	cificstatus.				check if any open tasks on the accounts associated with the ca	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the alg	void setCase (ToDoCase toDoCase); void setCaseOriginalStatus (CaseStatus caseStatus); Bool getShouldAutoTransition (); String getNextCaseStatus(); String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.SetDPDOutstandingAmount	com.splwg.ccb.domain.collection.caseType.specialisedCollections.bankruptcy.SetDPDOutstandingAmount_Impl	Set the DPD and Outstanding amount to all associated	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
	orithm spot is to execute t	getNextTransCondition();			accounts on entering the status - EnterStatus - C1-SETDPD	

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**Collections Algorithm Description and Code**	**Algorithm Summary**
	Caseismovedintospecificst					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	aseto itasFK Characteristic					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algori	void setCase (ToDoCase toDoCase); void setCaseOriginalStatus (CaseStatus caseStatus); Bool getShouldAutoTransition(); String getNextCaseStatus(); String getNextTran	com.splwg.ccb.domain.collection.caseType.earlyCollections.CreditGrantorCannotLocateConsumer	com.splwg.ccb.domain.collection.caseType.earlyCollections.CreditGrantorCannotLocateConsumer_Impl	Automatically set for all borrowers the account the Cl	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
	thm spot is to execute the	sCondition();			l Code in skip tracing status on entering a case status EnterPr	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	is moved into specific status.					

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	FK Characteristic					
ResultTypePostProcessing Algorithm Spot	This Algo	void setActionEntity (String actionEntity); void setAct	com.splwg.ccb.domain.collection.caseType.earlyCollections.ConsumerNowLocated	com.splwg.ccb.domain.collection.caseType.earlyCollections.ConsumerNowLocated_Impl	This algorithm will	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
	rithm spot decides in which stat	ionSourceId (String actionSourceId); void setActionSourceStatusCode (String actionSourceStatusCd); void setActionId (String actionId); void setActionType (ActionType			set the given CIl Code for the party id provided as result	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	tionhastobemadebasedonprocessin	; boolean getIsProcessingComplete();			It typepostprocessingalgoC1-CGCLC	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	Thepurposeofthealg	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.Metro2AcctStatuscodeEnterProcessingAlgo	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.Metro2AcctStatuscodeEnterProcessingAlgo_Impl	Metro2Reporting - AccountStatusCode C1-ASCREPO	This algorithm is used for Metro 2 Reporting - Account Status Code

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
	orithm spotistoexecutet	getNextTransCondition()				

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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ase to it as FK Characteristic					
ResultTypePostProcessing Algorithm Spot	T	void setActionEnt	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.Metro2AcctStatusCodePostLiquidationPostProcessing	com.splwg.ccb.domain.collection.caseType.specialisedCollections.AssetRepo.algorithms.Metro2AcctStatusCodePostLiquidationPostProcessing_Impl	Metro2	This algorithm is used for Metro 2 Reportin

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	his Algorithm spot decides inw	ity (String actionEntity); void setActionSourceId (String actionSourceId); void setActionSourceStatusCode (String actionSourceStatusCd); void setActionId (String action			Reporting - Account Status Code post Liquidation C1-ASCLIQU	g - Account Status Code post Liquidation

Algorith m Spot	S p o t D e t a i l	Spot Interf ace Funti ons	Collections Algorithm Component	Collections Algorithm Impl	C ol le ct io n s Al g or it h m D es cr ip ti o n a n d C o d e	Algorit hm Summa ry
	i c h s t a t u s t r a n s it i o n h a s t o b e m a d	onTyp e (Actio nType action Type); void setRe sultTy pe (Resul tType result Type); boolea n getIsP roces singC omple te();				

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessing Algorithm Spot	This Algorithm spot decides	void setActionEntity (String actionEntity); void setActionSourceId (String actionSourceId); void setActionSourceStatusCode (String actionSourceStatu	com.splwg.ccb.domain.collection.caseType.earlyCollections.Metro2ComplianceCodePostProcessingAlgo	com.splwg.ccb.domain.collection.caseType.earlyCollections.Metro2ComplianceCodePostProcessingAlgo_Impl	Metro2 Reporting - Compliance condition code C1-COMCO	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
	in which status transition has to b	sCd); void setActionId (String actionId); void setActionType (ActionType actionType); void setResultType (ResultType resultType); boolean getIsProcessingComplete();			DE	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	f result.					
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type i	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition (); CaseStatus getNextCaseStatus(); String getNe	com.splwg.ccb.domain.collection.caseType.earlyCollections.UpdateDisputeMonitor	com.splwg.ccb.domain.collection.caseType.earlyCollections.UpdateDisputeMonitor_Impl	MonitoringAlgo For Dispute Resolved C1-DI	This algorithm is a Monitoring Algo For Dispute Resolved.Used for updating Dispute Flag to 'N'

Algorithm Spot	**SpotDetail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**CollectionsAlgorithmDescriptionandCode**	**Algorithm Summary**
	susedtoperform autotransitionp	xtTransCondition();			SMON	

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessing Algorithm Spot	This Algorithm spot decides	void setActionEntity (String actionEntity); void setActionSourceId (String actionSourceId); void setActionSourceStatusCode (String actionSourceStatu	com.splwg.ccb.domain.collection.caseType.earlyCollections.CaseCreationonFollowupPostProcessingAlgo	com.splwg.ccb.domain.collection.caseType.earlyCollections.CaseCreationonFollowupPostProcessingAlgo_Impl	CreateRequiredCaseonFollowUpResultPostprocessi	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. Queue to which the case should

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	in which status transition has to b	sCd); void setActionId (String actionId); void setActionType (ActionType actionType); void setResultType (ResultType resultType); boolean getIsProcessingComplete();			ng Algorithm C1-CRETCSFL	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
	f result .					
ResultTypePostProcessing Algorithm Spot	This Algorithm spot	void setActionEntity (String actionEntity); void setActionSourceId (String actionSourceId); void setActionSo	com.splwg.ccb.domain.collection.caseType.earlyCollections.CaseTransitionandTraskCreationPostProcessingAlgo	com.splwg.ccb.domain.collection.caseType.earlyCollections.CaseTransitionandTraskCreationPostProcessingAlgo_Impl	Generic Result Post Processing Algorit	Generic Result Post

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	decidesinwhichstatustransitio	urceStatusCode (String actionSourceStatusCd); void setActionId (String actionId); void setActionType (ActionType actionType); void setResultType (ResultType resultType);			hm for Case Transition and Task Creation Result Post proc	Processing Algorithm for Case Transition and Task Creation Transition the case to given

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	edon processing of result.					Status is configured and the current status is present in one of the Valid Current Statuses. Display an error

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						tatus. - Create Task of given Task Type and assign it to the give Task Queue if Task Type is configured. - Map the

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 the case If Task For =

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algorithm	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.earlyCollections.CopyCharacteristicsOnCaseCreate	com.splwg.ccb.domain.collection.caseType.earlyCollections.CopyCharacteristicsOnCaseCreate_Impl	CopyCaseCharacteristicsAlgorithmCaseTypeEnter	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_ALT_

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spotistoexecutethebusinesslo	getNextTransCondition()			StatusAlgorithm C1-COPYCHAR	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
	pecificstatus.					
CaseTypeAutoTransitionAlgorithmSpot	This algorithm	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition (); CaseStatus getNe	com.splwg.ccb.domain.collection.caseType.earlyCollections.DetermineContactIntensity	com.splwg.ccb.domain.collection.caseType.earlyCollections.DetermineContactIntensity_Impl	DetermineContactIntensity an	"Determine Contact Intensity and Contact Intensity Review Date: - If case is not on Hold. - And Busines

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	typeisusedtoperform autotrans	xtCaseStatus(); String getNextTransCondition();			dContactIntensityReviewDateCaseTypeAutoTransition	s Date >= Contact Intensity Review 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 POSTINGDATE or

Algorithm Spot	**Spot Detail**	**Spot Interface Funtions**	**Collections Algorithm Component**	**Collections Algorithm Impl**	**Collections Algorithm Description and Code**	**Algorithm Summary**
	g for a Case.					
ResultTypePostProcessing Algorithm Spot	This Algorithm sp	void setActionEntity (String actionEntity); void setActionSourceId (String actionSourceId); void	com.splwg.ccb.domain.collection.caseType.earlyCollections.HoldCasePostProcessingAlgo	com.splwg.ccb.domain.collection.caseType.earlyCollections.HoldCasePostProcessingAlgo_Impl	Hold Case fo	Hold Case for Days as provided in Characteristic Type provided in Hold Period or if that is blank Hold Period should be

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	o t d e c i d e s i n w h i c h s t a t u s t r a n s	setActionSourceStatusCode (String actionSourceStatusCd); void setActionId (String actionId); void setActionType (ActionType actionType); void setResultType (ResultType			r Days as provided in Characteristic Typ	referred from Hold Period parameter. And Hold Reason should be set as provided in characteristic type provided in Hold Reason or if that is blank Hold Reason should be referred from Hold Reason paramet

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	adebasedonprocessingofresult.				nHoldPeriodorifthatisblankHoldPeriod	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
					hould be set as provided in characteristic type provided in	

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used top	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.caseType.earlyCollections.InitiateContact	com.splwg.ccb.domain.collection.caseType.earlyCollections.InitiateContact_Impl	Transition to contact state if First Contact Date has	Transition to contact state if First Contact Date has reached If First Contact Date has reached (based on the parameters below) Or Account is Direct Debit

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	erform auto transition processing				reached and set the ReAllocation Switch Case Type Aut	and Immediate Transition if Direct Debit = Yes/No Transition to Contact RM status if Relationship Manager exists and Contact RM status has been specified Transitio

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						tion Switch = Y for the case post case transition Possible Values First Contact Calculation Parameter: DPD, DIA, Days Since Case Start Immediate Transition if Direct Debit: Y,N Validation Date :

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeEnterStatusAlgorithmSpot	The purpose of the algorithm	void setCase (ToDoCase toDoCase) void setCaseOriginalStatus (CaseStatus caseStatus) Bool getShouldAutoTransition() String getNextCaseStatus() String	com.splwg.ccb.domain.collection.caseType.earlyCollections.InitiateSkipTracing	com.splwg.ccb.domain.collection.caseType.earlyCollections.InitiateSkipTracing_Impl	Transition to skip tacing status if no telephone number exist	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 any of the account holder. If no contact

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	spot is to execute the business lo	getNextTransCondition()			s for any of the account holder Case Type - Enter Status Algo C	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
	pecific status.					
CaseTypeAutoTransitionAlgorithmSpot	This algorithm	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNe	com.splwg.ccb.domain.collection.caseType.earlyCollections.InitiateSkipTracingInvalidTelNumber	com.splwg.ccb.domain.collection.caseType.earlyCollections.InitiateSkipTracingInvalidTelNumber_Impl	Transition to skip tracing status	"Transition to skip review if 'X' number of consecutive failed contacts - If last X number

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	type is used to perform auto trans	xtCaseStatus(); String getNextTransCondition();			if 'X' number of consecutive calls fails CaseType - AutoTransit	of consecutive contacts has been unsuccessful, transition to Skip Tracing Status. Logic for considering unsuccessful contacts: If last X consecutive contacts with given contact methods have

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	g for a Case.					e transition Possible Values for Validation Date are POSTINGDATE and SYSTEMDATE"
CaseTypeAutoTransitionAlgorithmSpot	This Algorith	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition (); Case	com.splwg.ccb.domain.collection.caseType.earlyCollections.ParkSmallBalanceAccounts	com.splwg.ccb.domain.collection.caseType.earlyCollections.ParkSmallBalanceAccounts_Impl	Park accounts with small	Park accounts with small balances to a separate status so that no

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	m type is used to perform autotra	Status getNextCaseStatus(); String getNextTransCondition();			balances to a separate status Case Type - Auto Transition C1-E	contacts are initiated for the account. If Net Arrear Amount <= Small Balance Threshold And Net Arrear Amount > 0 Then transition to small balance status. Net Arrear Amount = (Overdue

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	ssing for a Case.					due Amount instead of Net Arrear Amount in the calculations. Set Re-Allocation Switch = Y for the case post case transition. Possible Values : Use Overdue Amount : Y,N

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessing Algorithm Spot	This Algorithm spot decides in	void setActionEntity (String actionEntity); void setActionSourceId (String actionSourceId); void setActionSourceStatusCode (String actionSourceStatusCd); void	com.splwg.ccb.domain.collection.caseType.earlyCollections.ResumeCollectionsPostProcessingAlgo	com.splwg.ccb.domain.collection.caseType.earlyCollections.ResumeCollectionsPostProcessingAlgo_Impl	Resume Collections Transits the case to Contact status Result	Resume Collections Transition the case to Contact RM Status if RM exists and Contact RM status has been configured Contact Alternate Status If Contact Alternate Flag = Y Else Contact Status Set Re-

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	which status transition has to be em	setActionId (String actionId); void setActionType (ActionType actionType); void setResultType (ResultType resultType); boolean getIsProcessingComplete();			TypePostProcessingAlgoC1-RESCOLL	Allocation Switch = Yes 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**
	It.					
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used t	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.caseType.earlyCollections.ResumeContactFromUnderResolution	com.splwg.ccb.domain.collection.caseType.earlyCollections.ResumeContactFromUnderResolution_Impl	ResumeContactFromUnderResolutionStat	Resume Contact From Under Resolution Status: - If there is no more active PTP on the account and - If the Net Arrear Amount > 0 Than transition the case to

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	operform autotransitionprocessi				usMove case to contact status if the Net Arrear Amount is gr	Contact RM Status if RM exists and Contact RM status has been configured Contact Alternate Status If Contact Alternate Flag = Y Else Contact Status Set Re-Allocation Switch = Y for the case

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
					on C1-ECRCFUR	nt - Unclear Amount) Use Overdue Amount can be Y/N or Yes/No
CaseTypeAutoTransitionAlgorithmSpot	This algorithm typ	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus();	com.splwg.ccb.domain.collection.caseType.earlyCollections.ResumeContactfromSmallBalance	com.splwg.ccb.domain.collection.caseType.earlyCollections.ResumeContactfromSmallBalance_Impl	This algorithm is used to resume co	This algorithm is used to resume contact from small balance status. If Net Arrear Amount > Small Balance

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	e is used to perform autotransitio	String getNextTransCondition();			ntact from small balance status. Case Type - Auto Transitio	Threshold Then transition the case to Contact RM Status if RM exists and Contact RM status has been configured Contact Alternate Status If Contact Alternate Flag = Y Else Contact Status

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	Case.					ue Amount instead of Net Arrear Amount in the calculations. Net Arrear Amount = (Overdue Amount - Unclear Amount) Possible Value: Overdue Amount : Y,N
CaseTypeAutoTransitionAlgorithmSpot	Th	void setCase (ToDoCase	com.splwg.ccb.domain.collection.caseType.earlyCollections.ScheduleContact	com.splwg.ccb.domain.collection.caseType.earlyCollections.ScheduleContact_Impl	This	Schedul

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	isalgorithm type is used to perfo	toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();			algorithm will Schedule Contact for the case as per the give	e Contact for the case as per intensity: - If case is not on Hold. - And Display Date <= Business Date or Display Date is Blank. - Set Display Date = Max ((Last Successful Contact

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	autotransitionprocessingforaCa				ensityCaseType-AutoTransitionC1-ECSC	Contact Intensity), Business Date) Consider Contact Intensity from Algorithm parameter if specified else picks up Contact Intensity from case level field. Logic for considering successful contact

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
ResultTypePostProcessingAlgorithm Spot	This Algorithm spot decides	void setActionEntity (String actionEntity); void setActionSourceId (String actionSourceId); void setActionSourceStatusCode (String actionSourceStatu	com.splwg.ccb.domain.collection.caseType.earlyCollections.SupervisorReferralPostProcessingAlgo	com.splwg.ccb.domain.collection.caseType.earlyCollections.SupervisorReferralPostProcessingAlgo_Impl	This algorithm will transfer the case to the given	Supervisor Referral Algorithm: - If case is present in one of the status's specified in 'Valid Current Status' than Proceed with further actions Else Display an error 'The

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	in which status transition has to b	sCd); void setActionId (String actionId); void setActionType (ActionType actionType); void setResultType (ResultType resultType); boolean getIsProcessingComplete();			status if the current staus of the case is present in Valid C	selected result <Result Type> is not allowed in current Status.' And don't proceed with further actions. - Transition the case to given Case Status. - Create Task of given Task Type and

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	fresult.				C1-ECRTS	n Switch = Y if Re-Allocate = Y Re-Allocate can be Y/N
CaseTypeAutoTransitionAlgorithmSpot	This algorithm typei	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNe	com.splwg.ccb.domain.collection.caseType.earlyCollections.TransitionToSuspendedStatus	com.splwg.ccb.domain.collection.caseType.earlyCollections.TransitionToSuspendedStatus_Impl	Transition to suspended status if the acco	If the Account has one of the Account Risk Indicators specified in the paramet

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	sused to perform autotransitionp	xtTransCondition();			unt has one of the warning indicator set CaseType-Auto	er Transition to Suspended status. Create a task if Task Type has been mentioned and assign it to the Specified Queue Set Re-Allocation Switch

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						t. 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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
						te' than transition the case to the Contact 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

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used to p	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition (); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.caseType.earlyCollections.TransitionToUnderResolutionStatus	com.splwg.ccb.domain.collection.caseType.earlyCollections.TransitionToUnderResolutionStatus_Impl	Transition to under resolution status if Net Arrear Amount	Transition to under resolution status if Net Arrear Amount <=0. - Transition the case to Under Resolution Status if Net Arrear Amount <= 0 or PTP is running on the account. - Set Re-Allocatio

Algorithm Spot	Spot Detail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	Collections Algorithm Description and Code	Algorithm Summary
	erform autotransition processing				<=0 CaseType-AutoTransition C1-ECTTURS	n Switch = Y for the case post case transition Net Arrear Amount = (Overdue Amount - 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
CaseTypeAutoTransitionAlgorithmSpot	This algorithm type is used to p	void setCase (ToDoCase toDoCase); Bool getShouldAutoTransition(); CaseStatus getNextCaseStatus(); String getNextTransCondition();	com.splwg.ccb.domain.collection.caseType.earlyCollections.ValidateContactCap	com.splwg.ccb.domain.collection.caseType.earlyCollections.ValidateContactCap_Impl	The algorithm will hold the case when the contact cap is	Check if the contact cap has reached for the case 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

Algorithm Spot	SpotDetail	Spot Interface Funtions	Collections Algorithm Component	Collections Algorithm Impl	CollectionsAlgorithmDescriptionandCode	Algorithm Summary
	erform autotransitionprocessing				reachedCaseType-AutoTransitionC1-ECVCC	of days >= Contact Cap Hold the case for Y number of days with the given Hold Reason.. Logic for considering successful contacts: All contacts with given contact methods that have Authenti

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