

Oracle® Financial Services Lending and Leasing Extensibility Guide



Release 14.12.0.0.0

F82313-01

August 2024

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Copyright © 2022, 2024, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

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

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

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

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	vii
Conventions	vii
Pre-requisite	vii
Assumptions	vii

1 Customize and Extend the Application

1.1 Understanding Customizing & Extending the Application	1-1
1.2 Understanding Customization Layers	1-2
1.3 Installing Customization Tools	1-4

2 Using JDeveloper for Customizations

2.1 About Using JDeveloper for Customization	2-1
2.2 About Customizing Oracle ADF Artifacts	2-2
2.3 Customizing Oracle ADF Artifacts with JDeveloper	2-4
2.4 ADF Customization Best Practices	2-5
2.4.1 ADF Customization Best Practices - Do's	2-6
2.4.2 ADF Customization Best Practices - Don'ts	2-7

3 Application Artifacts

3.1 About Customizing Oracle ADF Application Artifacts	3-2
3.2 Customizable Application Libraries	3-2
3.3 Enable JDeveloper for Customization	3-3
3.4 Customizing the Skin	3-4
3.5 Customizing or Adding Resource Bundles	3-4
3.6 Editing Existing Business Components	3-8
3.7 Editing Pages	3-12
3.8 Editing Task Flows	3-13
3.9 Create Custom Business Components	3-14
3.10 Create Custom Task Flows	3-15
3.11 Create Custom Pages	3-15

3.12	Editing the UI Shell Template	3-16
3.13	Replacing OFSLL Logo	3-16
3.14	Deploy ADF Customizations and Extensions	3-17
3.15	Deployment Options	3-18

4 Customize Database Objects

4.1	UI – Package Interaction Logic	4-2
4.2	UI Java Wrapper (U*JW)	4-2
4.3	Database Schema	4-3
4.4	Wrapper Engine model	4-4
4.5	Batch Job (BJ)	4-5
4.6	Engine Wrapper (EW)	4-5
4.7	Main Engine (EM)	4-5
4.8	Engine Function (EN)	4-5
4.9	Engine View	4-5
4.10	Common Features	4-5
4.11	Seed Data	4-6
4.12	Developer’s Tips	4-6

5 Create Custom BI Publisher Report/Letter

5.1	Creating a New Report	5-1
5.2	Create Report Layout	5-8
5.3	Create XML data	5-11
5.4	Add Dynamics to Report	5-17
5.5	Upload Report in BIP	5-21

6 Customize Existing Base BIP Reports

6.1	Customizing Existing Base BIP Reports	6-1
-----	---------------------------------------	-----

7 Customize Existing Base BIP Reports

7.1	Customizing Existing Base BIP Reports	7-1
-----	---------------------------------------	-----

8 Create Custom Correspondence

8.1	Creating Correspondence	8-1
-----	-------------------------	-----

9 Generate Correspondence

9.1	Generating Correspondence	9-1
-----	---------------------------	-----

10 Set up the output Format For BIP Reports

10.1	Setting up the output Format For BIP Reports	10-1
------	--	------

11 Naming Convention for Customized Objects

12 RESTful Web Services Extensibility

12.1	Generic Post Transaction (POST)	12-5
12.1.1	Producer related transaction	12-5
12.1.2	Other Transactions	12-6
12.2	Account On Boarding (POST)	12-9
12.3	Payment Posting (POST)	12-20
12.4	Account Detail (GET)	12-22
12.5	Scenario Analysis (POST)	12-25
12.6	Lookups (GET)	12-28
12.7	Application Search (GET)	12-30
12.8	Calculator (POST)	12-32
12.9	Application Entry (POST)	12-35
12.10	Account Search (GET)	12-44
12.11	Call Activity (POST)	12-45
12.12	Remarketing (PUT)	12-47
12.13	Invoice (POST)	12-49
12.14	Application Comment (GET/POST)	12-51
12.15	Account Comment (GET/POST)	12-53
12.16	Application Checklist (GET)	12-56
12.17	Outgoing File List (GET)	12-57
12.18	Outgoing File (POST)	12-59
12.19	Incoming File (GET)	12-60
12.20	Products (GET)	12-60
12.21	Assets (GET)	12-62
12.22	Assets (PUT)	12-64
12.23	Asset Valuation (GET/PUT/POST)	12-67
12.24	Asset Sub Types (GET)	12-70
12.25	Application Status Change (PUT)	12-71
12.26	Application Update (PUT)	12-72
12.27	Application ACH (POST)	12-81

12.28	Application Document upload/download/list Service (POST/GET/GET)	12-83
12.29	Account Document upload/download/list Service (POST/GET/GET)	12-85
12.30	Application Get Service (GET)	12-87
12.31	Scheduler Force ReSubmit Service (PUT)	12-87
12.32	Credit Limit Service (Customer/Business) [GET]	12-88
12.33	Business Comments Service (GET/POST)	12-90
12.34	Customer Comments Service (GET/POST)	12-92
12.35	Customer Preference Service (GET/POST/PUT)	12-94
12.36	Scenario Analysis Service (PUT)	12-97
12.37	Transaction Parameters Service (GET)	12-98
12.38	Asset Tracking Attribute Service(PUT)	12-100
12.39	Business Tracking Attribute Service(PUT)	12-101
12.40	Customer Tracking Attribute Service(PUT)	12-103
12.41	Account Tracking Attribute Service(PUT)	12-105
12.42	Credit Bureau Web Service(PUT)	12-107
12.43	Delete Account Web Service(DELETE)	12-109
12.44	New Customization for RESTful Web Service	12-111
12.45	Securitization Web Service (POST)	12-127
12.46	Calculate Parameter Update Service(PUT)	12-129
12.47	User Update Service (PUT)	12-130
12.48	Master Account Rolledup details service (GET)	12-132
12.49	Application Decision Fetch Service (GET/PUT)	12-133
12.50	Work Order Fetch Service (GET/PUT/POST)	12-136
12.51	Pricing Fetch Service (GET)	12-139
12.52	Account UDF Update Service (PUT)	12-140
12.53	Stipulation Fetch Service (GET)	12-141

A Appendix: Revision History

Preface

- [Audience](#)
- [Conventions](#)
- [Pre-requisite](#)
- [Assumptions](#)

Audience

This document is intended for administrators and developers who want to customize and extend the standard functionality provided by Oracle Financial Services Lending and Leasing Application. Administrators should have a basic understanding of Oracle Financial Services Lending and Leasing Application and Oracle Application Development Framework concepts. Developers should have a basic understanding of the Java programming language, web applications, Oracle JDeveloper, and Oracle Application Development Framework.

Conventions

The following conventions are used in this document:

Table 1 Conventions

Term	Refers to
Application	Oracle Financial Services Lending and Leasing
Customization application workspace	OracleFSLLEnterpriseApp/ OracleFSLLEnterpriseApp.jws provided as part of installer under /cust_lib folder

Pre-requisite

- You can find all the customizable libraries along with the necessary default projects as part of the product release installer bundle under /cust_lib folder.
- You need to download and install JDeveloper 12.2.1.0.0.

Assumptions

- It is assumed that the customization team is familiar with ADF and this document **OFSLLExtensibility Guide** has been referred in detail.
- Since ADF is Java/J2EE based framework, it is assumed that the generic best practices of Java/J2EE development are understood beforehand.

1

Customize and Extend the Application

This chapter provides an overview of how to customize and extend the application and, introduces the design time and runtime tools used in the process, such as Oracle JDeveloper, Oracle Business Intelligence (BI) Publisher and Oracle Enterprise Manager Fusion Middleware Control.

- [Understanding Customizing & Extending the Application](#)
- [Understanding Customization Layers](#)
- [Installing Customization Tools](#)
- [Understanding Customizing & Extending the Application](#)
- [Understanding Customization Layers](#)

The application contains customization layer that allows you to make customizations which affect only certain instances of an application. For example, the application has a layer for US region. When you customize an artifact, you can choose to make that customization available only for US region.
- [Installing Customization Tools](#)

For procedures for setting up JDeveloper for customizations, see chapter 3, **Using JDeveloper for Customizations**.

1.1 Understanding Customizing & Extending the Application

Oracle Financial Services Lending and Leasing application is based on Oracle Fusion Middleware. User interfaces are implemented using Oracle Application Development Framework (Oracle ADF) and standard Java technologies. Business intelligence frameworks provide a number of reporting capabilities. Each of these areas of the application can be customized and extended to suit your business needs.

Within this guide, the term customizing means to change a standard (existing) artifact. For example, you can add an attribute to a standard business object, or you can change what is displayed on a standard view page. The term extending means to create a completely new artifact, such as a custom business object or custom view page. For customizations and extensions of this application, there are two basic scenarios: personalization and design time customizations and extensions.

Personalization

Personalization refers to the changes that every end user of the application can make to certain artifacts in the user interface (UI) at runtime. These changes remain for that user each time that user logs into the application. Personalization includes changes based on user behavior (such as changing the width of a column in a table).

Design time customizations and extensions

Design time customizations and extensions include more complex changes, such as creating new business objects or creating new view pages, and they require deployment into the runtime environment. Design time customizations are done by Java developers using Oracle JDeveloper. The customizations are then uploaded or deployed to a running instance of the application.

Most customizations, whether a personalization an end user makes, or a change a developer makes using JDeveloper to create new source code, are stored in a business metadata repository. Because these customizations are kept separate from the base code, you can safely upgrade your application without overwriting or needing to redo your changes.

Customizations for the UI and for entity components are created in layers, meaning that you can create them for specific industry, or for specific region or sites, and the changes will be shown only when applicable. For more information about the metadata dictionary and customization layers, see section 2.2 - **Understanding Customization Layers**.

1.2 Understanding Customization Layers

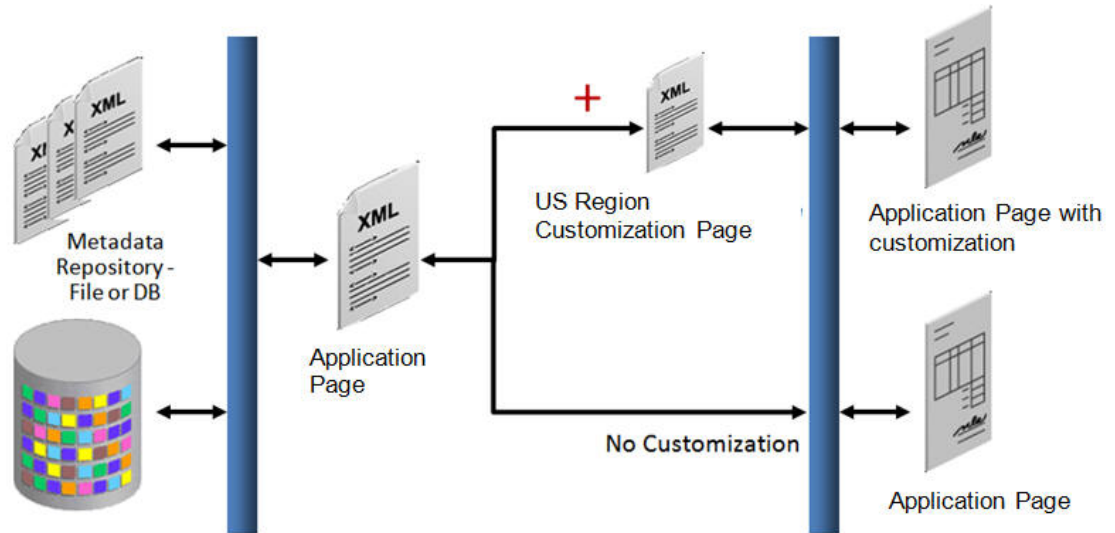
The application contains customization layer that allows you to make customizations which affect only certain instances of an application. For example, the application has a layer for US region. When you customize an artifact, you can choose to make that customization available only for US region.

Customizations you make are not saved to the base standard artifact. Instead, they are saved to an XML file that is stored in an Oracle Metadata Services (MDS) repository. This XML file acts like a list of instructions that determines how the artifact looks or behaves in the application, based on the layer that is controlling the current context. The MDS Customization Engine manages this process.

For example, say you want to customize the Applicant fragment by adding a new Passport field, but only for US region. Before you make your customization, you first select the layer to make your customization in, in this case the region layer whose value is US. When you make your customization by adding the new Passport field in the Application fragment, an XML file is generated with the instructions to add the field, but only in the region layer, and only when the value is US. The original page file remains untouched. The MDS Customization Engine then stores the XML file in an MDS repository.

Now, whenever someone logs into the application and requests an artifact, the MDS Customization Engine checks the repository for XML files that match the requested artifact and the given context, and if there is a match, it layers the instructions on top of the base artifact. In this example, whenever the Application page is requested (the artifact) by someone where US region customization is applied, before the page is rendered, the MDS Customization Engine pulls the corresponding XML file from the repository and layers it on top of the standard Application page, thereby adding the new field.

Figure 1-1 Customization Layers 1



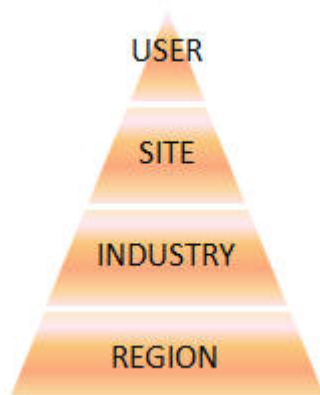
All users of the application can personalize the pages. Users can move elements around on a page, hide elements, and even add available elements to their page. When they do this personalization, the MDS Customization Engine creates an XML file specific to that user.

For example, say User 1 personalizes the Application page. There will then be an XML file stored in the repository, noting the changes that user made. When User 1 logs in, as in the previous example, the MDS Customization Engine pulls the XML file with the customizations from the repository and layers it on top of the standard Application page. In addition, the engine pulls the XML file with User 1's personalization's, allowing the user to see the personalization changes along with the US region changes. When other users log in, they do not see User 1's personalization changes.

The application has following customization layers:

- **Region:** When customizations are made in this layer, they affect users of the application for a specific region. This layer's XML files are added for everyone, whenever the artifact is requested.
- **Industry:** When customizations are made in this layer, they affect users of the application for a specific industry. This layer's XML files are added for everyone, whenever the artifact is requested.
- **Site:** Customizations made in the Site layer affect users at a particular location.
- **User:** This is where all personalization's are made. Users do not have to explicitly select this layer.

Figure 1-2 Customization Layers 2



These layers are applied in a hierarchy, and the highest layer in that hierarchy in the current context is considered the tip. Within the default customization layers, the Region layer is the base layer, and the User layer is the tip. If customizations are done to the same object, but in different layers, at runtime, the tip layer customizations take precedence. For example, if you customize the label for a field in the site layer and customize the same label in the industry layer using JDeveloper, the site layer customization will be displayed at runtime.

Because customizations are saved in these XML files, when you patch or upgrade your application, the base artifacts can be updated without touching your changes. The base artifact is replaced, and when the application is run after the patch or upgrade, the XML files are simply layered on top of the new version. You do not need to redo your customizations.

Before you create customizations, you must select the layer to which you want your customizations to be applied.

1.3 Installing Customization Tools

For procedures for setting up JDeveloper for customizations, see chapter 3, **Using JDeveloper for Customizations**.

2

Using JDeveloper for Customizations

This chapter describes how to configure JDeveloper for implementing customizations in the application.

- [About Using JDeveloper for Customization](#)
- [About Customizing Oracle ADF Artifacts](#)
- [Customizing Oracle ADF Artifacts with JDeveloper](#)
- [ADF Customization Best Practices](#)
- [About Using JDeveloper for Customization](#)
JDeveloper is used when it is needed to customize or create business objects or new pages. The procedures for each of these are different.
- [About Customizing Oracle ADF Artifacts](#)
The application is built using Oracle ADF artifacts, including the following:
- [Customizing Oracle ADF Artifacts with JDeveloper](#)
To customize ADF artifacts, open the default customization application workspace provided, using the **Customization Developer** role and customize the required artifacts.
- [ADF Customization Best Practices](#)
Listed below are some of the best practices that are captured during ADF customization. This list is iterative and includes the critical points discussed and captured from previous customizations.

2.1 About Using JDeveloper for Customization

JDeveloper is used when it is needed to customize or create business objects or new pages. The procedures for each of these are different.

New custom objects created in JDeveloper are not saved into the MDS Repository, and so are done in a standard application workspace using the **Default** role. However, when you customize standard objects, those customizations are saved into the MDS Repository, and so must be done using the **Customization Developer** role. Doing customizations using the customization developer role ensures that the changes are saved to upgrade-safe MDS Repository, and not written directly to the standard object. In future, when patch or upgrade Application, the customizations held in these metadata files will not be touched, and so, it need not be re-done.

When customizing ADF artifacts, a special customization application workspace can be created; using the Default role, for this application a default customization application workspace (`/OracleFSLLEnterpriseApp/OracleFSLLEnterpriseApp.jws`) is provided. This workspace includes all the artifacts that can be customized. This customization workspace can be configured, so that when customizations are tested and deployed, they appear to be part of native Oracle Financial Services Lending and Leasing Application.

Using the default workspace, it is possible to switch roles to customization developer and customize the ADF artifacts required. After completion, the artifacts are packaged and deployed in the workspace to the Oracle Financial Services Lending and Leasing environment.

Often, there is a need to perform both customizations (customizing an existing standard object) and extensions (creating a new object). For example, suppose it is needed to create a new business object and expose that new object in an existing application module. First, because a new custom business object is being created, first a standard application workspace is created and then entity object is created. After completion, the workspace is packaged as an ADF Library, and placed into a directory. Next, using the default workspace provided, the new entity object library and the library that contains the application module to which we need to add the entity object is added. After both are imported, User should log in using the customization developer role and make the customizations to the application module. After customizations are complete, User would deploy the customizations to the test environment.

 **Note:**

Before running JDeveloper in customization mode for the application, see Section 4.3 **Enable JDeveloper for Customization** for pre-configuration requirement.

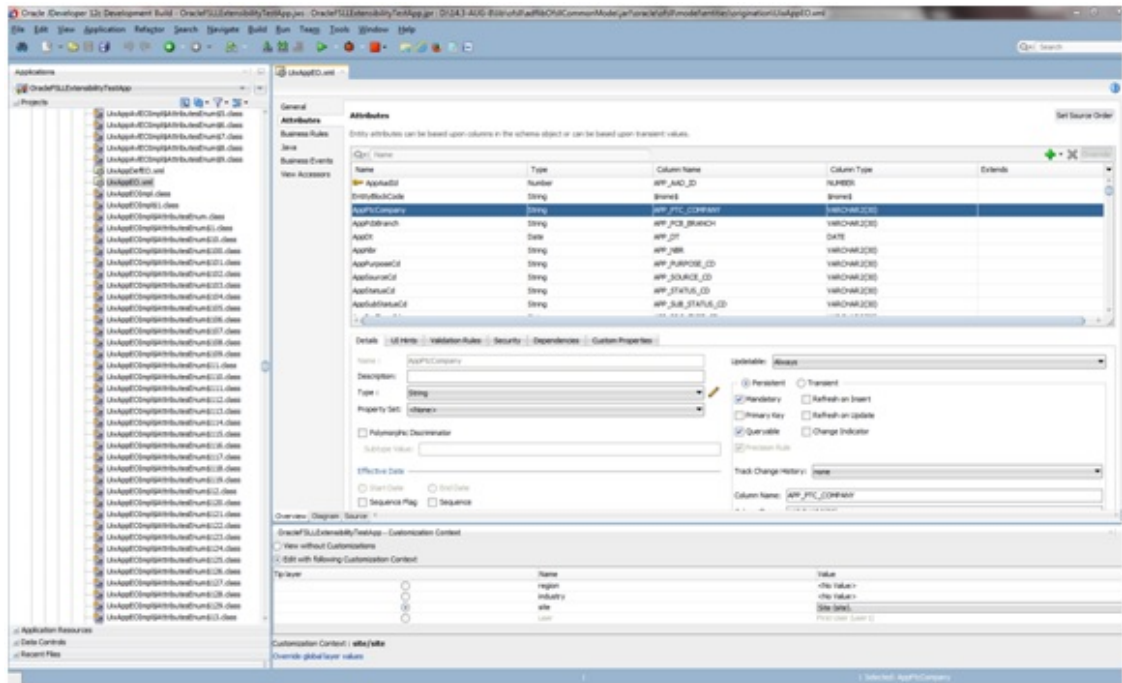
2.2 About Customizing Oracle ADF Artifacts

The application is built using Oracle ADF artifacts, including the following:

- **Application modules:** An application module is the transactional component that UI clients use to work with application data. It defines an updateable data model along with top-level procedures and functions (called service methods) related to a logical unit of work that is related to an end-user task.
- **Entity objects:** An entity object represents a row in a database table and simplifies modifying its data by handling all data manipulation language (DML) operations. It can encapsulate business logic to ensure that the required business rules are consistently enforced. An entity object can be associated with others to reflect relationships in the underlying database schema, to create a layer of business domain objects, and to reuse in multiple applications.
- **View objects:** A view object represents a SQL query and simplifies working with its results. The SQL language is used to join, filter, sort, and aggregate data into the shape required by the end-user task to be represented in the user interface. This includes the ability to link a view object with other view objects to create master-detail hierarchies of any complexity. When end users modify data in the user interface, view objects collaborate with entity objects to consistently validate and save the changes.
- **Task flows:** Task flows define the flow of control throughout an application. They also can be included in a page as a region, where users can navigate through a series of page fragments, without leaving the original page.
- **JSPX pages and page fragments:** The view layer of the application consists of a small number of pages per application. These pages then contain task flows, which in turn contain a number of page fragments.

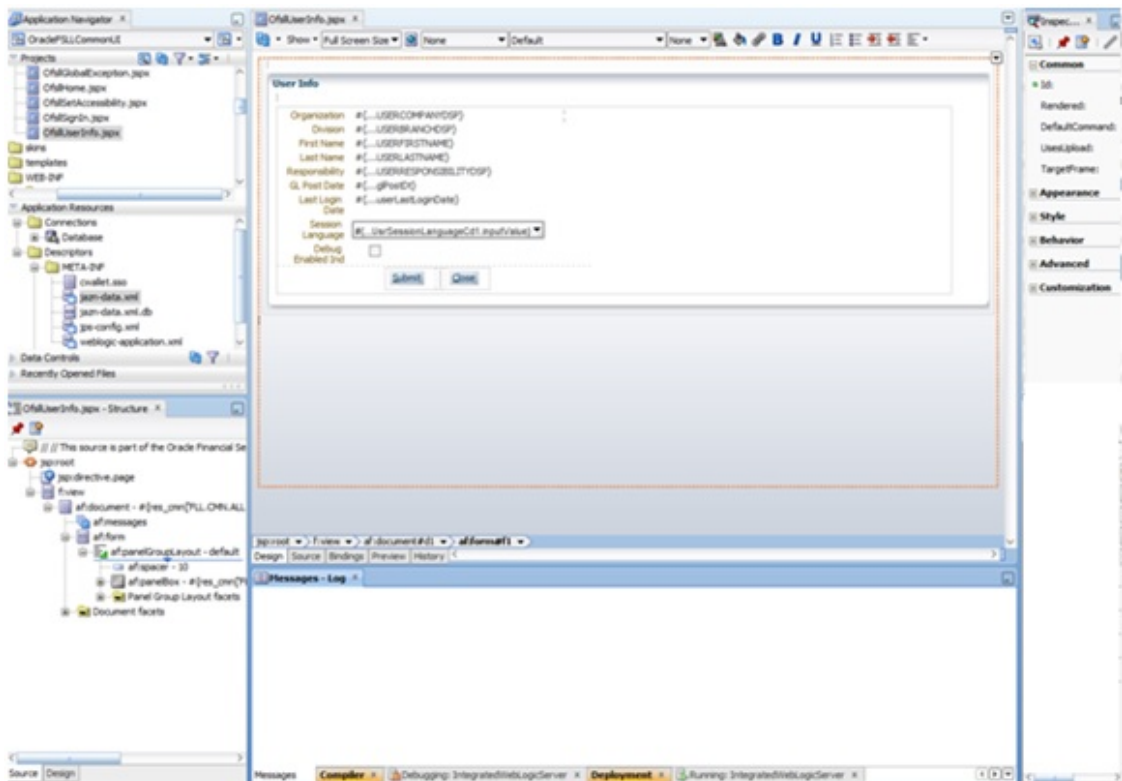
When Oracle ADF artifacts are customized, it generally happens in an overview editor that allows making customizations declaratively. For example, below figure shows the editor for an entity object. Among other things, validation can be set or UI display changes can be done.

Figure 2-1 Oracle ADF artifacts 1



For JSPX pages, a WYSIWYG environment is displayed where changes can be made using the Design tab in the editor window or structure window.

Figure 2-2 Oracle ADF artifacts 2



2.3 Customizing Oracle ADF Artifacts with JDeveloper

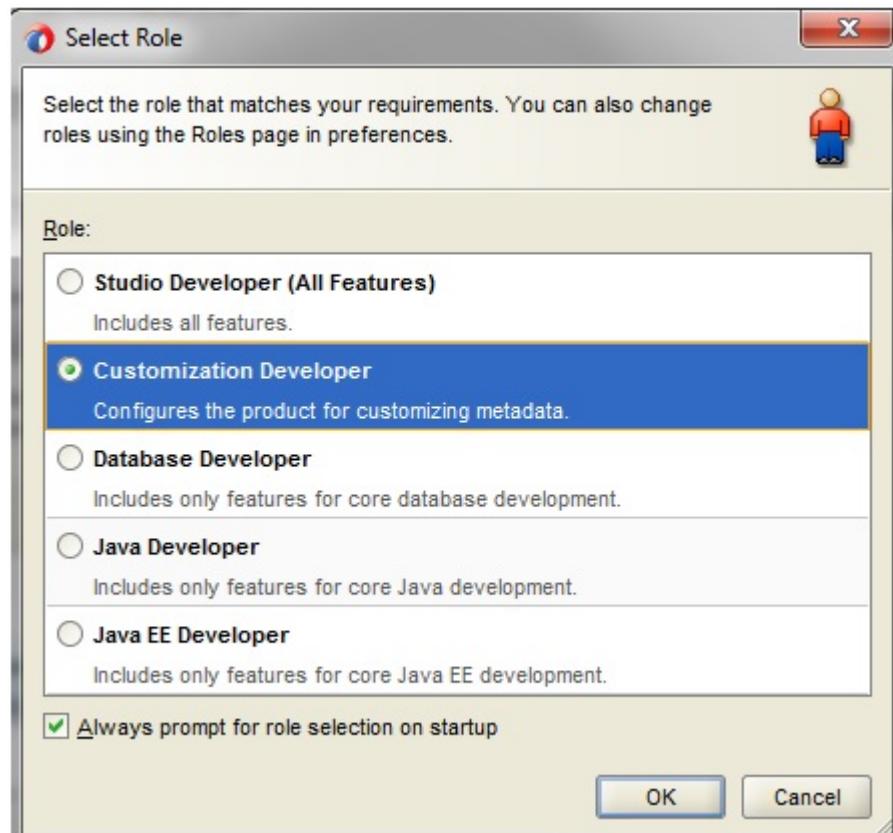
To customize ADF artifacts, open the default customization application workspace provided, using the **Customization Developer** role and customize the required artifacts.

Customizing the Artifacts

Users need to switch to the **Customization Developer** role before they can begin customizing.

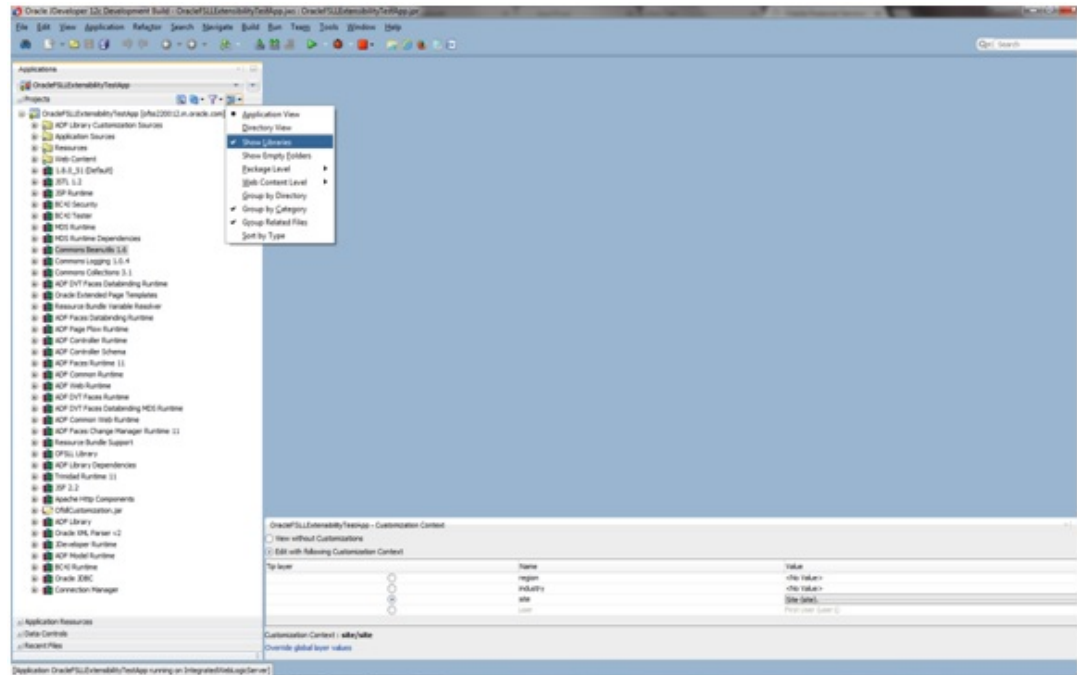
1. Restart JDeveloper and select the Customization Developer role.

Figure 2-3 Customizing the Artifacts 1



The artifacts from the imported library are displayed in the Application Navigator pane, under the ADF Library Customizations node, and the artifact selected to customize opens in the editor window.

Figure 2-4 Customizing the Artifacts 2



- In the Customization Context window (by default, displayed at the bottom of JDeveloper), select the layer that you want the customizations written to.
Note the following:

In case you want to change the value from customization.properties, you can follow the below steps:

- Step 1:** Extract the OfsllCustomization.jar using the following command. `Jar -xvf OfsllCustomization.jar`
- Step 2:** Modify the value in customization.properties
- Step 3:** Remove the old OfsllCustomization.jar, to build the jar again, please issue the following command `Jar -cvf OfsllCustomization.jar customization.properties oracle META-INF adf-loc.jar`

2.4 ADF Customization Best Practices

Listed below are some of the best practices that are captured during ADF customization. This list is iterative and includes the critical points discussed and captured from previous customizations.

- [ADF Customization Best Practices - Do's](#)
- [ADF Customization Best Practices - Don'ts](#)
- [ADF Customization Best Practices - Do's](#)
- [ADF Customization Best Practices - Don'ts](#)

2.4.1 ADF Customization Best Practices - Do's

- If there is a custom AM (Application Module) created as part of custom application, then you can nest it under base root application module. It's recommended to keep base application module as the main root module and route all custom transaction through it.
- You always need not explicitly create AM instances in backing beans. Instead create custom method in AMImpl class, expose them as client methods, and use operation bindings to call such methods.
- Use **finally** block to close expensive objects like RowSetIterators, Statements and Connections, wherever applicable.
- Every **request** that changes data in database should be complete i.e. commit or rollback must happen at the end of request.
- Try to reuse View Objects (VOs). This is true for data display and list of value VOs.
- New VOs should be tuned according to the business use case by tuning the fetch size and fetch mode properties.
- Use Bind Variables in VOs instead of hardcoded values in WHERE Clause, even for static values.
- If new custom bounded taskflows are created to show data in new tabs then make sure to control the taskflow activation and fetch data only when needed. For example, on click of tab rather than launch of page itself (assuming that tab is not the first tab).
- Write layer appropriate code i.e. avoid business logic in view layer. Another indicator to know would be, if you are importing jbo.* classes in view layer (backing bean) then you are misplacing the code and it actually belongs to model layer.
- Test with AM pooling off to make sure there is no adverse impact of passivation of AM, especially to transient and user entered values.
- Ensure that the ID for each component in the fragment is less than or equal to 7 characters in length.
- When bindings are created in the backing bean for any of the component in the fragment, ensure to use component references and not the component directly. Refer to below examples on the usage.
- Example of correct usage:

```
private ComponentReference<RichInputText> scraOrderRefNbr;
public RichInputText getScraOrderRefNbr() {
    return scraOrderRefNbr == null ? null :
    scraOrderRefNbr.getComponent();
}
public void setScraOrderRefNbr(RichInputText scraOrderRefNbr)
{
    this.scraOrderRefNbr =
    ComponentReference.newUIComponentReference(scraOrderRefNbr);
}
```

- Example of wrong usage:

```
private RichInputText doNotUse;
public void setDoNotUse(RichInputText doNotUse) {
    this.doNotUse = doNotUse;
}
```

```
}  
public RichInputText getDoNotUse() {  
    return doNotUse;  
}
```

- Any calls to DB packages should be done from the AM Impl method (Not applicable for pre/post insert/update which will be done directly at the respective Entity Impls) and that method should be invoked from the backing bean through operation binding, so never call the DB packages directly from the backing bean.

2.4.2 ADF Customization Best Practices - Don'ts

- Do not use value change listener to call database package that changes data in database table. Since same database connection is not guaranteed on every request, this could lead to uncommitted data hanging and probably undesirable results for other users who got that database connection assigned to their request.
- Do not perform commit or rollback after read-only database calls.
- Do not store component bindings/references in session scope beans. Use backingbean or request scope beans.
- Avoid making multiple DB calls for a single operation which would badly affect the performance of the system.
- Do not use `system.out.println()` or `printStackTrace()` anywhere in the code.
- Do not bind all the components in a fragment to the backing bean. Instead bind only the components on which you are going to perform operations programmatically.

3

Application Artifacts

This chapter describes how to use Oracle JDeveloper to customize and extend application artifacts defined by Oracle Application Development Framework (Oracle ADF) in Oracle Financial Services Lending and Leasing Application.

- [About Customizing Oracle ADF Application Artifacts](#)
- [Customizable Application Libraries](#)
- [Enable JDeveloper for Customization](#)
- [Customizing the Skin](#)
- [Customizing or Adding Resource Bundles](#)
- [Editing Existing Business Components](#)
- [Editing Pages](#)
- [Editing Task Flows](#)
- [Create Custom Business Components](#)
- [Create Custom Task Flows](#)
- [Create Custom Pages](#)
- [Editing the UI Shell Template](#)
- [Replacing OFSLL Logo](#)
- [Deploy ADF Customizations and Extensions](#)
- [Deployment Options](#)
- [About Customizing Oracle ADF Application Artifacts](#)

With the customization features provided by Oracle Metadata Services (MDS), developers can customize the application using JDeveloper, making modifications to suit the needs of a particular group, such as a specific region or industry or site.
- [Customizable Application Libraries](#)

All customization in the application would be done on the ADF Libraries. List of libraries that can be customized and set of default projects that can be used for building the projects are:
- [Enable JDeveloper for Customization](#)

Before running the JDeveloper in Customization Developer role, JDeveloper needs to be configured with following settings:
- [Customizing the Skin](#)

One method of customizing skin is opening the bundled `OracleFSSLCommonSkin/OracleFSSL CommonSkin.jws` project in Oracle ADF Skin Editor Application and customizes the skin details. Once the skin details are customized the same can be bundled as ADF library and deployed to the application server.
- [Customizing or Adding Resource Bundles](#)

One method of customizing label is by overriding values for existing keys defined in the resource bundle, but new keys cannot be added.

- [Editing Existing Business Components](#)
Before you start customizing business objects, it has to be determined which business objects need customizing. Then when customizing ADF artifacts, JDeveloper has to be launched in the **Customization Developer** role, and the appropriate layer selected.
- [Editing Pages](#)
JDeveloper can be used to implement customizations on the pages that are used in the application. When editing a page in JDeveloper, JDeveloper must be launched in the Customization role.
- [Editing Task Flows](#)
JDeveloper can be used to implement customizations on the task flows that are used in the application.
- [Create Custom Business Components](#)
JDeveloper can be used to extend the application by creating custom business components.
- [Create Custom Task Flows](#)
JDeveloper can be used to create custom task flows that can be included in the application.
- [Create Custom Pages](#)
JDeveloper can be used to create custom pages that can be included in the application. When creating custom pages in JDeveloper, JDeveloper must be launched in the Default role.
- [Editing the UI Shell Template](#)
To edit the UI Shell template in JDeveloper, in the Customization Developer role, select the `OFSLLUiShellTemplate.jsf` file and open and changes can be made as necessary.
- [Replacing OFSLL Logo](#)
OFSLL Logo can be changed through one time customization.
- [Deploy ADF Customizations and Extensions](#)
After customizing existing artifacts, JDeveloper can be used to deploy the customizations to Oracle Weblogic Server.
- [Deployment Options](#)

3.1 About Customizing Oracle ADF Application Artifacts

With the customization features provided by Oracle Metadata Services (MDS), developers can customize the application using JDeveloper, making modifications to suit the needs of a particular group, such as a specific region or industry or site.

Using JDeveloper, you can implement customizations on existing artifacts that are provided. The application can also be extended with new custom artifacts that are packaged into a JAR file, and integrated using customizations on the existing application.

However customizations to the application require a lower level approach, for which JDeveloper needs to be used.

3.2 Customizable Application Libraries

All customization in the application would be done on the ADF Libraries. List of libraries that can be customized and set of default projects that can be used for building the projects are:

Table 3-1 List of libraries

Library Name	Description
adflibOfsllCommonModel.jar	Contains all the application Business Objects such as entity object, view object and application module.
adflibOfsllCommonUI.jar	Contains all the User Interface fragments (JSFF) and taskflows (TFs) and all re-useable templates.

Table 3-2 Default projects

Project Name	Description
OracleFSLLEnterpriseApp/ OracleFSLLEnterpriseApp.jws	Enterprise EAR Application deployment project. This is the default customization main project used to bundle all the libraries into an EAR.
OracleFSLLCommonSkin/ OracleFSLLCommonSkin.jws	Application Skin project, containing images and CSS file. The skin project changes can be handled through Oracle ADF Skin Editor.
OracleFSLLCustomization/ OracleFSLLCustomization.jws	Customization project containing the customization layer values i.e. region layer, Industry layer and site layer key value pair.

 **Note:**

- Above projects are available as part of the application installer bundle under / cust_lib folder.
- The customizable libraries can be extracted out of /core_as/*.ear file. Extract the *.war out of *.ear and the libraries are under /WEB-INF/lib folder.
- Currently existing menu items cannot be customized as well as new menu items cannot be added.

3.3 Enable JDeveloper for Customization

Before running the JDeveloper in Customization Developer role, JDeveloper needs to be configured with following settings:

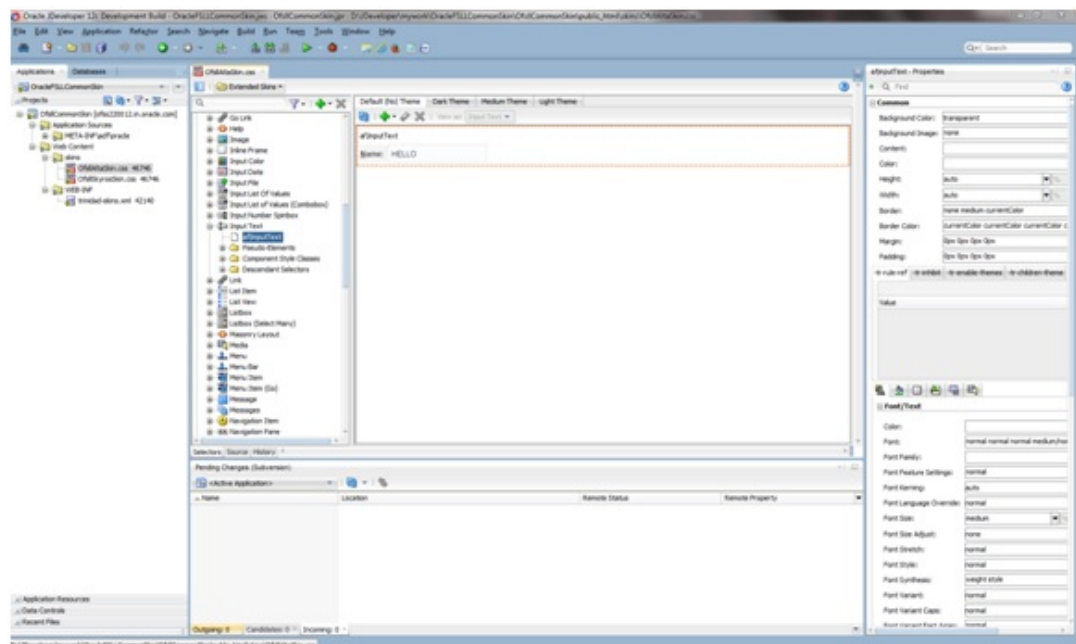
1. Open JDeveloper in Default role and open the OracleFSLLCustomization/ OracleFSLLCustomization.jws project and edit the customization.properties file with appropriate values for Region key layer, Industry key layer and Site key layer.
2. Rebuild the OfslCustomization.jar using the default deployment profile.
3. Edit the /cust_lib/CustomizationLayerValues.xml in Notepad and update the Region key layer, Industry key layer and Site key layer with the values added as per required customization.properties.
4. Copy the CustomizationLayerValues.xml onto JDeveloper installation location under \$JDEV_HOME/jdeveloper/jdev.

3.4 Customizing the Skin

One method of customizing skin is opening the bundled `OracleFSLCommonSkin/OracleFSLCommonSkin.jws` project in Oracle ADF Skin Editor Application and customizes the skin details. Once the skin details are customized the same can be bundled as ADF library and deployed to the application server.

1. Open the OracleFSLCommonSkin Project in Oracle ADF Skin Editor Application.
2. Select the component through selectors structure which needs to be customized.
3. Go to Property Inspector and make necessary changes.

Figure 3-1 Customizing the Skin



4. Make ADF Library JAR through deployment profile defined with this project.
5. Copy the JAR into **OracleFSLEnterpriseApp** to build the EAR.

Note:

Skin can be customized using Oracle ADF Skin Editor which can be downloaded from Oracle site. If the default skin family name is changed then `trinidad-config.xml` available in `OracleFSLEnterpriseApp` needs to be changed with new skin family name.

3.5 Customizing or Adding Resource Bundles

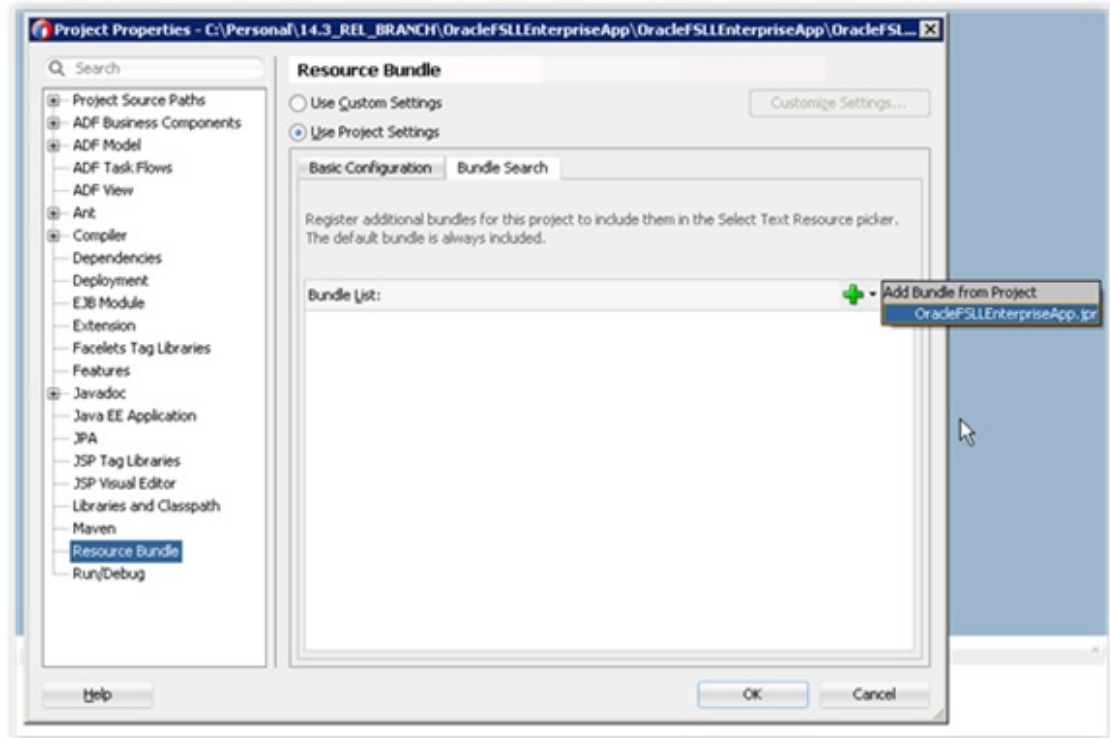
One method of customizing label is by overriding values for existing keys defined in the resource bundle, but new keys cannot be added.

Because a new key cannot be defined in the shipped resource bundle, a new override bundle needs to be created. This can be accomplished in JDeveloper by creating an XLIFF file from

the New Gallery. After the file is generated, new keys and their associated text in the XLIFF file can be entered.

To make the newly created resource bundle available for customization, the resource bundle needs to be registered with the customization project. Newly created resource bundle can be present in customization project or as a separate project. To register the resource bundle with customization project, package it into an ADF Library JAR file, and import the JAR file into the customization project.

Figure 3-2 Adding Resource Bundles 1



Step to override a message bundle which already exists in model or ui jar, is shown below:

1. Open the JDeveloper in Default role, select the application, click on Application (Menu) ▢ Application Properties. Select Resource Bundles, add the resource bundles here from jars present under lib folder and check **Overridden** check box. This will register the all selected bundles with `adf-config.xml` file.

Figure 3-3 Adding Resource Bundles 2

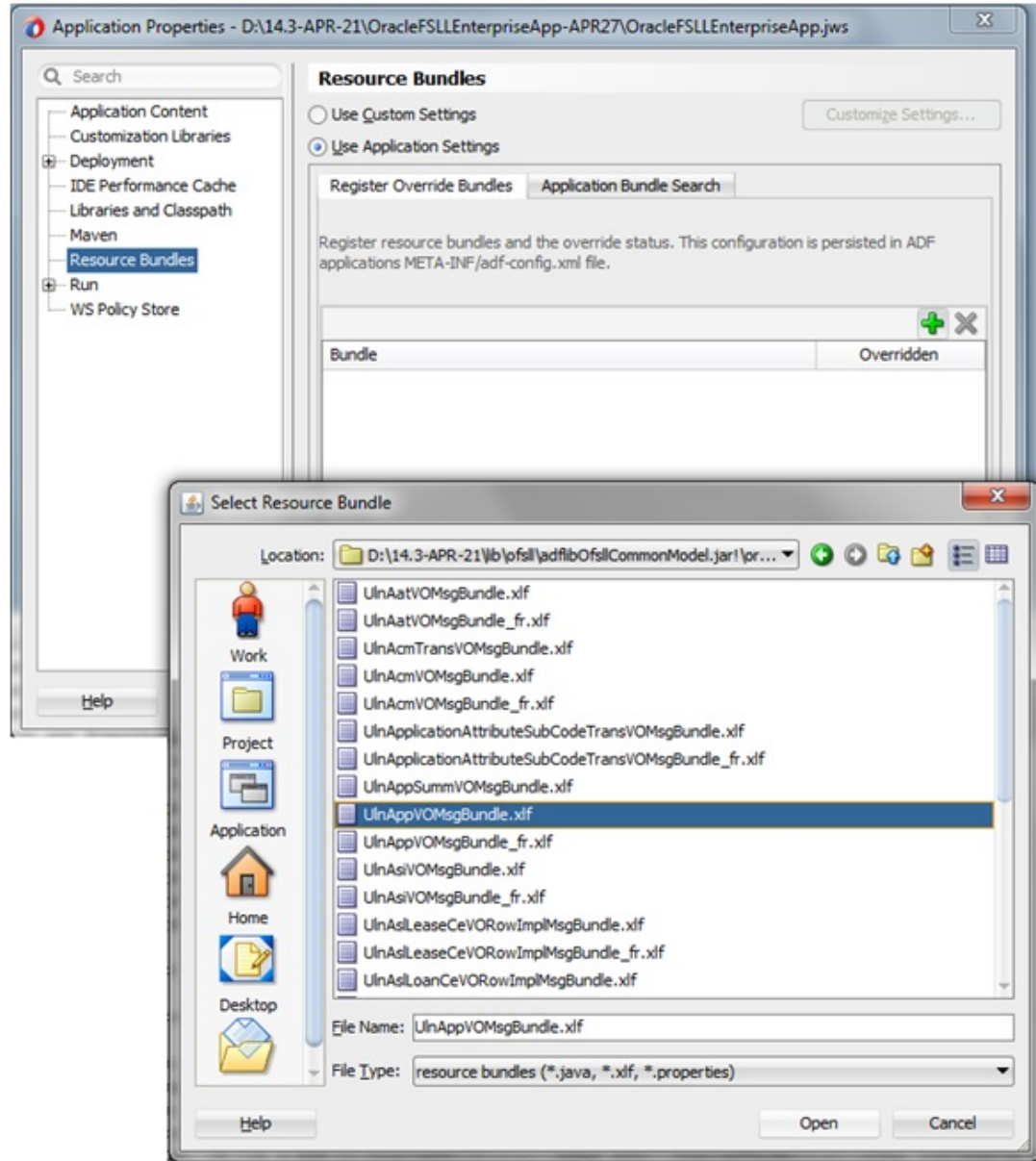
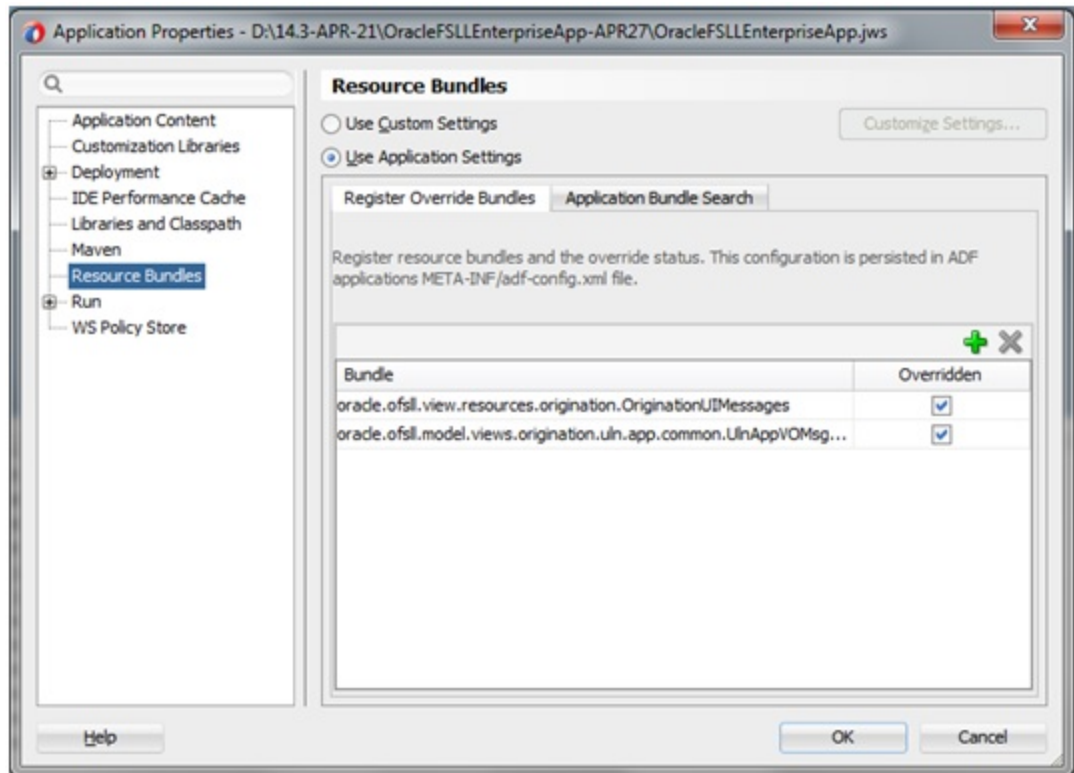
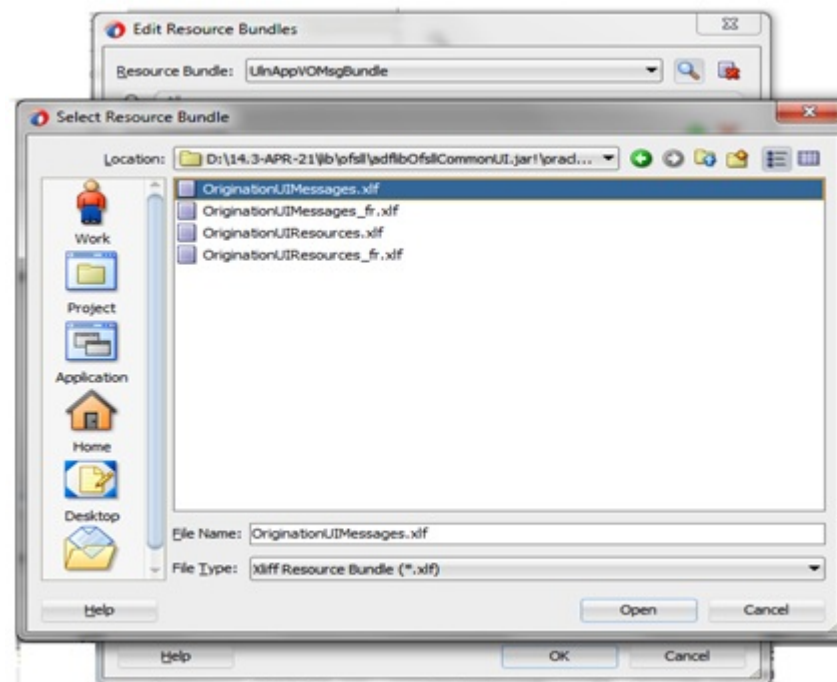


Figure 3-4 Adding Resource Bundles 3



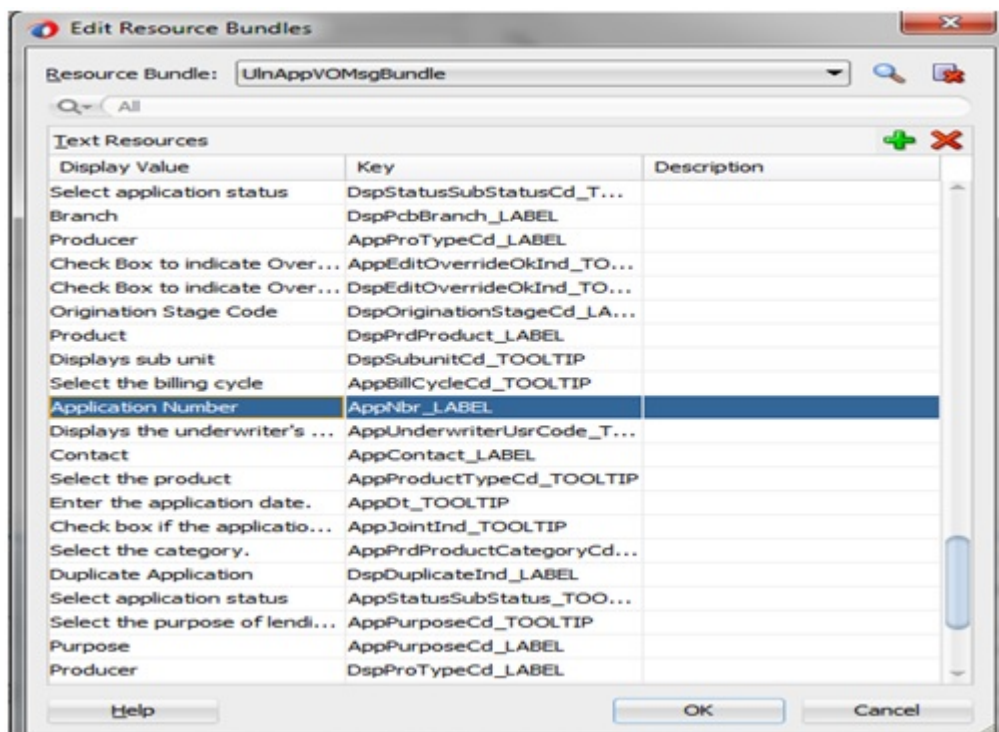
2. The Edit or Override Resource Bundles, Go to Customization Role, Select **Edit Resource Bundle** present under Application Menu. Navigate to the XLIFF Bundle that needs to be overridden from the jar under lib folder.

Figure 3-5 Adding Resource Bundles 4



3. Change Display Value for the Key as per requirement.

Figure 3-6 Adding Resource Bundles 5



4. Once changes are submitted, the override resource bundle folder would be created with overridden values.

Figure 3-7 Adding Resource Bundles 6



3.6 Editing Existing Business Components

Before you start customizing business objects, it has to be determined which business objects need customizing. Then when customizing ADF artifacts, JDeveloper has to be launched in the **Customization Developer** role, and the appropriate layer selected.

Task: Edit Attributes

The properties of an attribute can be customized from an entity object or view object using JDeveloper. When an entity object opened or viewed in the overview editor, the attributes of the object can be seen on click in the Attributes tab. When an attribute is selected, its properties are displayed in the Property Inspector.

It is not necessary to modify the page after customizing the properties of an existing attribute. Customizations to existing attributes are automatically reflected on the pages that show them.

However, if an attribute is modified so that it requires a different UI component, it must also be updated in the page. For example, if a list of values (LOV) is added to an attribute, the page needs to be edited to hide the existing UI component that displays the attribute, and a new UI component added that can display the LOV.

Note that some attribute properties defined in the entity object can be overridden in the view object. For example, the label text for a field can be defined in an entity object and subsequently given a different label in the consuming view object. Then pages that use the view object display the label from the view object.

Figure 3-8 Editing Business Components 1

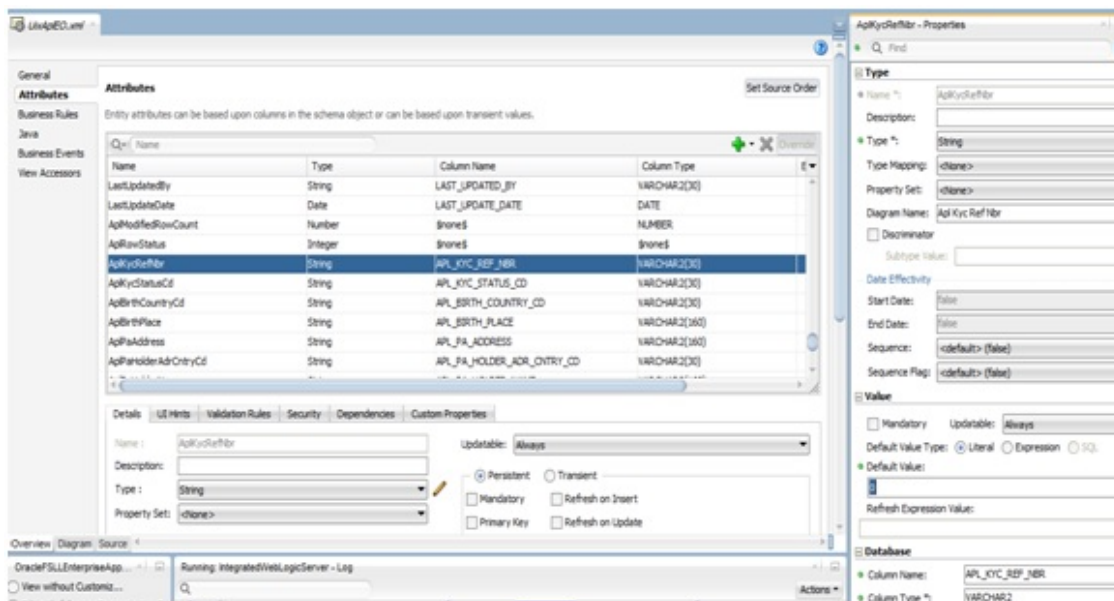
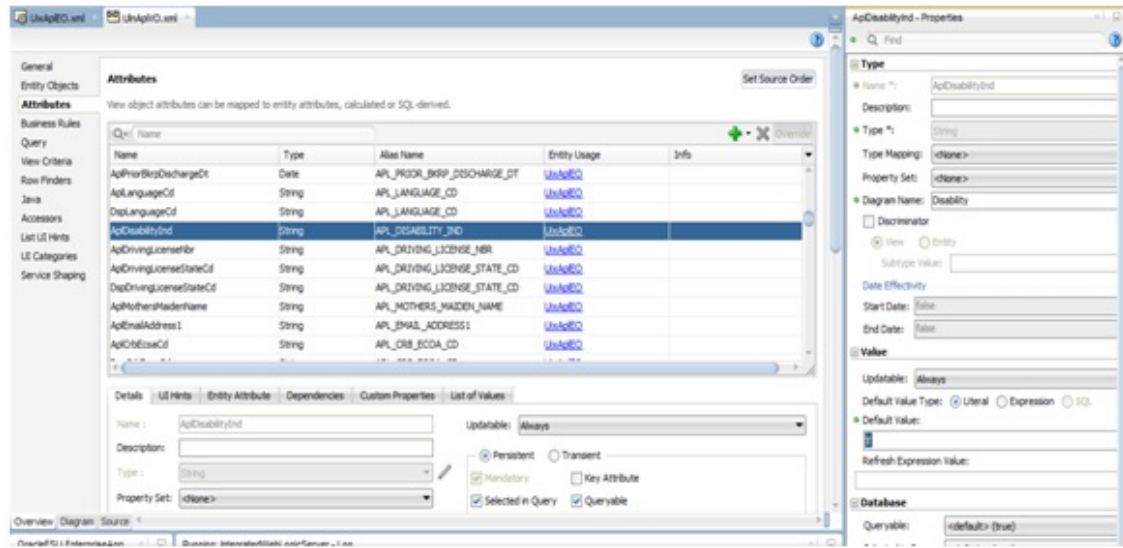


Figure 3-9 Editing Business Components 2



Task: Add Attributes

Custom attributes can be added to an entity object or view object using JDeveloper. To do this, JDeveloper must be launched in the **Customization Developer** role, a layer selected. Open an entity object or view object in the overview editor, and click the Attributes tab to see the attributes of the object. To add a custom attribute, click the Add icon.

To store the custom attribute in the database, first create the column that will be used to store it.

To display the custom attributes in the application, the pages also needs to be customized to display them.

Figure 3-10 Editing Business Components 3

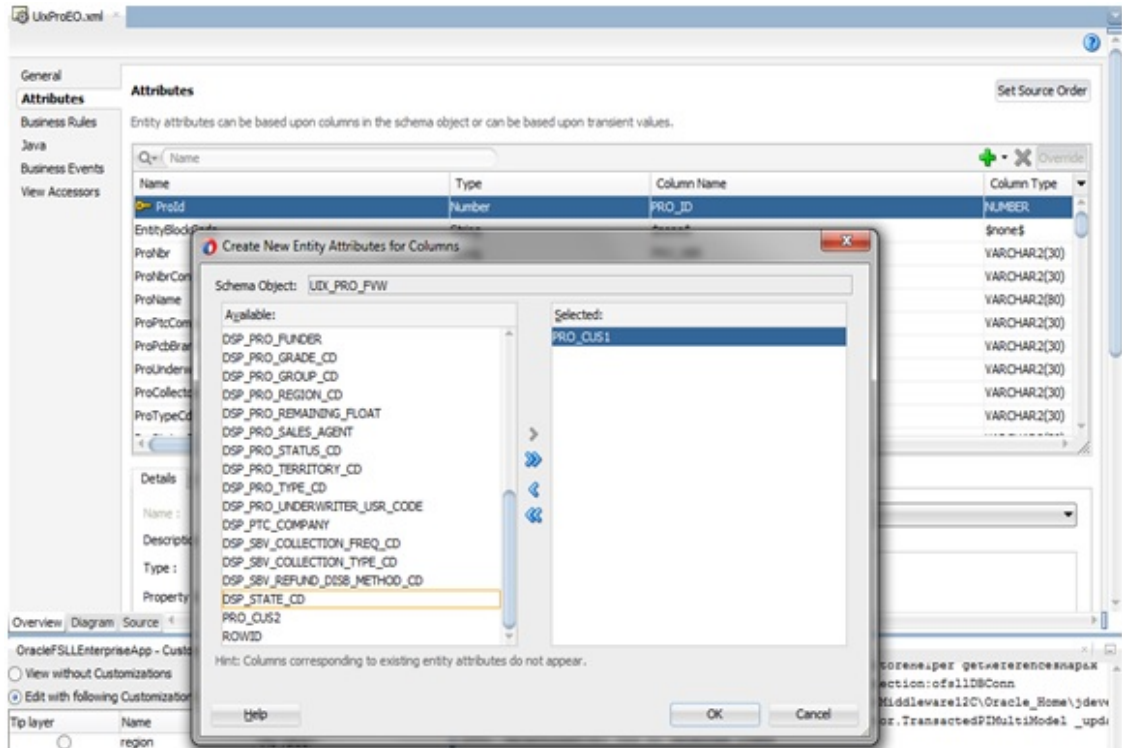
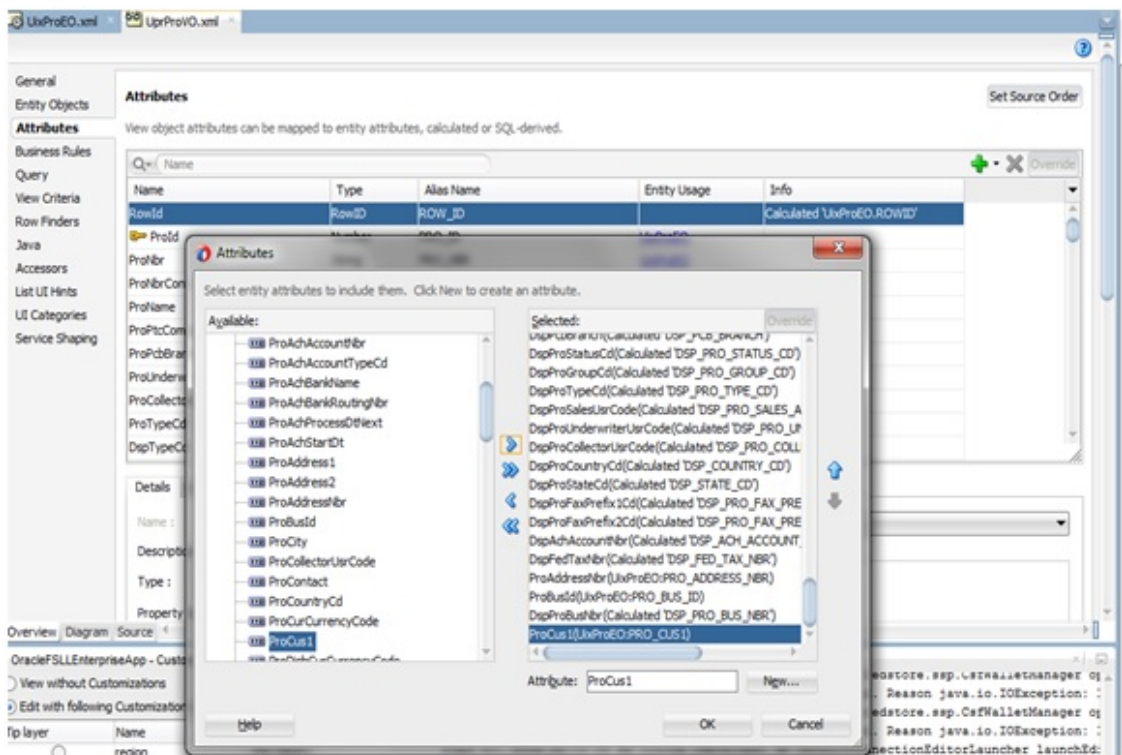


Figure 3-11 Editing Business Components 4

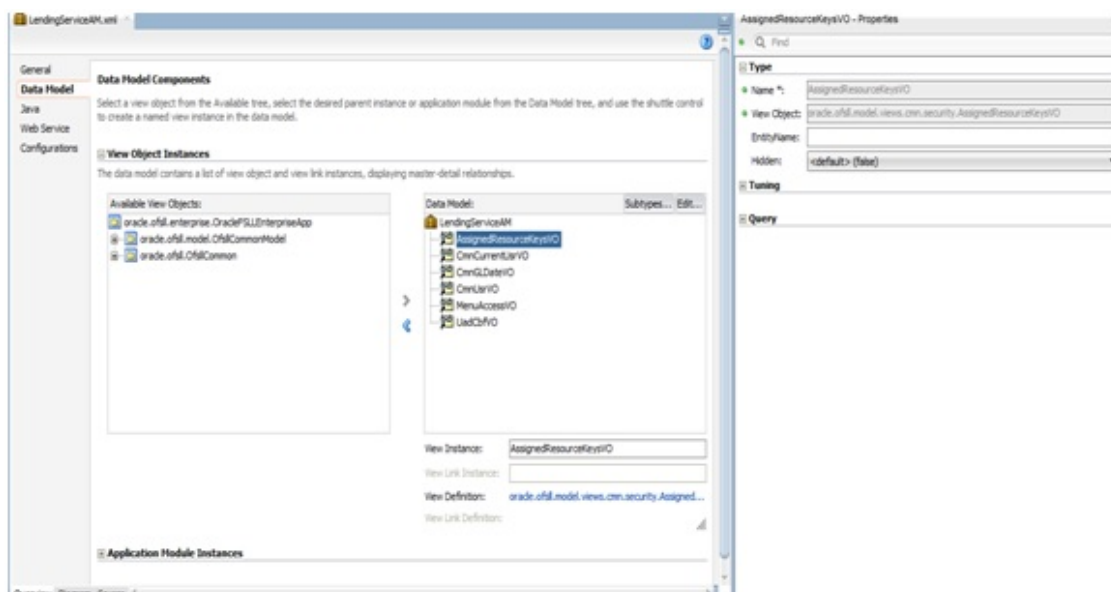


Task: Edit Entity Objects

In JDeveloper, edit entity objects using the overview editor. In the Application Navigator, right-click an entity object, and choose **Open**. Then click on the navigator tabs to view and edit the various features of the entity object.

Task: Edit Application Modules

In JDeveloper, edit application modules using the overview editor. In the Application Navigator, right-click an application module, and choose **Open**.

Figure 3-12 Editing Business Components 5

In JDeveloper, the following kinds of customizations can be made on an application module:

- Add new custom properties. This is done on the General page of the overview editor.
- Add new view object and application module instances. This is done on the Data Model page of the overview editor.
- Add newly created subtype view objects. This is done on the Data Model page of the overview editor.
- Add new application module configurations. This is done on the Configurations page of the overview editor.

Once the changes are applied, the MDS file is created based on the customization layer value shown below.

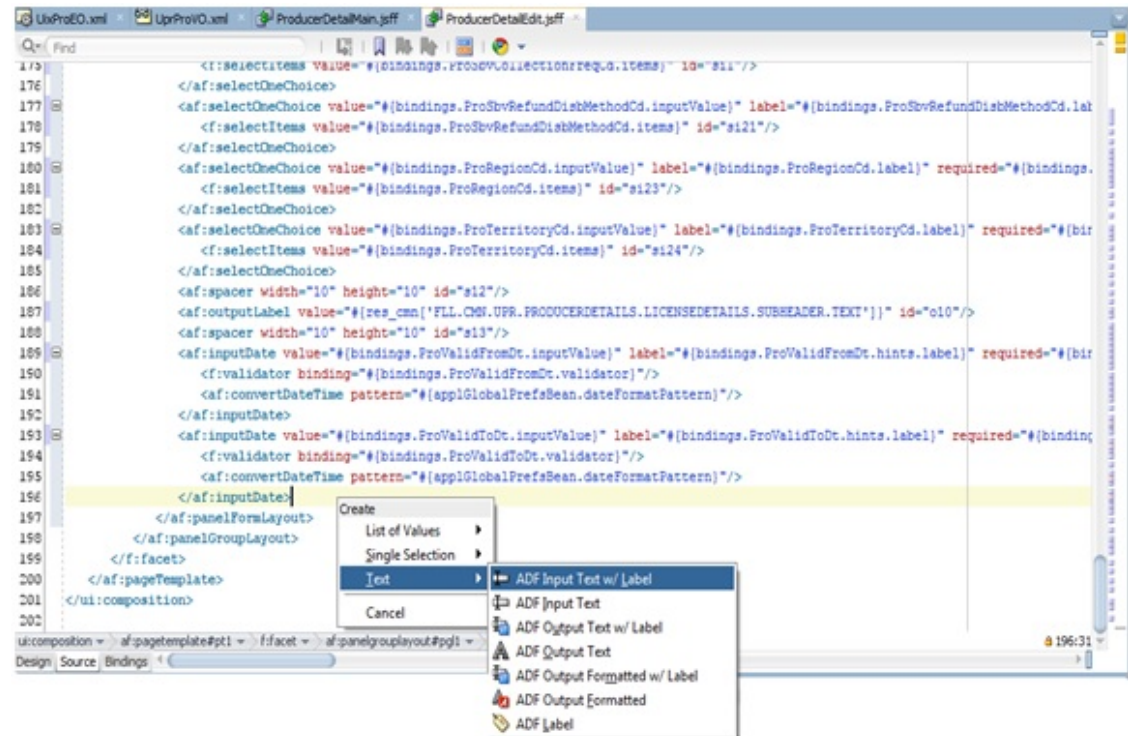
3.7 Editing Pages

JDeveloper can be used to implement customizations on the pages that are used in the application. When editing a page in JDeveloper, JDeveloper must be launched in the Customization role.

Task: Edit Pages

In the Application Navigator, right-click the page that has to be customized and choose **Open**. Either new component can be added or existing components properties can be changed via property inspector.

Figure 3-13 Edit Pages



3.8 Editing Task Flows

JDeveloper can be used to implement customizations on the task flows that are used in the application.

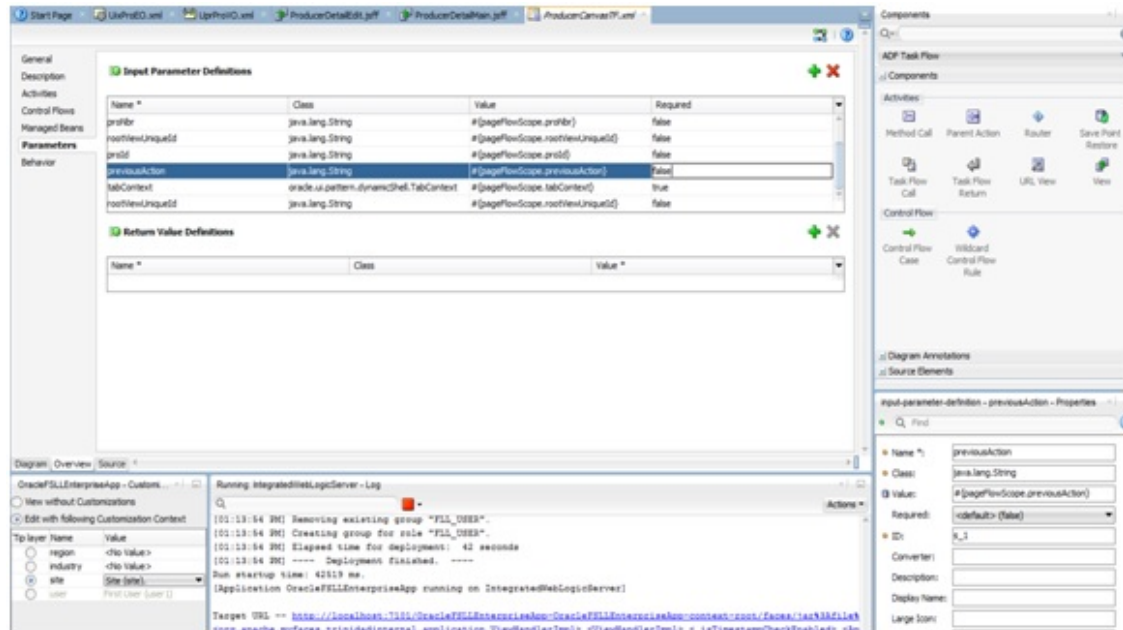
A task flow is a set of ADF Controller activities, control flow rules, and managed beans that interact to allow a user to complete a task. Although conceptually similar, a task flow is not the same as a human task, a task in the worklist, or a process flow.

A bounded task flow can be rendered in a JSF page or page fragment (`.jsff`) by using an ADF region. This is typically done to allow reuse of the task flow, as necessary, throughout the application. If a bounded task flow is modified, the changes apply to any ADF region that uses the task flow.

Task: Edit Task Flows

In JDeveloper, the task flow diagram editor is used to implement customizations on existing task flows. In the Application Navigator, right-click the task flow that has to be customized and choose **Open**. The page is displayed in the diagram editor, where changes can be made to the existing activities and control flow cases, or create new custom ones. And in the Overview editor also changes can be made.

Figure 3-14 Edit Task Flows



3.9 Create Custom Business Components

JDeveloper can be used to extend the application by creating custom business components.

When creating custom business components in JDeveloper, JDeveloper must be launched in the **Default** role. This role is used for creating new custom objects that needs to be added to the application. The same workspace that was created for customization can be used. After the custom business components are created, switch to the **Customization Developer** role, to make changes to existing artifacts to integrate the new custom artifacts into the application.

Task: Create Custom Entity Objects

An entity object represents a row in a database table, and encapsulates the business logic and database storage details of business entities.

In JDeveloper, entity objects can be created using the Create Entity Object wizard, which can be launched from the New Gallery. In the Application Navigator, right-click the project that has to be added to the entity object, and choose **New**. Then in the New Gallery, expand Business Tier, click **ADF Business Components**, choose Entity Object, and click **OK**. Follow the prompts in the wizard to create an entity object.

Task: Create Custom View Objects

A view object represents a SQL query and also collaborates with entity objects to consistently validate and save the changes when end users modify data in the UI.

In JDeveloper, view objects can be created using the Create View Object wizard, which can be launched from the New Gallery. In the Application Navigator, right-click the project that has to be added to the view object, and choose **New**. Then in the New Gallery, expand Business Tier, click **ADF Business Components**, choose View Object, and click **OK**. Follow the prompts in the wizard to create a view object.

Task: Create Custom Application Modules

An application module encapsulates an active data model and the business functions for a logical unit of work related to an end-user task.

In JDeveloper, application modules can be created using the Create Application Module wizard, which can be launched from the New Gallery. In the Application Navigator, right-click the project that has to be added to the application module, and choose **New**. Then in the New Gallery, expand Business Tier, click **ADF Business Components**, choose Application Module, and click **OK**. Follow the prompts in the wizard to create an application module.

Task: Add Validation

In JDeveloper, declarative validation rules can be created for entity objects and view objects to help ensure the integrity of the data. To do this, open the entity object or view object in the overview editor, and click the **Business Rules navigation** tab. Then select the attribute for which validation needs to be provided, click the **Create new validator** icon, and use the Add Validation Rule dialog to configure the rule.

3.10 Create Custom Task Flows

JDeveloper can be used to create custom task flows that can be included in the application.

A task flow is a set of ADF Controller activities, control flow rules, and managed beans that interact to allow a user to complete a task. Although conceptually similar, a task flow is not the same as a human task, a task in the worklist, or a process flow.

Task: Create a Custom Task Flow

A custom task flow can be created in JDeveloper using the New Gallery, and then its activities defined using the task flow diagram editor. In the Application Navigator, right-click the project that has to be added to the task flow, and choose **New**. Then in the New Gallery, expand Web Tier, and click **JSF/Facelets**. Then select **ADF Task Flow**, and click **OK**. In the Create Task Flow dialog, specify the details about the type of task flow that needs to be created. Click **OK** and the task flow is created and displayed in the diagram editor.

3.11 Create Custom Pages

JDeveloper can be used to create custom pages that can be included in the application. When creating custom pages in JDeveloper, JDeveloper must be launched in the Default role.

When creating the page (or dropping a view activity onto a task flow), it can be created either as a JSF JSP or as a JSF JSP fragment. JSF fragments provide a simple way to create reusable page content in a project, and are used for task flows as regions on a page. When a JSF page fragment is modified, the JSF pages that consume the page fragment are automatically updated.

After extending the application with custom pages, it is required to make sure that security for the new pages are implemented appropriately and that the new pages are deployed so that they are accessible from the application.

Task: Create a Custom Page

In JDeveloper, pages can be created either by double-clicking a view activity in a task flow or by using the New Gallery. In the Application Navigator, right-click the project to which the page has to be added to, and choose **New**. Then in the New Gallery, expand Web Tier, and click **JSF/Facelets**. Then select either Page or ADF Page Fragment, and click **OK**.

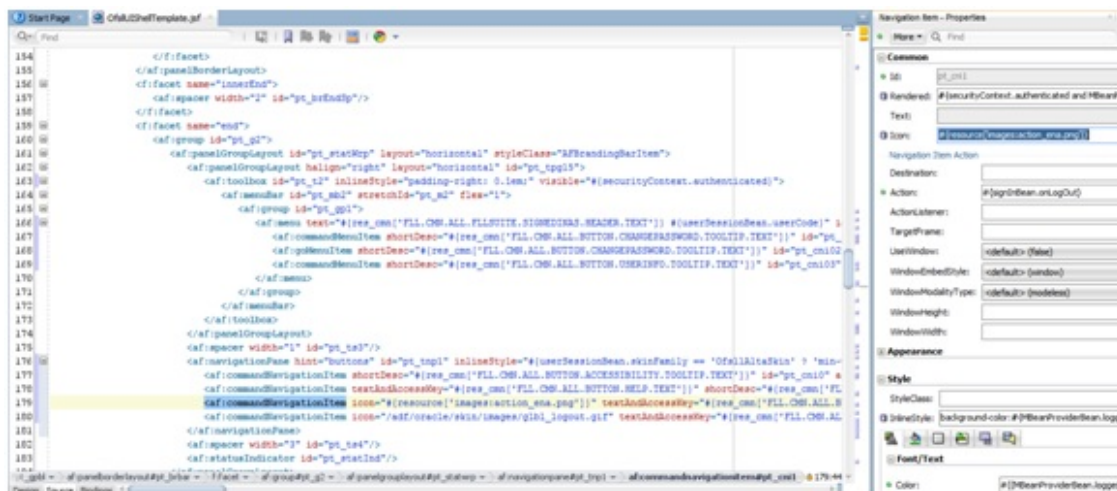
Task: Add a Custom Page to a Task Flow

If the page is created by double-clicking a view activity in a task flow, it is already added to the task flow. If it is created using the New Gallery, it can be added to a task flow by dragging the page from the Application Navigator and dropping it in the task flow diagram editor. Then connect the page using a control flow.

3.12 Editing the UI Shell Template

To edit the UI Shell template in JDeveloper, in the Customization Developer role, select the `OfsllUiShellTemplate.jsf` file and open and changes can be made as necessary.

Figure 3-15 Edit the UI Shell Template



3.13 Replacing OFSLL Logo

OFSLL Logo can be changed through one time customization.

The image path of OFSLL logo is referenced in both UI shell template and sign-in template through the `OfsllHome.jsf` and `OfsllSignIn.jsf` respectively and the same can be customized by making changes to these JSF files.

1. In the customization project, create a folder under `public_html` directory and add the new logo file.
2. Launch JDeveloper in Customization mode and load the customization project.
3. Open `OfsllHome.jsf` from ofsll library.

Locate the line

```
<f:attribute name="logoImagePath" value="/adf/oracle/skin/images/ofsll_logo.gif"/>
```

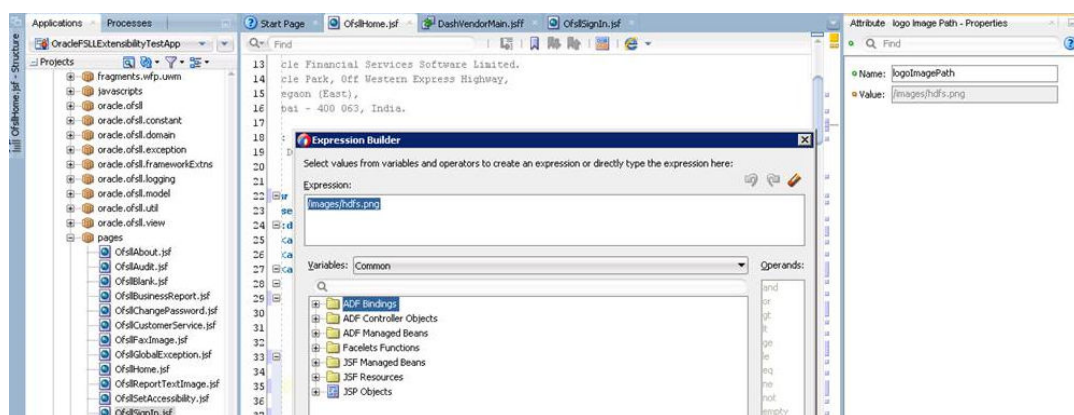
in the code.

Figure 3-16 Replacing OFSLL Logo 1



4. From property inspector's expression builder, change the value to refer the new logo file and click **ok**. Customization file for `OfsslHome.jsf` gets created and refers to the new logo file at runtime.
5. To refer the new logo in sign in page too, repeat steps 3 and 4 in `OfsslSignIn.jsf`.

Figure 3-17 Replacing OFSLL Logo 2



3.14 Deploy ADF Customizations and Extensions

After customizing existing artifacts, JDeveloper can be used to deploy the customizations to Oracle Weblogic Server.

The default customization workspace as described in Section 3.1, **About Using JDeveloper for Customization**, contains a MAR profile. By default, the name of the MAR profile is `application_name_customizations`. It will automatically include the customizations that are implemented. This profile can be used to package the customizations for deployment.

When customizations are packaged from the customization workspace, the MAR file should include only library customizations. Do not include the User Metadata or HTML Root Dir for Project in the MAR profile, unless explicitly directed to do so by product documentation.

If the application is extended with new custom artifacts, JDeveloper can be used to package them into an ADF Library JAR and place them into the proper location within the application directory structure.

Task: Deploy the Customizations

JDeveloper can be used to deploy the customizations directly or to create a MAR, and then load the MAR using WLST commands or the WebLogic Server Administration Console.

When customizations are deployed on ADF Business Component objects (such as entity objects and view objects), the server must be restarted for the customizations to be picked up.

Task: Package New Artifacts into ADF Library

If the application is extended with new custom artifacts (or new artifacts are supplied with), these artifacts must be packaged into an ADF library JAR and place the JAR files in the proper location within the application.

The ADF library JAR for the new model artifacts (such as entity objects and view objects) should be placed into the `/APP-INF/lib` directory. The ADF Library JAR for the new user interface artifacts (such as pages) should be placed in the `/WEB-INF/lib` directory.

3.15 Deployment Options

The Deployment or EAR creation of the application would be done through OracleFSLLEnterpriseApp project. In this Project, JPR has the necessary deployment profiles available. Deployment of the application on to Weblogic Server is defined as per **Install UI Components to Application Server** document.

Note:

- In **Customization Developer** role, the project creates the MAR deployment profile for customization deployment.
- MAR deployment is same as EAR deployment.

4

Customize Database Objects

The following section details how to customize database objects.

- [UI – Package Interaction Logic](#)
- [UI Java Wrapper \(U*JW\)](#)
- [Database Schema](#)
- [Wrapper Engine model](#)
- [Batch Job \(BJ\)](#)
- [Engine Wrapper \(EW\)](#)
- [Main Engine \(EM\)](#)
- [Engine Function \(EN\)](#)
- [Engine View](#)
- [Common Features](#)
- [Seed Data](#)
- [Developer's Tips](#)
- [UI – Package Interaction Logic](#)

OFSLL uses the Oracle Fusion Middleware based ADF user interface. Below mentioned image show how OFSLL user interfaces interacts with the Java wrapper.
- [UI Java Wrapper \(U*JW\)](#)

If the java wrapper engine needs to be customized, follow the steps given below:
- [Database Schema](#)

Oracle Financial Services Lending and Leasing has the below mentioned Database Objects.
- [Wrapper Engine model](#)

Below mentioned is the naming convention for Wrapper Engine model used in Oracle Financial Services Lending and Leasing.
- [Batch Job \(BJ\)](#)

Batch Job **cannot** be customized, it has to be developed as a new job.
- [Engine Wrapper \(EW\)](#)

Engine Wrapper **cannot** be customized.
- [Main Engine \(EM\)](#)

To customize the main engine, follow the steps given below:
- [Engine Function \(EN\)](#)

To customize an engine function, follow the steps given below:
- [Engine View](#)

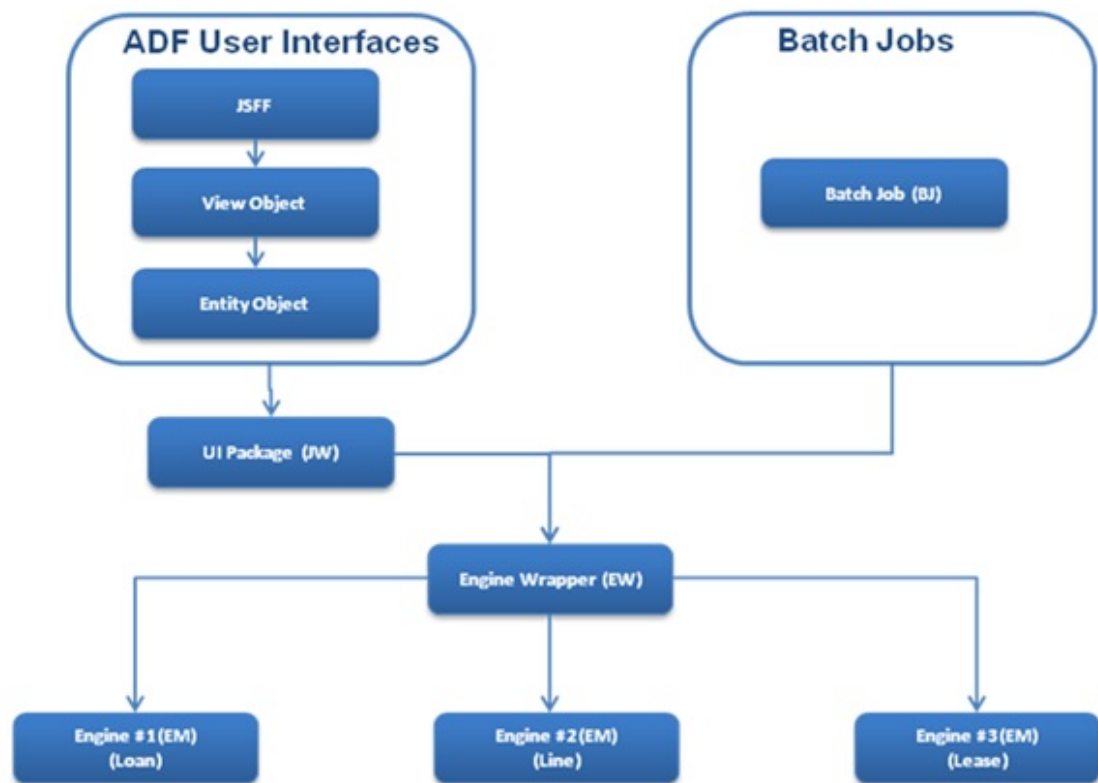
To customize an Engine View (EVW), follow the steps given below:
- [Common Features](#)

- **Seed Data**
Oracle Financial Services Lending and Leasing Seed data tables are classified in following three categories:
- **Developer's Tips**
Suppose the account number generation needs to be customized different from what OFSLL generates. Requirement is to replace the baseline format with its own format (like ACC-NNNNNNN).

4.1 UI – Package Interaction Logic

OFSLL uses the Oracle Fusion Middleware based ADF user interface. Below mentioned image show how OFSLL user interfaces interacts with the Java wrapper.

Figure 4-1 UI – Package Interaction Logic



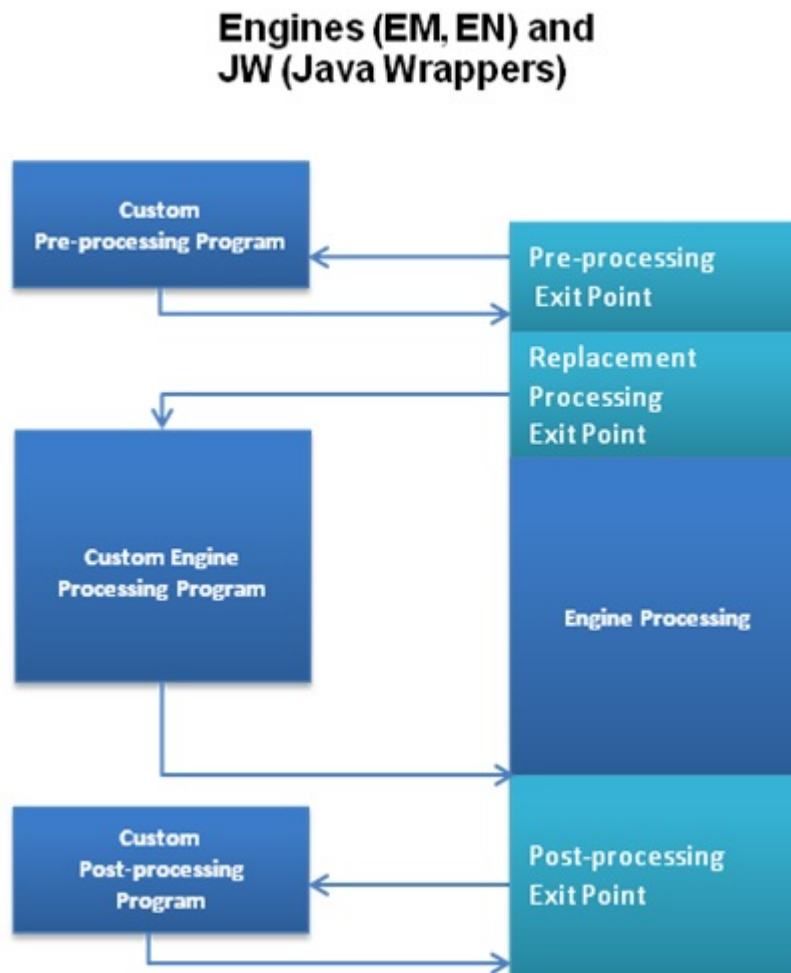
4.2 UI Java Wrapper (U*JW)

If the java wrapper engine needs to be customized, follow the steps given below:

1. Select the Exit point for the customization.
2. Rename the exit point package file name with `_xyz`. Do not change Package name.
3. Change the variable CV from NON_CUSTOMIZED to CUSTOMIZED depending upon the exit point before, replace or after.
4. Write the required customized engine library and call it in the java wrapper Exit Points Package (EX).

Engines (EM, EN) and JW (Java Wrappers)

Figure 4-2 Business Logic Engine (Process)

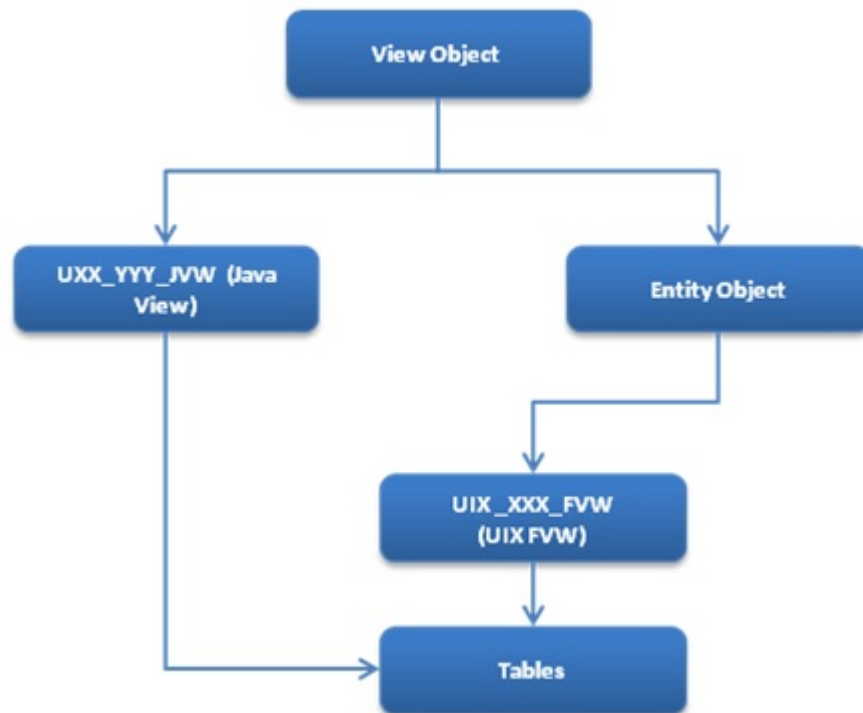


4.3 Database Schema

Oracle Financial Services Lending and Leasing has the below mentioned Database Objects.

- Table
- Table Column
- Sequence
- Index
- View
 - FVW – User Interface Views
 - JWV – Java Interface Views
 - EVW – Engine/Wrapper Signature Views
 - PL/SQL Programs

Figure 4-3 Database Schema



4.4 Wrapper Engine model

Below mentioned is the naming convention for Wrapper Engine model used in Oracle Financial Services Lending and Leasing.

Table 4-1 Wrapper Engine model

XXXYYY_ZZ_ABC_99	XXXYYY_ZZ_ABC_99
XXX Module or Engine	A System
YYY Function	0-Common
ZZ Program Type	1-Consumer
EM Engine Main	2-Commercial
EN Engine Function	(Always 0 for Wrapper)
EW Engine Wrapper	B Product Type
EL Engine Library	0-Common
EX Engine User Exits	1-Loan
JW Java Wrapper	2-Lease
BJ Batch Job	3-WFP
BL Batch Job Library	(Always 0 for Wrapper)
CL Common Library	C Product Sub Type
	0-Common
	1-Closed Ended
	2-Open Ended
	(Always 0 for Wrapper)

Table 4-1 (Cont.) Wrapper Engine model

XXXXYY_ZZ_ABC_99	XXXXYY_ZZ_ABC_99
	99 Running Sequence Number
	Starting 01 to 99

4.5 Batch Job (BJ)

Batch Job **cannot** be customized, it has to be developed as a new job.

4.6 Engine Wrapper (EW)

Engine Wrapper **cannot** be customized.

4.7 Main Engine (EM)

To customize the main engine, follow the steps given below:

- Select the Exit point for the customization.
- Rename the exit point package file name with `_xyz`. Do not change Package name.
- Change the variable CV from NON_CUSTOMIZED to CUSTOMIZED depending upon the exit point before, replace or after.
- Write your customized engine and call it in the Engine Exit Points Package (EX).

4.8 Engine Function (EN)

To customize an engine function, follow the steps given below:

- Select the Exit point for the customization.
- Rename the exit point package file name with `_xyz`. Do not change Package name.
- Change the variable CV from NON_CUSTOMIZED to CUSTOMIZED depending upon the exit point before, replace or after.
- Write your customized engine function and call it in the Engine Exit Points Package (EX).

4.9 Engine View

To customize an Engine View (EVW), follow the steps given below:

- Do Not modify the OFSLL Base Engine View Script.
- Create a copy of the OFSLL Base Engine View Script, rename and modify that Engine View Script.

Do **not** modify the OFSLL Base Engine View Name.

4.10 Common Features

- Error Logging

- Alert Log
- Debugging
 - Debug Log
- Version Control Header in each code unit

4.11 Seed Data

Oracle Financial Services Lending and Leasing Seed data tables are classified in following three categories:

System

- Only Oracle Financial Software Services Ltd. can change/update this data.

Combination

- Oracle Financial Software Services Ltd. or customer can change/update this data. It is recommended to identify all the new customized seed data records with a customer identifier in the primary key.

Demo

- Oracle Financial Software Services Ltd. provides the demo data as sample demo configurations. Customer can change/update/delete this data, this data should not be used for production configurations.

All seed data tables have two Primary Keys - one is user defined codes and the other is a system generated sequence number.

All seed data tables have a system defined indicator to indicate whether a record is system defined.

All seed data are stored in files and checked in the version control systems and sent as merged statements in patch for changed (added or modified) data.

4.12 Developer's Tips

Suppose the account number generation needs to be customized different from what OFSLL generates. Requirement is to replace the baseline format with its own format (like ACC-NNNNNNN).

Locate the procedure that generates the account number.

- Procedure `set_acc_nbr` from program `aaiacc_en_111_01.pkb` generates account number in **YYMMNNNNNNND** format.

Identify the exit point package having the **set_acc_nbr_xxx** procedures where xxx is bfr – before, afr - after and rep – replace.

- `aaiprc_ex_111_01.pks` and `aaiprc_ex_111_01.pkb`
- Create new package with name as `xyzaaiacc_en_111_01.pkb`. Add procedure to create account number in new format.
- Copy “`aaiprc_ex_111_01.pks`” to “`xyzaaiprc_ex_111_01.pks`”
- Modify “`xyzaaiprc_ex_111_01.pks`”, change constant `3CV_SET_ACC_NBR_REP`
`CONSTANT VARCHAR2(30) := cmncon_cl_000_01.NOT_CUSTOMIZED;`

5

Create Custom BI Publisher Report/Letter

The following section details the steps to create new custom BI publisher report/letter.

Pre-Requisites

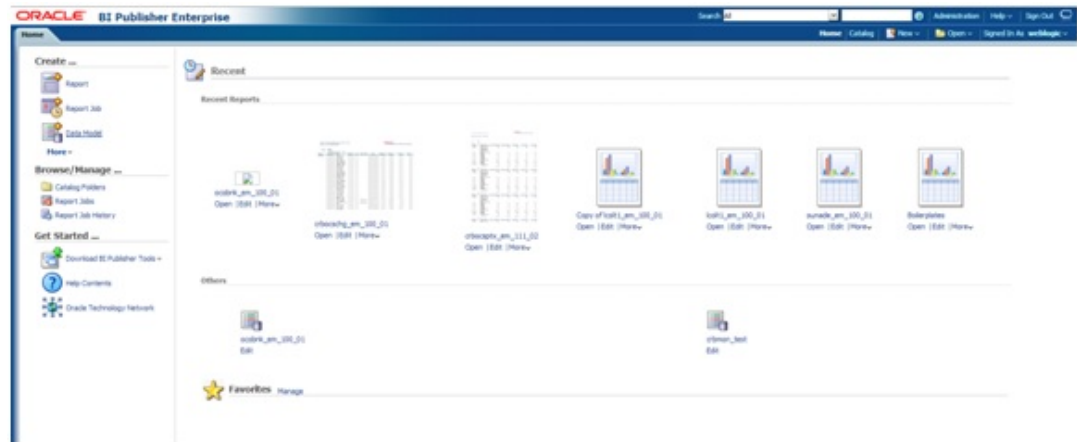
1. Changes to base reports not allowed.
2. Basic knowledge on BIP and BIP client.
3. The reports should be placed into the same folder structure as indicated below:
 - For Reports > Shared Folders/oracle/fll/xmlp/reports
 - For Letters > Shared Folders/oracle/fll/xmlp/letters
 - For Correspondences > Shared Folders/oracle/fll/xmlp/correspondence
- [Creating a New Report](#)
Follow the below steps to create new report.
- [Create Report Layout](#)
Follow the below steps to create report layout.
- [Create XML data](#)
Create XML data to create the layout on local editor (MS Word).
- [Add Dynamics to Report](#)
You can use table wizard to give more dynamic report as indicated.
- [Upload Report in BIP](#)
You can upload the report in BIP as indicated.

5.1 Creating a New Report

Follow the below steps to create new report.

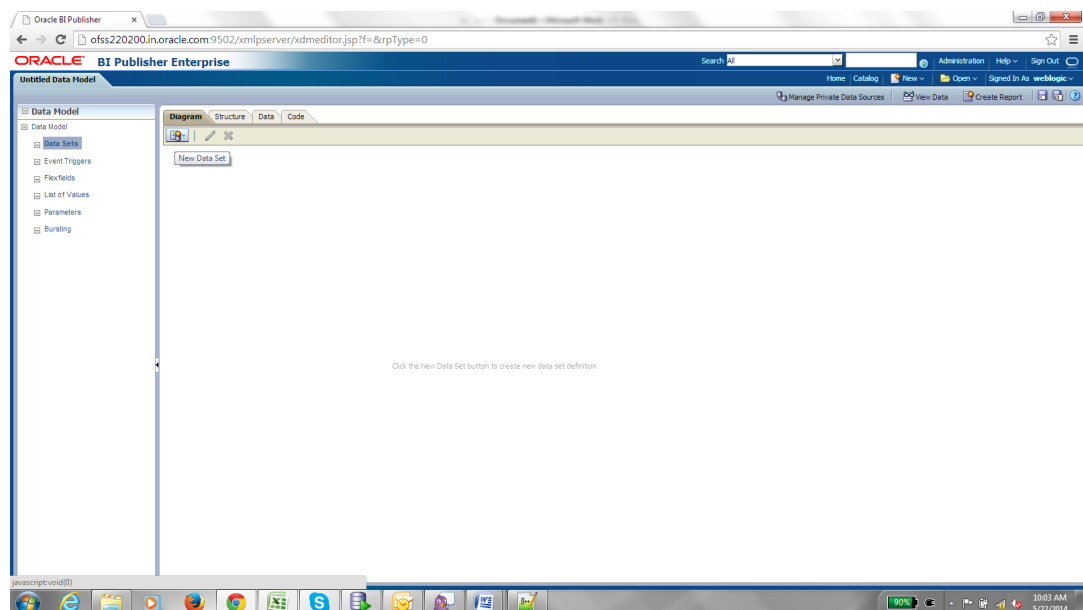
1. Login into BIP console.
2. To Create a new report first create the data model. Click on Data Model on left.

Figure 5-1 Creating a New Report 1



3. On clicking Data Model, it will open the following Screen.

Figure 5-2 Creating a New Report 2



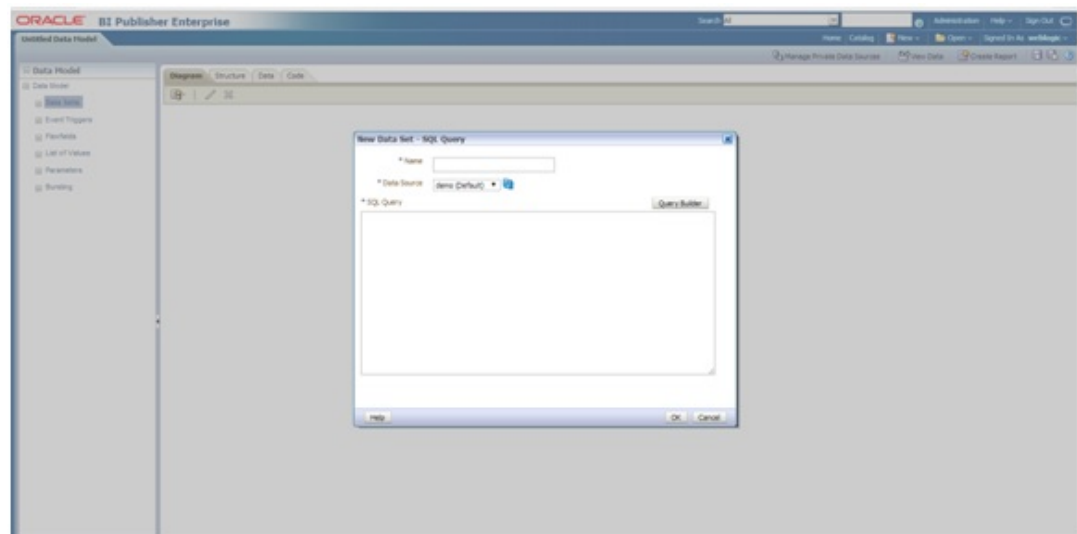
4. To create a New Data Set, Click on **New Data Set** and select the type **SQL Query**.

Figure 5-3 Creating a New Report 3



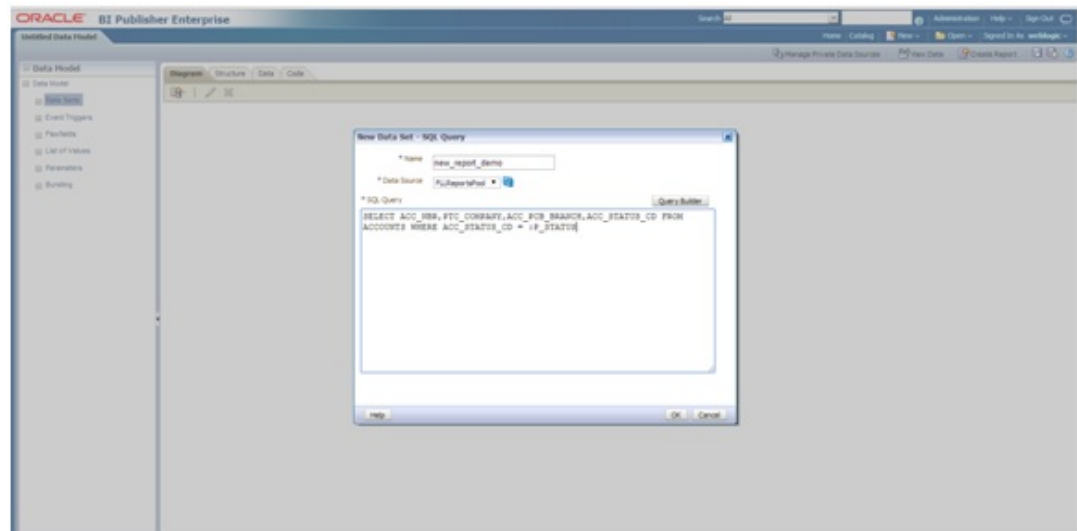
The following screen is displayed.

Figure 5-4 Creating a New Report 4



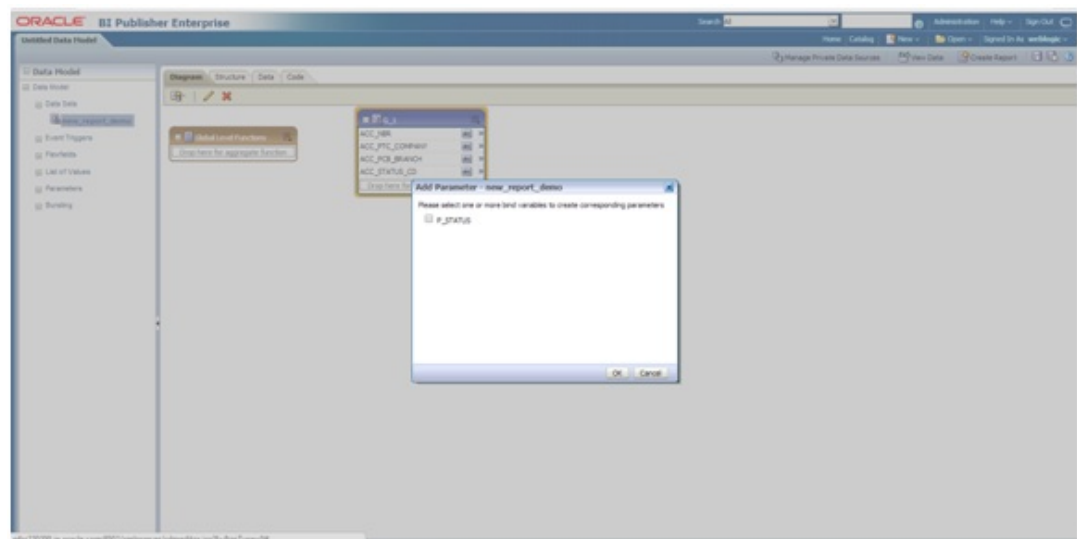
5. Enter the new data model name (Use the same name while creating the report layout).
6. Select the Data Source.
7. Add the sql query. It depends on what data needed on the report and what should be the parameters.

Figure 5-5 Creating a New Report 5



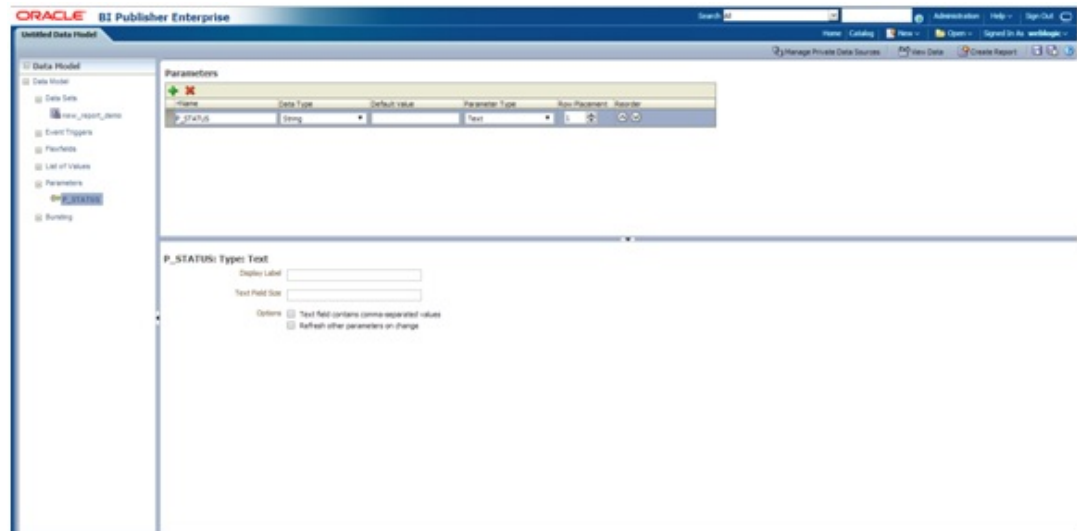
8. Click **OK**. A confirmation dialog is displayed to create the parameter. Select the parameter and press **OK**.

Figure 5-6 Creating a New Report 6



9. The parameter is created as indicated below.

Figure 5-7 Creating a New Report 7



10. Save the data model. Select the directory in which you would need to save the details. For Reports save in reports directory and for Letters save it in Letters directory.
11. You can also save with the same name and specify the description of the report.

Figure 5-8 Creating a New Report 8

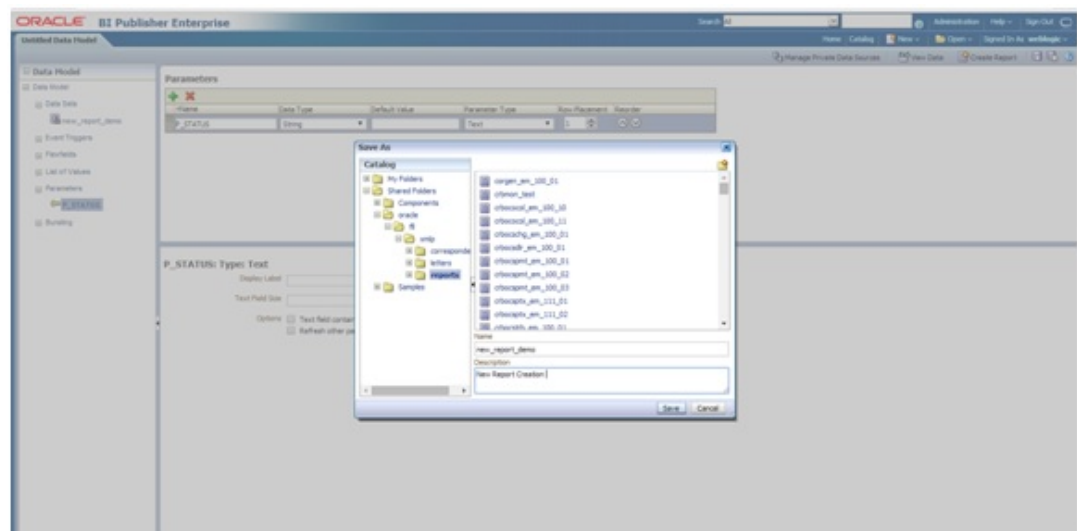
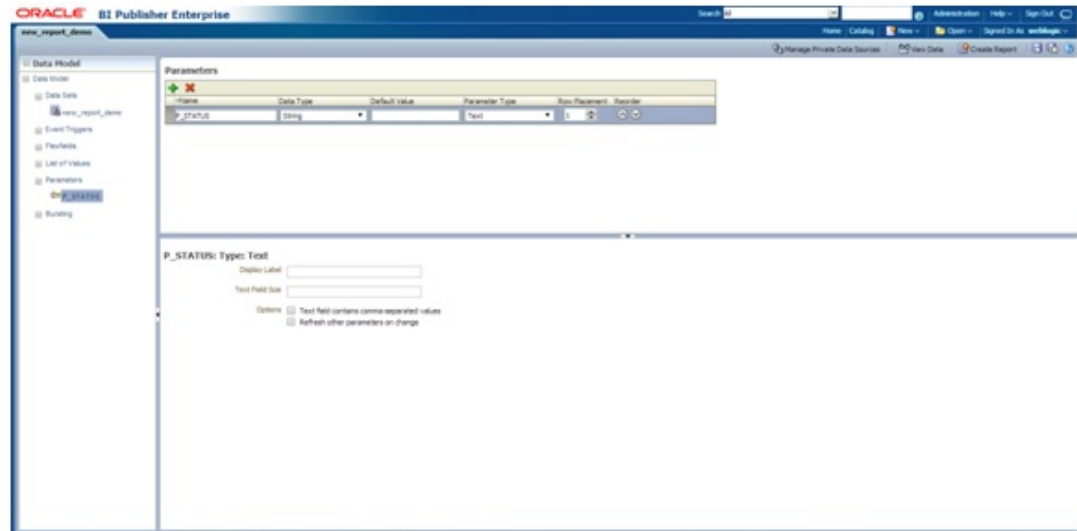


Figure 5-9 Creating a New Report 9



12. Once the Data Model is created, create the sample data as indicated below.
13. Click on **View Data** on right side of the page.

Figure 5-10 Creating a New Report 10

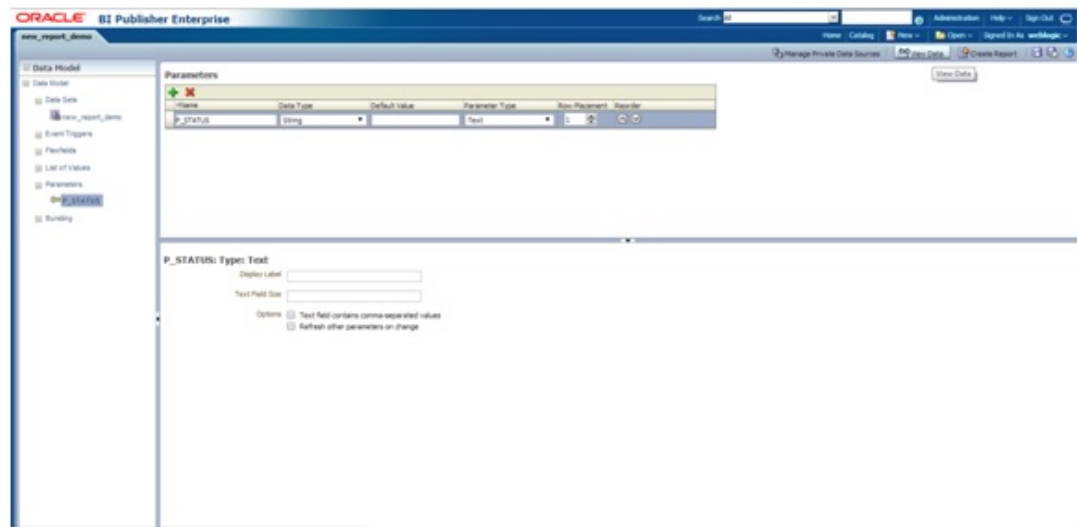
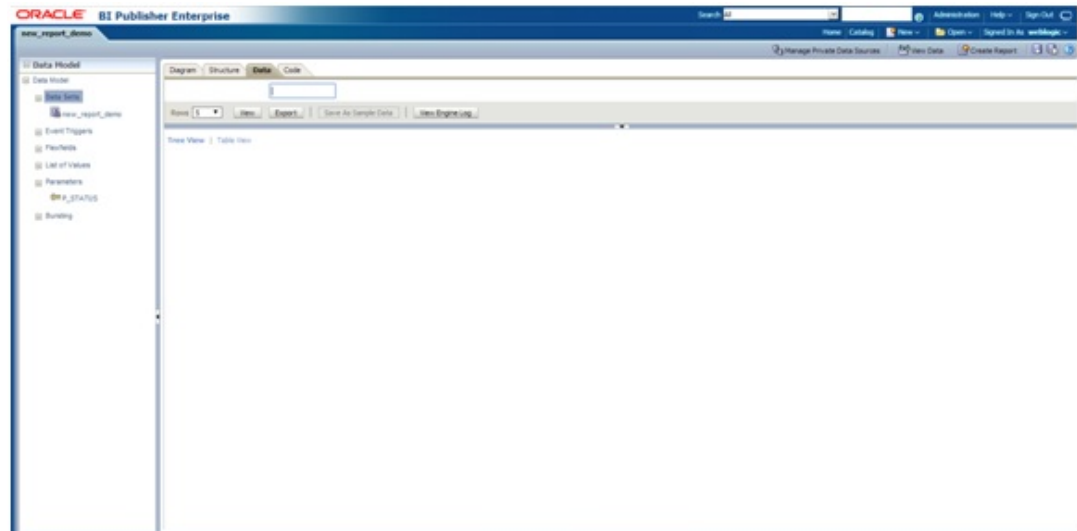
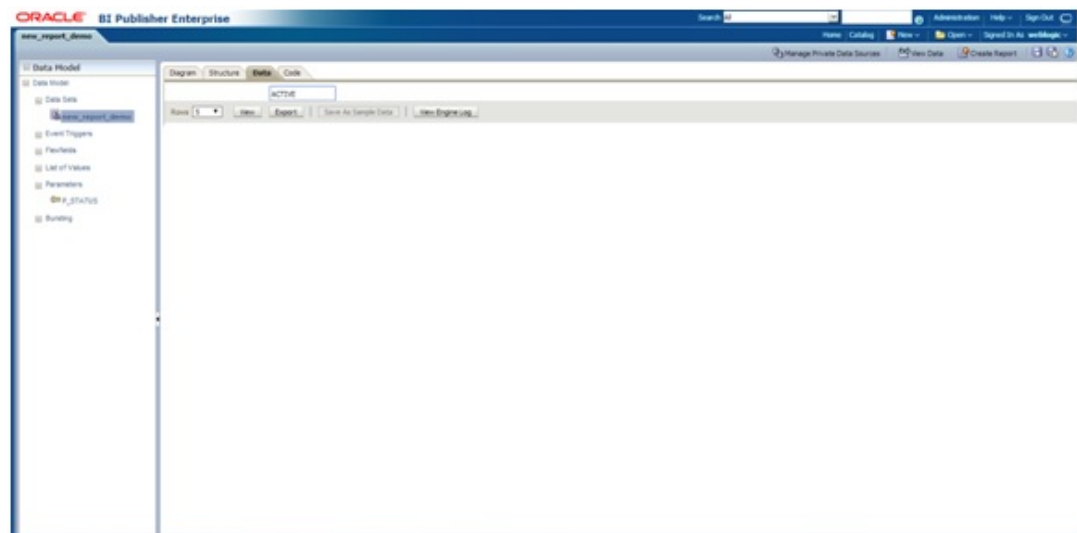


Figure 5-11 Creating a New Report 11



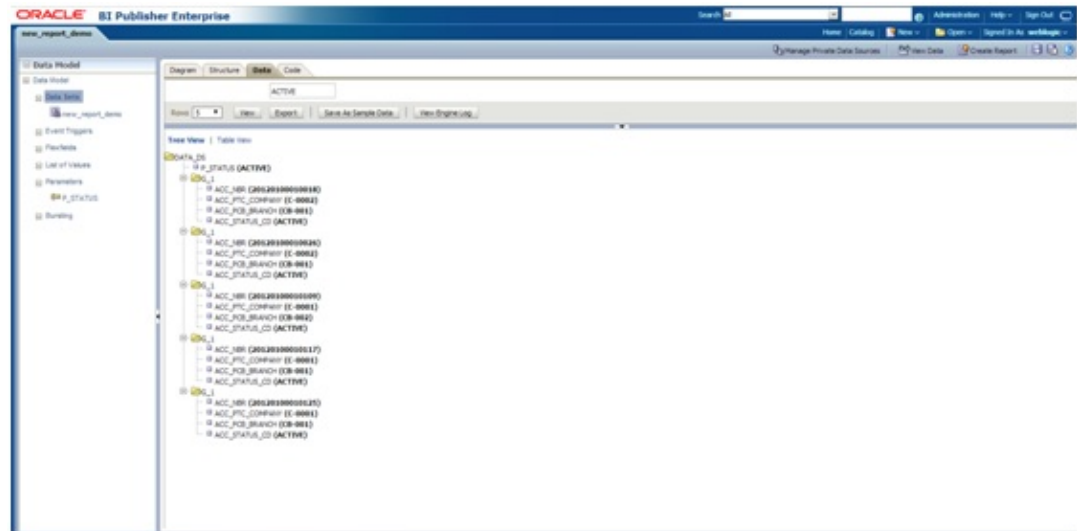
14. Specify the following value in Parameter (P_STATUS).

Figure 5-12 Creating a New Report 12



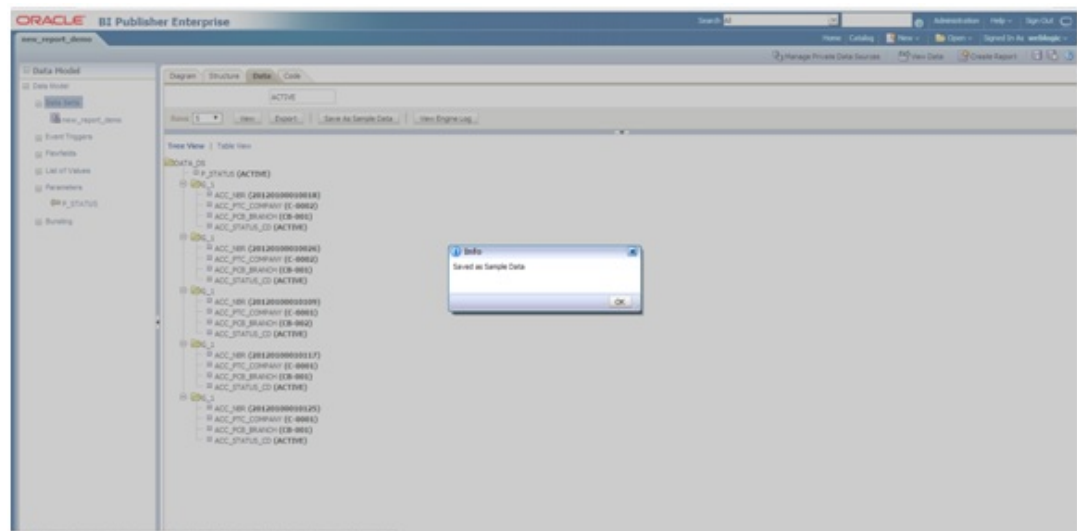
15. Click **View**. The sample data is created as indicated.

Figure 5-13 Creating a New Report 13



16. Click on **Save As Sample Data**.

Figure 5-14 Creating a New Report 14



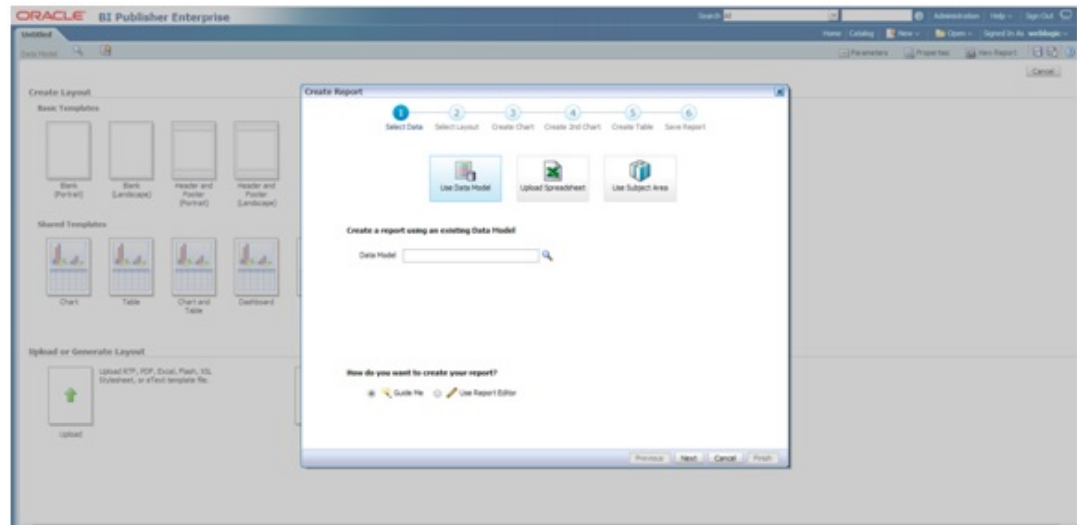
17. The Data model is created and can be use to create the report layout.

5.2 Create Report Layout

Follow the below steps to create report layout.

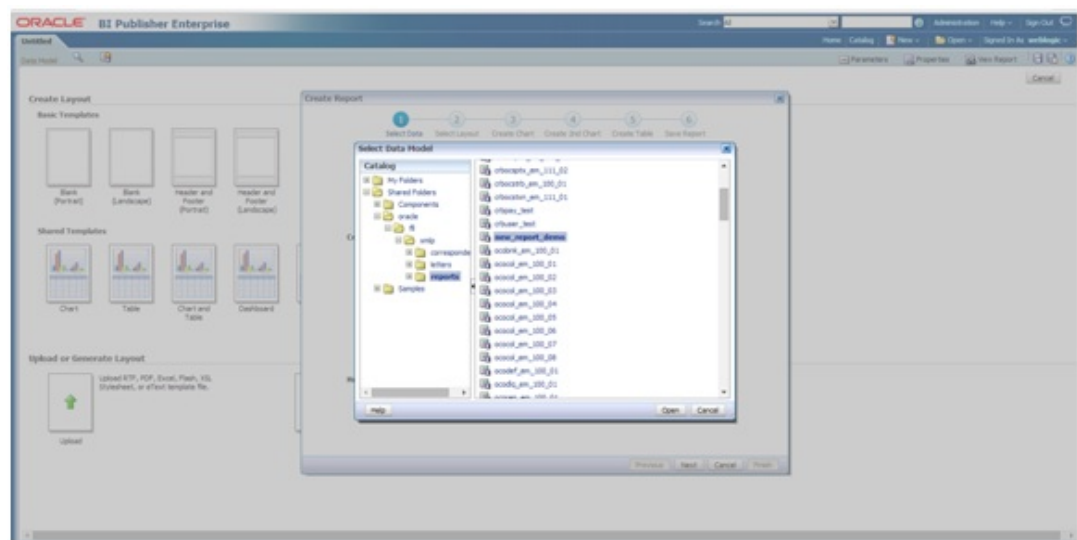
1. Navigate to Home Page and create the report layout as indicated below.
2. Click on Create > Report on LHS panel.

Figure 5-15 Creating Report Layout 1



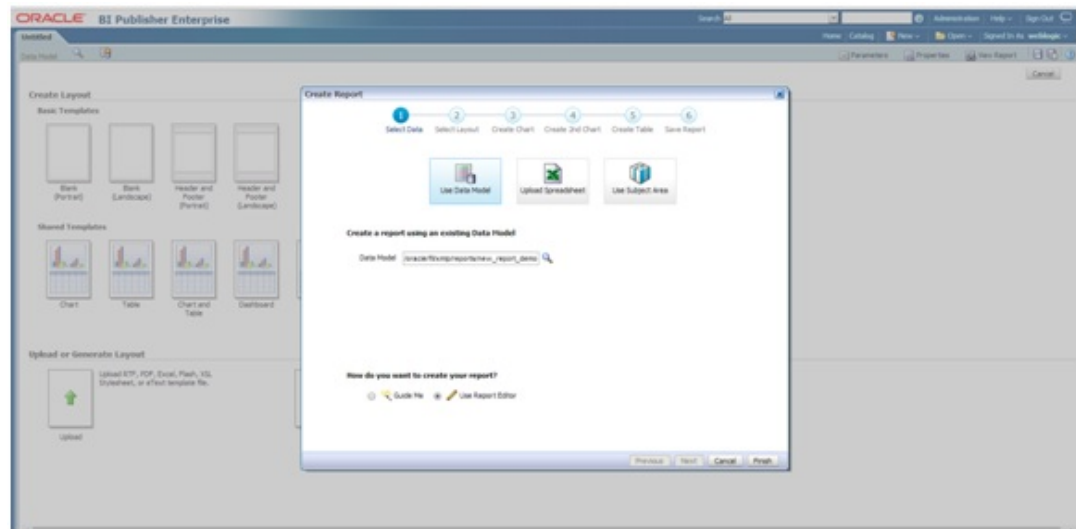
3. Select the **Use Data Model** and click to browse the existing data model , and select the data model just created.

Figure 5-16 Creating Report Layout 2



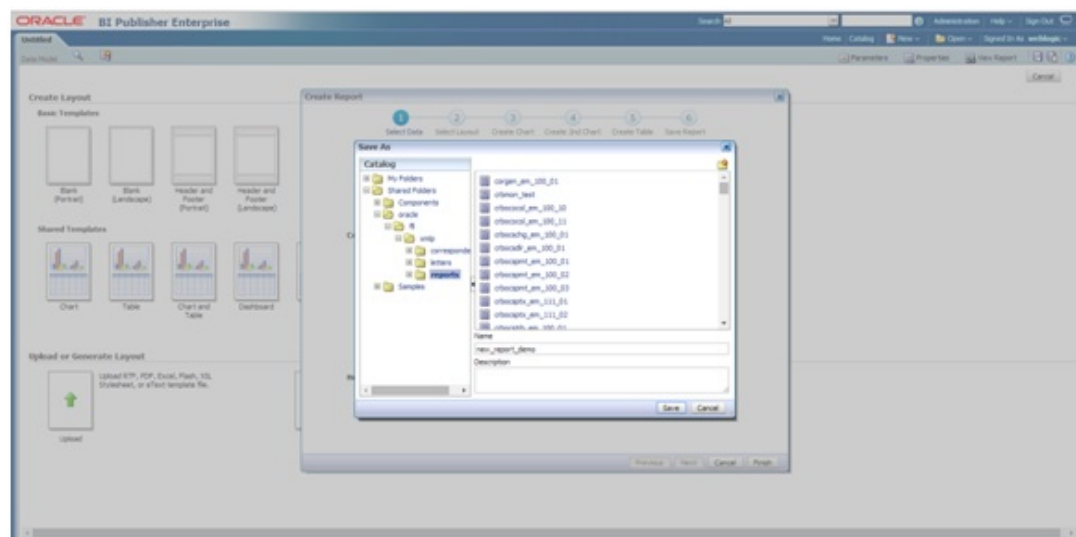
4. Click **Open**.

Figure 5-17 Creating Report Layout 3



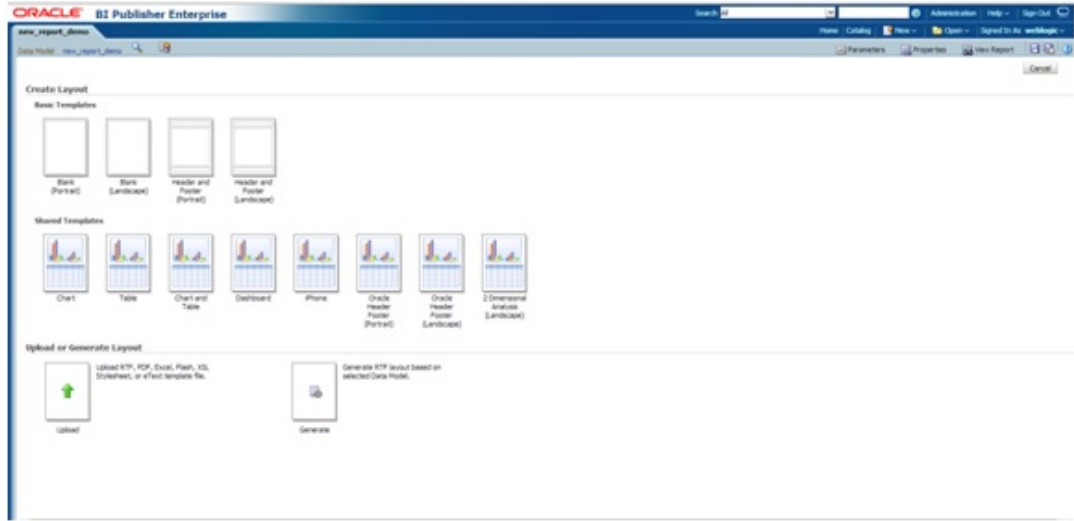
5. Select **Use report Editor** option (If user have the BIP Client installed on his machine , he can create and the .rtf layout from MS Word).
6. Click **Finish** and save the report with the same name as data model.

Figure 5-18 Creating Report Layout 4



The below screen indicates the saved report.

Figure 5-19 Creating Report Layout 5

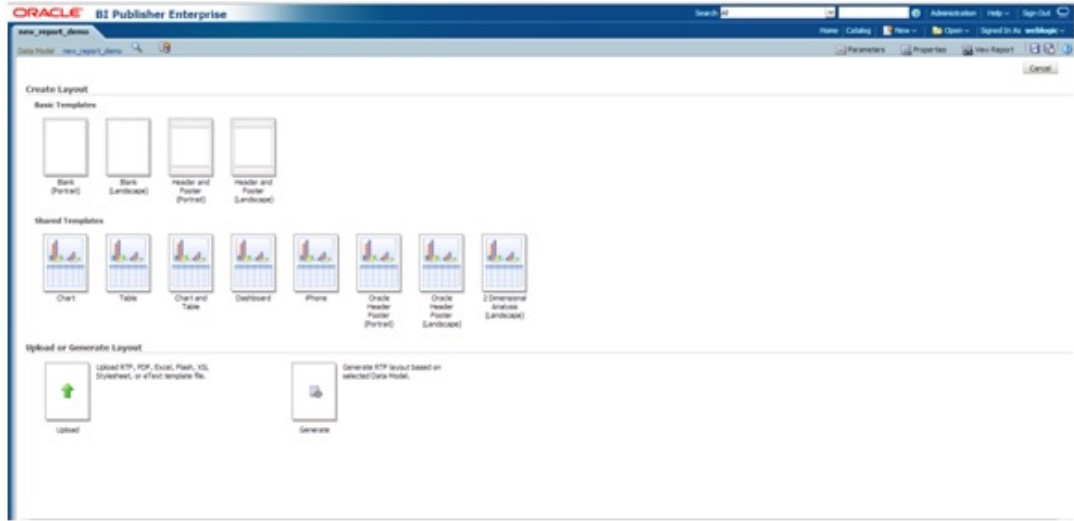


5.3 Create XML data

Create XML data to create the layout on local editor (MS Word).

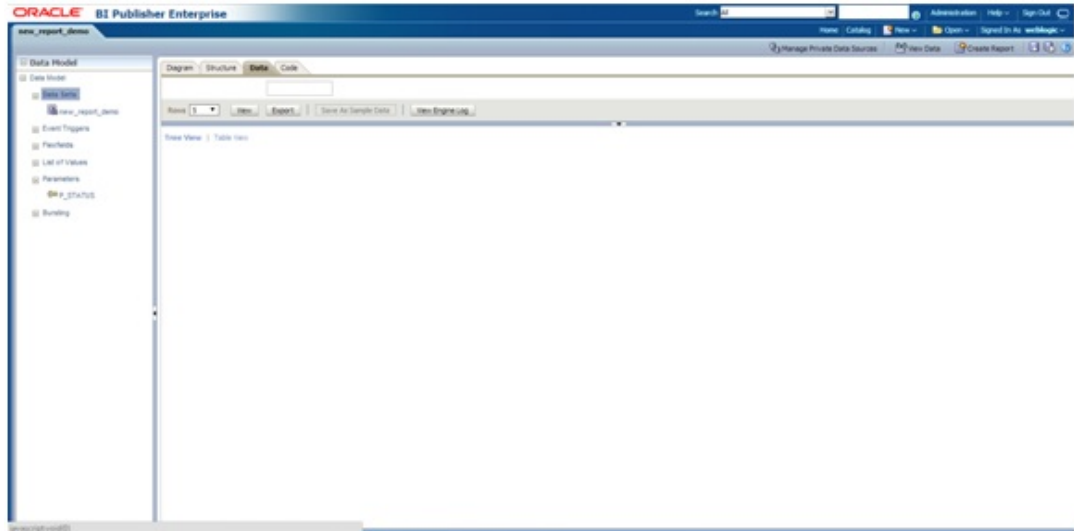
1. Navigate to Home Page and select the data model created.

Figure 5-20 Creating XML data 1



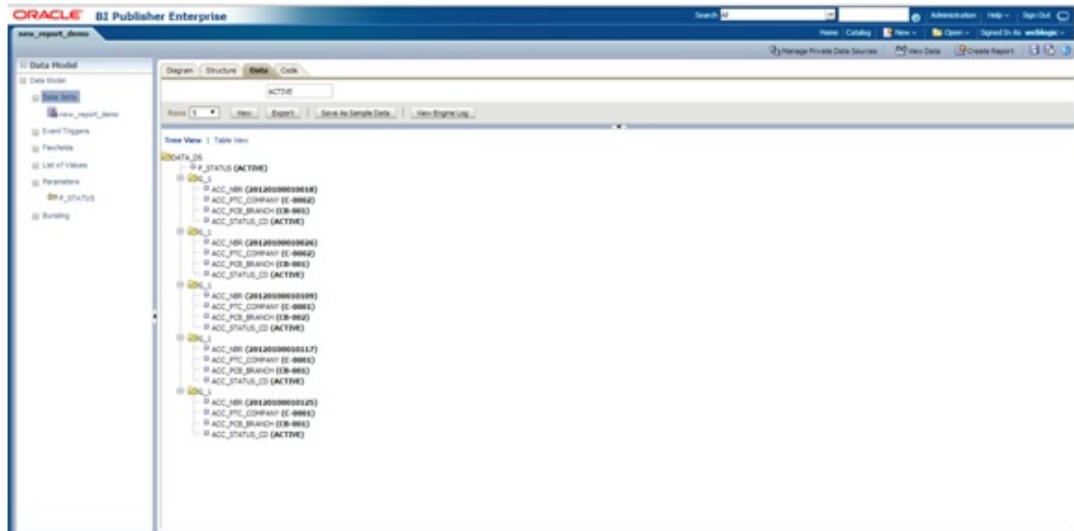
2. Select **Data** Tab.

Figure 5-21 Creating XML data 2



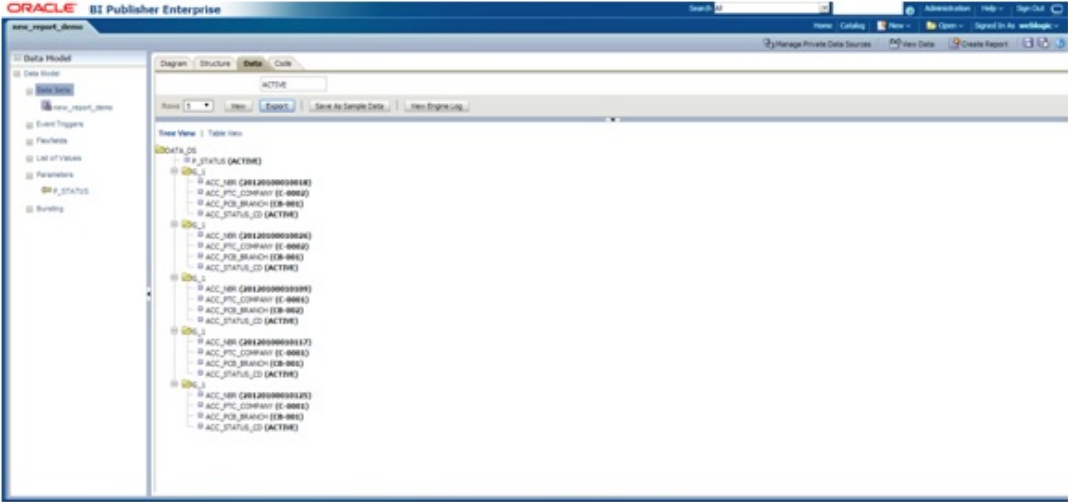
- 3. Enter the Parameter Value and click **View**.

Figure 5-22 Creating XML data 3



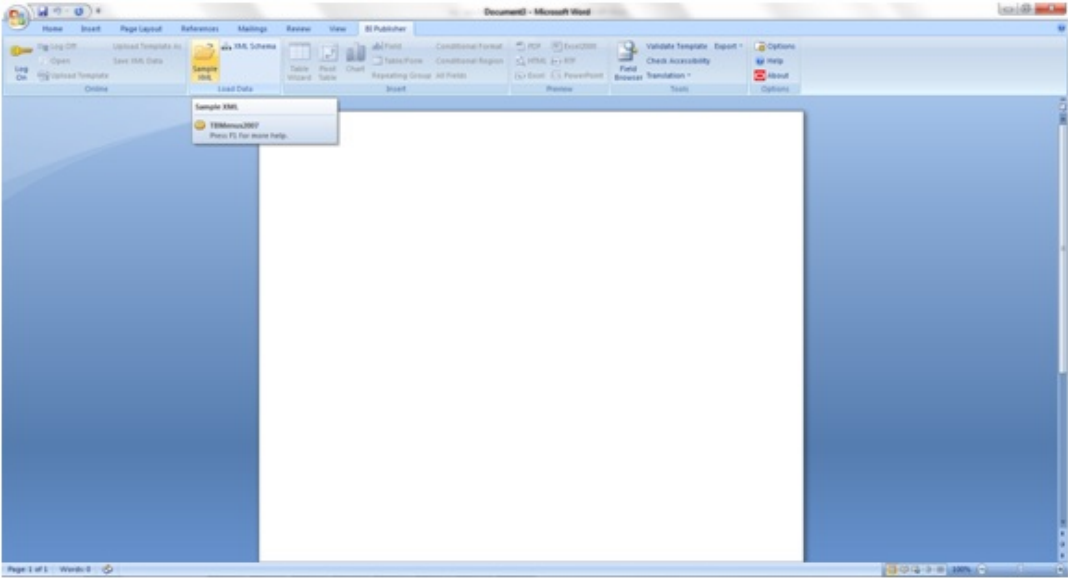
- 4. Click **Export** (it will create the XML file locally in download folder).

Figure 5-23 Creating XML data 4



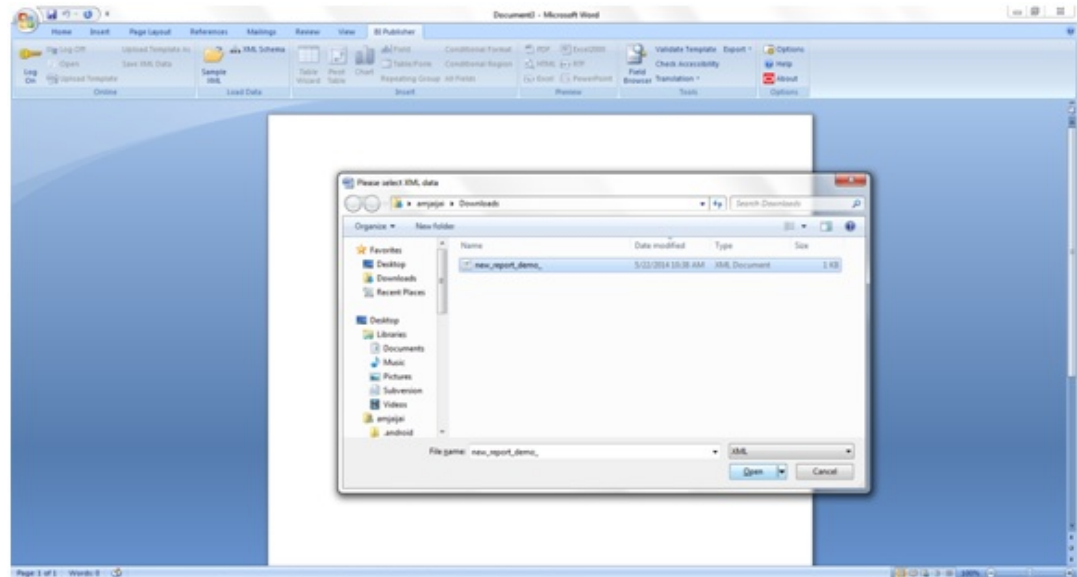
- 5. Open MS Word (Ensure that the BIP client is installed on User's Machine).
- 6. Access BI Publisher tab and click **Sample data**.

Figure 5-24 Creating XML data 5



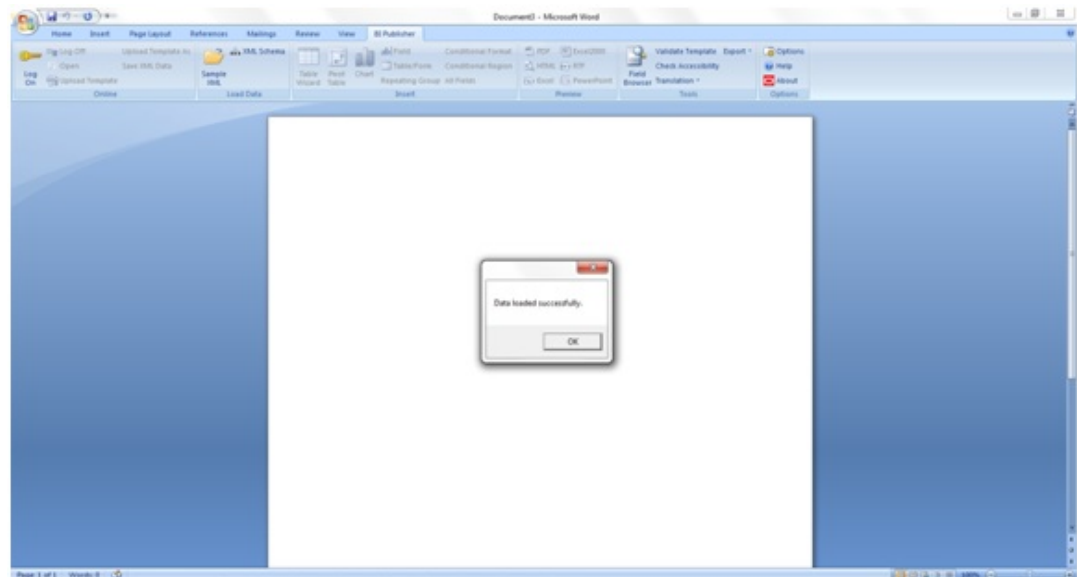
- 7. Select the same XML file from download.

Figure 5-25 Creating XML data 6



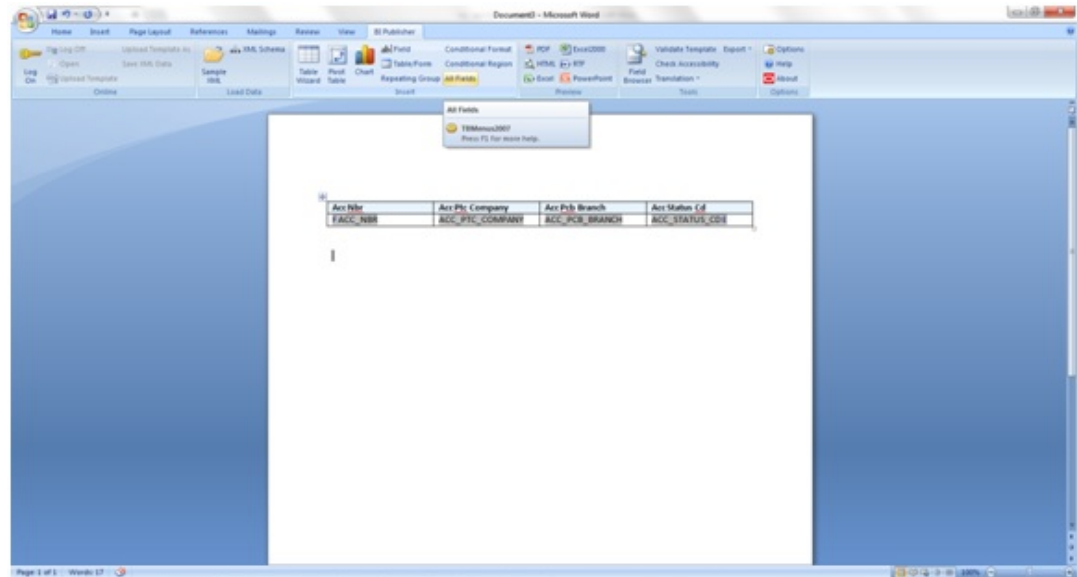
8. Select **Open**. Required data is loaded.

Figure 5-26 Creating XML data 7



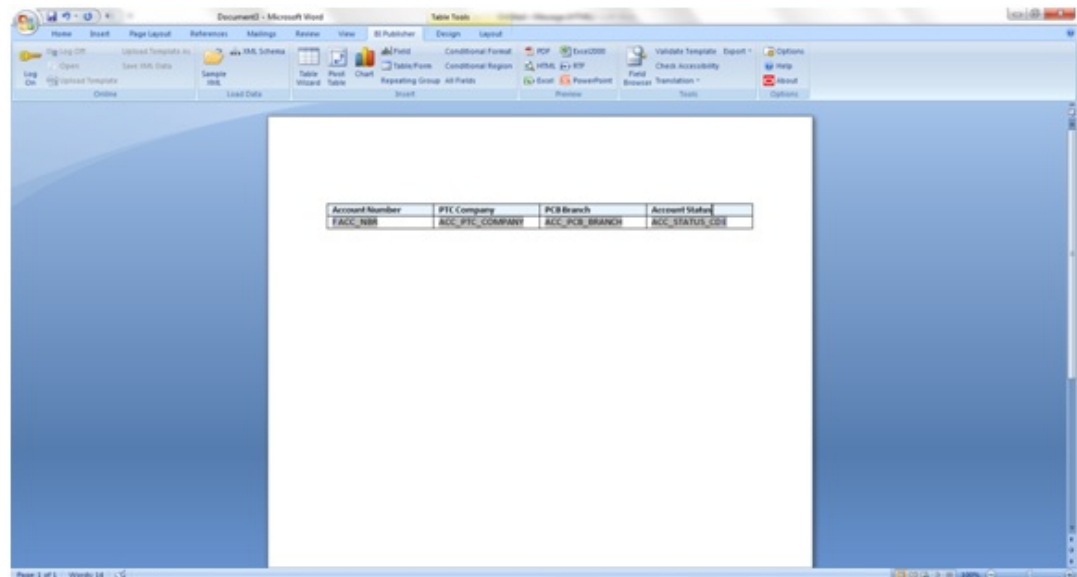
9. You can use the available option is MS word to create the desired layout.
10. First use **All Fields** option. This creates all the fields from sample xml and create the layout as follows in tabular form.

Figure 5-27 Creating XML data 8



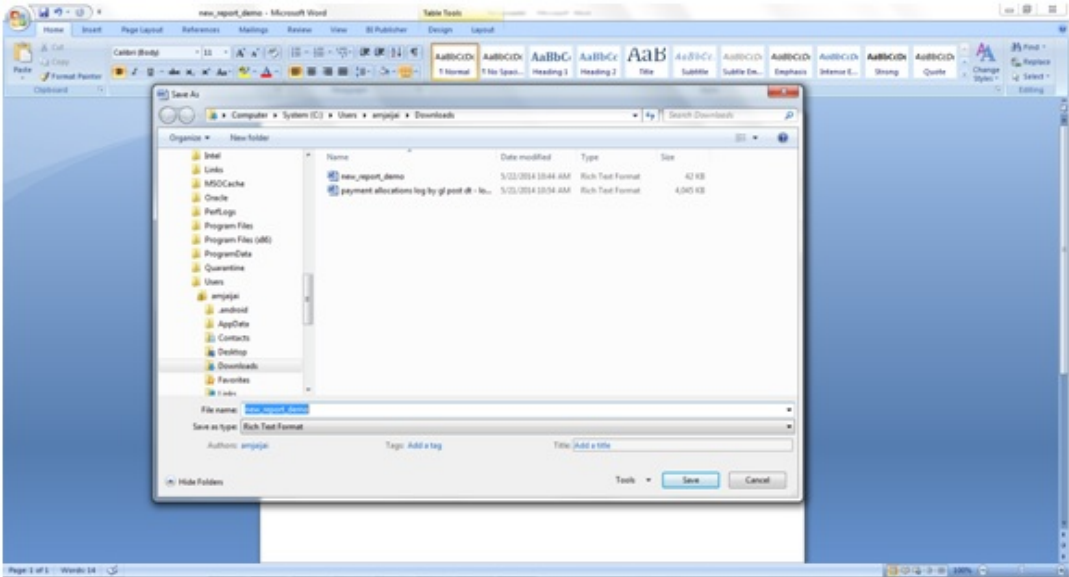
11. You can modify the headings, add new static text, logos and so on.

Figure 5-28 Creating XML data 9



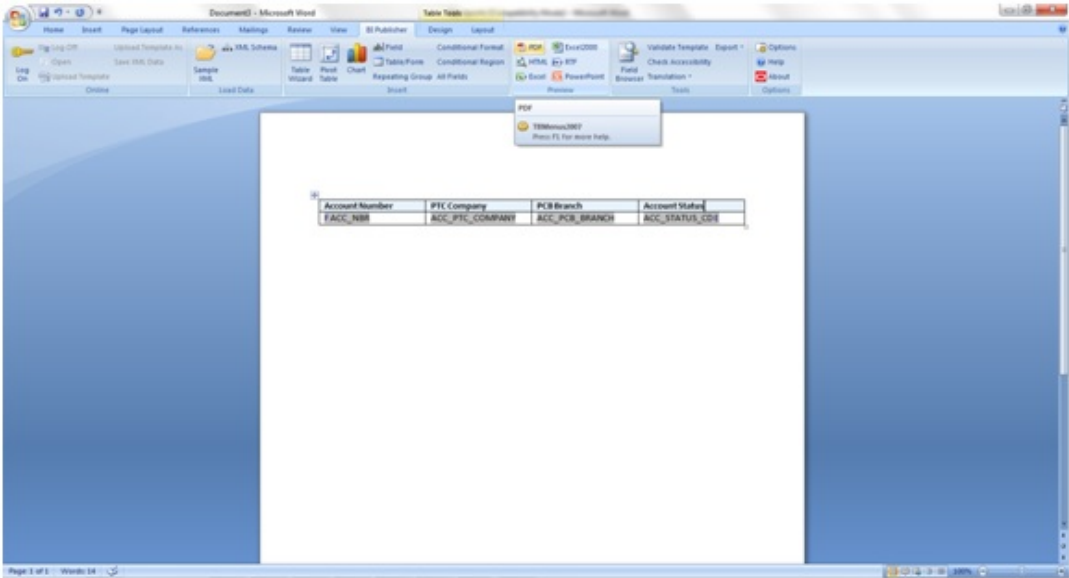
12. Save the layout with the same name as the data model and report in BIP.

Figure 5-29 Creating XML data 10



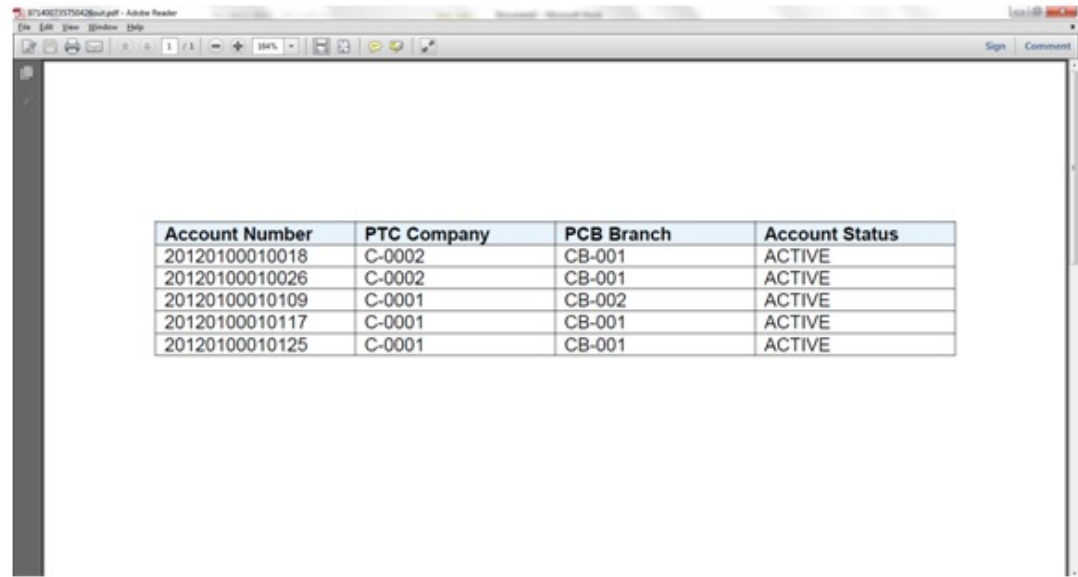
- 13. To preview the layout, click **PDF** in preview tab.

Figure 5-30 Creating XML data 11



- 14. The report is displayed in PDF format.

Figure 5-31 Creating XML data 12



The screenshot shows a window titled 'ST14021070426out.pdf - Adobe Reader'. The window displays a table with the following data:

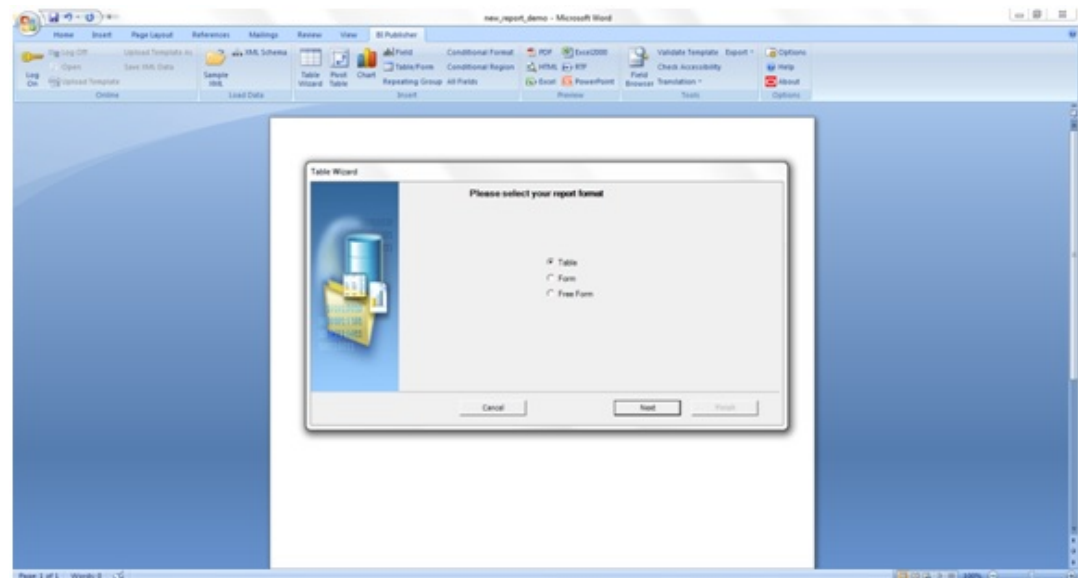
Account Number	PTC Company	PCB Branch	Account Status
20120100010018	C-0002	CB-001	ACTIVE
20120100010026	C-0002	CB-001	ACTIVE
20120100010109	C-0001	CB-002	ACTIVE
20120100010117	C-0001	CB-001	ACTIVE
20120100010125	C-0001	CB-001	ACTIVE

5.4 Add Dynamics to Report

You can use table wizard to give more dynamic report as indicated.

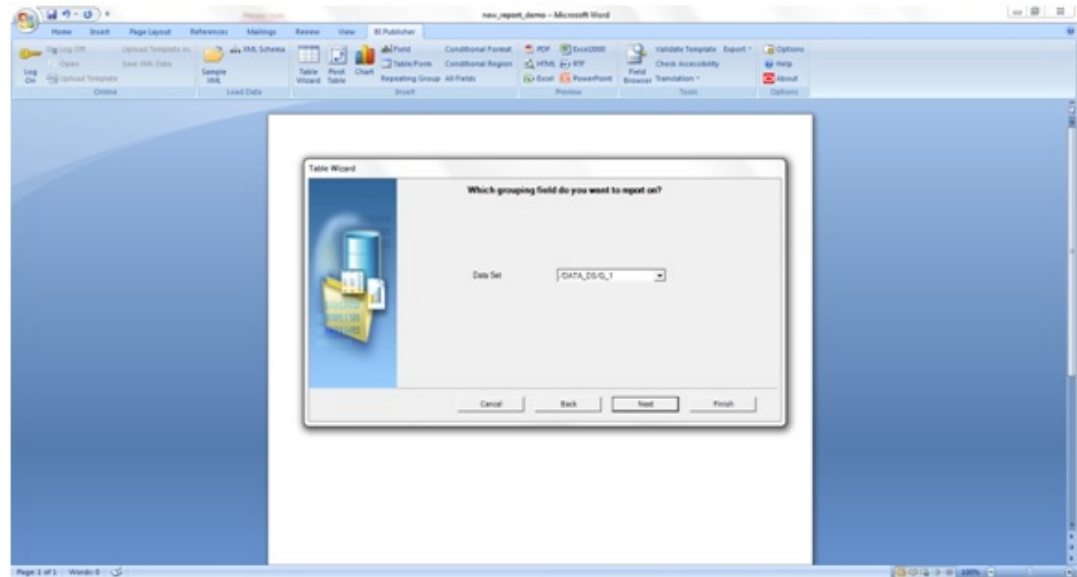
1. Select the Table Wizard from BI Publisher tab.

Figure 5-32 Adding Dynamics to Report 1



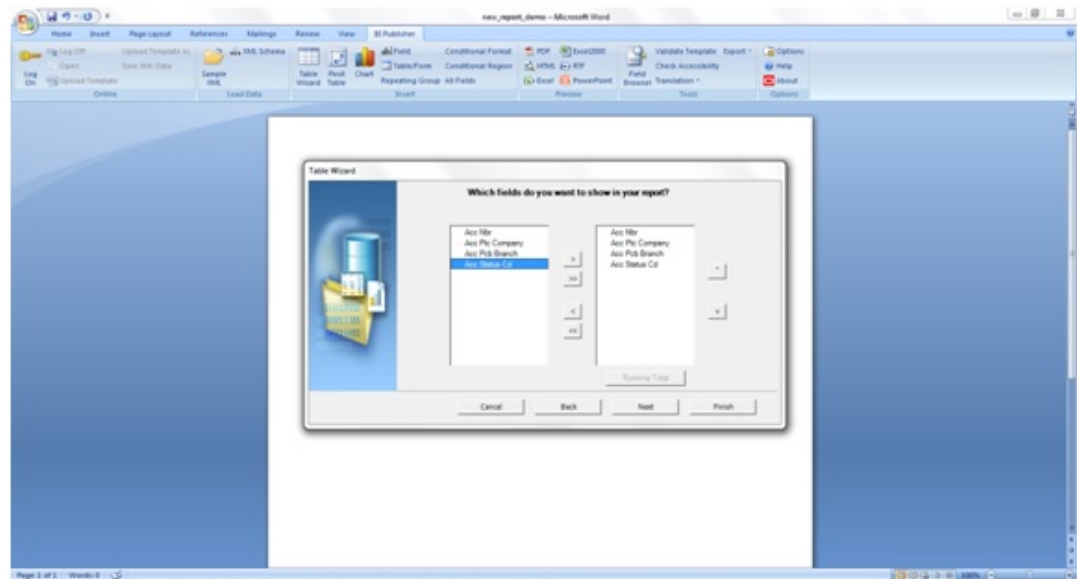
2. Select the Table and press **Next**.

Figure 5-33 Adding Dynamics to Report 2



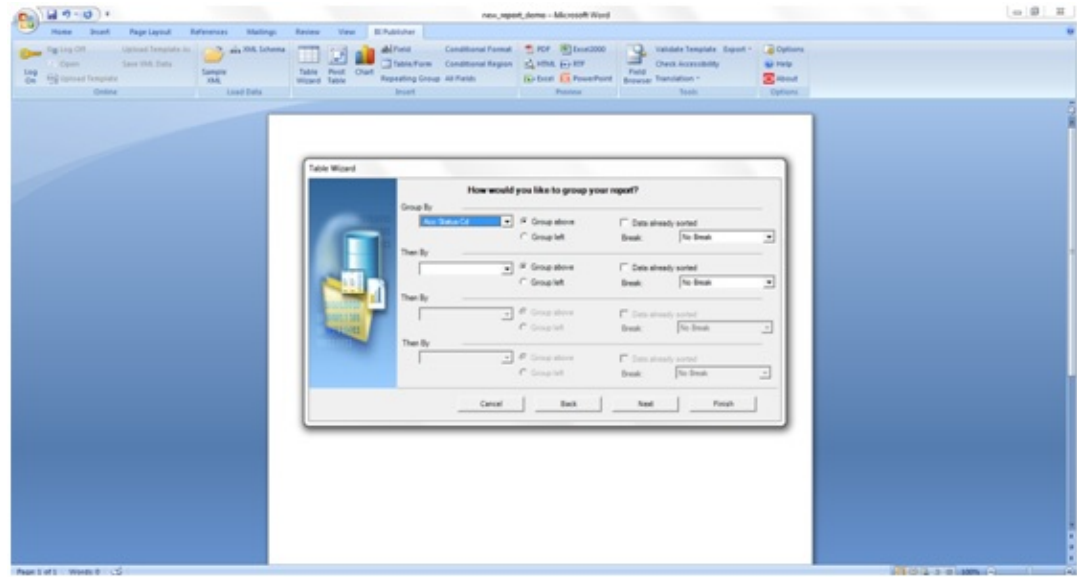
3. Use the same Data SET.

Figure 5-34 Adding Dynamics to Report 3



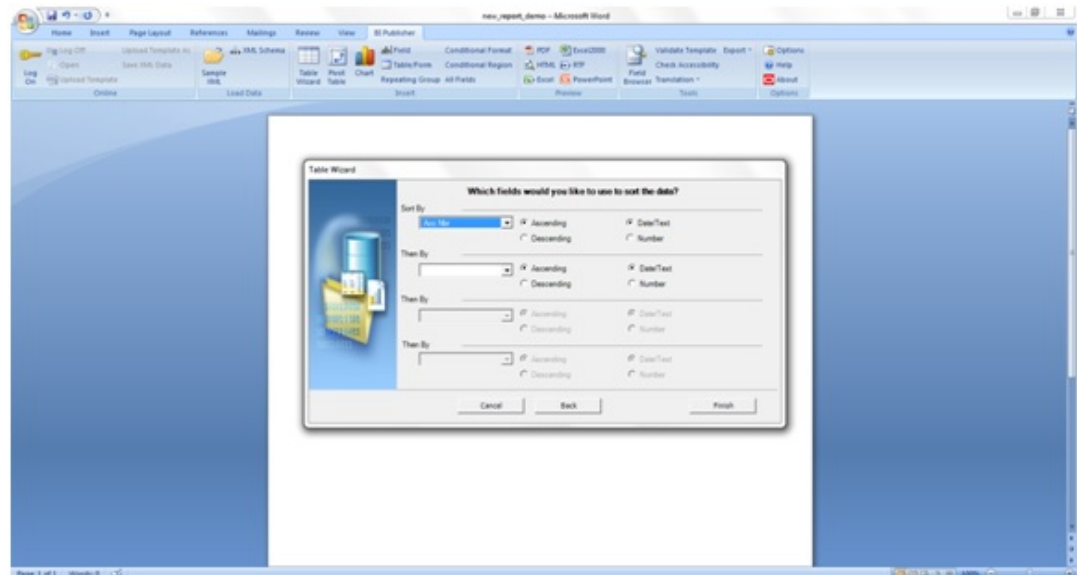
4. Select all the fields and press **Next**.
5. You can create the group as follows:

Figure 5-35 Adding Dynamics to Report 4



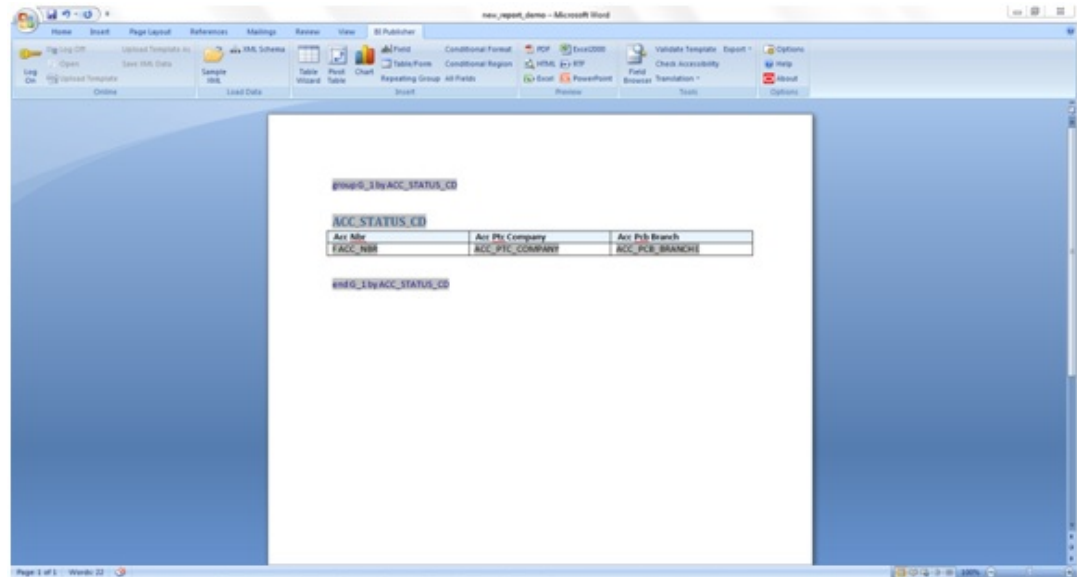
6. Click **Next**. You can give the sorting option as indicated.

Figure 5-36 Adding Dynamics to Report 5



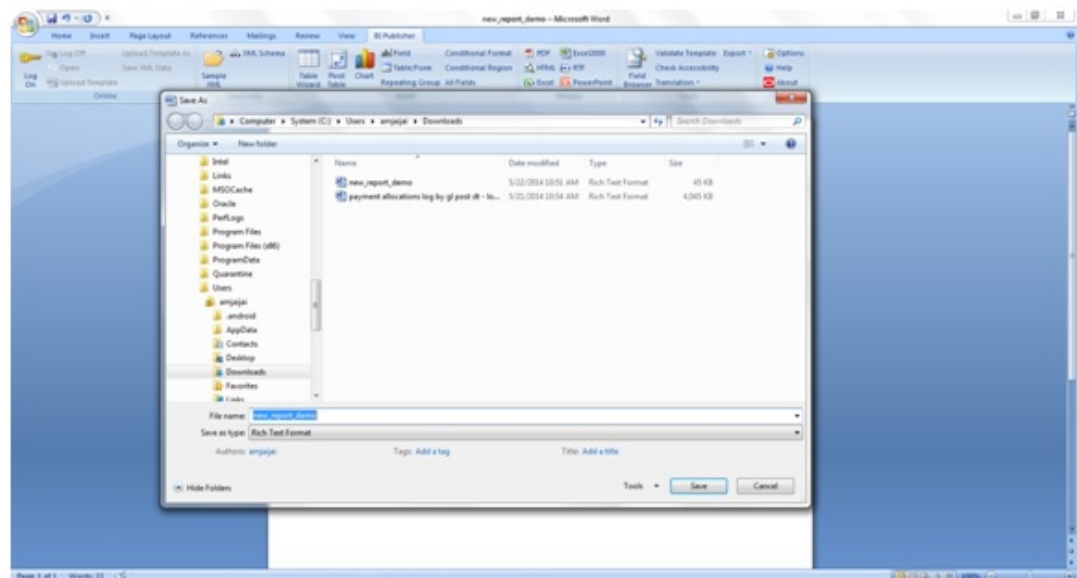
7. Click **Next**. The wizard creates the layout as indicated.

Figure 5-37 Adding Dynamics to Report 6



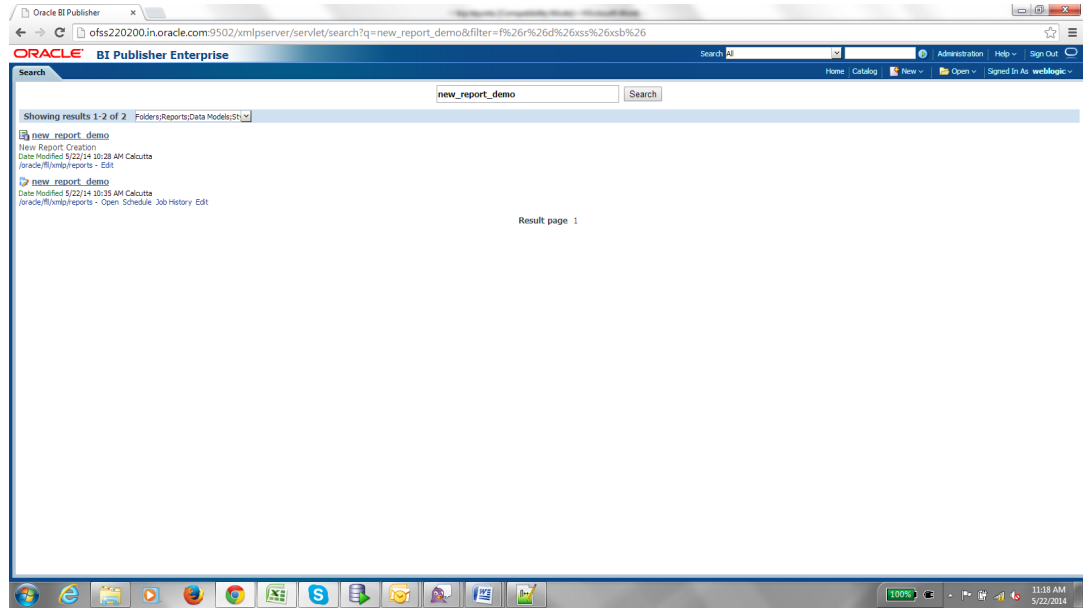
8. You can add, modify static Text, logo in the layout and Save the layout with the same name of datamodel or report.

Figure 5-38 Adding Dynamics to Report 7



9. Preview of the report layout is as indicated.

Figure 5-41 Uploading Report in BIP 2



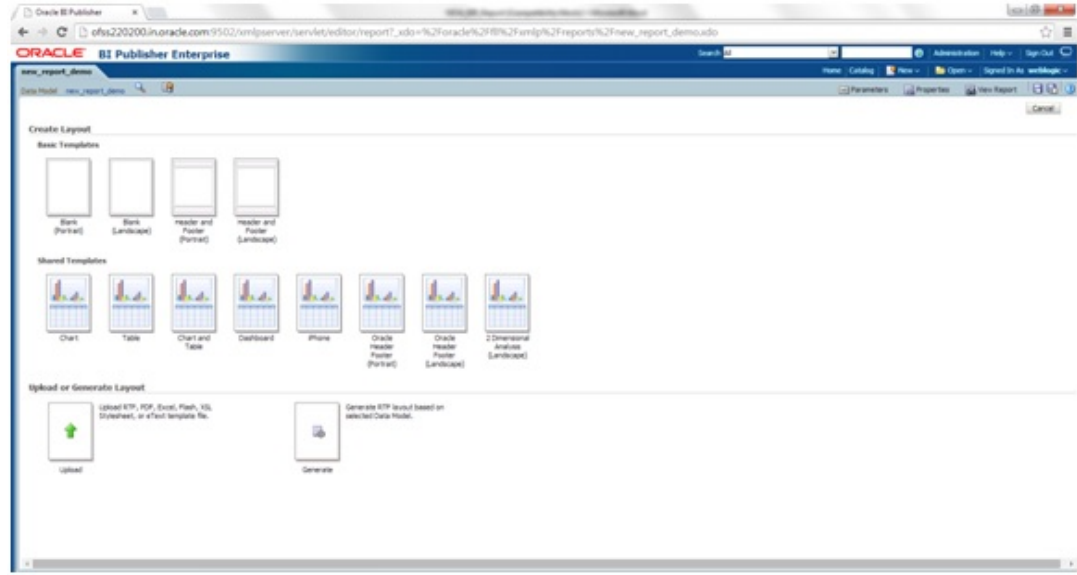
3. Click **Edit** on New report layout.

Figure 5-42 Uploading Report in BIP 3



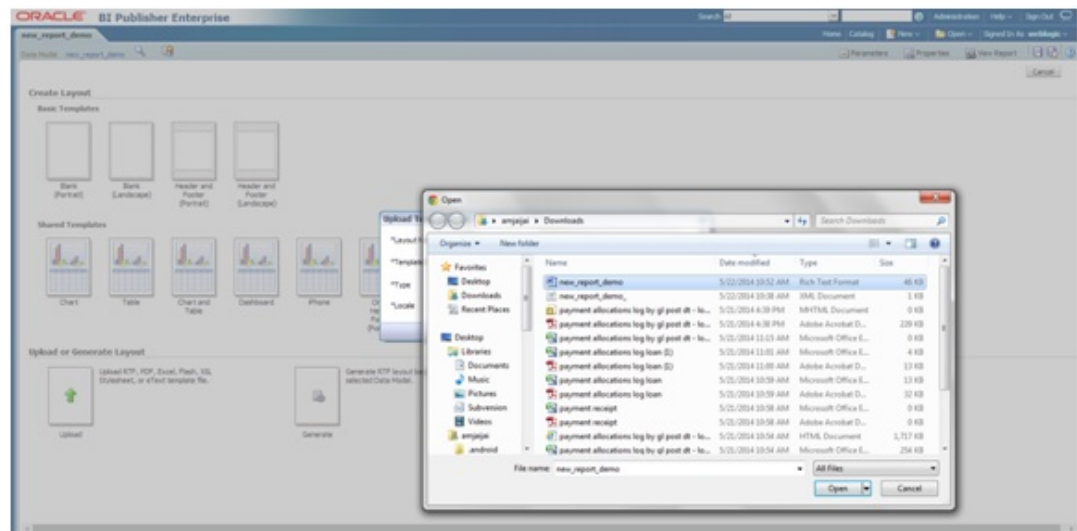
4. The following screen is displayed and allows you to upload the layout.

Figure 5-43 Uploading Report in BIP 4



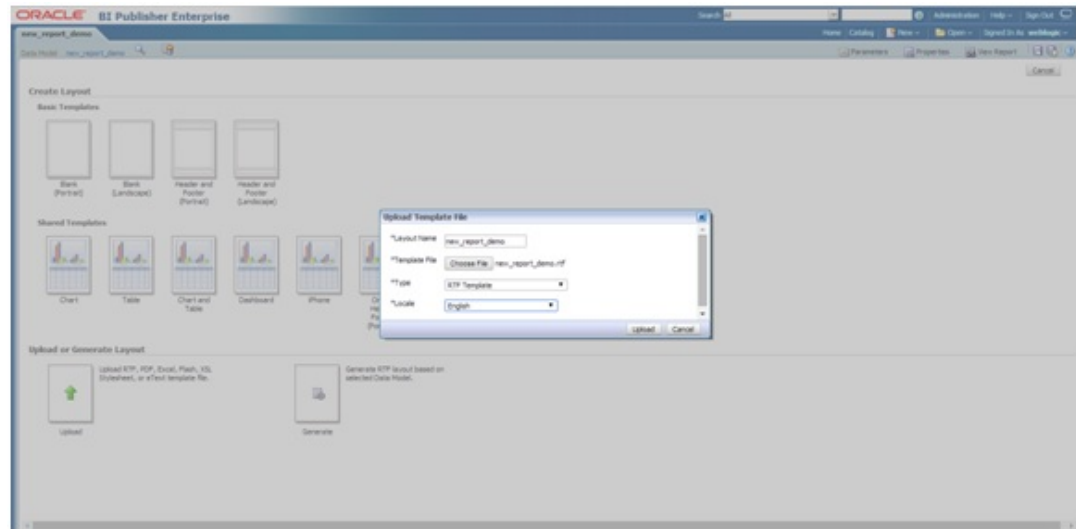
5. Click **Upload**.
6. Specify the same layout name (`new_report_demo`) and choose the file.

Figure 5-44 Uploading Report in BIP 5



7. Select the type as **RTF Template** and locale as **English**.

Figure 5-45 Uploading Report in BIP 6



8. Click **Upload**.

Figure 5-46 Uploading Report in BIP 7



9. Click **Save**. The report is created successfully.
 - Check the properties for caching and View a List for Output Format.
 - Click on View Report to verify the report.
 - Enter the parameter Value and press **Apply**.

Figure 5-47 Uploading Report in BIP 8

The screenshot shows the Oracle BI Publisher Enterprise interface. The main content area displays a table titled "Account Status | ACTIVE". The table has three columns: "Account Number", "FTC Company", and "FCB Branch". The data is as follows:

Account Number	FTC Company	FCB Branch
2012010010018	C-0002	CB-001
2012010010026	C-0002	CB-001
2012010010109	C-0001	CB-002
2012010010117	C-0001	CB-001
2012010010125	C-0001	CB-001
2012010010133	C-0001	CB-002
2012010010141	C-0002	CB-001
2012010010172	C-0001	CB-002
2012010010190	C-0001	CB-001
2012010010240	C-0001	CB-001
2012010010258	C-0001	CB-001
2012010010266	C-0001	CB-002
2012020010421	C-0001	CB-002
2012020010588	C-0002	CB-002
2012020010596	C-0001	CB-002
2012090011836	C-0002	CB-001
2012090011844	C-0002	CB-001
2012090011852	C-0002	CB-001
2012090011860	C-0002	CB-001
2012090011875	C-0002	CB-001
2012090011883	C-0002	CB-001
2012090011918	C-0002	CB-001
2012090011934	C-0002	CB-001
2012100011977	C-0002	CB-001
2012100011985	C-0002	CB-001
2012100011999	C-0002	CB-001
2012100012004	C-0002	CB-001
2011120050044	C-0001	CB-001
2011120050052	C-0001	CB-001
2011120050060	C-0001	CB-001
2011120050068	C-0001	CB-001
2011120050094	C-0001	CB-001
2011120050101	C-0001	CB-001
2011120050107	C-0001	CB-001

Note:

These are steps to create a new Report /Letter and then user can setup report/ letter accordingly from OFSLL. Please see section 7 and 8 for the Steps to setup report and letter from UI (OFSLL).

6

Customize Existing Base BIP Reports

To customize a report, follow the steps given below:

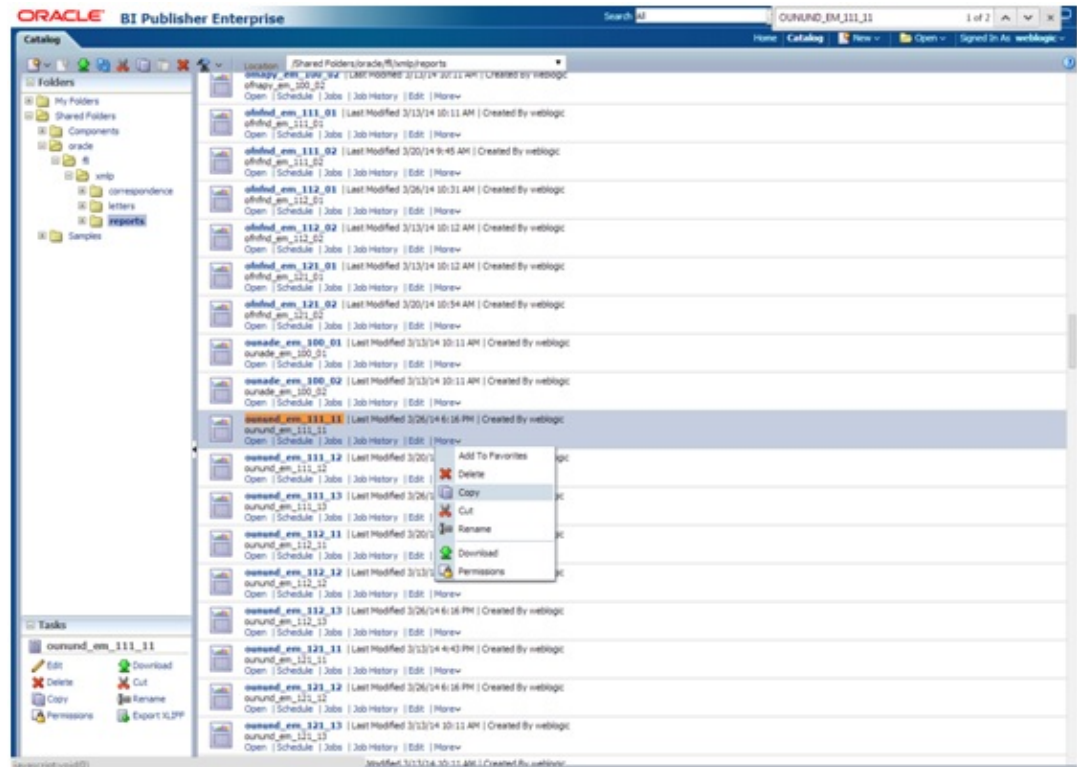
- Do not modify the OFSLL Base Report.
- Create a copy of the OFSLL Base Report, rename and modify that report. A new report also can be created. Name the report as xyz_<report_name>.
- Register the new report and its parameters in OFSLL using reports setup.
- [Customizing Existing Base BIP Reports](#)

6.1 Customizing Existing Base BIP Reports

Pre-Requisites

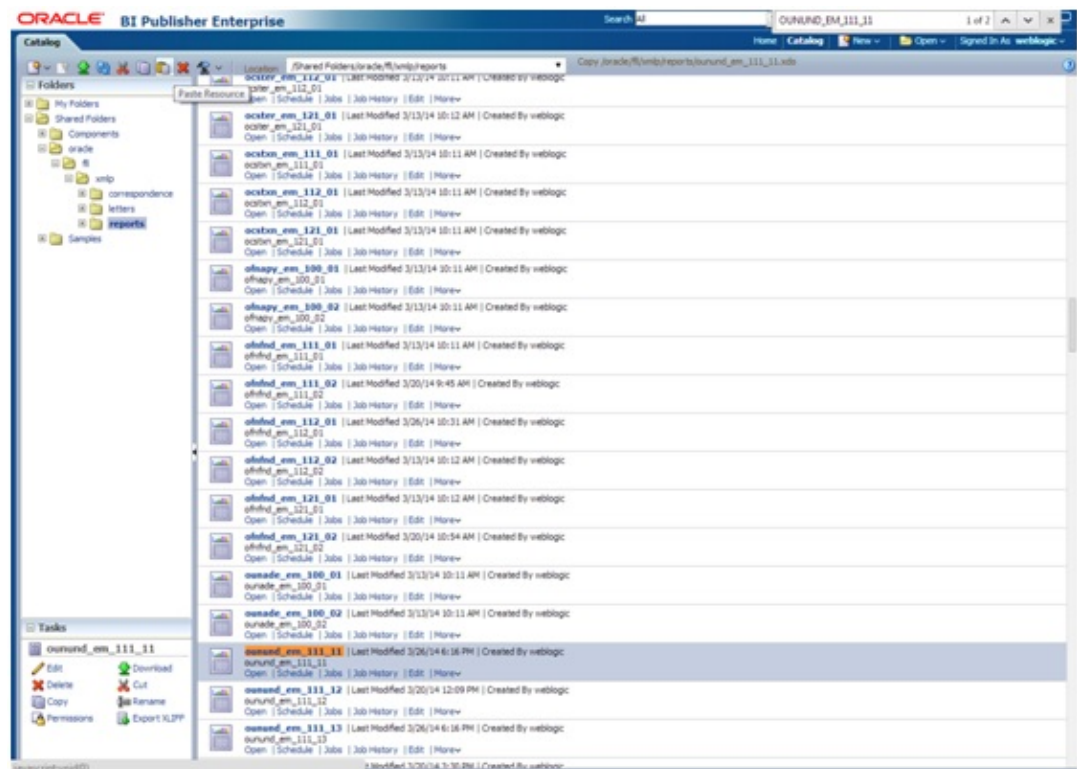
1. Please make sure that we should not change any base reports.
2. The reports should be placed into the same folder structure.
 - i.e. For Reports → Shared Folders/oracle/fll/xmlp/reports
 - For Letters → Shared Folders/oracle/fll/xmlp/letters
 - For Correspondences → Shared Folders/oracle/fll/xmlp/correspondence
3. Consider a base report OUNUND_EM_111_11 (UNDERWRITING STATUS BY MONTH AND PRODUCER LOAN).
4. Let us assume we will do some customizations on the base report and create a new report called XYZOUNUND_EM_111_11 (Here XYZ is bank code).
5. Search for the base report and press More → Copy as shown in the image below.

Figure 6-1 Customizing Base Reports 1



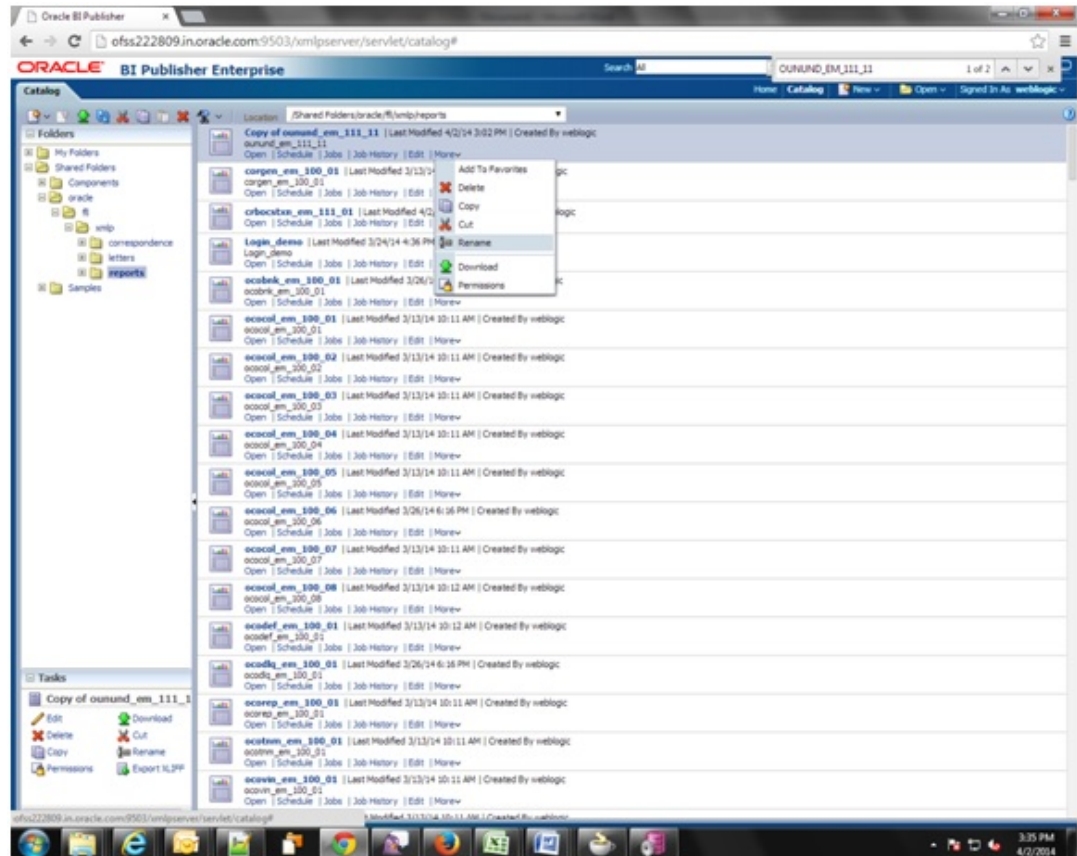
- After pressing copy go to the folder where you want to paste the new report and press the **Paste Resource** button as shown in the image below.

Figure 6-2 Customizing Base Reports 2



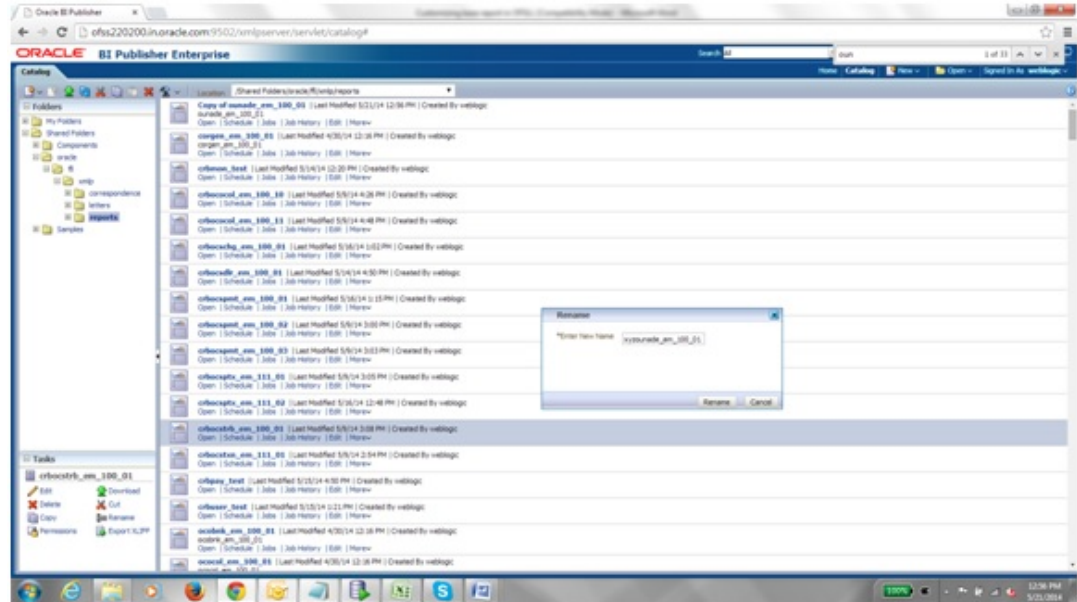
7. On pressing Paste Resource button, a new report will be created in the directory with name as Copy of ounund_em_111_11
8. Select the particular report (Copy of ounund_em_111_11) and press More ▾ Rename as shown in the image below.

Figure 6-3 Customizing Base Reports 3



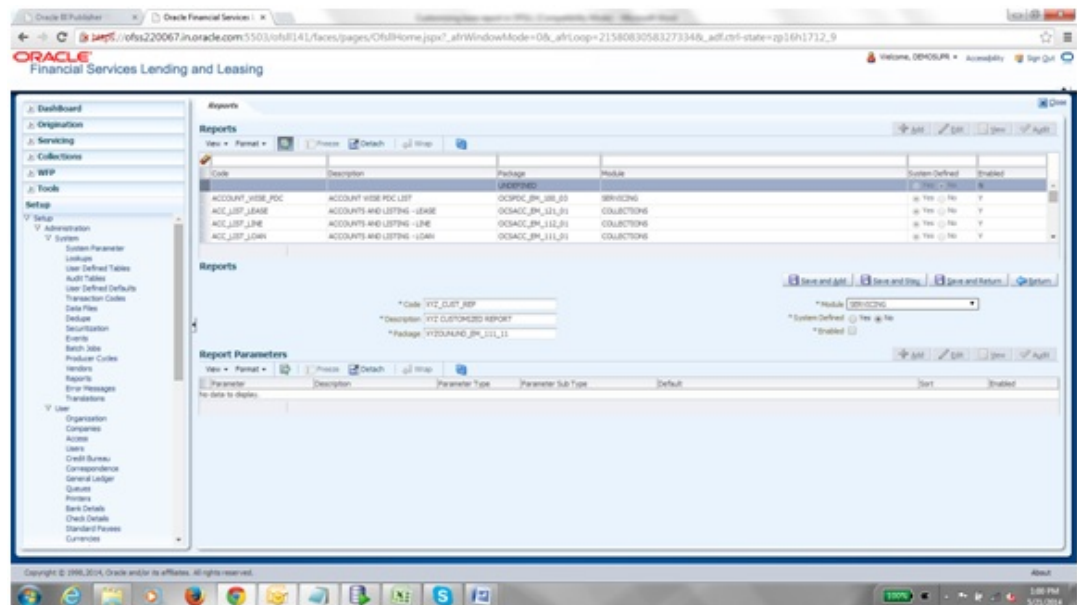
9. Enter the new name as xyzounund_em_111_11 and press Rename button as shown in the image below.

Figure 6-4 Customizing Base Reports 4



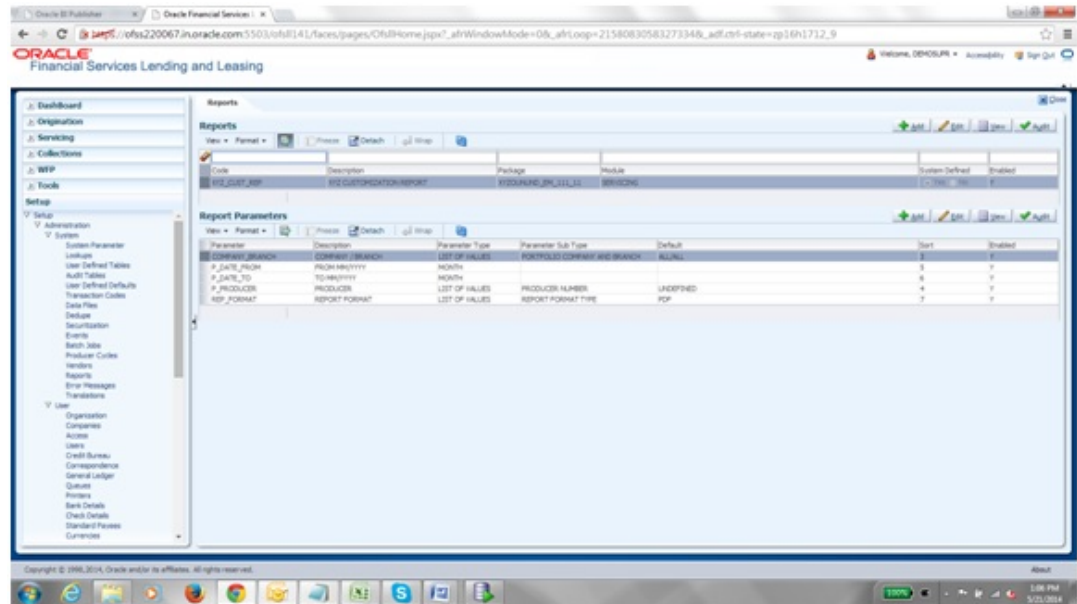
10. Do similar Copy, Paste and rename activity for the data model side also for the particular report which will be with the same name as of the report in the same directory.
11. After the new report is ready i.e. xyzound_em_111_11. We can now do our customizations whatever is required on this new report.
12. So after Completing from bi publisher side we need to make an entry in the database for the new report to be available in the front end application.
13. Go to Setup System Reports in the front end and add a new record with package name as XYZOUNUND_EM_111_11 as shown in the image below.

Figure 6-5 Customizing Base Reports 5



14. After Save and return please add the report parameters for the particular report as shown in the image below.

Figure 6-6 Customizing Base Reports 6



- Now open User Access Reports Tab and select the newly created report and add the responsibility to it in Reports user access definition screen as shown in the image below.

Figure 6-7 Customizing Base Reports 7

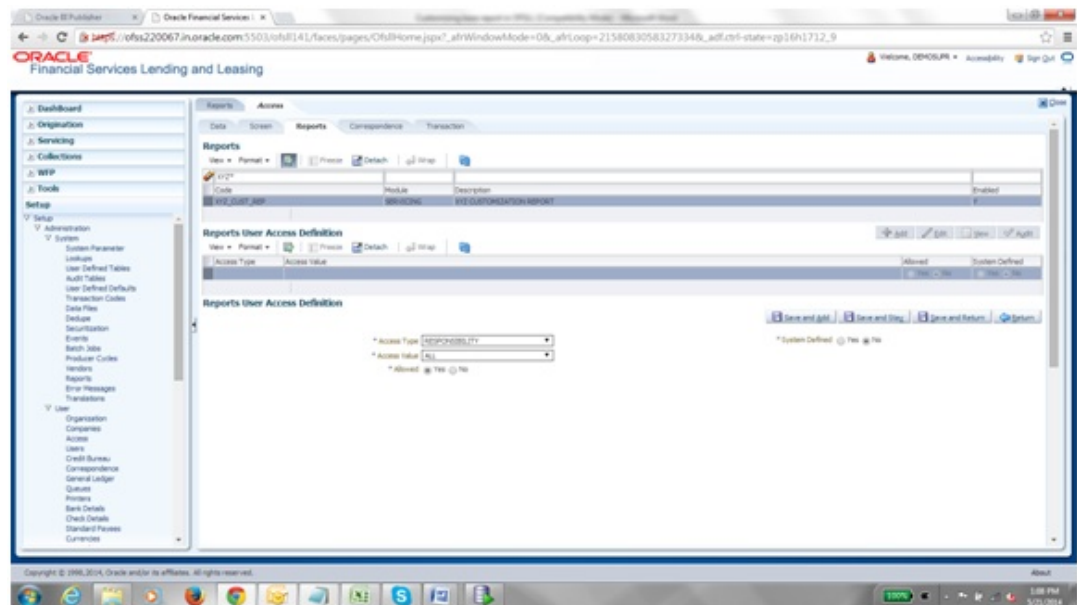
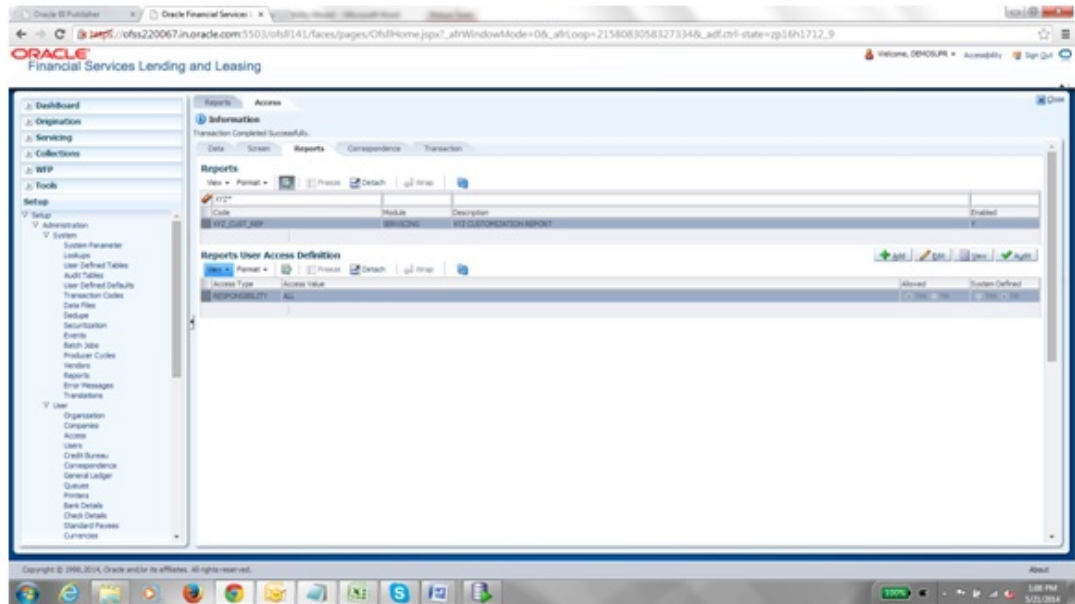
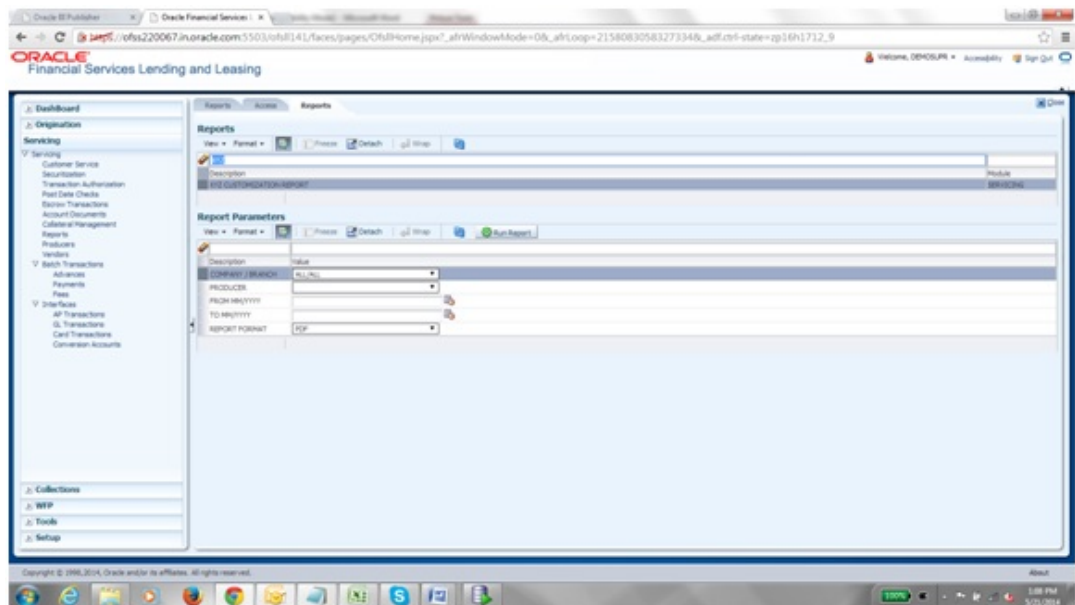


Figure 6-8 Customizing Base Reports 8



- Now go to Servicing > Reports screen and you will be able to find the new report in the list as shown in the image below.

Figure 6-9 Customizing Base Reports 9



Note:

There is no impact on customization reports when a new base patch applied in the system. All customized report will not be override, removed or modified.

7

Customize Existing Base BIP Reports

To customize a report, follow the steps given below:

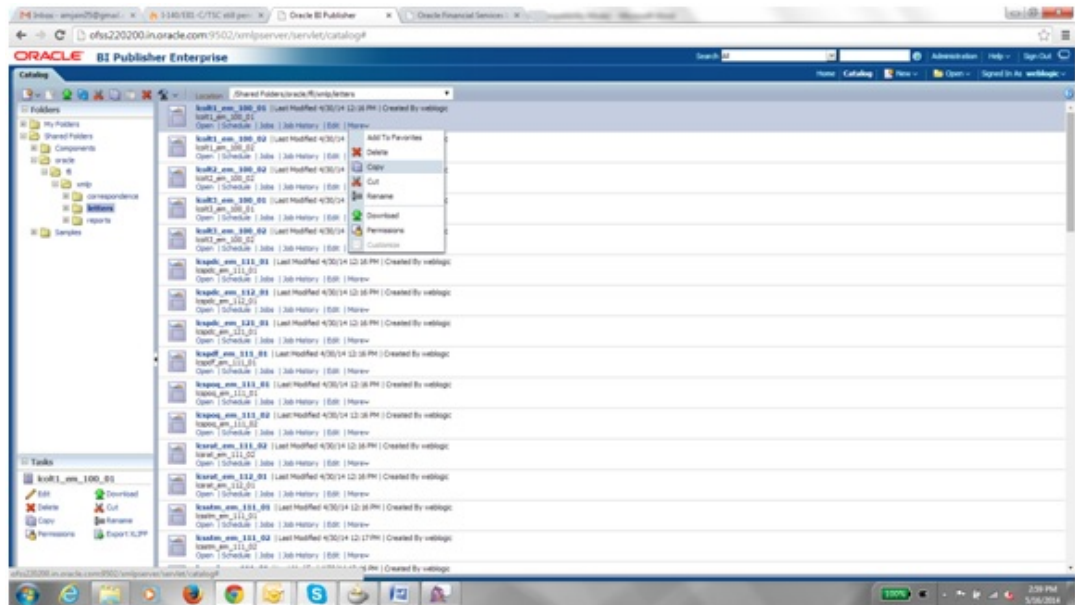
- Do not modify the OFSLL Base Report.
- Create a copy of the OFSLL Base Report, rename and modify that report. A new report also can be created. Name the report as xyz_<report_name>.
- Register the new report and its parameters in OFSLL using reports setup.
- [Customizing Existing Base BIP Reports](#)

7.1 Customizing Existing Base BIP Reports

Pre-Requisites

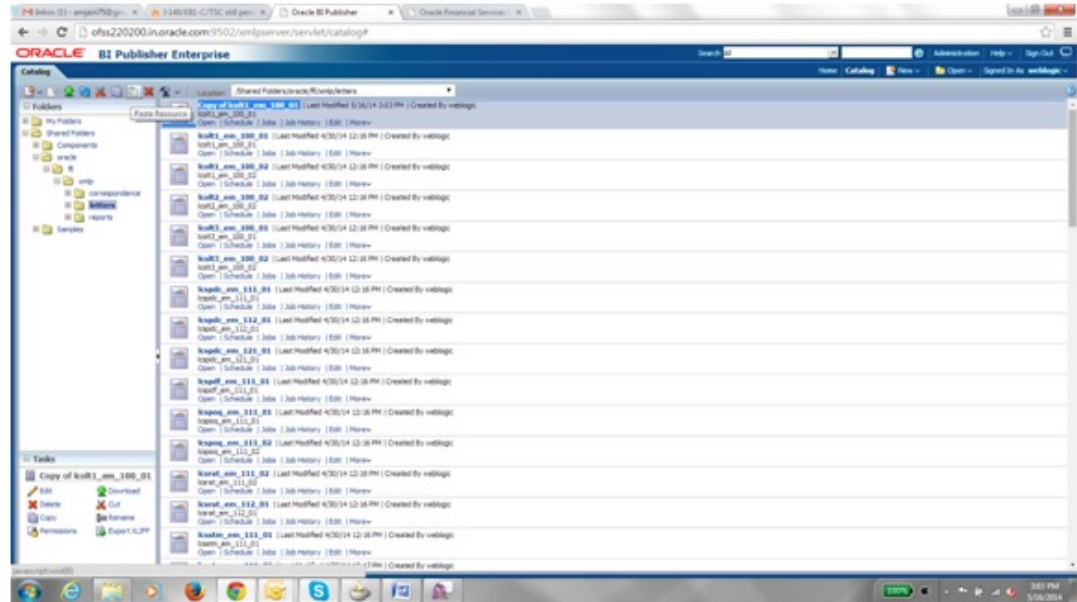
1. Please make sure that we should not change any base letters.
2. The letters should be placed into the same folder structure.
For Letters → Shared Folders/oracle/fll/xmp/letters
3. Consider a base report lcolt1_em_100_01 (Collection Letter).
4. Let us assume we will do some customizations on the base report and create a new report called xyzlcolt1_em_100_01(Here XYZ is bank /customer code).
5. Search for the base letter and press **More** → **Copy** as shown in the image below.

Figure 7-1 Customizing Base Letters 1



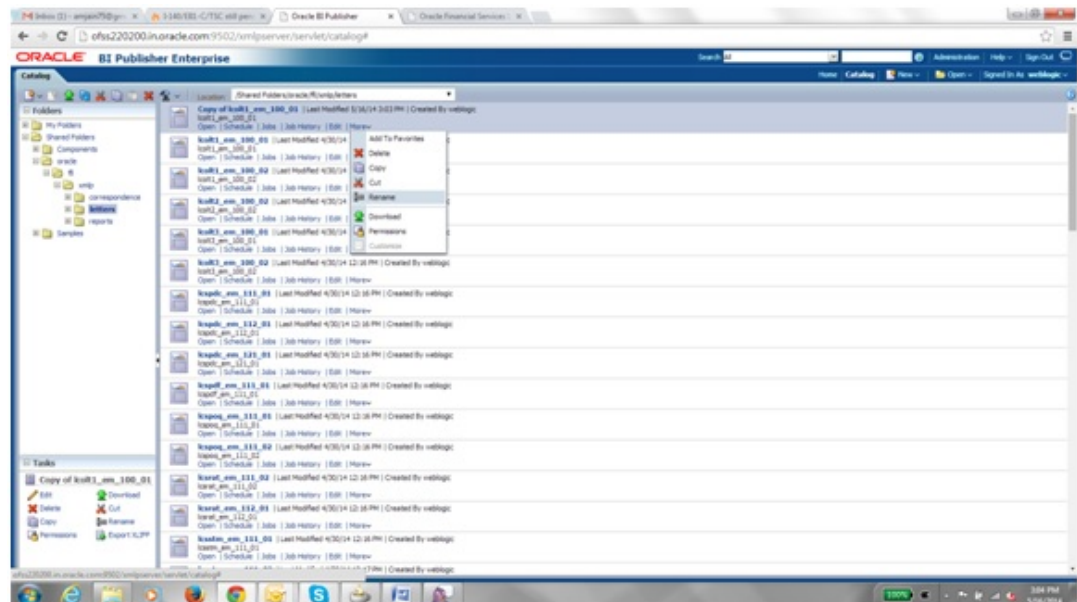
6. After pressing copy go to the folder where you want to paste the new report and press the **Paste Resource** button as shown in the image below.

Figure 7-2 Customizing Base Letters 2



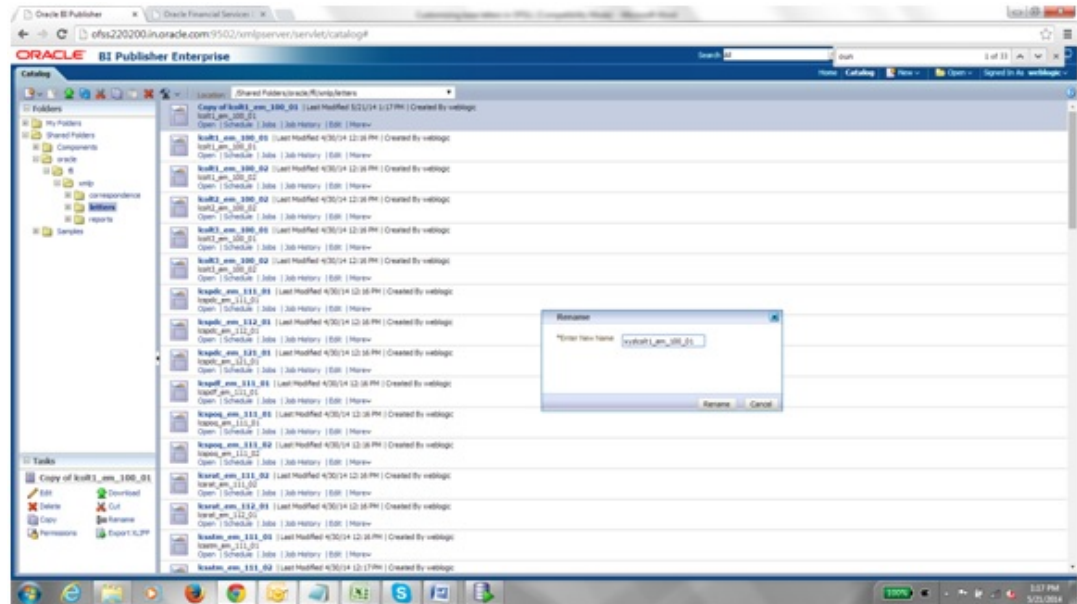
7. On pressing Paste Resource button, a new report will be created in the directory with name as Copy of lcolt1_em_100_01
8. Select the particular report (Copy of lcolt1_em_100_01) and press **More** **Rename** as shown in the image below

Figure 7-3 Customizing Base Letters 3



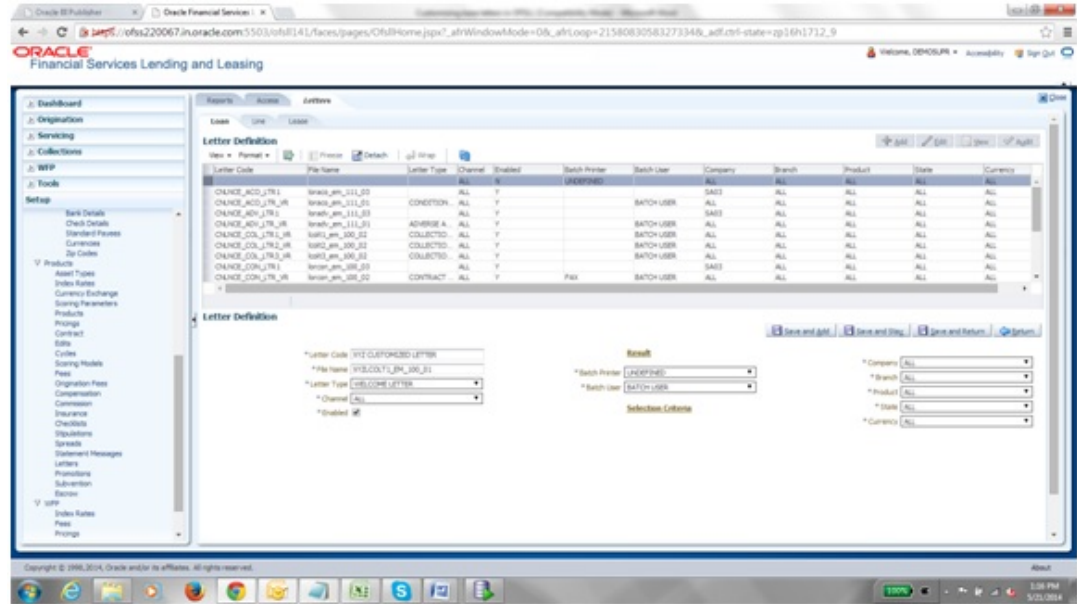
9. Enter the new name as xyzlcolt1_em_100_01 and press **Rename** button as shown in the image below.

Figure 7-4 Customizing Base Letters 4



10. Do similar Copy, Paste and rename activity for the data model side also for the particular report which will be with the same name as of the report in the same directory.
11. After the new report is ready i.e. xyzcolt1_em_100_01. We can now do our customizations whatever is required on this new report.
12. So after Completing from bi publisher side we need to make an entry in the database for the new report to be available in the front end application.
13. Go to Setup → Products → Letters in the front end and add a new record with package name as xyzcolt1_em_100_01 as shown in the image below. Only one letter can be saved only for following combination.
 - Letter Type
 - Company
 - Branch
 - Product
 - State
 - Currency
14. Save and return.

Figure 7-5 Customizing Base Letters 5



Note:

There is no Impact on customization letters when a new base patch applied in the system. All customized letters will not be override, removed or modified.

8

Create Custom Correspondence

The Correspondence screen enables you to define who will receive the documents you created on the Document Definition page by creating correspondence sets. Each document must belong to a set, and a set can have more than one document.

You can set up the various documents and the data fields that the system compiles together when creating a correspondence. The system provides two different document formats: Word or XFDF: XML-based format.



Note:

Oracle Financial Services Software assumes that the user is familiar with Word and the Merge Document command. If the user is creating e-form documents with XFDF, then Oracle Financial Services Software assumes that person is familiar with Adobe forms.

- [Creating Correspondence](#)

8.1 Creating Correspondence

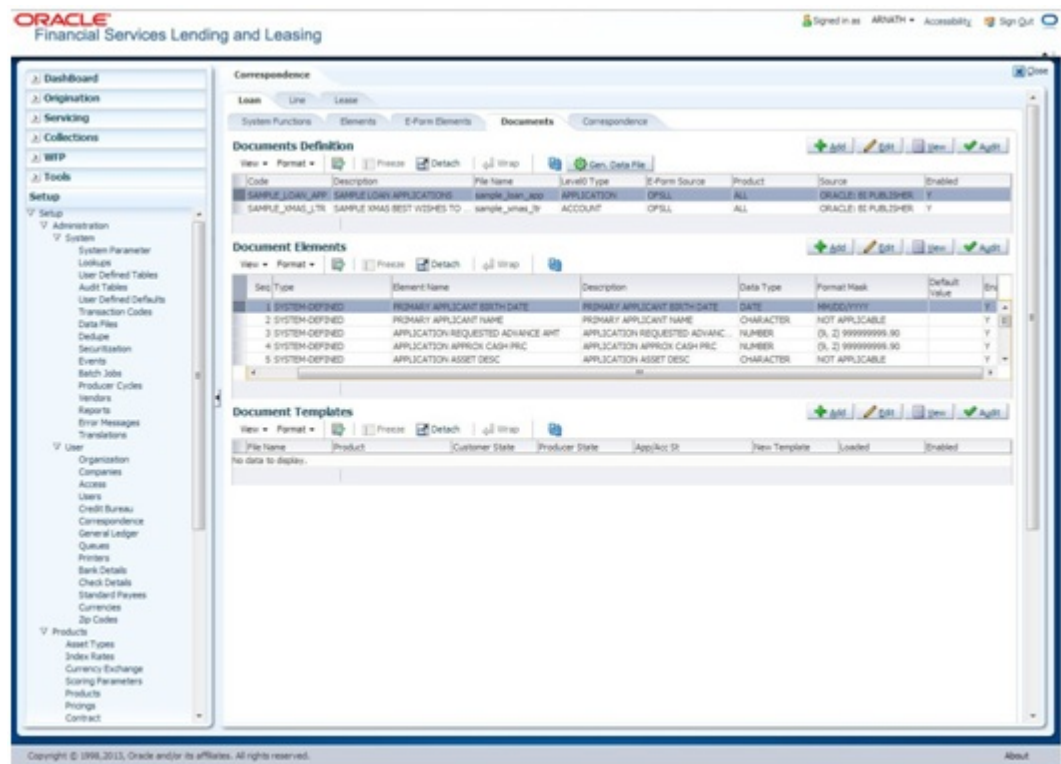
1. On the Oracle Financial Services Lending and Leasing home page, click Setup > Setup > Administration > User > Correspondence > Loan/Line/Lease > Documents
2. In the Document definition block, add a record. For example: SAMPLE_LOAN_APP A brief description is given below:

Table 8-1 Document definition block

Field	Do this
Code	Specify the document code to define the name for the new document.
Description	Specify the document description for the new document. This entry appears in the Correspondence section on the Request page, when you generate an ad hoc correspondence.
File Name	Specify the document file name for the resulting file (Word or XFDF document).
Level0 Type	Select the level0 type from the drop-down list.
E-form Source	Select the element e-form source from the drop-down list.
Product	Select the document product from the drop-down list.
Source	Select the document source type from the drop-down list.
Enabled	Check this box to enable the document definition.

- In the Document **Elements** section, add the elements required in the correspondence.

Figure 8-1 Creating Correspondence 1



Brief description of the fields are given below:

Table 8-2 Description of the fields

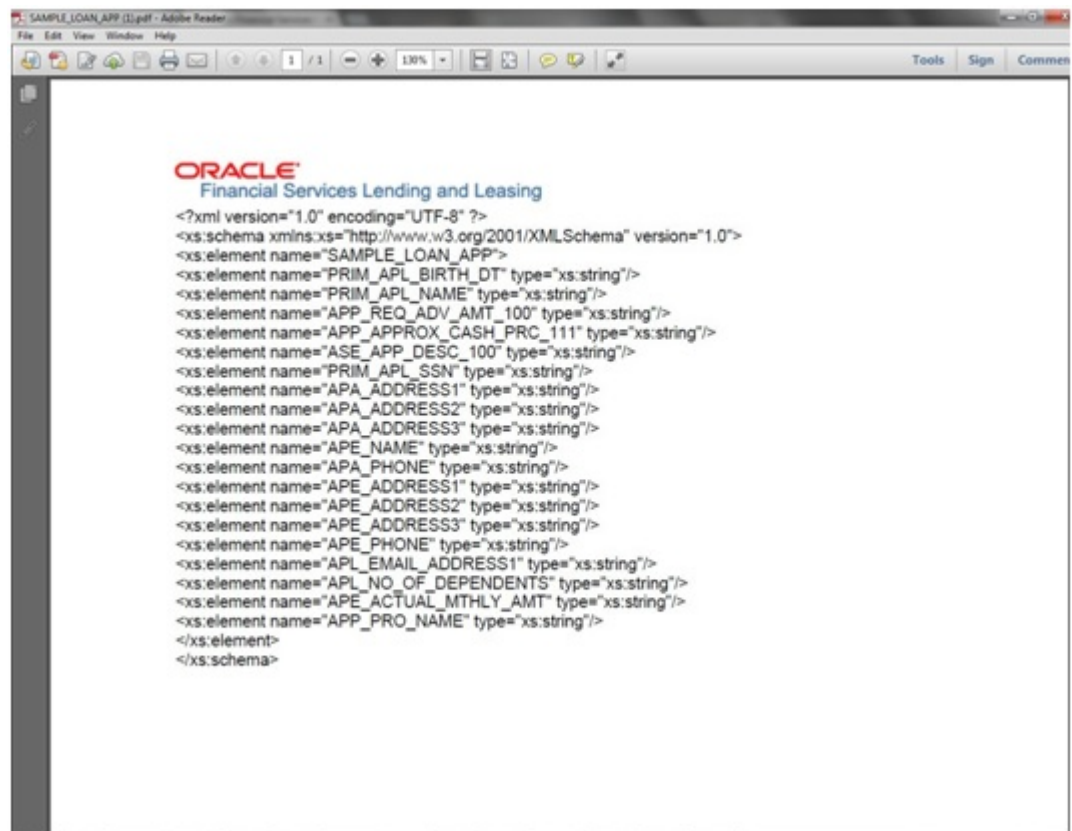
Field	Do this
Seq	Specify the sequence number to order the document elements.
Type	Select element type from the following from the drop-down list. This list provides the following options: System-defined – If you select, the value is supplied by the system and cannot be changed in the Correspondence Request page. Constant. User Defined Element – If you select, you can choose the value and change it in the Correspondence Request screen. User Defined Constant – If you choose, you can choose the value, but you cannot change it in the Correspondence Request screen. Translated Element – If a document contains an e-form element and you do not select this option, then the value will not be translated.

Table 8-2 (Cont.) Description of the fields

Field	Do this
Element Name	Select the element name from the drop-down list.
Description	Specify element description. Notes: 1. Check that the element name does not have blank spaces or special characters, such as the forward slash "/" or backward slash "\". 2. If the element is system-defined, then the system will automatically complete this field.
Data Type	Select the element data type from the drop-down list.
Format Mask	Select the element format mask from the drop-down list.
Default Value	Specify the element default value.
Enabled	Check this box to include the element in the document.

4. Click on **Gen.Data** File to generate PDF file of the report.

Figure 8-2 Creating Correspondence 2



5. Copy and save the content in the pdf file as an xml file. The saved xml file should have the same name as entered in the Code column of Document Definition section. For Example: SAMPLE_LOAN_APP.xml.

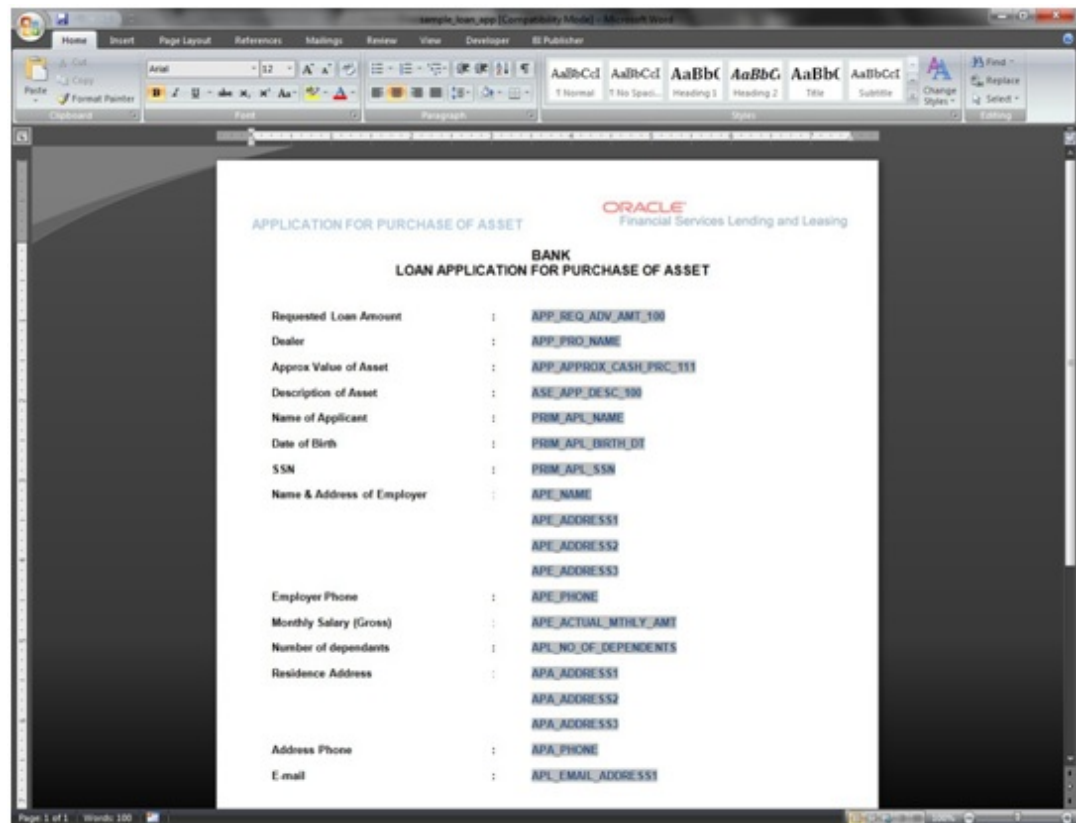
- Open MS Word.

 **Note:**

Oracle Financial Services Software assumes that BIP Desktop Tool is installed and the user is familiar with the BIP Report Tool.

- In BI Publisher Tab in MS Word, click on Sample XML and import the saved xml file. For Example: SAMPLE_LOAN_APP.xml.
- Create the template by inserting required elements tag.

Figure 8-3 Creating Correspondence 3



- The template created in MS Word should be saved with .rtf extension. For Example: SAMPLE_LOAN_APP.rtf

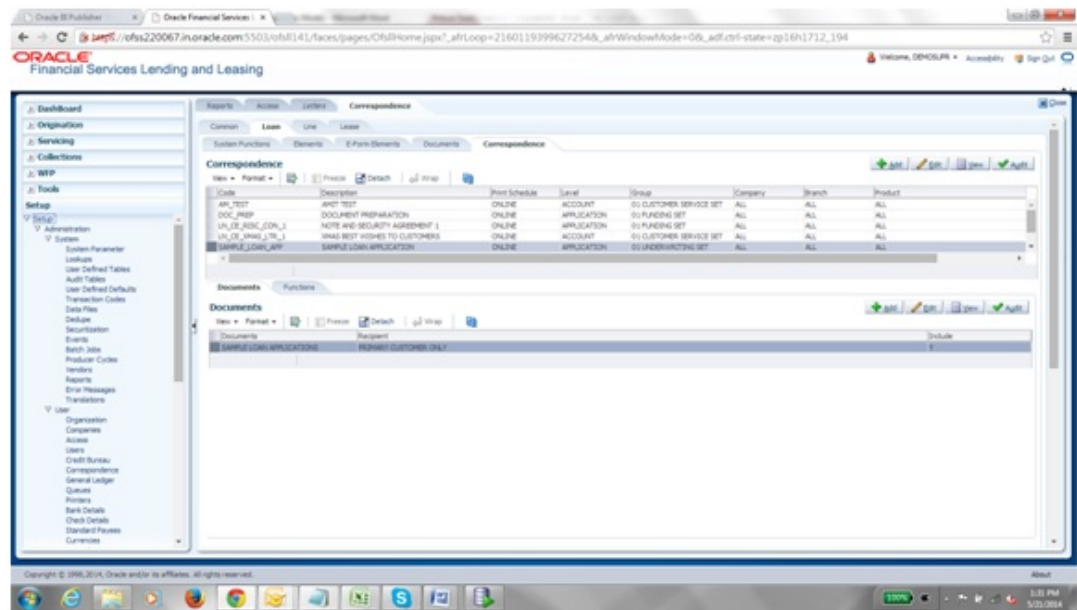
 **Note:**

The .xml and .rtf file should be saved with the same name as entered in the **Code** column of Document Definition section.

- Upload the rtf template in the BIP and create the data model with SQL query as `select CDO_XML_DOCUMENT from correspondence_docs where cdo_id = :docId.`
- After the data model creation, launch the correspondence screen and click **Correspondence** tab.

- You can setup a correspondence with the created doc.

Figure 8-4 Creating Correspondence 4



9

Generate Correspondence

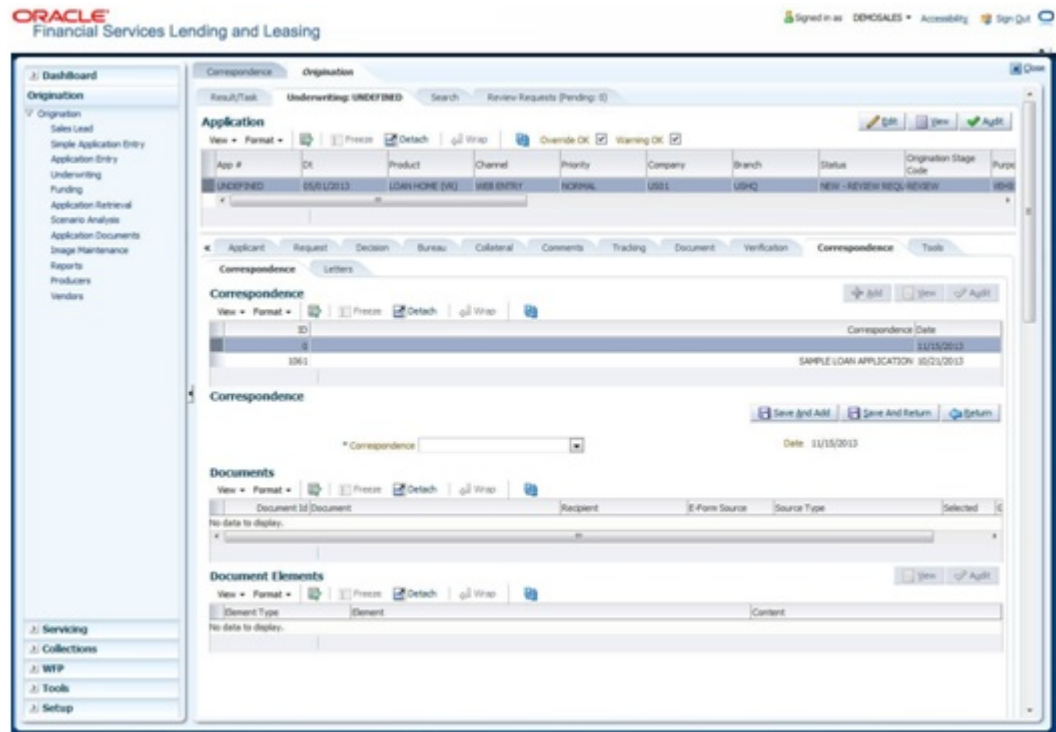
You can generate a correspondence once the respective correspondence is created in the database.

- [Generating Correspondence](#)

9.1 Generating Correspondence

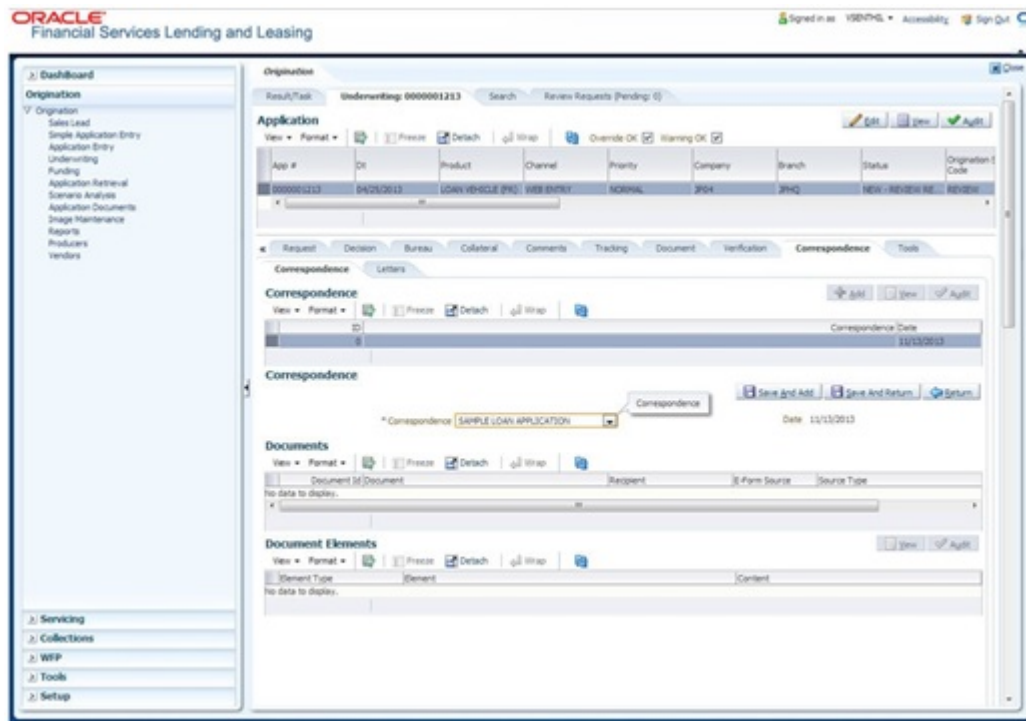
1. On the Oracle Financial Services Lending and Leasing home page, click Origination → Origination → Underwriting
2. Open the application for which the correspondence should be generated.
3. Click **Correspondence** tab. In the Correspondence section, click on **Add**.

Figure 9-1 Generating Correspondence 1



4. Select the created Correspondence. Click **Save and Add** to save and add a new record.
5. Click to **Save and Return** to save and return to main screen. Click **Return** to return to main screen without modifications.

Figure 9-2 Generating Correspondence 2



6. Click **Generate** to generate the selected correspondence and **View Correspondence** to view the Correspondence in PDF format.

Figure 9-3 Generating Correspondence 3



 **Note:**

There is no Impact on customization letters when a new base patch applied in the system. All customized letters will not be override, removed or modified.

10

Set up the output Format For BIP Reports

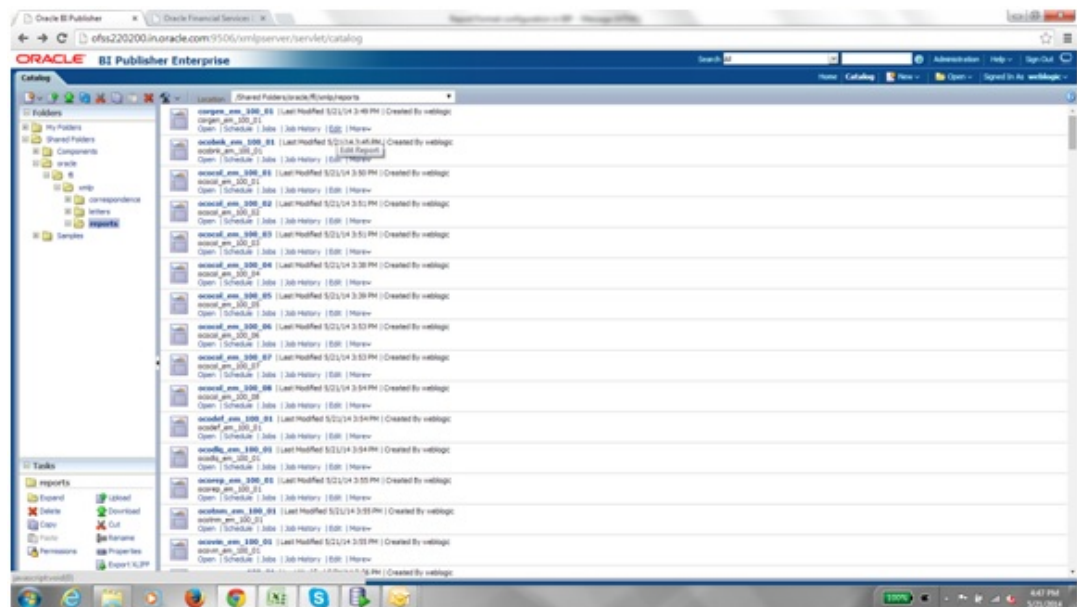
The following section details the steps to set up the output format for BIP reports.

- [Setting up the output Format For BIP Reports](#)

10.1 Setting up the output Format For BIP Reports

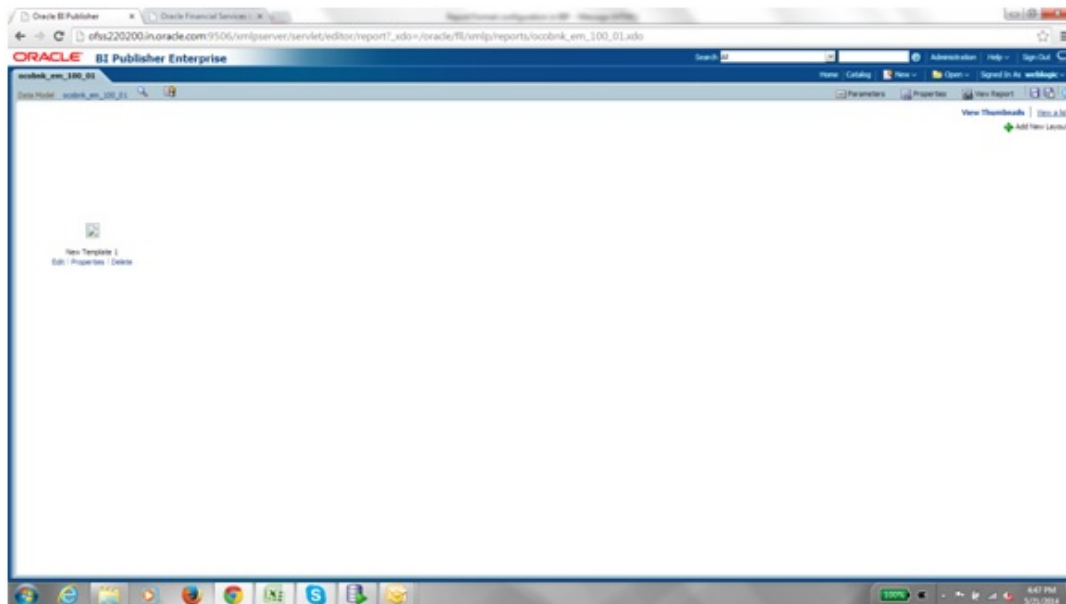
1. Go to catalog and select the desired report (XDO). Press **Edit**.

Figure 10-1 Setting up the output Format 1



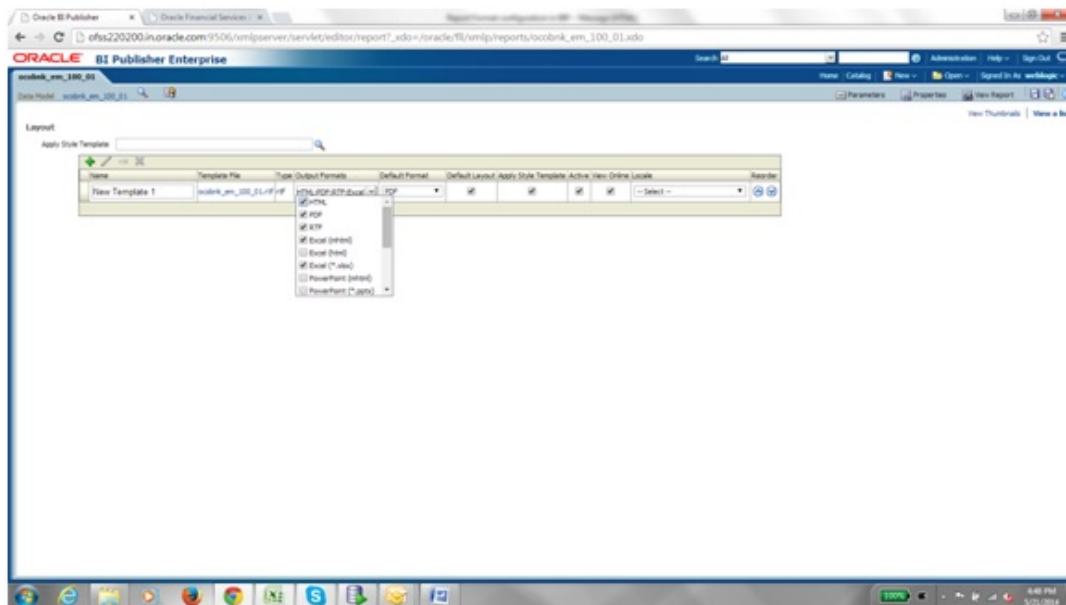
2. On **Edit** it will open the following page. On right corner there is **View a List** link , click on it.

Figure 10-2 Setting up the output Format 2



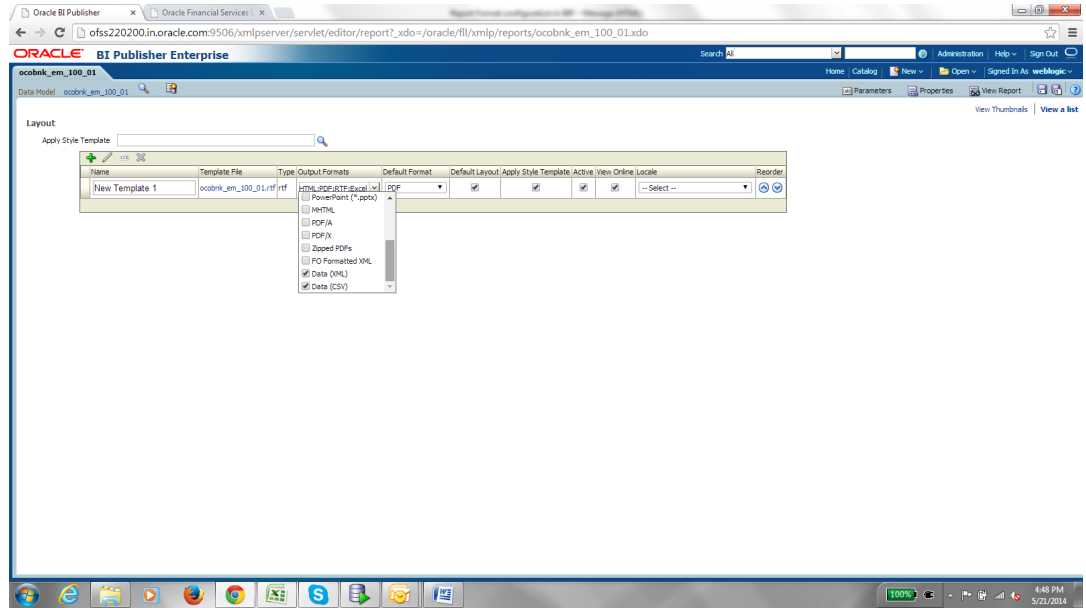
3. It will open the Layout format, click on the **Output Formats** dropdown. Select the following formats HTML, PDF, RTF, EXCEL (.xlsx).

Figure 10-3 Setting up the output Format 3



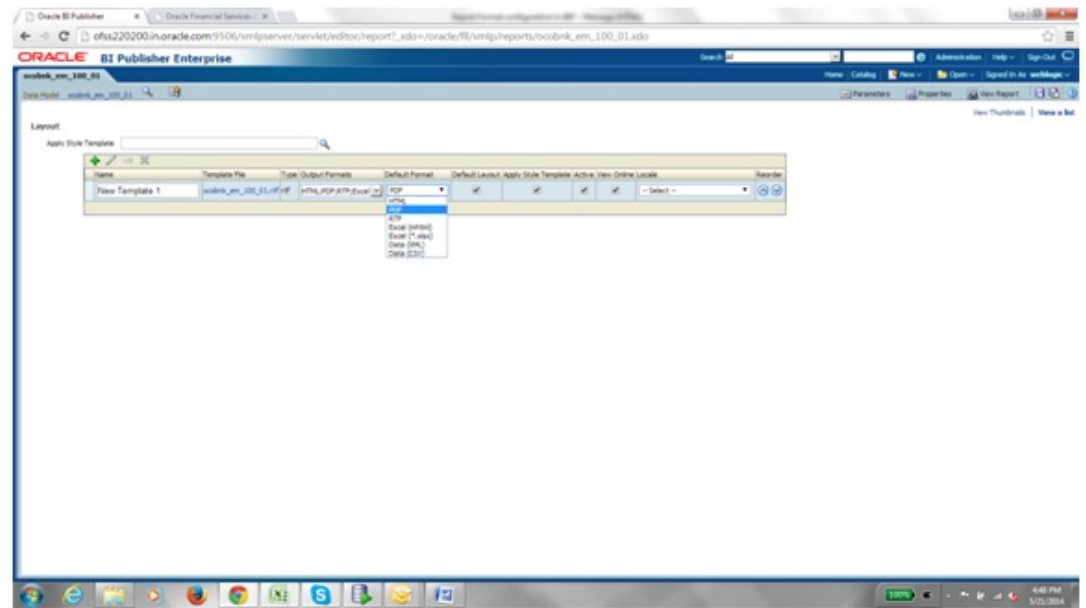
4. And in the end of drop down select DATA(XML) and DATA(CSV).

Figure 10-4 Setting up the output Format 4



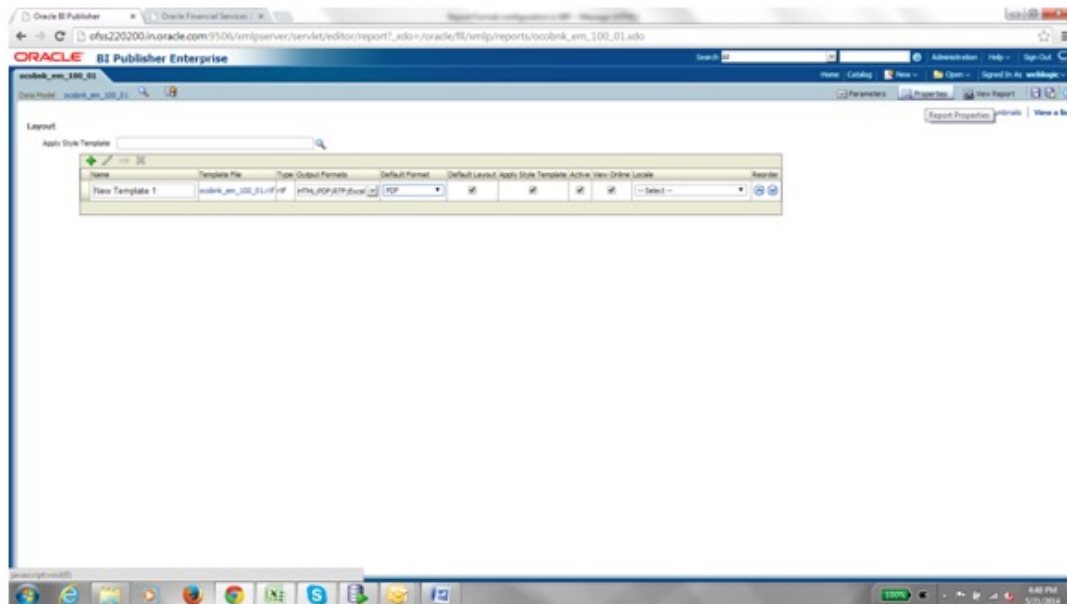
5. Also please select the Default Format as PDF

Figure 10-5 Setting up the output Format 5



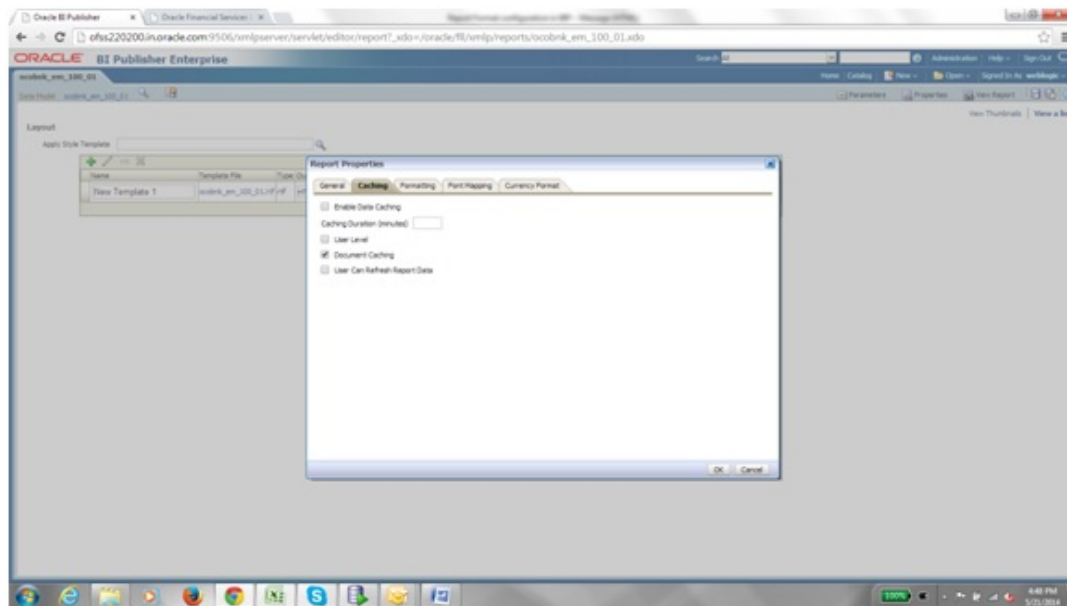
6. Once Output Format is done, I would recommend to make the Caching **False** for online reports as follows. On Right side, Click on the **Properties** Button.

Figure 10-6 Setting up the output Format 6



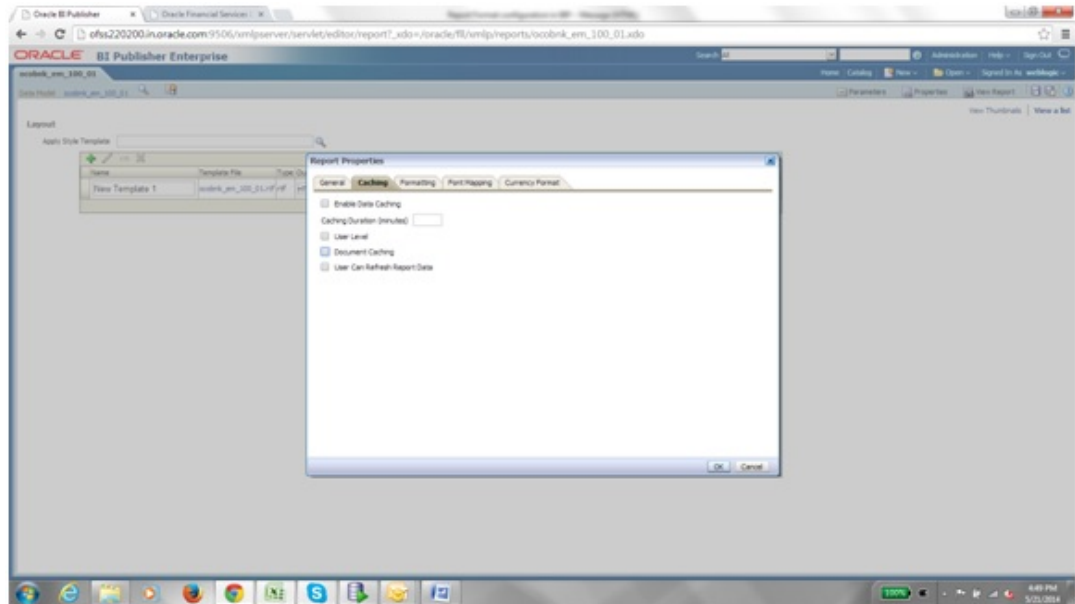
- 7. Select the TAB Caching, check if any of the caching is selected.

Figure 10-7 Setting up the output Format 7



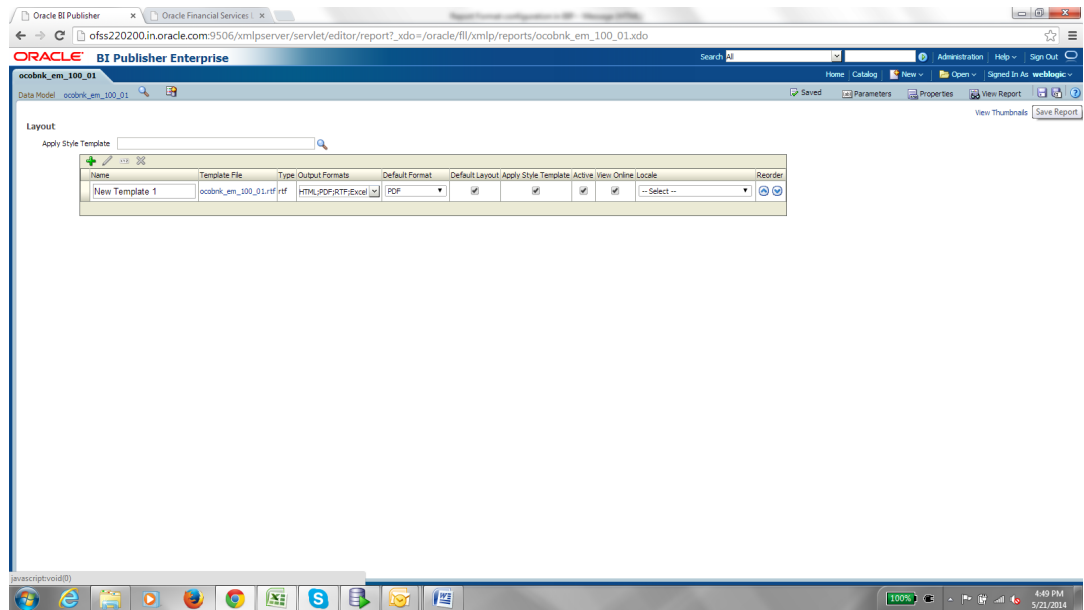
- 8. Unchecked all the caching if any of them are checked, By Default Document Caching is always true, make it false.

Figure 10-8 Setting up the output Format 8



9. Save the changes.

Figure 10-9 Setting up the output Format 9



11

Naming Convention for Customized Objects

The following table details the naming convention for customized objects.

Table 11-1 Naming Convention

Object	Naming Convention	Comment
New Table	<table_name>_xyz	Same as Column Naming Convention
New View	<view_name>_xyz	Same as Column Naming Convention
New Column in OFSLL Base Version Table	abc_<column_name>_xyz	
New Column in OFSLL Base Version View	abc_<column_name>_xyz	
New Sequence	abc_seqnum_xyz	
New Unique Index	abc_udx_xyz/ abc_udx2_xyz	
New Non Unique Index	abc_idx_xyz/ abc_idx2_xyz	
New System Parameter (Seed Data)	<system_parmeter>_xyz	
New Lookup Type (Seed Data)	<lookup_type>_xyz	
New Lookup Code (Seed Data)	<lookup_code>_xyz	
New Other (Seed Data)	<seed_code>_xyz	
New Correspondence Function	<function_name>_xyz	
New Correspondence Element	<element_name>_xyz	
New Package Name (EM/EN)	xyz<package_name>	
New Package File Name(EM/EN)	xyz<package_name>	
New Package Name (EX)	<package_name>	
New Package File Name(EX)	xyz<package_name>	
New Report File Name	xyz<report_name>	
View File Name	xyz<view file name>	

- Where 'xyz' is Customer Unique Id
- Signature of Base OFSLL Package Functions, Package Procedures, Reports, Correspondences and Faxes should not be changed.
- No New Functions or Procedures should be added to OFSLL Base Packages.
- List of Objects with exceptions must be published.

When checking-in custom code in version control software, follow the guidelines given below:

1. Instead of putting all the code in one directory, follow the Base Engine directory structure.
2. For New custom Engine Create a New Engine directory.
3. Follow the naming convention for the files. All package files should start with three-character client name.

e.g.: ulnapp_el_100_01.pkb will become : xyzulnapp_el_100_01.pkb for XYZ Bank.

uln_eww.sql will become : xyzuln_eww.sql for XYZ Bank.

DDL scripts should end with the three-character client name.

e.g. crt_vw_applications.sql will become crt_vw_applications_xyz.sql for XYZ Bank.

12

RESTful Web Services Extensibility

Refer to the following section for details on extensibility for the below RESTful Web Services:

- [Generic Post Transaction \(POST\)](#)
- [Account On Boarding \(POST\)](#)
- [Payment Posting \(POST\)](#)
- [Account Detail \(GET\)](#)
- [Scenario Analysis \(POST\)](#)
- [Lookups \(GET\)](#)
- [Application Search \(GET\)](#)
- [Calculator \(POST\)](#)
- [Application Entry \(POST\)](#)
- [Account Search \(GET\)](#)
- [Call Activity \(POST\)](#)
- [Remarketing \(PUT\)](#)
- [Invoice \(POST\)](#)
- [Application Comment \(GET/POST\)](#)
- [Account Comment \(GET/POST\)](#)
- [Application Checklist \(GET\)](#)
- [Outgoing File List \(GET\)](#)
- [Outgoing File \(POST\)](#)
- [Incoming File \(GET\)](#)
- [Products \(GET\)](#)
- [Assets \(GET\)](#)
- [Assets \(PUT\)](#)
- [Asset Valuation \(GET/PUT/POST\)](#)
- [Asset Sub Types \(GET\)](#)
- [Application Status Change \(PUT\)](#)
- [Application Update \(PUT\)](#)
- [Application ACH \(POST\)](#)
- [Application Document upload/download/list Service \(POST/GET/GET\)](#)
- [Account Document upload/download/list Service \(POST/GET/GET\)](#)
- [Application Get Service \(GET\)](#)
- [Scheduler Force ReSubmit Service \(PUT\)](#)

- [Credit Limit Service \(Customer/Business\) \[GET\]](#)
- [Business Comments Service \(GET/POST\)](#)
- [Customer Comments Service \(GET/POST\)](#)
- [Customer Preference Service \(GET/POST/PUT\)](#)
- [Scenario Analysis Service \(PUT\)](#)
- [Transaction Parameters Service \(GET\)](#)
- [Asset Tracking Attribute Service\(PUT\)](#)
- [Business Tracking Attribute Service\(PUT\)](#)
- [Customer Tracking Attribute Service\(PUT\)](#)
- [Account Tracking Attribute Service\(PUT\)](#)
- [Credit Bureau Web Service\(PUT\)](#)
- [Delete Account Web Service\(DELETE\)](#)
- [New Customization for RESTful Web Service](#)
- [Securitization Web Service \(POST\)](#)
- [Calculate Parameter Update Service\(PUT\)](#)
- [Master Account Rolledup details service \(GET\)](#)
- [Application Decision Fetch Service \(GET/PUT\)](#)
- [Pricing Fetch Service \(GET\)](#)
- [Account UDF Update Service \(PUT\)](#)
- [Stipulation Fetch Service \(GET\)](#)
- [Work Order Fetch Service \(GET/PUT/POST\)](#)
- [Generic Post Transaction \(POST\)](#)
Based on the type of element names in the below mentioned tables there is a sub element and child element along with their data types. If any custom field is required which is of date type, TransactionDateParameter should be used along with other fields and their values.
- [Account On Boarding \(POST\)](#)
Below mentioned table has element name which indicates which type of custom data is passed in that enclosing the name and its value in key name and key value respectively.
- [Payment Posting \(POST\)](#)
Below mentioned table has element name which indicates which type of custom data is passed in that enclosing the name and its value in keyname and keyvalue respectively.
- [Account Detail \(GET\)](#)
Below mentioned table has element name which indicates which type of custom data is passed in that enclosing the name and its value in name and value respectively.
- [Scenario Analysis \(POST\)](#)
Below mentioned table has element name which indicates which type of custom data is passed by enclosing the name and its value in keyname and keyvalue respectively.
- [Lookups \(GET\)](#)
Below mentioned table has element name which indicates which type of custom data is passed by enclosing the name and its value in keyname and keyvalue respectively.

- [Application Search \(GET\)](#)
Below mentioned table has element name which indicates In response which type of custom data is passed by enclosing the name and its value in KeyName and KeyValue respectively.
- [Calculator \(POST\)](#)
Below mentioned table has element name which indicates which type of custom data is passed by enclosing the name and its value in KeyName and KeyValue respectively.
- [Application Entry \(POST\)](#)
Below mentioned table has element name which indicates which type of custom data is passed in that enclosing the name and its value in keyname and keyvalue respectively.
- [Account Search \(GET\)](#)
Below are the package details for account search web service
- [Call Activity \(POST\)](#)
Below mentioned are the custom fields which are part of response alone.
- [Remarketing \(PUT\)](#)
Below mentioned table has element name which indicates which type of custom data is passed by enclosing the name and its value in keyname and keyvalue respectively.
- [Invoice \(POST\)](#)
Below mentioned table has element name which indicates which type of custom data is passed by enclosing the name and its value in keyname and keyvalue respectively.
- [Application Comment \(GET/POST\)](#)
Below mentioned are the custom fields. These fields will be part of response in case of GET method and will be part of request in case of POST method.
- [Account Comment \(GET/POST\)](#)
Below mentioned are the custom fields. These fields will be part of response in case of GET method and will be part of request in case of POST method.
- [Application Checklist \(GET\)](#)
Below mentioned are the custom fields. These fields will be part of response.
- [Outgoing File List \(GET\)](#)
Below mentioned are the custom fields. These fields will be part of response.
- [Outgoing File \(POST\)](#)
Below are the package details for outgoing file web service
- [Incoming File \(GET\)](#)
Below are the package details for incoming file web service
- [Products \(GET\)](#)
Below mentioned are the custom fields. These fields will be part of response.
- [Assets \(GET\)](#)
Below mentioned are the custom fields. These fields will be part of response.
- [Assets \(PUT\)](#)
Below mentioned are the custom fields. These fields will be part of response for PUT services.
- [Asset Valuation \(GET/PUT/POST\)](#)
Below mentioned are the custom fields. These fields will be part of response for GET/PUT/ POST services.
- [Asset Sub Types \(GET\)](#)
Below mentioned are the custom fields. These fields will be part of response.

- [Application Status Change \(PUT\)](#)
Below the custom packages.
- [Application Update \(PUT\)](#)
Below mentioned table has element name which indicates which type of custom data is passed in the custom element block, enclosing the name and its value in keyname and keyvalue respectively.
- [Application ACH \(POST\)](#)
Below mentioned table has element name which indicates which type of custom data is passed in the custom element block, enclosing the name and its value in keyname and keyvalue respectively.
- [Application Document upload/download/list Service \(POST/GET/GET\)](#)
Below are the package details for application document upload/download web service
- [Account Document upload/download/list Service \(POST/GET/GET\)](#)
Below are the package details for Account document upload/download web service
- [Application Get Service \(GET\)](#)
Below are the package details for application get web service
- [Scheduler Force ReSubmit Service \(PUT\)](#)
Below are the package details for force resubmit job set web service
- [Credit Limit Service \(Customer/Business\) \[GET\]](#)
Below mentioned are the custom fields. These fields will be part of response.
- [Business Comments Service \(GET/POST\)](#)
Below mentioned are the custom fields. These fields will be part of response for both GET and POST service and will be part of request for POST service.
- [Customer Comments Service \(GET/POST\)](#)
Below mentioned are the custom fields. These fields will be part of response for both GET and POST service and will be part of request for POST service.
- [Customer Preference Service \(GET/POST/PUT\)](#)
Below mentioned are the custom fields. These fields will be part of response for GET, PUT and POST service and will be part of request for POST and PUT service.
- [Scenario Analysis Service \(PUT\)](#)
Below mentioned are the custom fields. These fields will be part of both request and response.
- [Transaction Parameters Service \(GET\)](#)
Below mentioned are the custom fields. These fields will be part of response.
- [Asset Tracking Attribute Service\(PUT\)](#)
Below mentioned are the custom fields. These fields will be part of both request and response.
- [Business Tracking Attribute Service\(PUT\)](#)
Below mentioned are the custom fields. These fields will be part of both request and response.
- [Customer Tracking Attribute Service\(PUT\)](#)
Below mentioned are the custom fields. These fields will be part of both request and response.
- [Account Tracking Attribute Service\(PUT\)](#)
Below mentioned are the custom fields. These fields will be part of both request and response.

- [Credit Bureau Web Service\(PUT\)](#)
Below mentioned are the custom fields. These fields will be part of both request and response.
- [Delete Account Web Service\(DELETE\)](#)
Below mentioned are the custom fields. These fields will be part of both request and response.
- [New Customization for RESTful Web Service](#)
Below are the list of services introduced and it follows new naming standard for custom elements in request/response.
- [Securitization Web Service \(POST\)](#)
Below mentioned are the custom fields. These fields will be part of both request and response.
- [Calculate Parameter Update Service\(PUT\)](#)
- [User Update Service \(PUT\)](#)
- [Master Account Rolledup details service \(GET\)](#)
- [Application Decision Fetch Service \(GET/PUT\)](#)
Below mentioned are the custom fields. These fields will be part of both request/response.
- [Work Order Fetch Service \(GET/PUT/POST\)](#)
Below mentioned are the custom fields. These fields will be part of both request/response.
- [Pricing Fetch Service \(GET\)](#)
Below mentioned are the custom fields. These fields will be part of both request/response.
- [Account UDF Update Service \(PUT\)](#)
Below mentioned are the custom fields. These fields will be part of both request/response.
- [Stipulation Fetch Service \(GET\)](#)
Below mentioned are the custom fields. These fields will be part of both request/response.

12.1 Generic Post Transaction (POST)

Based on the type of element names in the below mentioned tables there is a sub element and child element along with their data types. If any custom field is required which is of date type, TransactionDateParameter should be used along with other fields and their values.

Same is applicable for other data types.

- [Producer related transaction](#)
- [Other Transactions](#)

12.1.1 Producer related transaction

Table 12-1 Producer related Transaction

Element name	Sub Element	Child Element	Data Type	Values(Example)
TransactionDateParameter				
	ParameterDetails			
		ParameterName	String	NA
		ParameterValue	Date(YYYY-MM-DDTHH:MM:SS)	NA

Table 12-1 (Cont.) Producer related Transaction

Element name	Sub Element	Child Element	Data Type	Values(Example)
TransactionStringParameter	ParameterDetails			
		ParameterName	String	ACC_NBR
		ParameterValue	String	123654
	ParameterDetails			
		ParameterName	String	PTX_COMMENT
		ParameterValue	String	Test Comment
	ParameterDetails			
		ParameterName	String	PTX_REFERENCE
		ParameterValue	String	Test Reference
	TransactionNumberParameter	ParameterDetails		
		ParameterName	String	PTX_AMT
		ParameterName	Number	150

12.1.2 Other Transactions

Table 12-2 Other Transactions

Element name	Sub Element	Child Element	Data Type	Values
TransactionDateParameter	ParameterDetails			
		ParameterName	String	
		ParameterValue	Date(YYYY-MM-DDTHH:MM:SS)	
	ParameterDetails			
TransactionStringParameter	ParameterDetails			
		ParameterName	String	
		ParameterValue	String	
	ParameterDetails			
TransactionNumberParameter	ParameterDetails			
		ParameterName	String	
		ParameterValue	Number	

Sample format

```
<?xml version="1.0" encoding="UTF-8"?>
<PostTransactionRequest>
  <UserCode></UserCode>
  <TransactionDetails>
```

```

<TransactionType></TransactionType>
<EntityReferenceNumber></EntityReferenceNumber>
<TransactionCode></TransactionCode>
<TransactionDateParameter>
  <ParameterDetails>
<ParameterName>DateParamName</ParameterName>
<ParameterValue> DateParamValue</ParameterValue>
  </ParameterDetails>
</TransactionDateParameter>
<TransactionStringParameter>
  <ParameterDetails><ParameterName> StringParamName</ParameterName>
  <ParameterValue> StringParamValue</ParameterValue>
</ParameterDetails>
</TransactionStringParameter>
<TransactionNumberParameter>
  <ParameterDetails><ParameterName> NumberParamName</ParameterName>
  <ParameterValue> NumberParamValue</ParameterValue>
</ParameterDetails>
</TransactionNumberParameter>
<Result>
  <ResultId></ResultId>
  <Status></Status>
  <StatusDetails></StatusDetails>
</Result>
</TransactionDetails>
</PostTransactionRequest>

```

Below are the package details for generic post transaction

```

xcsupd_ew_100_01.xcsupd_ew_100_01 (iv_txn_tab_t IN xws_att_str_tab_t,
iv_txn_result_rec_t OUT NOCOPY xcs_txn_result_rec_t);
  xcsupd_em_100_01.post_txn (iv_txn_tab_t IN xws_att_str_tab_t,
iv_txn_result_rec_t OUT NOCOPY xcs_txn_result_rec_t);
  xcsupd_en_100_01.post_batch_txn (iv_txn_tab_t IN xws_att_str_tab_t,
iv_txn_result_rec_t OUT NOCOPY xcs_txn_result_rec_t);

```

You can do the customization on xcsupd_en_100_01. post_batch_txn();

Below are the exit points:-

BEFORE:

```

xcsupd_ex_100_01.cv_post_batch_txn_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
xcsupd_ex_100_01.post_batch_txn_bfr (iv_txn_tab_t, iv_txn_inp_rec_t,
iv_txn_result_rec_t);

```

REPLACE:

```

xcsupd_ex_100_01.cv_post_batch_txn_rep = cmncon_cl_000_01.CUSTOMIZED THEN
xcsupd_ex_100_01.post_batch_txn_rep (iv_txn_tab_t, iv_txn_inp_rec_t,
iv_txn_result_rec_t);

```

AFTER:

```
xcsupd_ex_100_01.cv_post_batch_txn_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsupd_ex_100_01.post_batch_txn_afr (iv_txn_tab_t, iv_txn_inp_rec_t,  
iv_txn_result_rec_t);
```

You can do the customization on xcsupd_en_100_01. post_txns();

Below are the exit points:-

BEFORE:

```
xcsupd_ex_100_01.cv_post_txns_bfr= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsupd_ex_100_01.post_txns_bfr (iv_txn_tab_t, iv_txn_inp_rec_t,  
iv_txn_result_rec_t);
```

REPLACE:

```
xcsupd_ex_100_01. cv_post_txns_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsupd_ex_100_01. post_txns_rep (iv_txn_tab_t, iv_txn_inp_rec_t,  
iv_txn_result_rec_t);
```

AFTER:

```
xcsupd_ex_100_01.cv_post_txns_afr= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsupd_ex_100_01. post_txns_afr (iv_txn_tab_t, iv_txn_inp_rec_t,  
iv_txn_result_rec_t);
```

You can do the customization on xcsupd_em_100_01. post_txn();

Below are the exit points:-

BEFORE:

```
xcsupd_ex_100_01. cv_post_txn_bfr= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsupd_ex_100_01. post_txn_bfr (iv_txn_tab_t, iv_txn_inp_rec_t,  
iv_txn_result_rec_t);
```

REPLACE:

```
xcsupd_ex_100_01. cv_post_txn_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsupd_ex_100_01. post_txn_rep (iv_txn_tab_t, iv_txn_inp_rec_t,  
iv_txn_result_rec_t);
```

AFTER:

```
xcsupd_ex_100_01. cv_post_txn_afr= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsupd_ex_100_01. post_txn_afr (iv_txn_tab_t, iv_txn_inp_rec_t,  
iv_txn_result_rec_t);
```

IN parameter is Tab Type object:

```
TYPE xws_att_str_rec_t AS OBJECT (
  ATT_NAME VARCHAR2(30),
  ATT_VALUE VARCHAR2(4000))
```

OUT parameter is Rec Type object:

```
xcs_txn_result_rec_t AS OBJECT (
  XTR_BMT_ID          NUMBER
  , XTR_TXN_RES       XTR_TXN_RES_TAB_T
  , XTR_BMT_STATUS    VARCHAR2(30)
  , XTR_TXN_ERROR     VARCHAR2(2000))
xtr_txn_res_rec_t AS OBJECT (
  XTR_TXN_RESULT     VARCHAR2(4000))
```

IN OUT parameter is Rec Type object:

```
XWS_GEN_TXN_REC_T AS OBJECT (
ACC_NBR              VARCHAR2(50),
TCD_CODES            XWS_TXN_CODES_REC_T,
TXN_AMT              VARCHAR2(30),
BMT_SLOT_NO         NUMBER,
CREATED_BY           VARCHAR2(30),
BMT_BATCH_TXN_IND   VARCHAR2(30),
USER_CODE            VARCHAR2(30),
STANDARD_TXN_IND    VARCHAR2(30),
EMAIL_CONFIRMATION_IND VARCHAR2(30),
EMAIL_TEMPLATE_NAME VARCHAR2(100),
TXN_DATE             DATE,
COMMENTS             VARCHAR2(2000),
BMT_ID               NUMBER,
BMT_AAD_ID           NUMBER,
ACC_PRODUCT_TYPE_CD VARCHAR2(30),
ACC_FUNDING_TYPE_CD VARCHAR2(30),
ACC_MASTER_ACC_IND  VARCHAR2(30),
BMT_STATUS_CD       VARCHAR2(30),
BMT_STATUS           VARCHAR2(80),
POST_TRD             VARCHAR2(30))
```

12.2 Account On Boarding (POST)

Below mentioned table has element name which indicates which type of custom data is passed in that enclosing the name and its value in key name and key value respectively.

Table 12-3 Account On Boarding

Element name	Sub Element	Data Type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Double)

Table 12-3 (Cont.) Account On Boarding

Element name	Sub Element	Data Type
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
<StringData>
  <KeyName>Middle Name</KeyName>
  <KeyValue>Singh</KeyValue>
</StringData>
<NumberData>
  <KeyName>Age</KeyName>
  <KeyValue>25</KeyValue>
</NumberData>
<DateData>
  <KeyName>FiestPmtDate</KeyName>
  <KeyValue>2016-07-14T11:53:40</KeyValue>
</DateData>
</Custom>

```

Below are the package details for Account on Boarding

```

acxprc_ew_100_01.acxprc_ew_100_01 (iv_app_rec IN OUT NOCOPY acx_acc_rec_t,
ov_res_rec IN OUT NOCOPY acx_res_rec_t);
acxprc_em_100_01.process_account (iv_app_rec IN OUT NOCOPY acx_acc_rec_t,
ov_res_rec IN OUT NOCOPY acx_res_rec_t);

```

Lookup Validations- involves the following packages related to

Application details □

```

acxapp_en_100_01.chk_app(iv_app_rec.app_rec,ov_res_rec);

```

Applicant details □

```

acxapl_en_100_01.chk_apl(iv_app_rec.app_apl(j),ov_res_rec);

```

Application Address details □

```

acxapl_en_100_01.chk_apa(iv_app_rec.app_apl(j).apl_apa(k), ov_res_rec);

```

Applicant Employment details □

```

acxapl_en_100_01.chk_ape(iv_app_rec.app_apl(j).apl_ape(k), ov_res_rec);

```

Applicant Telecoms details []

```
acxapl_en_100_01.chk_apt(iv_app_rec.app_apl(j).apl_apt(k), ov_res_rec);
```

Applicant Field investigation []

```
acxafi_en_100_01.chk_afi(iv_app_rec.app_apl(j).apl_afi(k), ov_res_rec);
```

Business Applicant details []

```
acxbsd_en_100_01.chk_bsd(iv_app_rec.app_bsd, ov_res_rec);
```

Business Applicant Affiliates []

```
acxbsd_en_100_01.chk_bsl(iv_app_rec.app_bsd.bsd_bsl(i), ov_res_rec);
```

Business Applicant Partners []

```
acxbsd_en_100_01.chk_bsp(iv_app_rec.app_bsd.bsd_bsp(i), ov_res_rec);
```

Business Applicant Address []

```
acxbsd_en_100_01.chk_bsa(iv_app_rec.app_bsd.bsd_bsa(i), ov_res_rec);
```

Business Applicant Telecoms []

```
acxbsd_en_100_01.chk_bst(iv_app_rec.app_bsd.bsd_bst(i), ov_res_rec);
```

Assets []

```
acxase_en_100_01.chk_ase(iv_app_rec.app_ase(i), ov_res_rec, lv_axn_rec);
```

Asset Valuation []

```
acxase_en_100_01.chk_avl(iv_app_rec.app_ase(i).ase_avl(j), ov_res_rec);
```

Asset Attributes []

```
acxase_en_100_01.chk_atr(iv_app_rec.app_ase(i).ase_avl(j).atr(k), ov_res_rec);
```

Asset Tracking []

```
acxase_en_100_01.chk_atk(iv_app_rec.app_ase(i).ase_atk(j), ov_res_rec);
```

Seller details []

```
acxsdi_en_100_01.chk_sdi(iv_app_rec.app_sdi(i), ov_res_rec);
```

Seller details Address []

```
acxsdi_en_100_01.chk_sda(iv_app_rec.app_sdi(i).sda(j),ov_res_rec);
```

Contract details []

```
acxsdi_en_100_01.chk_sda(iv_app_rec.app_sdi(i).sda(j),ov_res_rec);
```

Repayment Change Schedule []

```
acxcon_en_100_01.chk_acs(iv_app_rec.app_con.con_rpmt.app_acs(i),ov_res_rec);
```

Trade In []

```
acxcon_en_100_01.chk_apd(iv_app_rec.app_con.con_apd(i), ov_res_rec)
```

Subvention []

```
acxcon_en_100_01.chk_asn(iv_app_rec.app_con.con_asn(i), ov_res_rec)
```

ACH []

```
acxcon_en_100_01.chk_aac(iv_app_rec.app_con.con_aac(i), ov_res_rec);
```

PDC []

```
acxcon_en_100_01.chk_pdc(iv_app_rec.app_con.con_pdc(i), ov_res_rec);
```

References []

```
acxcon_en_100_01.chk_aar(iv_app_rec.app_con.con_aar(i), ov_res_rec);
```

Insurances []

```
acxcon_en_100_01.chk_ins(iv_app_rec.app_con.con_ins(i), ov_res_rec);
```

Repayment Options []

```
acxcon_en_100_01.chk_aro(iv_app_rec.app_con.con_rpmt.app_aro, ov_res_rec);
```

Recourse []

```
acxcon_en_100_01.chk_recourse(iv_app_rec.app_con.con_rec,ov_res_rec);
```

Deriving the product data []

```
acxsel_el_100_01.select_product( iv_app_rec.app_rec.app_prd_product,lv_prd_rec  
);
```


Deriving the contract data

```
acxsel_en_111_01.sel_con_dtls ( iv_app_rec.app_con, lv_prd_rec);  
acxsel_en_112_01.sel_con_dtls ( iv_app_rec.app_con, lv_prd_rec);  
acxsel_en_121_01.sel_con_dtls ( iv_app_rec.app_con, lv_prd_rec);
```

Insertion in to the iTables

```
acxins_en_100_01.ins(iv_app_rec, ov_res_rec, lv_axn_rec,iv_app_rec.str_attr ,  
iv_app_rec.num_attr,iv_app_rec.date_attr);
```

Edits Validation

```
aceval_ew_100_01.aceval_ew_100_01(lv_app_rec ,lv_edi_rec);
```

Account Creation

```
acraai_ew_100_01.acraai_ew_100_01(lv_con_rec);
```

Any error occurs in the process

```
acxprc_el_100_01.insert_error(ov_res_rec , lv_axn_rec);
```

You can do the customization on the following packages

```
acxprc_em_100_01
```

BEFORE:

```
acxprc_em_100_01.CV_PROCESS_ACCOUNT_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
    acxprc_ex_100_01.process_account_bfr(iv_app_rec ,ov_res_rec);
```

REPLACE:

```
acxprc_ex_100_01.CV_PROCESS_ACCOUNT_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
    acxprc_ex_100_01.process_account_bfr(iv_app_rec ,ov_res_rec);
```

AFTER:

```
acxprc_ex_100_01.CV_PROCESS_ACCOUNT_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
    acxprc_ex_100_01.process_account_afr(iv_app_rec,ov_res_rec);
```

acxins_en_100_01

For this package, all the procedures are having the before, replace and after exit points:-

The procedures are:-

Main procedure that calls other procedures to insert the payload data:-

```
ins(iv_app_rec IN OUT NOCOPY acx_acc_rec_t ,ov_res_rec IN OUT NOCOPY  
acx_res_rec_t ,iv_axn_rec IN acx_axn_evw%ROWTYPE ,iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t ,iv_ext_rec_num IN OUT NOCOPY  
xws_att_num_tab_t ,iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application details:-

```
insert_app(iv_app_rec IN OUT NOCOPY acx_app_rec_t,iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the applicant details:-

```
insert_apl(iv_apl_rec IN OUT NOCOPY acx_apl_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the applicant telecoms details:-

```
insert_apl(iv_apl_rec IN OUT NOCOPY acx_apl_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the applicant address details:-

```
insert_apa(iv_apa_rec IN OUT NOCOPY acx_apa_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the applicant employment details:-

```
insert_ape(iv_ape_rec IN OUT NOCOPY acx_ape_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the applicant tracking details:-

```
insert_alt(iv_alt_rec IN OUT NOCOPY acx_alt_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the business details details:-

```
insert_bsd(iv_bsd_rec IN OUT NOCOPY acx_bsd_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the business affiliates details:-

```
insert_bsl(iv_bsl_rec IN OUT NOCOPY acx_bsl_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the business partners details:-

```
insert_bsp(iv_bsp_rec IN OUT NOCOPY acx_bsp_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the business address details:-

```
insert_bsa(iv_bsa_rec IN OUT NOCOPY acx_bsa_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the business telecoms details:-

```
insert_bst(iv_bst_rec IN OUT NOCOPY acx_bst_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application assets details:-

```
insert_ase(iv_ase_rec IN OUT NOCOPY acx_ase_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application asset valuations details:-

```
insert_avl(iv_avl_rec IN OUT NOCOPY acx_avl_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application asset attributes details:-

```
insert_atr(iv_atr_rec IN OUT NOCOPY acx_atr_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application asset tracking details:-

```
insert_atk(iv_atk_rec IN OUT NOCOPY acx_atk_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application asset tracking attribute details:-

```
insert_ata(iv_ata_rec IN OUT NOCOPY acx_ata_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application seller details:-

```
insert_sdi(iv_sdi_rec IN OUT NOCOPY acx_sdi_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application seller address details:-

```
insert_sda(iv_sda_rec IN OUT NOCOPY acx_sda_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application repayment schedule details:-

```
insert_apc(iv_apc_rec IN OUT NOCOPY acx_apc_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application payment change schedule details:-

```
insert_acs(iv_acs_rec IN OUT NOCOPY acx_acs_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application contract details:-

```
insert_acd(iv_acd_rec IN OUT NOCOPY acx_acd_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application contract insurance details:-

```
insert_acd_ins(iv_app_rec IN OUT NOCOPY acx_acc_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application insurance details:-

```
insert_ins(iv_ins_rec IN OUT NOCOPY acx_ins_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application tradein details:-

```
insert_apd(iv_apd_rec IN OUT NOCOPY acx_apd_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application subvention details:-

```
insert_asn(iv_asn_rec IN OUT NOCOPY acx_asn_rec_t, ov_res_rec IN OUT NOCOPY  
acx_res_rec_t, iv_ext_rec_str IN OUT NOCOPY xws_att_str_tab_t, iv_ext_rec_num  
IN OUT NOCOPY xws_att_num_tab_t, iv_ext_rec_dt IN OUT NOCOPY  
xws_att_date_tab_t);
```

Procedure to insert the application subvention_details details:-

```
insert_asl(iv_asl_rec IN OUT NOCOPY acx_asl_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application ach details:-

```
insert_aac(iv_aac_rec IN OUT NOCOPY acx_aac_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application post dated check details:-

```
insert_pdc(iv_pdc_rec IN OUT NOCOPY acx_pdc_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application references details:-

```
insert_aar(iv_aar_rec IN OUT NOCOPY acx_aar_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application contract fees details:-

```
insert_afe(iv_afe_rec IN OUT NOCOPY acx_afe_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the application comment details:-

```
insert_acm(iv_acm_rec IN OUT NOCOPY acx_acm_rec_t, iv_ext_rec_str IN OUT  
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,  
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to update the application details:-

```
update_con(iv_con_rec IN OUT NOCOPY acx_con_rec_t, iv_aro_rec IN OUT NOCOPY
acx_aro_rec_t, iv_ext_rec_str IN OUT NOCOPY xws_att_str_tab_t, iv_ext_rec_num
IN OUT NOCOPY xws_att_num_tab_t, iv_ext_rec_dt IN OUT NOCOPY
xws_att_date_tab_t);
```

Procedure to insert the application tracking attribute details:-

```
insert_aat(iv_aat_rec IN OUT NOCOPY acx_aat_rec_t, iv_ext_rec_str IN OUT
NOCOPY xws_att_str_tab_t, iv_ext_rec_num IN OUT NOCOPY xws_att_num_tab_t,
iv_ext_rec_dt IN OUT NOCOPY xws_att_date_tab_t);
```

Procedure to insert the applicant field investigation details:-

```
insert_afd(iv_afd_rec IN OUT NOCOPY acx_afd_rec_t);
```

Procedure to insert the applicant field investigation_details details:-

```
insert_afd(iv_afd_rec IN OUT NOCOPY acx_afd_rec_t);
```

acxsel_en_111_01 (LOAN)

BEFORE

```
acxsel_ex_111_01.CV_SEL_CON_DTLS_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
    acxsel_ex_111_01.sel_con_dtls_bfr (iv_con_rec,iv_prd_rec);
```

REPLACE

```
acxsel_ex_111_01.CV_SEL_CON_DTLS_REP = cmncon_cl_000_01.CUSTOMIZED THEN
    acxsel_ex_111_01.sel_con_dtls_rep (iv_con_rec,iv_prd_rec);
```

AFTER

```
acxsel_ex_111_01.CV_SEL_CON_DTLS_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
    acxsel_ex_111_01.sel_con_dtls_afr (iv_con_rec,iv_prd_rec);
```

Similarly the exit points have been added for the line and lease products also for selecting contract details in acxsel_en_112_01 and acxsel_en_121_01

Acraai_en_111_01 (LOAN)

BEFORE

```
acraai_ex_111_01.CV_CREATE_ACCOUNT_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
    acraai_ex_111_01.create_account_bfr (iv_con_rec);
```

REPLACE

```
acraai_ex_111_01.CV_CREATE_ACCOUNT_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
    acraai_ex_111_01.create_account_rep (iv_con_rec);
```

AFTER

```
acraai_ex_111_01.CV_CREATE_ACCOUNT_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
    acraai_ex_111_01.create_account_afr (iv_con_rec);
```

BEFORE

```
acraai_ex_111_01.CV_SET_ACC_NBR_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
    acraai_ex_111_01.set_acc_nbr_bfr (iv_con_rec,iv_acc_aad_id,iv_acc_nbr);
```

REPLACE

```
acraai_ex_111_01.CV_SET_ACC_NBR_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
    acraai_ex_111_01.set_acc_nbr_rep (iv_con_rec,iv_acc_aad_id,iv_acc_nbr);
```

AFTER

```
acraai_ex_111_01.CV_SET_ACC_NBR_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
    acraai_ex_111_01.set_acc_nbr_afr (iv_con_rec,iv_acc_aad_id,iv_acc_nbr);
```

acraai_en_111_02

BEFORE

```
acraai_ex_111_01.CV_LOAD_CURRENT_ACC_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
    acraai_ex_111_01.load_current_acc_bfr (iv_con_rec,iv_acc_aad_id);
```

REPLACE

```
acraai_ex_111_01.CV_LOAD_CURRENT_ACC_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
    acraai_ex_111_01.load_current_acc_rep (iv_con_rec,iv_acc_aad_id);
```

AFTER

```
acraai_ex_111_01.CV_LOAD_CURRENT_ACC_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
    acraai_ex_111_01.load_current_acc_afr (iv_con_rec,iv_acc_aad_id);
```

BEFORE

```
acraai_ex_111_01.cv_convert_new_acc_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
    acraai_ex_111_01.convert_new_acc_bfr (iv_con_rec,iv_acc_aad_id);
```

REPLACE

```
acraai_ex_111_01.cv_convert_new_acc_rep = cmncon_cl_000_01.CUSTOMIZED THEN
  acraai_ex_111_01.convert_new_acc_rep (iv_con_rec,iv_acc_aad_id);
```

AFTER

```
acraai_ex_111_01.cv_convert_new_acc_afr = cmncon_cl_000_01.CUSTOMIZED THEN
  acraai_ex_111_01.convert_new_acc_afr (iv_con_rec,iv_acc_aad_id);
```

Similarly the exit points have been added for the line and lease products also for inserting the account details in acraai_en_112_01,acraai_en_112_02 and acxsel_en_121_01,acraai_en_121_02.

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT
(
  ext_key_name VARCHAR2(30),
  ext_key_value DATE
);
/
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT
(
  ext_key_name VARCHAR2(30),
  ext_key_value NUMBER
);
/
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT
(
  ext_key_name VARCHAR2(30),
  ext_key_value VARCHAR2(4000)
);
/
```

12.3 Payment Posting (POST)

Below mentioned table has element name which indicates which type of custom data is passed in that enclosing the name and its value in keyname and keyvalue respectively.

Table 12-4 Payment Posting

Element name	Sub Element	Data Type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Double)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  <CustomUserDefinedStringData>
    <KeyName>Middle Name</KeyName>
    <KeyValue>Singh</KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName>Age</KeyName>
    <KeyValue>25</KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName>FirstPmtDate</KeyName>
    <KeyValue>2016-07-14T11:53:40</KeyValue>
  </CustomUserDefinedDateData>
</Custom>

```

Below are the package details for Payment posting

```

xftpmt_ew_100_01. xftpmt_ew_100_01 ( iv_pmt_axn_rec IN xbt_pmt_axn_rec_t,
iv_pmt_axn_result_tab_t OUT NOCOPY xbt_pmt_result_tab_t)
xftpmt_em_100_01.post_pmt ( iv_pmt_axn_rec IN xbt_pmt_axn_rec_t,
iv_pmt_axn_result_tab_t OUT NOCOPY xbt_pmt_result_tab_t)

```

You can do the customization on xftpmt_em_100_01.post_pmt ();

Below are the exit points:-

BEFORE:

```

xftpmt_ex_100_01.cv_post_pmt_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
xftpmt_ex_100_01.post_pmt_bfr ( iv_pmt_axn_rec IN xbt_pmt_axn_rec_t,
iv_pmt_axn_result_tab_t OUT NOCOPY xbt_pmt_result_tab_t)

```

REPLACE

```

xftpmt_ex_100_01.cv_post_pmt_rep = cmncon_cl_000_01.CUSTOMIZED THEN
xftpmt_ex_100_01.post_pmt_rep ( iv_pmt_axn_rec IN xbt_pmt_axn_rec_t,
iv_pmt_axn_result_tab_t OUT NOCOPY xbt_pmt_result_tab_t)

```

AFTER :-

```

xftpmt_ex_100_01.cv_post_pmt_afr = cmncon_cl_000_01.CUSTOMIZED THEN
xftpmt_ex_100_01.post_pmt_afr ( iv_pmt_axn_rec IN xbt_pmt_axn_rec_t,
iv_pmt_axn_result_tab_t OUT NOCOPY xbt_pmt_result_tab_t)

```

Extensible parameters are Tab Type object

```

CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT
(
  ATT_NAME          VARCHAR2 (30)
  , ATT_VALUE       DATE);
/

```

```

CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT
(
  ATT_NAME          VARCHAR2 (30)
  , ATT_VALUE       NUMBER);
);
/
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT
(
  ATT_NAME          VARCHAR2 (30)
  , ATT_VALUE       VARCHAR2 (4000));
);
/

```

12.4 Account Detail (GET)

Below mentioned table has element name which indicates which type of custom data is passed in that enclosing the name and its value in name and value respectively.

Table 12-5 Account Detail

Element name	Sub Element	Data Type
CustomerAttributes	Name	String
	Value	String
CustomElements	Name	String
	CustomData	
CustomData	Uniqueld	Number (Double)
	CustomAttributes	
CustomAttributes	Name	String
	Value	String
AccountAttributes	Name	String
	Value	String
AddressAttributes	Name	String
	Value	String
BusinessAttributes	Name	String
	Value	String
TelecomAttributes	Name	String
	Value	String
AffiliateAttributes	Name	String
	Value	String
PartnersAttributes	Name	String
	Value	String
BusinessAddressAttributes	Name	String
	Value	String
TransactionAttributes	Name	String
	Value	String
AllocationsAttributes	Name	String
	Value	String
CollateralAttributes	Name	String

Table 12-5 (Cont.) Account Detail

Element name	Sub Element	Data Type
	Value	String
DetailsAttributes	Name	String
	Value	String
StatementAttributes	Name	String
	Value	String
StatementMessagesAttributes	Name	String
	Value	String
AchAttributes	Name	String
	Value	String
CreditCardAttributes	Name	String
	Value	String

Sample XML

```
<CustomerDetails>
  <CustomerID>1001</CustomerID>
  <FirstName>METRO</FirstName>
  <LastName>FILE</LastName>
  <CustomerAttributes>
    <name>Last Name</name>
    <value>kulkarni</value>
  </CustomerAttributes>
</CustomerDetails>
```

Below are the package details for Account Details

```
xcsprc_ew_100_01( iv_usr_code IN VARCHAR2,
                 iv_acc_nbr IN accounts.acc_nbr%TYPE,
                 iv_action IN VARCHAR2,
                 iv_acc IN OUT NOCOPY xcs_acc_rec_t
               )
xcsprc_em_100_01.get_account_information (iv_usr_code IN VARCHAR2,
                 iv_acc_nbr IN accounts.acc_nbr%TYPE,
                 iv_action IN VARCHAR2,
                 iv_acc IN OUT NOCOPY xcs_acc_rec_t
               )
xcsacc_en_100_01.get_account_details(iv_acc_nbr IN accounts.acc_nbr%TYPE,
                 iv_action IN VARCHAR2,
                 iv_acc IN OUT NOCOPY xcs_acc_rec_t
               )
xcsacc_el_100_01.populate_account_response(iv_acc IN OUT NOCOPY xcs_acc_rec_t)
```

You can do the customization on `xcsacc_en_100_01.get_account_details()`;

BEFORE:-

```
IF xcsprc_ex_100_01.cv_get_account_details_bfr = cmncon_cl_000_01.CUSTOMIZED
THEN
```

```

        xcsprc_ex_100_01.get_account_details_bfr(iv_acc_nbr, iv_action, iv_acc);
    END IF;

```

REPLACE

```

IF xcsprc_ex_100_01.cv_get_account_details_rep = cmncon_cl_000_01.CUSTOMIZED
THEN
    xcsprc_ex_100_01.get_account_details_rep(iv_acc_nbr, iv_action, iv_acc);
ELSE

```

AFTER

```

IF xcsprc_ex_100_01.cv_get_account_details_afr = cmncon_cl_000_01.CUSTOMIZED
THEN
    xcsprc_ex_100_01.get_account_details_afr (iv_acc_nbr, iv_action, iv_acc);
END IF;

```

You can do the customization on `xcsprc_em_100_01.get_account_information ()`;

BEFORE

```

IF xcsprc_ex_100_01.cv_get_account_information_bfr =
cmncon_cl_000_01.CUSTOMIZED THEN
    xcsprc_ex_100_01.get_account_information_bfr (iv_acc_nbr, iv_action,
iv_acc);
END IF;

```

REPLACE

```

IF xcsprc_ex_100_01.cv_get_account_information_rep =
cmncon_cl_000_01.CUSTOMIZED THEN
    xcsprc_ex_100_01.get_account_information_rep (iv_acc_nbr, iv_action,
iv_acc);
ELSE

```

AFTER

```

IF xcsprc_ex_100_01.cv_get_account_information_afr =
cmncon_cl_000_01.CUSTOMIZED THEN
    xcsprc_ex_100_01.get_account_information_afr (iv_acc_nbr, iv_action,
iv_acc);
END IF;

```

Extensible parameters are tab type object

```

xws_custom_rec_t AS OBJECT(
    TAB_NAME      VARCHAR2(80),
    TAB_DATA      xws_custom_tab2_t);
xws_custom_rec2_t AS OBJECT(
    CUSTOM_ID     NUMBER,
    CUSTOM_DATA   xws_att_str_tab_t);
xws_att_str_rec_t AS OBJECT (

```

```
ATT_NAME      VARCHAR2 (30)
, ATT_VALUE   VARCHAR2 (4000);
```

12.5 Scenario Analysis (POST)

Below mentioned table has element name which indicates which type of custom data is passed by enclosing the name and its value in keyname and keyvalue respectively.

Table 12-6 Scenario Analysis

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Double)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <CustomUserDefinedStringData>
    <KeyName>Middle Name</KeyName>
    <KeyValue>Oracle</KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName>Age</KeyName>
    <KeyValue>27</KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName>FirstPmtDate</KeyName>
    <KeyValue>2016-07-14</KeyValue>
  </CustomUserDefinedDateData>
</Custom>
```

Sample JSON

```
{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "Middle Name",
      "KeyValue": "Oracle"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "Age",
      "KeyValue": "27"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "FirstPmtDate",
      "KeyValue": "2016-07-14"
    }
  }
}
```

```

    }
}

```

Below are the package details for scenario analysis web service

Engine wrapper package:

```

xsaprc_ew_100_01.xsaprc_ew_100_01(iv_xsa_asa_rec IN OUT xae_cal_rec_t,
iv_calculator_type IN VARCHAR2,
iv_action          IN VARCHAR2,
iv_result          OUT VARCHAR2,
iv_err_desc        OUT VARCHAR2)

```

Wrapper package is used for two services (POST & GET), both services differentiated by `iv_action`.

Engine main packages:

POST:

```

xsaprc_em_100_01.create_asa( iv_xsa_asa_rec IN OUT xae_cal_rec_t,
iv_calculator_type          IN VARCHAR2,
iv_result                   OUT VARCHAR2,
iv_err_desc                  OUT VARCHAR2)

```

Below are the exit points for post service

BEFORE:

```

xsaprc_ex_100_01.post_prc_bfr ( iv_xsa_asa_rec          IN OUT xae_cal_rec_t,
iv_calculator_type IN VARCHAR2,
iv_result           OUT VARCHAR2,
iv_err_desc         OUT VARCHAR2)

```

REPLACE:

```

xsaprc_ex_100_01.post_prc_rep ( iv_xsa_asa_rec          IN OUT xae_cal_rec_t,
iv_calculator_type IN VARCHAR2,
iv_result           OUT VARCHAR2,
iv_err_desc         OUT VARCHAR2)

```

AFTER:

```

xsaprc_ex_100_01.post_prc_afr ( iv_xsa_asa_rec          IN OUT xae_cal_rec_t,
iv_calculator_type IN VARCHAR2,
iv_result           OUT VARCHAR2,
iv_err_desc         OUT VARCHAR2)

```

GET:

```

xsaprc_em_100_01.get_asa( iv_xsa_asa_rec IN OUT xae_cal_rec_t,
iv_result                   OUT VARCHAR2,
iv_err_desc                  OUT VARCHAR2)

```

Below are the exit points for get service**BEFORE:**

```
xsaprc_ex_100_01.get_prc_bfr ( iv_xsa_asa_rec IN OUT  xae_cal_rec_t,  
iv_result      OUT VARCHAR2,  
iv_err_desc    OUT VARCHAR2)
```

REPLACE :

```
xsaprc_ex_100_01.get_prc_rep ( iv_xsa_asa_rec IN OUT  xae_cal_rec_t,  
iv_result      OUT VARCHAR2,  
iv_err_desc    OUT VARCHAR2)
```

AFTER:

```
xsaprc_ex_100_01.get_prc_afr ( iv_xsa_asa_rec IN OUT  xae_cal_rec_t,  
iv_result      OUT VARCHAR2,  
iv_err_desc    OUT VARCHAR2)
```

IN parameters:

```
xae_cal_rec_t --- Rec Type Object  
iv_calculator_type --- VARCHAR2  
iv_action --- VARCHAR2
```

OUT parameters:

```
xae_cal_rec_t --- Rec Type Object  
iv_result --- VARCHAR2,  
iv_err_desc --- VARCHAR2
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT  
(  
ATT_NAME          VARCHAR2 (30)  
ATT_VALUE         DATE);  
/  
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT  
(  
ATT_NAME          VARCHAR2 (30)  
, ATT_VALUE      NUMBER);  
);  
/  
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT  
(  
ATT_NAME          VARCHAR2 (30)  
, ATT_VALUE      VARCHAR2 (4000));  
);  
/
```

12.6 Lookups (GET)

Below mentioned table has element name which indicates which type of custom data is passed by enclosing the name and its value in keyname and keyvalue respectively.

Table 12-7 Lookups

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Double)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <CustomUserDefinedStringData>
    <KeyName>StringName</KeyName>
    <KeyValue>StringValue</KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName>NumberName</KeyName>
    <KeyValue>NumberValue</KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName>DateName</KeyName>
    <KeyValue>DateValue</KeyValue>
  </CustomUserDefinedDateData>
</Custom>
```

Sample JSON

```
{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  }
}
```


Below are the package details for Lookup web service

```

xlkprc_ew_100_01.xlkprc_ew_100_01      ( iv_req_rec_t IN
xlk_req_rec_t,iv_res_rec_t IN OUT NOCOPY xlk_resp_rec_t)
xlkprc_em_100_01.xlkprc_em_100_01 (iv_req_rec_t IN
xlk_req_rec_t,iv_res_rec_t IN OUT xlk_resp_rec_t)
xlkprc_em_100_01.get_lookup ( iv_lkt_type IN      VARCHAR2,iv_res_rec_t IN
OUT      XLK_TYPE_REC_T)

```

You can do the customization on xlkprc_em_100_01.xlkprc_em_100_01();**BEFORE:-**

```

IF xlkprc_ex_100_01.cv_xlkprc_em_100_01_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
    xlkprc_ex_100_01.xlkprc_em_100_01_bfr (iv_req_rec_t, iv_res_rec_t);
END IF;

```

REPLACE:-

```

IF xlkprc_ex_100_01.cv_xlkprc_em_100_01_rep = cmncon_cl_000_01.CUSTOMIZED THEN
    xlkprc_ex_100_01.xlkprc_em_100_01_rep (iv_req_rec_t, iv_res_rec_t);
ELSE

```

AFTER:-

```

IF xlkprc_ex_100_01.cv_xlkprc_em_100_01_afr = cmncon_cl_000_01.CUSTOMIZED THEN
    xlkprc_ex_100_01.xlkprc_em_100_01_afr (iv_req_rec_t, iv_res_rec_t);
END IF;

```

You can also do the customization on xlkprc_em_100_01.get_lookup();**BEFORE:-**

```

IF xlkprc_ex_100_01.cv_get_lookup_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
    xlkprc_ex_100_01.get_lookup_bfr (iv_lkt_type, iv_res_rec_t);
END IF;

```

REPLACE:-

```

IF xlkprc_ex_100_01.cv_get_lookup_rep = cmncon_cl_000_01.CUSTOMIZED THEN
    xlkprc_ex_100_01.get_lookup_rep (iv_lkt_type, iv_res_rec_t);
ELSE

```

AFTER:-

```

IF xlkprc_ex_100_01.cv_get_lookup_afr = cmncon_cl_000_01.CUSTOMIZED THEN
    xlkprc_ex_100_01.get_lookup_afr (iv_lkt_type, iv_res_rec_t);
END IF;

```

Extensible parameters are rec type object

```
xws_att_str_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30)
  , ATT_VALUE    VARCHAR2(4000));
xws_att_num_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30)
  , ATT_VALUE    NUMBER);
xws_att_date_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30)
  , ATT_VALUE    DATE);
```

12.7 Application Search (GET)

Below mentioned table has element name which indicates In response which type of custom data is passed by enclosing the name and its value in KeyName and KeyValue respectively.

Table 12-8 Application Search

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Double)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample Response XML

```
<Custom>
  <CustomUserDefinedStringData>
    <KeyName>Middle Name</KeyName>
    <KeyValue>Oracle</KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName>Age</KeyName>
    <KeyValue>27</KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName>FirstPmtDate</KeyName>
    <KeyValue>2016-07-14</KeyValue>
  </CustomUserDefinedDateData>
</Custom>
```

Sample Response JSON

```
{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "Middle Name",
      "KeyValue": "Oracle"
    },
  },
}
```

```

        "CustomUserDefinedNumberData": {
            "KeyName": "Age",
            "KeyValue": "27"
        },
        "CustomUserDefinedDateData": {
            "KeyName": "FirstPmtDate",
            "KeyValue": "2016-07-14"
        }
    }
}
}

```

Below are the package details for ApplicationSearch web service

```

xaeque_ew_100_01 (iv_que_rec      IN      xae_que_rec_t, iv_response_tab OUT
NOCOPY xae_que_resp_tab_t)
xaeque_em_100_01.get_application_summary (iv_que_rec IN
xae_que_rec_t, iv_response_tab OUT NOCOPY xae_que_resp_tab_t)
xaeque_en_100_01.get_apl_search_summary(iv_que_rec IN
xae_que_rec_t, iv_response_tab OUT NOCOPY xae_que_resp_tab_t)

```

You can do the customization on xaeque_en_100_01.get_apl_search_summary();

BEFORE:-

```

IF xaeque_ex_100_01.cv_get_apl_search_summary_bfr =
cmncon_cl_000_01.CUSTOMIZED THEN
    xaeque_ex_100_01.get_apl_search_summary_bfr (iv_que_rec, iv_response_tab);
END IF;

```

REPLACE:-

```

IF xaeque_ex_100_01.cv_get_apl_search_summary_rep =
cmncon_cl_000_01.CUSTOMIZED THEN
    xaeque_ex_100_01.get_apl_search_summary_rep (iv_que_rec, iv_response_tab);
ELSE

```

AFTER:-

```

IF xaeque_ex_100_01.cv_get_apl_search_summary_afr =
cmncon_cl_000_01.CUSTOMIZED THEN
    xaeque_ex_100_01.get_apl_search_summary_afr (iv_que_rec, iv_response_tab);
END IF;

```

You can also do the customization on xaeque_em_100_01.get_application_summary();

BEFORE:-

```

IF xaeque_ex_100_01.cv_get_application_summary_bfr =
cmncon_cl_000_01.CUSTOMIZED THEN
    xaeque_ex_100_01.get_application_summary_bfr (iv_que_rec,
iv_response_tab);
END IF;

```

REPLACE:-

```

IF xaeque_ex_100_01.cv_get_application_summary_rep =
cmncon_cl_000_01.CUSTOMIZED THEN
    xaeque_ex_100_01.get_application_summary_rep (iv_que_rec,
iv_response_tab);
ELSE

```

AFTER:-

```

IF xaeque_ex_100_01.cv_get_application_summary_afr =
cmncon_cl_000_01.CUSTOMIZED THEN
    xaeque_ex_100_01.get_application_summary_afr (iv_que_rec,
iv_response_tab);
END IF;

```

Extensible parameters are rec type objects

```

xws_att_str_rec_t AS OBJECT (
    ATT_NAME          VARCHAR2(30)
    , ATT_VALUE       VARCHAR2(4000));
xws_att_num_rec_t AS OBJECT (
    ATT_NAME          VARCHAR2(30)
    , ATT_VALUE       NUMBER);
xws_att_date_rec_t AS OBJECT (
    ATT_NAME          VARCHAR2(30)
    , ATT_VALUE       DATE);

```

12.8 Calculator (POST)

Below mentioned table has element name which indicates which type of custom data is passed by enclosing the name and its value in KeyName and KeyValue respectively.

Table 12-9 Calculator

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Double)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample Input/Output XML

```

<Custom>
  <CustomUserDefinedStringData>
    <KeyName>First Name</KeyName>
    <KeyValue>ABC</KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedStringData>

```

```

        <KeyName>Last Name</KeyName>
        <KeyValue>DEF</KeyValue>
    </CustomUserDefinedStringData>
    <CustomUserDefinedNumberData>
        <KeyName>Phone Number</KeyName>
        <KeyValue>123456987</KeyValue>
    </CustomUserDefinedNumberData>
    <CustomUserDefinedNumberData>
        <KeyName>Age</KeyName>
        <KeyValue>27</KeyValue>
    </CustomUserDefinedNumberData>
    <CustomUserDefinedDateData>
        <KeyName>BirthDate</KeyName>
        <KeyValue>2017-02-24T15:00:43+05:30</KeyValue>
    </CustomUserDefinedDateData>
    <CustomUserDefinedDateData>
        <KeyName>ContractDate</KeyName>
        <KeyValue>2017-02-24T15:00:51+05:30</KeyValue>
    </CustomUserDefinedDateData>
</Custom>

```

Sample Input/Output JSON

```

"Custom" : {
  "CustomUserDefinedStringData" : [
    {
      "KeyName" : "First Name",
      "KeyValue" : "Test"
    },
    {
      "KeyName" : "Last Name",
      "KeyValue" : "ABC"
    }
  ],
  "CustomUserDefinedNumberData" : [
    {
      "KeyName" : "Age",
      "KeyValue" : "26"
    },
    {
      "KeyName" : "Phone Number",
      "KeyValue" : "123456987"
    }
  ],
  "CustomUserDefinedDateData" : [
    {
      "KeyName" : "BirthDate",
      "KeyValue" : "2017-02-24T14:52:44+05:30"
    },
    {
      "KeyName" : "ContractDate",
      "KeyValue" : "2017-02-24T14:54:22+05:30"
    }
  ]
}

```

Below are the package details for Calculator web service**Wrapper package:**

```

xaecal_ew_100_01.xaecal_ew_100_01 (iv_cal_rec IN OUT NOCOPY xae_cal_rec_t,
iv_calculator_type IN VARCHAR2,
iv_result      OUT NUMBER,
iv_err_desc    OUT VARCHAR2).

```

Main engine package has been modified:

```

xaecal_em_100_01. xaecal_em_100_01 (iv_cal_rec IN OUT NOCOPY xae_cal_rec_t,
iv_calculator_type IN VARCHAR2,
iv_result      OUT NUMBER,
iv_err_desc    OUT VARCHAR2).

```

Below are the exit point packages added.**BEFORE:-**

```

xaecal_ex_100_01.CV_CAL_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaecal_ex_100_01.cal_bfr(iv_cal_rec ,iv_calculator_type ,iv_result ,iv_err_desc);

```

REPLACE:-

```

xaecal_ex_100_01.CV_CAL_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaecal_ex_100_01.cal_rep(iv_cal_rec ,iv_calculator_type ,iv_result ,iv_err_desc);

```

AFTER:-

```

xaecal_ex_100_01.CV_CAL_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaecal_ex_100_01.cal_afr(iv_cal_rec ,iv_calculator_type ,iv_result ,iv_err_desc);

```

Extensible parameters are Tab Type object

```

CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT
(
  ATT_NAME      VARCHAR2 (30)
  , ATT_VALUE    DATE);
/
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT
(
  ATT_NAME      VARCHAR2 (30)
  , ATT_VALUE    NUMBER);
);
/
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT
(
  ATT_NAME      VARCHAR2 (30)
  , ATT_VALUE    VARCHAR2 (4000));

```

```
);  
/
```

12.9 Application Entry (POST)

Below mentioned table has element name which indicates which type of custom data is passed in that enclosing the name and its value in keyname and keyvalue respectively.



Note:

This block is unbounded and is part of other blocks . For Example: Address Block.

Table 12-10 Application Entry

Element name	Sub Element	Data Type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Double)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML FORMAT

```
<CustomFields>  
  <CustomUserDefinedStringData>  
    <KeyName>KEY_STRING</KeyName>  
    <KeyValue>VALUE_STRING</KeyValue>  
  </CustomUserDefinedStringData>  
  <CustomNumberDataTypes>  
    <KeyName>KEY_NUMBER</KeyName>  
    <KeyValue>VALUE_NUMBER</KeyValue>  
  </CustomNumberDataTypes>  
  <CustomDateDataTypes>  
    <KeyName>KEY_DATE</KeyName>  
    <KeyValue>VALUE_DATE</KeyValue>  
  </CustomDateDataTypes>  
</CustomFields>
```

Below are the package details for Calculator web service

Wrapper engine:-

```
xaeprc_ew_100_02. xaeprc_ew_100_02  
( iv_app_rec IN OUT NOCOPY xae_app_rest_rec_t  
,ov_error_rec IN OUT NOCOPY xae_error_rec_t)
```

Main Engine:-

```
xaeprc_em_100_02.submit ( iv_app_rec IN OUT NOCOPY xae_app_rest_rec_t
,ov_error_rec IN OUT NOCOPY xae_error_rec_t)
```

Lookup Validations- involves the following packages related to

```
Applicant -> xaeapl_en_100_02
Applicant Address -> xaeapa_en_100_02
Applicant Employment -> xaeape_en_100_02
Applicant Telecom -> xaeapt_en_100_02
Applicant Financial -> xaeapf_en_100_02
Applicant liability -> xaeapb_en_100_02
Applicant other income information -> xaeapi_en_100_02
Decision trade in -> xaeapd_en_100_02
Collateral valuation -> xaeavl_en_100_02
Business details -> xaebds_en_100_02
Business address -> xaebsa_en_100_02
Business partners -> xaebsp_en_100_02
Business affiliates -> xaebsl_en_100_02
Business telecom -> xaebst_en_100_02
Business financials -> xaebsf_en_100_02
Business liabilities -> xaebsb_en_100_02
```

Below is the application entry main insert package

```
xaeins_en_100_02.ins (iv_app_rec IN OUT NOCOPY
xae_app_rest_rec_t,ov_error_rec IN OUT NOCOPY xae_error_rec_t,iv_axn_rec IN
xae_axn_evw%ROWTYPE)
```

You can do the customization on the following packages**xaeprc_em_100_02.submit****BEFORE:-**

```
xaeprc_ex_100_03.CV_SUBMIT_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeprc_ex_100_03.submit_bfr ( iv_app_rec IN OUT NOCOPY
xae_app_rest_rec_t,ov_error_rec IN OUT NOCOPY xae_error_rec_t)
```

REPLACE:-

```
xaeprc_ex_100_03.CV_SUBMIT_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaeprc_ex_100_03.submit_rep ( iv_app_rec IN OUT NOCOPY
xae_app_rest_rec_t,ov_error_rec IN OUT NOCOPY xae_error_rec_t)
```

AFTER:-

```
xaeprc_ex_100_03.CV_SUBMIT_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeprc_ex_100_03.submit_afr ( iv_app_rec IN OUT NOCOPY
xae_app_rest_rec_t,ov_error_rec IN OUT NOCOPY xae_error_rec_t)
```

xaeins_en_100_02.ins

BEFORE:-

```
xaeprc_ex_100_04.CV_INS_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeprc_ex_100_04.ins_bfr (iv_app_rec IN OUT NOCOPY
xae_app_rest_rec_t,ov_error_rec IN OUT NOCOPY xae_error_rec_t,iv_axn_rec IN
xae_axn_evw%ROWTYPE)
```

REPLACE:-

```
xaeprc_ex_100_04.CV_INS_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaeprc_ex_100_04.ins_rep(iv_app_rec IN OUT NOCOPY
xae_app_rest_rec_t,ov_error_rec IN OUT NOCOPY xae_error_rec_t,iv_axn_rec IN
xae_axn_evw%ROWTYPE)
```

AFTER:-

```
xaeprc_ex_100_04.CV_INS_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeprc_ex_100_04.ins_afr(iv_app_rec IN OUT NOCOPY
xae_app_rest_rec_t,ov_error_rec IN OUT NOCOPY xae_error_rec_t,iv_axn_rec IN
xae_axn_evw%ROWTYPE)
```

```
xaeapp_en_100_02. chk(
    iv_app_rec IN OUT NOCOPY xae_app_rest_rec_t,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapp_ex_100_02.CV_CHK_BFR =cmncon_cl_000_01.CUSTOMIZED THEN
xaeapp_ex_100_02.chk_bfr(iv_app_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapp_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapp_ex_100_02.chk_rep(iv_app_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapl_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapl_ex_100_02.chk_afr(iv_app_rec ,ov_error_rec ,iv_axn_rec);
xaeapl_en_100_02. chk(
    iv_apl_rec IN OUT NOCOPY xae_apl_rest_rec_t,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE);
```

BEFORE:-

```
xaeapl_ex_100_02.CV_CHK_BFR =cmncon_cl_000_01.CUSTOMIZED THEN
xaeapl_ex_100_02.chk_bfr(iv_apl_rec, ov_error_rec, iv_axn_rec);
```

REPLACE:-

```
xaeapl_ex_100_02.CV_CHK_REP =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapl_ex_100_02.chk_rep(iv_apl_rec, ov_error_rec, iv_axn_rec);
```

AFTER:-

```
xaeapl_ex_100_02.CV_CHK_AFR =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapl_ex_100_02.chk_afr(iv_apl_rec, ov_error_rec, iv_axn_rec );  
xaeapa_en_100_02. chk(  
    iv_apa_rec IN OUT NOCOPY xae_apa_rest_rec_t,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapa_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapa_ex_100_02.chk_bfr(iv_apa_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapa_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapa_ex_100_02.chk_rep(iv_apa_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapa_ex_100_02.CV_CHK_AFR =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapa_ex_100_02.chk_afr(iv_apa_rec ,ov_error_rec ,iv_axn_rec);  
xaeape_en_100_02. chk(  
    iv_ape_rec IN OUT NOCOPY xae_ape_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeape_ex_100_02.CV_CHK_BFR =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeape_ex_100_02.chk_bfr(iv_ape_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeape_ex_100_02.CV_CHK_REP =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeape_ex_100_02.chk_rep(iv_ape_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeape_ex_100_02.CV_CHK_AFR =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeape_ex_100_02.chk_afr(iv_ape_rec ,ov_error_rec ,iv_axn_rec);  
xaeapi_en_100_02. chk(  
    iv_api_rec IN OUT NOCOPY xae_api_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapi_ex_100_02.CV_CHK_BFR =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapi_ex_100_02.chk_bfr(iv_api_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapi_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapi_ex_100_02.chk_rep(iv_api_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapi_ex_100_02.CV_CHK_AFR =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapi_ex_100_02.chk_afr(iv_api_rec ,ov_error_rec ,iv_axn_rec);  
xaeapt_en_100_02.chk (  
    iv_apt_rec IN OUT NOCOPY xae_apt_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapt_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapt_ex_100_02.chk_bfr(iv_apt_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapt_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapt_ex_100_02.chk_rep(iv_apt_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapt_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapt_ex_100_02.chk_afr(iv_apt_rec ,ov_error_rec ,iv_axn_rec);  
xaeapf_en_100_02.chk (  
    iv_apf_rec IN OUT NOCOPY xae_apf_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapf_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapf_ex_100_02.chk_bfr(iv_apf_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapf_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapf_ex_100_02.chk_rep(iv_apf_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapf_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapf_ex_100_02.chk_afr(iv_apf_rec ,ov_error_rec ,iv_axn_rec);
xaeapb_en_100_02.chk(
    iv_apb_rec IN OUT NOCOPY xae_apb_rest_rec_t ,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapb_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapb_ex_100_02.chk_bfr(iv_apb_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapb_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapb_ex_100_02.chk_rep(iv_apb_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapb_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapb_ex_100_02.chk_afr(iv_apb_rec ,ov_error_rec ,iv_axn_rec);
xaease_en_100_02.chk(
    iv_ase_rec IN OUT NOCOPY xae_ase_rest_rec2_t ,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaease_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaease_ex_100_02.chk_bfr(iv_ase_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaease_ex_100_02.CV_CHK_REP =cmncon_cl_000_01.CUSTOMIZED THEN
xaease_ex_100_02.chk_rep(iv_ase_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaease_ex_100_02.CV_CHK_REP =cmncon_cl_000_01.CUSTOMIZED THEN
xaease_ex_100_02.chk_afr(iv_ase_rec ,ov_error_rec ,iv_axn_rec);
xaeavl_en_100_02.chk(
    iv_avl_rec IN OUT NOCOPY xae_avl_rest_rec2_t ,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeavl_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeavl_ex_100_02.chk_bfr(iv_avl_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeavl_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeavl_ex_100_02.chk_rep(iv_avl_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeavl_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeavl_ex_100_02.chk_afr(iv_avl_rec ,ov_error_rec ,iv_axn_rec);  
xaeapd_en_100_02.chk (  
    iv_apd_rec IN OUT NOCOPY xae_apd_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapd_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapd_ex_100_02.chk_bfr(iv_apd_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapd_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapd_ex_100_02.chk_rep(iv_apd_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapd_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapd_ex_100_02.chk_afr(iv_apd_rec ,ov_error_rec ,iv_axn_rec);  
xaebsd_en_100_02.chk (  
    iv_bus_app_rec IN OUT NOCOPY xae_bsd_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebsd_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebsd_ex_100_02.chk_bfr(iv_bus_app_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebsd_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebsd_ex_100_02.chk_rep(iv_bus_app_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebsd_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebsd_ex_100_02.chk_afr(iv_bus_app_rec ,ov_error_rec ,iv_axn_rec);  
xaebsa_en_100_02.chk (  
    iv_bsa_rec IN OUT NOCOPY xae_bsa_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebasa_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebasa_ex_100_02.chk_bfr(iv_bsa_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebasa_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebasa_ex_100_02.chk_rep(iv_bsa_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebasa_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebasa_ex_100_02.chk_afr(iv_bsa_rec ,ov_error_rec ,iv_axn_rec);  
xaebasp_en_100_02.chk (  
    iv_bsp_rec IN OUT NOCOPY xae_bsp_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t ,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebasp_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebasp_ex_100_02.chk_bfr(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebasp_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebasp_ex_100_02.chk_rep(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebasp_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebasp_ex_100_02.chk_afr(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);  
xaebasl_en_100_02.chk (  
    iv_bsl_rec IN OUT NOCOPY xae_bsl_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t ,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebasl_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebasl_ex_100_02.chk_bfr(iv_bsl_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebasl_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebasl_ex_100_02.chk_rep(iv_bsl_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebst_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebst_ex_100_02.chk_afr(iv_bst_rec ,ov_error_rec ,iv_axn_rec);
xaebst_en_100_02.chk(
    iv_bst_rec IN OUT NOCOPY xae_bst_rest_rec_t ,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebst_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebst_ex_100_02.chk_bfr(iv_bst_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebst_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaebst_ex_100_02.chk_rep(iv_bst_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebst_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebst_ex_100_02.chk_afr(iv_bst_rec ,ov_error_rec ,iv_axn_rec);
xaebst_en_100_02.chk(
    iv_bst_rec IN OUT NOCOPY xae_bst_rest_rec_t ,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebst_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebst_ex_100_02.chk_bfr(iv_bst_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebst_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaebst_ex_100_02.chk_rep(iv_bst_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebst_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebst_ex_100_02.chk_afr(iv_bst_rec ,ov_error_rec ,iv_axn_rec);
xaebst_en_100_02.chk(
    iv_bst_rec IN OUT NOCOPY xae_bst_rest_rec_t ,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebst_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebst_ex_100_02.chk_bfr(iv_bst_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebbsb_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaebbsb_ex_100_02.chk_rep(iv_bsb_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebbsb_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebbsb_ex_100_02.chk_afr(iv_bsb_rec ,ov_error_rec ,iv_axn_rec);
```

Extensible parameters are Tab Type object**DATE:**

```
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT
(
    ATT_NAME VARCHAR2 (30)
    , ATT_VALUE DATE);
/
```

NUMBER:

```
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT
(
    ATT_NAME VARCHAR2 (30)
    , ATT_VALUE NUMBER);
);
/
```

STRING:

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT
(
    ATT_NAME VARCHAR2 (30)
    , ATT_VALUE VARCHAR2 (4000));
);
/
```

12.10 Account Search (GET)

Below are the package details for account search web service

Wrapper Engine package: -

```
xcsacs_ew_100_01. xcsacs_ew_100_01 (iv_csh IN xcs_csh_rec_t,iv_acs OUT NOCOPY
xcs_acs_tab_t);
```

Main Engine package: -

```
xcsacs_em_100_01. get_account_summary (iv_csh IN xcs_csh_rec_t,iv_acs OUT
NOCOPY xcs_acs_tab_t);
xcsacs_en_100_01. get_acc_search_summary (iv_csh IN xcs_csh_rec_t,iv_acs OUT
NOCOPY xcs_acs_tab_t);
```

Below are the Exit point package details

BEFORE:-

```
xcsacs_ex_100_01.cv_get_acc_search_summary_bfr =cmncon_cl_000_01.CUSTOMIZED
THEN
    xcsacs_ex_100_01.get_acc_search_summary_bfr(iv_csh, iv_acs);
```

REPLACE:-

```
xcsacs_ex_100_01.cv_get_acc_search_summary_rep =cmncon_cl_000_01.CUSTOMIZED
THEN
    xcsacs_ex_100_01.get_acc_search_summary_rep(iv_csh, iv_acs);
```

AFTER:-

```
xcsacs_ex_100_01.cv_get_acc_search_summary_afr =cmncon_cl_000_01.CUSTOMIZED
THEN
    xcsacs_ex_100_01.get_acc_search_summary_afr(iv_csh, iv_acs);
```

Extensible parameters are Tab Type object**DATE:**

```
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT
(
    ATT_NAME VARCHAR2 (30)
    , ATT_VALUE DATE);
/
```

NUMBER:

```
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT
(
    ATT_NAME VARCHAR2 (30)
    , ATT_VALUE NUMBER);
);
/
```

STRING:

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT
(
    ATT_NAME VARCHAR2 (30)
    , ATT_VALUE VARCHAR2 (4000));
);
/
```

12.11 Call Activity (POST)

Below mentioned are the custom fields which are part of response alone.

Table 12-11 Call Activity

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String

Table 12-11 (Cont.) Call Activity

Element name	Sub element	Data type
	KeyValue	Number (Double)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  <CustomUserDefinedStringData>
    <KeyName>StringName</KeyName>
    <KeyValue>StringValue</KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName>NumberName</KeyName>
    <KeyValue>NumberValue</KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName>DateName</KeyName>
    <KeyValue>DateValue</KeyValue>
  </CustomUserDefinedDateData>
</Custom>

```

Sample JSON

```

{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  }
}

```

Below are the package details for account search web service

Wrapper Engine package:-

```

xcscac_ew_100_01. xcscac_ew_100_01 (iv_cac_rec_t IN
xcs_csc_rec_t,iv_cac_result_tab_t OUT NOCOPY xcs_cac_result_tab_t);

```

Main Engine package:-

```
xcscac_em_100_01.post_cac (iv_cac_rec_t IN xcs_csc_rec_t, Iv_cac_result_tab_t
OUT NOCOPY xcs_cac_result_tab_t);
```

Below are the Exit point package details**BEFORE:-**

```
xcscac_ex_100_01.cv_post_cac_bfr =cmncon_cl_000_01.CUSTOMIZED THEN
    xcscac_ex_100_01. post_cac_bfr(iv_cac_rec_t, iv_cac_result_tab_t);
```

REPLACE:-

```
xcscac_ex_100_01.cv_post_cac_rep =cmncon_cl_000_01.CUSTOMIZED THEN
    xcscac_ex_100_01. post_cac_rep (iv_cac_rec_t,iv_cac_result_tab_t);
```

AFTER:-

```
xcscac_ex_100_01.cv_post_cac_aftr =cmncon_cl_000_01.CUSTOMIZED THEN
    xcscac_ex_100_01.post_cac_aftr (iv_cac_rec_t,iv_cac_result_tab_t);
```

Extensible parameters are Tab Type object**DATE:**

```
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT
(
    ATT_NAME          VARCHAR2 (30)
    , ATT_VALUE       DATE);
/
```

NUMBER:

```
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT
(
    ATT_NAME          VARCHAR2 (30)
    , ATT_VALUE       NUMBER);
);
/
```

STRING:

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT
(
    ATT_NAME          VARCHAR2 (30)
    , ATT_VALUE       VARCHAR2 (4000));
);
/
```

12.12 Remarketing (PUT)

Below mentioned table has element name which indicates which type of custom data is passed by enclosing the name and its value in keyname and keyvalue respectively.

Table 12-12 Remarketing

Element name	Sub element	Data type
CustomUserDefinedStringData		
	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData		
	KeyName	String
	KeyValue	Number (Double)
CustomUserDefinedDateData		
	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  <CustomUserDefinedStringData>
    <KeyName>Middle Name</KeyName>
    <KeyValue>Oracle</KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName>Age</KeyName>
    <KeyValue>27</KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName>FirstPmtDate</KeyName>
    <KeyValue>2016-07-14</KeyValue>
  </CustomUserDefinedDateData>
</Custom>

```

Sample JSON

```

{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "Middle Name",
      "KeyValue": "Oracle"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "Age",
      "KeyValue": "27"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "FirstPmtDate",
      "KeyValue": "2016-07-14"
    }
  }
}

```

Below are the package details for products web service

Engine wrapper package:

```
gaiprc_ew_100_01.gaiprc_ew_100_01(iv_gai_rec IN OUT gai_rec_t,iv_service_type  
IN VARCHAR2)
```

Engine main packages:

GET:

```
gaiprc_em_100_01.get_remarketing(iv_gai_rec IN OUT gai_rec_t);
```

Below are the exit points for get service

BEFORE:

```
gaiprc_ex_100_01.get_remarketing_bfr (iv_gai_rec IN OUT gai_rec_t)
```

REPLACE:

```
gaiprc_ex_100_01.get_remarketing_rep (iv_gai_rec IN OUT gai_rec_t)
```

AFTER:

```
gaiprc_ex_100_01.get_remarketing_afr (iv_gai_rec IN OUT gai_rec_t)
```

PUT:

```
gaiprc_em_100_01.put_remarketing (iv_gai_rec IN OUT gai_rec_t);
```

Below are the exit points for get service.

BEFORE:

```
gaiprc_ex_100_01.put_remarketing_bfr(iv_gai_rec IN OUT gai_rec_t)
```

REPLACE:

```
gaiprc_ex_100_01.put_remarketing_rep (iv_gai_rec IN OUT gai_rec_t)
```

AFTER:

```
gaiprc_ex_100_01.put_remarketing_afr (iv_gai_rec IN OUT gai_rec_t)
```

12.13 Invoice (POST)

Below mentioned table has element name which indicates which type of custom data is passed by enclosing the name and its value in keyname and keyvalue respectively.

Table 12-13 Invoice

Element name	Sub element	Data type
CustomUserDefinedStringData		
	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData		
	KeyName	String
	KeyValue	Number (Double)
CustomUserDefinedDateData		
	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <CustomUserDefinedStringData>
    <KeyName>Middle Name</KeyName>
    <KeyValue>Oracle</KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName>Age</KeyName>
    <KeyValue>27</KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName>FirstPmtDate</KeyName>
    <KeyValue>2016-07-14</KeyValue>
  </CustomUserDefinedDateData>
</Custom>
```

Sample JSON

```
{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "Middle Name",
      "KeyValue": "Oracle"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "Age",
      "KeyValue": "27"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "FirstPmtDate",
      "KeyValue": "2016-07-14"
    }
  }
}
```

Below are the package details for products web service

Engine wrapper package:

```
gaiprc_ew_100_01.gaiprc_ew_100_01(iv_gai_rec IN OUT gai_rec_t,iv_service_type  
IN VARCHAR2)
```

Engine main packages:

GET:

```
gaiprc_em_100_02.get_invoice(iv_gai_rec IN OUT gai_rec_t);
```

Below are the exit points for get service

BEFORE:

```
gaiprc_ex_100_02.get_invoice_bfr (iv_gai_rec IN OUT gai_rec_t)
```

REPLACE:

```
gaiprc_ex_100_02.get_invoice_rep (iv_gai_rec IN OUT gai_rec_t)
```

AFTER:

```
gaiprc_ex_100_02.get_invoice_afr (iv_gai_rec IN OUT gai_rec_t)
```

POST:

```
gaiprc_em_100_02.post_invoice (iv_gai_rec IN OUT gai_rec_t);
```

Below are the exit points for get service

BEFORE:

```
gaiprc_ex_100_02.post_invoice_bfr (iv_gai_rec IN OUT gai_rec_t)
```

REPLACE:

```
gaiprc_ex_100_02.post_invoice_rep (iv_gai_rec IN OUT gai_rec_t)
```

AFTER:

```
gaiprc_ex_100_02.post_invoice_afr (iv_gai_rec IN OUT gai_rec_t)
```

12.14 Application Comment (GET/POST)

Below mentioned are the custom fields. These fields will be part of response in case of GET method and will be part of request in case of POST method.

Table 12-14 Application Comment

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <CustomUserDefinedStringData>
    <KeyName>Middle Name</KeyName>
    <KeyValue>Oracle</KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName>Age</KeyName>
    <KeyValue>27</KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName>FirstPmtDate</KeyName>
    <KeyValue>2016-07-14</KeyValue>
  </CustomUserDefinedDateData>
</Custom>
```

Sample JSON

```
{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  }
}
```

Below are the package details for application comment web service

Wrapper Engine package: -

```
xaexcm_ew_100_01. xaexcm_ew_100_01 (iv_req_rec_t IN xae_acm_req_t,
iv_res_rec_t IN OUT xae_acm_res_rec_t);
```

Main Engine package:-

```
xaexcm_em_100_01. xaexcm_em_100_01 (iv_req_rec_t IN
xae_acm_req_t, iv_res_rec_t IN OUT xae_acm_res_rec_t);
```

Below are the Exit point package details for xaexcm_em_100_01.xaexcm_em_100_01());**BEFORE:-**

```
xaexcm_ex_100_01.cv_xaexcm_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
xaexcm_ex_100_01.xaexcm_em_100_01_bfr(iv_req_rec_t, iv_res_rec_t);
```

REPLACE:-

```
xaexcm_ex_100_01.cv_xaexcm_rep = cmncon_cl_000_01.CUSTOMIZED THEN
xaexcm_ex_100_01.xaexcm_em_100_01_rep(iv_req_rec_t, iv_res_rec_t);
```

AFTER:-

```
xaexcm_ex_100_01.cv_xaexcm_afr = cmncon_cl_000_01.CUSTOMIZED THEN
xaexcm_ex_100_01.xaexcm_em_100_01_afr(iv_req_rec_t, iv_res_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
ATT_NAME VARCHAR2(30),
ATT_VALUE VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
ATT_NAME VARCHAR2(30),
ATT_VALUE NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
ATT_NAME VARCHAR2(30),
ATT_VALUE DATE );
```

12.15 Account Comment (GET/POST)

Below mentioned are the custom fields. These fields will be part of response in case of GET method and will be part of request in case of POST method.

Table 12-15 Account Comment

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String

Table 12-15 (Cont.) Account Comment

Element name	Sub element	Data type
	KeyValue	Number (Decimal)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <CustomUserDefinedStringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </CustomUserDefinedDateData>
</Custom>
```

Sample JSON

```
{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  }
}
```

Below are the package details for applicationupdate web service

Wrapper Engine package:-

```
xcsxcm_ew_100_01. xcsxcm_ew_100_01 (iv_acm_rec_t IN OUT xcs_acm_axn_rec_t);
```

Main Engine package:-

```
xcsxcm_em_100_01.post_acm (iv_acm_rec_t IN OUT xcs_acm_axn_rec_t);  
xcsxcm_em_100_01.get_acm(iv_acm_rec_t IN OUT xcs_acm_axn_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01.post_acm();**BEFORE:-**

```
xcsxcm_ex_100_01.cv_post_acm_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsxcm_ex_100_01.post_acm_bfr(iv_acm_rec_t);
```

REPLACE:-

```
xcsxcm_ex_100_01.cv_post_acm_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsxcm_ex_100_01.post_acm_rep(iv_acm_rec_t);
```

AFTER:-

```
xcsxcm_ex_100_01.cv_post_acm_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsxcm_ex_100_01.post_acm_afr(iv_acm_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01.get_acm();**BEFORE:-**

```
xcsxcm_ex_100_01.cv_get_acm_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsxcm_ex_100_01.get_acm_bfr(iv_acm_rec_t);
```

REPLACE:-

```
xcsxcm_ex_100_01.cv_get_acm_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsxcm_ex_100_01.get_acm_rep(iv_acm_rec_t);
```

AFTER:-

```
xcsxcm_ex_100_01.cv_get_acm_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsxcm_ex_100_01.get_acm_afr(iv_acm_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   VARCHAR2(4000));  
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   NUMBER );  
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   DATE );
```

12.16 Application Checklist (GET)

Below mentioned are the custom fields. These fields will be part of response.

Table 12-16 Application Checklist

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <CustomUserDefinedStringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </CustomUserDefinedDateData>
</Custom>
```

Sample JSON

```
{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  }
}
```

Below are the package details for application checklist web service

Wrapper Engine package:-

```
xaexck_ew_100_01.xaexck_ew_100_01 (iv_req_rec_t IN xae_ack_req_t,
iv_res_rec_t IN OUT xae_ack_res_t);
```

Main Engine package:-

```
xaexck_em_100_01.xaexck_em_100_01 (iv_req_rec_t IN xae_ack_req_t,iv_res_rec_t
IN OUT xae_ack_res_t);
```

Below are the Exit point package details for xaexck_em_100_01.xaexck_em_100_01());

BEFORE:-

```
xaexck_ex_100_01.cv_xaexck_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
xaexck_ex_100_01.xaexck_em_100_01_bfr (iv_req_rec_t, iv_res_rec_t);
```

REPLACE:-

```
xaexck_ex_100_01.cv_xaexck_rep = cmncon_cl_000_01.CUSTOMIZED THEN
xaexck_ex_100_01.xaexck_em_100_01_rep (iv_req_rec_t, iv_res_rec_t);
```

AFTER:-

```
xaexck_ex_100_01.cv_xaexck_afr = cmncon_cl_000_01.CUSTOMIZED THEN
xaexck_ex_100_01.xaexck_em_100_01_afr (iv_req_rec_t, iv_res_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
ATT_NAME VARCHAR2(30),
ATT_VALUE VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
ATT_NAME VARCHAR2(30),
ATT_VALUE NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
ATT_NAME VARCHAR2(30),
ATT_VALUE DATE );
```

12.17 Outgoing File List (GET)

Below mentioned are the custom fields. These fields will be part of response.

Table 12-17 Outgoing File List

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)

Table 12-17 (Cont.) Outgoing File List

Element name	Sub element	Data type
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  <CustomUserDefinedStringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </CustomUserDefinedDateData>
</Custom>

```

Sample JSON

```

{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  }
}

```

Below are the package details for outgoing file list web service

Wrapper Engine package:-

```

xpfopf_ew_100_01.xpfopf_ew_100_01 (iv_req_rec_t IN xws_opf_req_rec_t,
iv_res_rec_t IN OUT xws_opf_resp_rec_t);

```

Main Engine package:-

```
xpfopf_em_100_01.xpfopf_em_100_01 (iv_req_rec_t IN xws_opf_req_rec_t,  
iv_res_rec_t IN OUT xws_opf_resp_rec_t);
```

Below are the Exit point package details for xpfopf_em_100_01.xpfopf_em_100_01();**BEFORE:-**

```
xpfopf_ex_100_01.cv_xpfopf_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
xpfopf_ex_100_01.xpfopf_em_100_01_bfr (iv_req_rec_t, iv_res_rec_t);
```

REPLACE:-

```
xpfopf_ex_100_01.cv_xpfopf_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
xpfopf_ex_100_01.xpfopf_em_100_01_rep (iv_req_rec_t, iv_res_rec_t);
```

AFTER:-

```
xpfopf_ex_100_01.cv_xpfopf_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
xpfopf_ex_100_01.xpfopf_em_100_01_afr (iv_req_rec_t, iv_res_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   VARCHAR2(4000));  
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   NUMBER );  
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   DATE );
```

12.18 Outgoing File (POST)

Below are the package details for outgoing file web service

Wrapper Engine package:-

```
xpfopf_ew_100_01.xpfopf_ew_100_01 (iv_req_rec_t IN xws_opf_req_rec_t,  
iv_res_rec_t IN OUT xws_opf_resp_rec_t);
```

Main Engine package:-

```
xpfopf_em_100_01.xpfopf_em_100_01 (iv_req_rec_t IN xws_opf_req_rec_t,  
iv_res_rec_t IN OUT xws_opf_resp_rec_t);
```

Below are the Exit point package details for xpfopf_em_100_01.xpfopf_em_100_01();

BEFORE:-

```
xpfopf_ex_100_01.cv_xpfopf_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xpfopf_ex_100_01.xpfopf_em_100_01_bfr (iv_req_rec_t, iv_res_rec_t);
```

REPLACE:-

```
xpfopf_ex_100_01.cv_xpfopf_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
    xpfopf_ex_100_01.xpfopf_em_100_01_rep (iv_req_rec_t, iv_res_rec_t);
```

AFTER:-

```
xpfopf_ex_100_01.cv_xpfopf_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xpfopf_ex_100_01.xpfopf_em_100_01_afr (iv_req_rec_t, iv_res_rec_t);
```

12.19 Incoming File (GET)

Below are the package details for incoming file web service

Wrapper Engine package:-

```
xfpif_ew_100_01.xfpif_ew_100_01 (iv_req_rec_t IN OUT xws_ipf_req_rec_t);
```

Main Engine package:-

```
xfpif_em_100_01.xfpif_em_100_01 (iv_req_rec_t IN OUT xws_ipf_req_rec_t);
```

Below are the Exit point package details for xpfopf_em_100_01.xpfopf_em_100_01();

BEFORE:-

```
xfpif_ex_100_01.cv_xfpif_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xfpif_ex_100_01.xfpif_em_100_01_bfr(iv_req_rec_t);
```

REPLACE:-

```
xfpif_ex_100_01.cv_xfpif_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
    xfpif_ex_100_01.xfpif_em_100_01_rep(iv_req_rec_t);
```

AFTER:-

```
xfpif_ex_100_01.cv_xfpif_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xfpif_ex_100_01.xfpif_em_100_01_afr(iv_req_rec_t);
```

12.20 Products (GET)

Below mentioned are the custom fields. These fields will be part of response.

Table 12-18 Products

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <CustomUserDefinedStringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </CustomUserDefinedDateData>
</Custom>
```

Sample JSON

```
{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  }
}
```

Below are the package details for products web service

Wrapper Engine package:-

```
xpdprc_ew_100_01.xpdprc_ew_100_01 (iv_req_rec_t IN xpd_req_rec_t,
iv_res_rec_t IN OUT xpd_resp_rec_t);
```

Main Engine package:-

```
xpdprc_em_100_01.xpdprc_em_100_01 (iv_req_rec_t IN xpd_req_rec_t,
iv_res_rec_t IN OUT xpd_resp_rec_t);
```

Below are the Exit point package details for xpdprc_em_100_01.xpdprc_em_100_01());

BEFORE:-

```
xpdprc_ex_100_01.cv_xpdprc_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
  xpdprc_ex_100_01.xpdprc_em_100_01_bfr (iv_req_rec_t, iv_res_rec_t);
```

REPLACE:-

```
xpdprc_ex_100_01.cv_xpdprc_rep = cmncon_cl_000_01.CUSTOMIZED THEN
  xpdprc_ex_100_01.xpdprc_em_100_01_rep (iv_req_rec_t, iv_res_rec_t);
```

AFTER:-

```
xpdprc_ex_100_01.cv_xpdprc_afr = cmncon_cl_000_01.CUSTOMIZED THEN
  xpdprc_ex_100_01.xpdprc_em_100_01_afr (iv_req_rec_t, iv_res_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     DATE );
```

12.21 Assets (GET)

Below mentioned are the custom fields. These fields will be part of response.

Table 12-19 Assets

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)

Table 12-19 (Cont.) Assets

Element name	Sub element	Data type
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  <CustomUserDefinedStringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </CustomUserDefinedDateData>
</Custom>

```

Sample JSON

```

{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  }
}

```

Below are the package details for assets web service

Wrapper Engine package:-

```

xstprc_ew_100_01.xstprc_ew_100_01 (iv_req_rec_t IN xst_req_rec_t,
iv_res_rec_t IN OUT xst_resp_rec_t);

```

Main Engine package:-

```
xstprc_em_100_01.get_asset_types (iv_req_rec_t IN xst_req_rec_t,
iv_res_rec_t IN OUT xst_resp_rec_t);
```

Below are the Exit point package details for xstprc_em_100_01.get_asset_types ();**BEFORE:-**

```
xstprc_ex_100_01.cv_get_asset_types_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
xstprc_ex_100_01.get_asset_types_bfr (iv_req_rec_t, iv_res_rec_t);
```

REPLACE:-

```
xstprc_ex_100_01.cv_get_asset_types_rep = cmncon_cl_000_01.CUSTOMIZED THEN
xstprc_ex_100_01.get_asset_types_rep (iv_req_rec_t, iv_res_rec_t);
```

AFTER:-

```
xstprc_ex_100_01.cv_get_asset_types_afr = cmncon_cl_000_01.CUSTOMIZED THEN
xstprc_ex_100_01.get_asset_types_afr (iv_req_rec_t, iv_res_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
ATT_NAME    VARCHAR2(30),
ATT_VALUE   VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
ATT_NAME    VARCHAR2(30),
ATT_VALUE   NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
ATT_NAME    VARCHAR2(30),
ATT_VALUE   DATE );
```

12.22 Assets (PUT)

Below mentioned are the custom fields. These fields will be part of response for PUT services.

Table 12-20 Assets

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  < StringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </ StringData>
  <NumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </NumberData>
  <DateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </DateData>
</Custom>

```

Sample JSON

```

"Custom": {
  "StringData": [
    {
      "KeyName": " StringName ",
      "KeyValue": " StringValue "
    }
  ],
  "NumberData": [
    {
      "KeyName": " NumberName ",
      "KeyValue": NumberValue
    }
  ],
  "DateData": [
    {
      "KeyName": " DateName ",
      "KeyValue": " DateValue"
    }
  ]
}

```

Below are the package details for assets valuation web service

Wrapper Engine package:-

```
xcsase_ew_100_01.xcsase_ew_100_01(iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

Main Engine package:-

```
xcsase_em_100_01.xcsase_em_100_01(iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

Main Engine package calls below procedures as per service type:

```
put_asset(iv_xcs_ase_rec);
```

Below are the Exit point package details for xcsase_em_100_01.xcsase_em_100_01:**BEFORE:-**

```
xcsase_ex_100_01.CV_xcsase_ex_100_01_BFR= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.xcsase_ex_100_01_bfr (iv_xcs_ase_rec IN OUT
xcs_ase_req_rec_t);
```

REPLACE:-

```
xcsase_ex_100_01.CV_xcsase_ex_100_01_REP= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.xcsase_ex_100_01_rep (iv_xcs_ase_rec IN OUT
xcs_ase_req_rec_t);
```

AFTER:-

```
xcsase_ex_100_01.CV_xcsase_ex_100_01_AFR= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.xcsase_ex_100_01_afr (iv_xcs_ase_rec IN OUT
xcs_ase_req_rec_t);
```

Below are the Exit point package details for PUT asset:**BEFORE:-**

```
xcsase_ex_100_01.CV_put_asset_BFR= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.put_asset_bfr (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

REPLACE:-

```
xcsase_ex_100_01.CV_put_asset_REP= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.get_put_rep (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

AFTER:-

```
xcsase_ex_100_01.CV_put_asset_AFR= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.put_asset_afr (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE   VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE   NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE   DATE );
```

12.23 Asset Valuation (GET/PUT/POST)

Below mentioned are the custom fields. These fields will be part of response for GET/PUT/POST services.

Table 12-21 Asset Valuation

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  < StringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </ StringData>
  <NumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </NumberData>
  <DateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </DateData>
</Custom>
```

Sample JSON

```
"Custom": {
  "StringData": [
    {
      "KeyName": " StringName ",
      "KeyValue": " StringValue "
    }
  ],
  "NumberData": [
    {
      "KeyName": " NumberName ",
      "KeyValue": NumberValue
    }
  ],
  "DateData": [
    {
      "KeyName": " DateName ",
      "KeyValue": " DateValue"
    }
  ]
}
```

```
]
}
```

Below are the package details for assets valuation web service

Wrapper Engine package:-

```
xcsase_ew_100_01.xcsase_ew_100_01(iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

Main Engine package:-

```
xcsase_em_100_01.xcsase_em_100_01(iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

Main Engine package calls below procedures as per service type:

```
GET: get_ase_avl(iv_xcs_ase_rec);
PUT: put_ase_avl(iv_xcs_ase_rec);
POST: post_ase_avl(iv_xcs_ase_rec);
```

Below are the Exit point package details for xcsase_em_100_01.xcsase_em_100_01:

BEFORE:-

```
xcsase_ex_100_01.CV_xcsase_ex_100_01_BFR= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.xcsase_ex_100_01_bfr (iv_xcs_ase_rec IN OUT
xcs_ase_req_rec_t);
```

REPLACE:-

```
xcsase_ex_100_01.CV_xcsase_ex_100_01_REP= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.xcsase_ex_100_01_rep (iv_xcs_ase_rec IN OUT
xcs_ase_req_rec_t);
```

AFTER:-

```
xcsase_ex_100_01.CV_xcsase_ex_100_01_AFR= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.xcsase_ex_100_01_afr (iv_xcs_ase_rec IN OUT
xcs_ase_req_rec_t);
```

Below are the Exit point package details for GET:

BEFORE:-

```
xcsase_ex_100_01.CV_get_ase_avl_BFR= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.get_ase_avl_bfr (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

REPLACE:-

```
xcsase_ex_100_01.CV_get_ase_avl_REP= cmncon_cl_000_01.CUSTOMIZED THEN
xcsase_ex_100_01.get_ase_avl_rep (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```


AFTER:-

```
xcsase_ex_100_01.CV_get_ase_avl_AFR= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsase_ex_100_01.get_ase_avl_afr (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

Below are the Exit point package details for PUT:

BEFORE:-

```
xcsase_ex_100_01.CV_put_ase_avl_BFR= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsase_ex_100_01.put_ase_avl_bfr (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

REPLACE:-

```
xcsase_ex_100_01.CV_put_ase_avl_REP= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsase_ex_100_01.put_ase_avl_rep (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

AFTER:-

```
xcsase_ex_100_01.CV_put_ase_avl_AFR= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsase_ex_100_01.put_ase_avl_afr (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

Below are the Exit point package details for POST:

BEFORE:-

```
xcsase_ex_100_01.CV_post_ase_avl_BFR= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsase_ex_100_01.post_ase_avl_bfr (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

REPLACE:-

```
xcsase_ex_100_01.CV_post_ase_avl_REP= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsase_ex_100_01.post_ase_avl_rep (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

AFTER:-

```
xcsase_ex_100_01.CV_post_ase_avl_AFR= cmncon_cl_000_01.CUSTOMIZED THEN  
xcsase_ex_100_01.post_ase_avl_afr (iv_xcs_ase_rec IN OUT xcs_ase_req_tab_t)
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   VARCHAR2(4000));  
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   NUMBER );  
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   DATE );
```

12.24 Asset Sub Types (GET)

Below mentioned are the custom fields. These fields will be part of response.

Table 12-22 Asset Sub Types

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <CustomUserDefinedStringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </CustomUserDefinedDateData>
</Custom>
```

Sample JSON

```
{
  "Custom": {
    "CustomUserDefinedStringData": {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    },
    "CustomUserDefinedNumberData": {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    },
    "CustomUserDefinedDateData": {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  }
}
```

Below are the package details for asset sub types web service

Wrapper Engine package:-

```
xsbprc_ew_100_01.xsbprc_ew_100_01 (iv_req_rec_t IN xsb_req_rec_t,  
iv_res_rec_t IN OUT xsb_resp_rec_t);
```

Main Engine package:-

```
xstprc_em_100_01.get_asset_sub_types (iv_req_rec_t IN xsb_req_rec_t,  
iv_res_rec_t IN OUT xsb_resp_rec_t);
```

Below are the Exit point package details for xsbprc_em_100_01.get_asset_sub_types ();

BEFORE:-

```
xsbprc_ex_100_01.cv_get_asset_sub_types_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
xsbprc_ex_100_01.get_asset_sub_types_bfr (iv_req_rec_t, iv_res_rec_t);
```

REPLACE:-

```
xsbprc_ex_100_01.cv_get_asset_sub_types_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
xsbprc_ex_100_01.get_asset_sub_types_rep (iv_req_rec_t, iv_res_rec_t);
```

AFTER:-

```
xsbprc_ex_100_01.cv_get_asset_sub_types_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
xsbprc_ex_100_01.get_asset_sub_types_afr (iv_req_rec_t, iv_res_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   ARCHAR2(4000));  
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   NUMBER );  
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   DATE );
```

12.25 Application Status Change (PUT)

Below the custom packages.

Below are the package details for application status change web service

Wrapper Engine package:-

```
xaeasc_ew_100_01.xaeasc_ew_100_01 ( iv_req_rec_t IN  
xae_app_asc_rec_t,iv_res_rec_t IN OUT NOCOPY xae_app_asc_res_rec_t)
```

Main Engine package:-

```
xaeasc_em_100_01.app_status_change (iv_req_rec_t IN
xae_app_asc_rec_t,iv_res_rec_t IN OUT NOCOPY xae_app_asc_res_rec_t)
```

Below are the Exit point package details for xaeasc_em_100_01.app_status_change ();**BEFORE:-**

```
xaeasc_ex_100_01.cv_app_status_change_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
xaeasc_ex_100_01.app_status_change_bfr(iv_req_rec_t, iv_res_rec_t);
```

REPLACE:-

```
xaeasc_ex_100_01.cv_app_status_change_rep = cmncon_cl_000_01.CUSTOMIZED THEN
xaeasc_ex_100_01.app_status_change_rep(iv_req_rec_t, iv_res_rec_t);
```

AFTER:-

```
xaeasc_ex_100_01.cv_app_status_change_afr = cmncon_cl_000_01.CUSTOMIZED THEN
xaeasc_ex_100_01.app_status_change_afr(iv_req_rec_t, iv_res_rec_t);
```

12.26 Application Update (PUT)

Below mentioned table has element name which indicates which type of custom data is passed in the custom element block, enclosing the name and its value in keyname and keyvalue respectively.

**Note:**

This block is unbounded and is part of other blocks. Example: Address Block.

Table 12-23 Application Update

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML FORMAT

```
<CustomFields>
  <CustomUserDefinedStringData>
    <KeyName>str1234</KeyName>
    <KeyValue>str1234</KeyValue>
  </CustomUserDefinedStringData>
```

```

    <CustomUserDefinedNumberData>
      <KeyName>str1234</KeyName>
      <KeyValue>3.1415926535</KeyValue>
    </CustomUserDefinedNumberData>
    <CustomUserDefinedDateData>
      <KeyName>str1234</KeyName>
      <KeyValue>2012-12-13T12:12:12</KeyValue>
    </CustomUserDefinedDateData>
  </CustomFields>

```

Sample JSON

```

"CustomFields": {
  "CustomUserDefinedStringData": {
    "KeyName": "str1234",
    "KeyValue": "str1234"
  },
  "CustomUserDefinedNumberData": {
    "KeyName": "str1234",
    "KeyValue": "3.1415926535"
  },
  "CustomUserDefinedDateData": {
    "KeyName": "str1234",
    "KeyValue": "2012-12-13T12:12:12"
  }
}

```

Below are the package details for applicationupdate web service

Wrapper Engine package:-

```

xaeupd_ew_100_02.xaeupd_ew_100_02 (iv_app_upd_rec IN OUT NOCOPY
xae_app_rest_rec_t, ov_error_rec IN OUT NOCOPY xae_error_rec_t);

```

Main Engine package:-

```

xaeupd_em_100_02.update_app (iv_app_rec IN OUT NOCOPY xae_app_rest_rec_t,
ov_error_rec IN OUT NOCOPY xae_error_rec_t);

```

Lookup Validations- involves the following packages related to

```

Applicant -> xaeapl_en_100_02
Applicant Address -> xaeapa_en_100_02
Applicant Employment -> xaeape_en_100_02
Applicant Telecom -> xaeapt_en_100_02
Applicant Financial -> xaeapf_en_100_02
Applicant liability -> xaeapb_en_100_02
Applicant other income information -> xaeapi_en_100_02
Decision trade in -> xaeapd_en_100_02
Collateral valuation -> xaeavl_en_100_02
Business details -> xaebsd_en_100_02
Business address -> xaebsa_en_100_02
Business partners -> xaebsp_en_100_02
Business affiliates -> xaebsl_en_100_02

```

```
Business telecom -> xaebst_en_100_02
Business financials -> xaebst_en_100_02
Business liabilities -> xaebst_en_100_02
```

Below is the application update main package

```
xaeupd_en_100_02.ins (iv_app_rec IN OUT NOCOPY
xae_app_rest_rec_t,iv_error_rec IN OUT NOCOPY xae_error_rec_t,iv_axn_rec IN
OUT NOCOPY xae_axn_evw%ROWTYPE)
```

You can do the customization on the following packages

Below are the Exit point package details for xaeupd_em_100_02.update_app ();

BEFORE:-

```
xaeupd_ex_100_02.cv_update_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
xaeupd_ex_100_02.update_app_bfr (iv_app_rec,ov_error_rec);
```

REPLACE:-

```
xaeupd_ex_100_02.cv_update_rep = cmncon_cl_000_01. CUSTOMIZED THEN
xaeupd_ex_100_02.update_app_rep (iv_app_rec, ov_error_rec);
```

AFTER:-

```
xaeupd_ex_100_02.cv_update_afr = cmncon_cl_000_01.CUSTOMIZED THEN
xaeupd_ex_100_02.update_app_afr (iv_app_rec, ov_error_rec);
xaeapl_en_100_02. chk (
iv_apl_rec IN OUT NOCOPY xae_apl_rest_rec_t,
ov_error_rec IN OUT NOCOPY xae_error_rec_t,
iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE);
```

BEFORE:-

```
xaeapl_ex_100_02.CV_CHK_BFR =cmncon_cl_000_01.CUSTOMIZED THEN
xaeapl_ex_100_02.chk_bfr(iv_apl_rec, ov_error_rec, iv_axn_rec);
```

REPLACE:-

```
xaeapl_ex_100_02.CV_CHK_REP =cmncon_cl_000_01.CUSTOMIZED THEN
xaeapl_ex_100_02.chk_rep(iv_apl_rec, ov_error_rec, iv_axn_rec);
```

AFTER:-

```
xaeapl_ex_100_02.CV_CHK_AFR =cmncon_cl_000_01.CUSTOMIZED THEN
xaeapl_ex_100_02.chk_afr(iv_apl_rec, ov_error_rec, iv_axn_rec );
xaeapa_en_100_02. chk (
iv_apa_rec IN OUT NOCOPY xae_apa_rest_rec_t,
ov_error_rec IN OUT NOCOPY xae_error_rec_t,
iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapa_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapa_ex_100_02.chk_bfr(iv_apa_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapa_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapa_ex_100_02.chk_rep(iv_apa_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapa_ex_100_02.CV_CHK_AFR =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapa_ex_100_02.chk_afr(iv_apa_rec ,ov_error_rec ,iv_axn_rec);  
xaeape_en_100_02. chk(  
    iv_ape_rec IN OUT NOCOPY xae_ape_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeape_ex_100_02.CV_CHK_BFR =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeape_ex_100_02.chk_bfr(iv_ape_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeape_ex_100_02.CV_CHK_REP =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeape_ex_100_02.chk_rep(iv_ape_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeape_ex_100_02.CV_CHK_AFR =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeape_ex_100_02.chk_afr(iv_ape_rec ,ov_error_rec ,iv_axn_rec);  
xaeapi_en_100_02. chk(  
    iv_api_rec IN OUT NOCOPY xae_api_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapi_ex_100_02.CV_CHK_BFR =cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapi_ex_100_02.chk_bfr(iv_api_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapi_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapi_ex_100_02.chk_rep(iv_api_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapi_ex_100_02.CV_CHK_AFR =cmncon_cl_000_01.CUSTOMIZED THEN
xaeapi_ex_100_02.chk_afr(iv_api_rec ,ov_error_rec ,iv_axn_rec);
xaeapt_en_100_02.chk(
    iv_apt_rec IN OUT NOCOPY xae_apt_rest_rec_t ,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapt_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapt_ex_100_02.chk_bfr(iv_apt_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapt_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapt_ex_100_02.chk_rep(iv_apt_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapf_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapf_ex_100_02.chk_afr(iv_apt_rec ,ov_error_rec ,iv_axn_rec);
xaeapf_en_100_02.chk(
    iv_apf_rec IN OUT NOCOPY xae_apf_rest_rec_t ,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapf_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapf_ex_100_02.chk_bfr(iv_apf_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapf_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapf_ex_100_02.chk_rep(iv_apf_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapb_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapb_ex_100_02.chk_afr(iv_apf_rec ,ov_error_rec ,iv_axn_rec);
xaeapb_en_100_02.chk( iv_apb_rec IN OUT NOCOPY xae_apb_rest_rec_t ,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapb_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaeapb_ex_100_02.chk_bfr(iv_apb_rec ,ov_error_rec ,iv_axn_rec);
```


REPLACE:-

```
xaeapb_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapb_ex_100_02.chk_rep(iv_apb_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapb_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapb_ex_100_02.chk_afr(iv_apb_rec ,ov_error_rec ,iv_axn_rec);  
xaease_en_100_02.chk (  
    iv_ase_rec IN OUT NOCOPY xae_ase_rest_rec2_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaease_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaease_ex_100_02.chk_bfr(iv_ase_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaease_ex_100_02.CV_CHK_REP =cmncon_cl_000_01.CUSTOMIZED THEN  
xaease_ex_100_02.chk_rep(iv_ase_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaease_ex_100_02.CV_CHK_REP =cmncon_cl_000_01.CUSTOMIZED THEN  
xaease_ex_100_02.chk_afr(iv_ase_rec ,ov_error_rec ,iv_axn_rec);  
xaeavl_en_100_02.chk (  
    iv_avl_rec IN OUT NOCOPY xae_avl_rest_rec2_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeavl_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeavl_ex_100_02.chk_bfr(iv_avl_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeavl_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeavl_ex_100_02.chk_rep(iv_avl_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeavl_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeavl_ex_100_02.chk_afr(iv_avl_rec ,ov_error_rec ,iv_axn_rec);  
xaeapd_en_100_02.chk (  
    iv_apd_rec IN OUT NOCOPY xae_apd_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaeapd_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapd_ex_100_02.chk_bfr(iv_apd_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaeapd_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapd_ex_100_02.chk_rep(iv_apd_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaeapd_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaeapd_ex_100_02.chk_afr(iv_apd_rec ,ov_error_rec ,iv_axn_rec);  
xaebsd_en_100_02.chk (  
    iv_bus_app_rec IN OUT NOCOPY xae_bsd_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebsd_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebsd_ex_100_02.chk_bfr(iv_bus_app_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebsd_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebsd_ex_100_02.chk_rep(iv_bus_app_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebsd_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebsd_ex_100_02.chk_afr(iv_bus_app_rec ,ov_error_rec ,iv_axn_rec);  
xaebsa_en_100_02.chk (  
    iv_bsa_rec IN OUT NOCOPY xae_bsa_rest_rec_t ,  
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebsa_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebsa_ex_100_02.chk_bfr(iv_bsa_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebsa_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebsa_ex_100_02.chk_rep(iv_bsa_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebbsa_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebbsa_ex_100_02.chk_afr(iv_bsa_rec ,ov_error_rec ,iv_axn_rec);
xaebbsen_100_02.chk(
iv_bsp_rec IN OUT NOCOPY xae_bsp_rest_rec_t ,
ov_error_rec IN OUT NOCOPY xae_error_rec_t,
iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebbsp_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebbsp_ex_100_02.chk_bfr(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebbsp_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaebbsp_ex_100_02.chk_rep(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebbsl_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebbsl_ex_100_02.chk_afr(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);
xaebbsl_en_100_02.chk(
iv_bsl_rec IN OUT NOCOPY xae_bsl_rest_rec_t ,
ov_error_rec IN OUT NOCOPY xae_error_rec_t,
iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebbsl_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebbsl_ex_100_02.chk_bfr(iv_bsl_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebbsl_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xaebbsl_ex_100_02.chk_rep(iv_bsl_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebbst_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebbst_ex_100_02.chk_afr(iv_bsl_rec ,ov_error_rec ,iv_axn_rec);
xaebbst_en_100_02.chk(
    iv_bst_rec IN OUT NOCOPY xae_bst_rest_rec_t ,
    ov_error_rec IN OUT NOCOPY xae_error_rec_t,
    iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebbst_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xaebbst_ex_100_02.chk_bfr(iv_bst_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebst_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebst_ex_100_02.chk_rep(iv_bst_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebst_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebst_ex_100_02.chk_afr(iv_bst_rec ,ov_error_rec ,iv_axn_rec);  
xaebst_en_100_02.chk (  
iv_bsf_rec IN OUT NOCOPY xae_bsf_rest_rec_t ,  
ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebst_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebst_ex_100_02.chk_bfr(iv_bsf_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebst_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebst_ex_100_02.chk_rep(iv_bsf_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebst_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebst_ex_100_02.chk_afr(iv_bsf_rec ,ov_error_rec ,iv_axn_rec);  
xaebst_en_100_02.chk (  
iv_bsb_rec IN OUT NOCOPY xae_bsb_rest_rec_t ,  
ov_error_rec IN OUT NOCOPY xae_error_rec_t,  
iv_axn_rec IN OUT NOCOPY xae_axn_evw%ROWTYPE)
```

BEFORE:-

```
xaebst_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebst_ex_100_02.chk_bfr(iv_bsb_rec ,ov_error_rec ,iv_axn_rec);
```

REPLACE:-

```
xaebst_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebst_ex_100_02.chk_rep(iv_bsb_rec ,ov_error_rec ,iv_axn_rec);
```

AFTER:-

```
xaebst_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN  
xaebst_ex_100_02.chk_afr(iv_bsb_rec ,ov_error_rec ,iv_axn_rec);
```

Extensible parameters are Tab Type object

```

CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     DATE );

```

12.27 Application ACH (POST)

Below mentioned table has element name which indicates which type of custom data is passed in the custom element block, enclosing the name and its value in keyname and keyvalue respectively.

**Note:**

This block is unbounded and is part of other blocks. Example: Address Block.

Table 12-24 Application ACH

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML FORMAT

```

<CustomFields>
  <CustomUserDefinedStringData>
    <KeyName>str1234</KeyName>
    <KeyValue>str1234</KeyValue>
  </CustomUserDefinedStringData>
  <CustomUserDefinedNumberData>
    <KeyName>str1234</KeyName>
    <KeyValue>3.1415926535</KeyValue>
  </CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName>str1234</KeyName>
    <KeyValue>2012-12-13T12:12:12</KeyValue>
  </CustomUserDefinedDateData>
</CustomFields>

```

Sample JSON

```
"CustomFields": {
  "CustomUserDefinedStringData": {
    "KeyName": "str1234",
    "KeyValue": "str1234"
  },
  "CustomUserDefinedNumberData": {
    "KeyName": "str1234",
    "KeyValue": "3.1415926535"
  },
  "CustomUserDefinedDateData": {
    "KeyName": "str1234",
    "KeyValue": "2012-12-13T12:12:12"
  }
}
```

Below are the package details for applicationupdate web service

Wrapper Engine package:-

```
xaeaac_ew_100_01. xaeaac_ew_100_01 (iv_xae_app_aac_rec_t IN OUT
xae_app_aac_rec_t);
```

Main Engine package:-

```
xaeaac_em_100_01. post_ach (iv_xae_app_aac_rec_t IN OUT xae_app_aac_rec_t);
xaeaac_em_100_01. get_ach (iv_xae_app_aac_rec_t IN OUT xae_app_aac_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01. post_ach();

BEFORE:-

```
xaeaac_ex_100_01.cv_post_ach_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
  xaeaac_ex_100_01.post_ach_bfr(iv_xae_app_aac_rec_t);
```

REPLACE:-

```
xaeaac_ex_100_01.cv_post_ach_rep = cmncon_cl_000_01.CUSTOMIZED THEN
  xaeaac_ex_100_01.post_ach_rep(iv_xae_app_aac_rec_t);
```

AFTER:-

```
xaeaac_ex_100_01.cv_post_ach_afr = cmncon_cl_000_01.CUSTOMIZED THEN
  xaeaac_ex_100_01.post_ach_afr(iv_xae_app_aac_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01. get_ach();

BEFORE:-

```
xaeaac_ex_100_01.cv_get_ach_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
  xaeaac_ex_100_01.get_ach_bfr(iv_xae_app_aac_rec_t);
```

REPLACE:-

```
xaeaac_ex_100_01.cv_get_ach_rep = cmncon_cl_000_01.CUSTOMIZED THEN
  xaeaac_ex_100_01.get_ach_rep(iv_xae_app_aac_rec_t);
```

AFTER:-

```
xaeaac_ex_100_01.cv_get_ach_afr = cmncon_cl_000_01.CUSTOMIZED THEN
  xaeaac_ex_100_01.get_ach_afr(iv_xae_app_aac_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     DATE );
```

12.28 Application Document upload/download/list Service (POST/GET/GET)

Below are the package details for application document upload/download web service

Wrapper Engine package:-

```
xaeado_ew_100_01. xaeado_ew_100_01 (iv_ado_rec IN OUT xae_ado_req_rec_t);
```

Main Engine package:-

```
xaeado_em_100_01. xaeado_em_100_01 (iv_ado_rec IN OUT xae_ado_req_rec_t);
xaeado_em_100_01. post_app_doc (iv_ado_rec IN OUT xae_ado_req_rec_t);
xaeado_em_100_01. get_app_doc (iv_ado_rec IN OUT xae_ado_req_rec_t);
xaeado_em_100_01. ins_ptt_doc_upload_temp (iv_ado_rec IN OUT
xae_ado_req_rec_t);
xaeado_em_100_01. write_blob (iv_ado_rec IN OUT NOCOPY xae_ado_req_rec_t,
iv_file_id IN NUMBER,
iv_dirname IN VARCHAR2 );
```

Below are the Exit point package details for xaeado_em_100_01. write_blob ();

BEFORE:-

```
xaeado_ex_100_01.cv_write_blob_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
  xaeado_ex_100_01.write_blob_bfr(iv_ado_rec, iv_file_id, iv_dirname);
```

REPLACE:-

```
xaeado_ex_100_01.cv_write_blob_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
    xaeado_ex_100_01.write_blob_rep(iv_ado_rec, iv_file_id, iv_dirname);
```

AFTER:-

```
xaeado_ex_100_01.cv_write_blob_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xaeado_ex_100_01.write_blob_afr(iv_ado_rec, iv_file_id, iv_dirname);
```

**Below are the Exit point package details for xaeado_em_100_01.
ins_ptt_doc_upload_temp();**

BEFORE:-

```
xaeado_ex_100_01.cv_ins_ptt_doc_upload_temp_bfr = cmncon_cl_000_01.CUSTOMIZED  
THEN  
    xaeado_ex_100_01.ins_ptt_doc_upload_temp_bfr(iv_ado_rec);
```

REPLACE:-

```
xaeado_ex_100_01.cv_ins_ptt_doc_upload_temp_rep= cmncon_cl_000_01.CUSTOMIZED  
THEN  
    xaeado_ex_100_01.ins_ptt_doc_upload_temp_rep(iv_ado_rec);
```

AFTER:-

```
xaeado_ex_100_01.cv_ins_ptt_doc_upload_temp_afr= cmncon_cl_000_01.CUSTOMIZED  
THEN  
    xaeado_ex_100_01.ins_ptt_doc_upload_temp_afr(iv_ado_rec);
```

Below are the Exit point package details for xaeado_em_100_01.post_app_doc ();

BEFORE:-

```
xaeado_ex_100_01.cv_post_app_doc_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xaeado_ex_100_01.post_app_doc_bfr(iv_ado_rec);
```

REPLACE:-

```
xaeado_ex_100_01.cv_post_app_doc_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
    xaeado_ex_100_01.post_app_doc_rep(iv_ado_rec);
```

AFTER:-

```
xaeado_ex_100_01.cv_post_app_doc_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xaeado_ex_100_01.post_app_doc_afr(iv_ado_rec);
```

Below are the Exit point package details for xaeado_em_100_01.get_app_doc ();

BEFORE:-

```
xaeado_ex_100_01.cv_get_app_doc_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
    xaeado_ex_100_01.get_app_doc_bfr(iv_ado_rec);
```

REPLACE:-

```
xaeado_ex_100_01.cv_get_app_doc_rep = cmncon_cl_000_01.CUSTOMIZED THEN
    xaeado_ex_100_01.get_app_doc_rep(iv_ado_rec);
```

AFTER:-

```
xaeado_ex_100_01.cv_get_app_doc_afr = cmncon_cl_000_01.CUSTOMIZED THEN
    xaeado_ex_100_01.get_app_doc_afr(iv_ado_rec);
```

12.29 Account Document upload/download/list Service (POST/GET/GET)

Below are the package details for Account document upload/download web service

Wrapper Engine package:-

```
xcsado_ew_100_01. xcsado_ew_100_01 (iv_ado_rec IN OUT xcs_ado_req_rec_t)
```

Main Engine package:-

```
xcsado_em_100_01. xcsado_em_100_01 (iv_ado_rec IN OUT xcs_ado_req_rec_t)
xcsado_em_100_01. post_acc_doc(iv_ado_rec IN OUT xcs_ado_req_rec_t);
xcsado_em_100_01. get_acc_doc (iv_ado_rec IN OUT xcs_ado_req_rec_t);
xcsado_em_100_01. ins_ptt_doc_upload_temp(iv_ado_rec IN OUT NOCOPY
xcs_ado_req_rec_t);
xaeado_em_100_01. write_blob (iv_ado_rec IN OUT NOCOPY xcs_ado_req_rec_t,
iv_file_id IN NUMBER,
iv_dirname IN VARCHAR2);
```

Below are the Exit point package details for xaeado_em_100_01. write_blob ();

BEFORE:-

```
xcsado_ex_100_01.cv_write_blob_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsado_ex_100_01.write_blob_bfr(iv_ado_rec, iv_file_id, iv_dirname);
```

REPLACE:-

```
xcsado_ex_100_01.cv_write_blob_rep = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsado_ex_100_01.write_blob_rep(iv_ado_rec, iv_file_id, iv_dirname);
```

AFTER:-

```
xcsado_ex_100_01.cv_write_blob_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcsado_ex_100_01.write_blob_afr(iv_ado_rec, iv_file_id, iv_dirname);
```

**Below are the Exit point package details for xcsado_em_100_01.
ins_ptt_doc_upload_temp());**

BEFORE:-

```
xcsado_ex_100_01.cv_ins_ptt_doc_upload_temp_bfr = cmncon_cl_000_01.CUSTOMIZED  
THEN  
    xcsado_ex_100_01.ins_ptt_doc_upload_temp_bfr(iv_ado_rec);
```

REPLACE:-

```
xcsado_ex_100_01.cv_ins_ptt_doc_upload_temp_rep = cmncon_cl_000_01.CUSTOMIZED  
THEN  
    xcsado_ex_100_01.ins_ptt_doc_upload_temp_rep (iv_ado_rec);
```

AFTER:-

```
xcsado_ex_100_01.cv_ins_ptt_doc_upload_temp_afr = cmncon_cl_000_01.CUSTOMIZED  
THEN  
    xcsado_ex_100_01.ins_ptt_doc_upload_temp_afr (iv_ado_rec);
```

Below are the Exit point package details for xcsado_em_100_01.post_acc_doc ();

BEFORE:-

```
xcsado_ex_100_01.cv_post_acc_doc_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcsado_ex_100_01.post_acc_doc_bfr(iv_ado_rec);
```

REPLACE:-

```
xcsado_ex_100_01.cv_post_acc_doc_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcsado_ex_100_01.post_acc_doc_rep(iv_ado_rec);
```

AFTER:-

```
xcsado_ex_100_01.cv_post_acc_doc_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcsado_ex_100_01.post_acc_doc_afr(iv_ado_rec);
```

Below are the Exit point package details for xcsado_em_100_01.get_acc_doc ();

BEFORE:-

```
xcsado_ex_100_01.cv_get_acc_doc_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcsado_ex_100_01.get_acc_doc_bfr (iv_ado_rec);
```

REPLACE:-

```
xcsado_ex_100_01.cv_get_acc_doc_rep = cmncon_cl_000_01.CUSTOMIZED THEN
  xcsado_ex_100_01.get_acc_doc_rep (iv_ado_rec);
```

AFTER:-

```
xcsado_ex_100_01.cv_get_acc_doc_afr = cmncon_cl_000_01.CUSTOMIZED THEN
  xcsado_ex_100_01.get_acc_doc_afr (iv_ado_rec);
```

12.30 Application Get Service (GET)

Below are the package details for application get web service

Wrapper Engine package:-

```
xaechk_ew_100_02. xaechk_ew_100_02 (iv_app_rec IN OUT NOCOPY
  xae_app_rest_rec_t,ov_error_rec IN OUT NOCOPY xae_error_rec_t);
```

Main Engine package:-

```
xaechk_en_100_02. check_status (iv_app_rec IN OUT NOCOPY
  xae_app_rest_rec_t,ov_error_rec IN OUT NOCOPY xae_error_rec_t)
xaechk_en_100_02. populate_application_response(iv_app_rec IN OUT NOCOPY
  xae_app_rest_rec_t,ov_error_rec IN OUT NOCOPY xae_error_rec_t)
```

Below are the Exit point package details for xaechk_en_100_02. check_status ();

BEFORE:-

```
xaechk_ex_100_02.CV_CHECK_STATUS_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
  xaechk_ex_100_02.check_status_bfr( iv_app_rec);
```

REPLACE:-

```
xaechk_ex_100_02.CV_CHECK_STATUS_REP = cmncon_cl_000_01.CUSTOMIZED THEN
  xaechk_ex_100_02.check_status_rep(iv_app_rec);
```

AFTER:-

```
xaechk_ex_100_02.CV_CHECK_STATUS_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
  xaechk_ex_100_02.check_status_afr(iv_app_rec);
```

12.31 Scheduler Force ReSubmit Service (PUT)

Below are the package details for force resubmit job set web service

Wrapper Engine package:-

```
xaejbs_ew_100_02.xaejbs_ew_100_02 (iv_jbs_rec_t IN OUT NOCOPY
job_jbs_rec_t,iv_axn_rec_t IN OUT NOCOPY xac_axn_rec_t);
```

Main Engine package:-

```
xaejbs_em_100_02. Submit (iv_jbs_rec_t IN OUT NOCOPY
job_jbs_rec_t,iv_axn_rec_t IN OUT NOCOPY xac_axn_rec_t);
```

Below are the Exit point package details for xaejbs_em_100_02. Submit ();

BEFORE:-

```
xaejbs_ex_100_01.cv_submit_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
    xaejbs_ex_100_01.submit_bfr (iv_jbs_rec_t, iv_axn_rec_t);
```

REPLACE:-

```
xaejbs_ex_100_01.cv_submit_rep = cmncon_cl_000_01.CUSTOMIZED THEN
    xaejbs_ex_100_01.submit_rep (iv_jbs_rec_t, iv_axn_rec_t);
```

AFTER:-

```
xaejbs_ex_100_01.cv_submit_afr = cmncon_cl_000_01.CUSTOMIZED THEN
    xaejbs_ex_100_01.submit_afr (iv_jbs_rec_t, iv_axn_rec_t);
```

12.32 Credit Limit Service (Customer/Business) [GET]

Below mentioned are the custom fields. These fields will be part of response.

Table 12-25 Credit Limit Service (Customer/Business)

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  < StringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </ StringData>
  <NumberData>
    <KeyName> NumberName </KeyName>
```

```

        <KeyValue> NumberValue </KeyValue>
    </NumberData>
    <DateData>
        <KeyName> DateName </KeyName>
        <KeyValue> DateValue</KeyValue>
    </DateData>
</Custom>

```

Sample JSON

```

"Custom": {
  "StringData": [
    {
      "KeyName": " StringName ",
      "KeyValue": " StringValue "
    }
  ],
  "NumberData": [
    {
      "KeyName": " NumberName ",
      "KeyValue": NumberValue
    }
  ],
  "DateData": [
    {
      "KeyName": " DateName ",
      "KeyValue": " DateValue"
    }
  ]
}

```

Below are the package details for applicationupdate web service

Wrapper Engine package:-

```
xcsctl_ew_100_01.xcsctl_ew_100_01 (iv_xcs_ctl_rec IN OUT xcs_ctl_rec_t);
```

Main Engine package:-

```
xcsctl_em_100_01.get_ctl (iv_xcs_ctl_rec IN OUT xcs_ctl_rec_t);
```

Below are the Exit point package details for xcsctl_em_100_01.get_ctl();

BEFORE:-

```
xcsctl_ex_100_01.cv_get_ctl_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
  xcsctl_ex_100_01.get_ctl_bfr(iv_xcs_ctl_rec_t);
```

REPLACE:-

```
xcsctl_ex_100_01.cv_get_ctl_rep = cmncon_cl_000_01.CUSTOMIZED THEN
  xcsctl_ex_100_01.get_ctl_rep(iv_xcs_ctl_rec_t);
```

AFTER:-

```
xcscrl_ex_100_01.cv_get_crl_afr = cmncon_cl_000_01.CUSTOMIZED THEN
  xcscrl_ex_100_01.get_crl_afr(iv_xcs_crl_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     DATE );
```

12.33 Business Comments Service (GET/POST)

Below mentioned are the custom fields. These fields will be part of response for both GET and POST service and will be part of request for POST service.

Table 12-26 Business Comments Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  < StringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </ StringData>
  <NumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </NumberData>
  <DateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </DateData>
</Custom>
```

Sample JSON

```
"Custom": {
  "StringData": [
    {
      "KeyName": " StringName ",
      "KeyValue": " StringValue "
    }
  ],
  "NumberData": [
    {
      "KeyName": " NumberName ",
      "KeyValue": NumberValue
    }
  ],
  "DateData": [
    {
      "KeyName": " DateName ",
      "KeyValue": " DateValue"
    }
  ]
}
```

Below are the package details for applicationupdate web service

Wrapper Engine package:-

```
xcsxcm_ew_100_01.xcsxcm_ew_100_01 (iv_acm_rec_t IN OUT xcs_acm_axn_rec_t);
```

Main Engine package:-

```
xcsxcm_em_100_01.post_acm (iv_acm_rec_t IN OUT xcs_acm_axn_rec_t);
xcsxcm_em_100_01.get_acm(iv_acm_rec_t IN OUT xcs_acm_axn_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01.post_acm();

BEFORE:-

```
xcsxcm_ex_100_01.cv_post_acm_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
  xcsxcm_ex_100_01.post_acm_bfr(iv_acm_rec_t);
```

REPLACE:-

```
xcsxcm_ex_100_01.cv_post_acm_rep = cmncon_cl_000_01.CUSTOMIZED THEN
  xcsxcm_ex_100_01.post_acm_rep(iv_acm_rec_t);
```

AFTER:-

```
xcsxcm_ex_100_01.cv_post_acm_afr = cmncon_cl_000_01.CUSTOMIZED THEN
  xcsxcm_ex_100_01.post_acm_afr(iv_acm_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01.get_acm();

BEFORE:-

```
xcsxcm_ex_100_01.cv_get_acm_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsxcm_ex_100_01.get_acm_bfr(iv_acm_rec_t);
```

REPLACE:-

```
xcsxcm_ex_100_01.cv_get_acm_rep = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsxcm_ex_100_01.get_acm_rep(iv_acm_rec_t);
```

AFTER:-

```
xcsxcm_ex_100_01.cv_get_acm_afr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsxcm_ex_100_01.get_acm_afr(iv_acm_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
    ATT_NAME      VARCHAR2(30),
    ATT_VALUE     VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
    ATT_NAME      VARCHAR2(30),
    ATT_VALUE     NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
    ATT_NAME      VARCHAR2(30),
    ATT_VALUE     DATE );
```

12.34 Customer Comments Service (GET/POST)

Below mentioned are the custom fields. These fields will be part of response for both GET and POST service and will be part of request for POST service.

Table 12-27 Customer Comments Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  < StringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </ StringData>
  <NumberData>
    <KeyName> NumberName </KeyName>
```



```

        <KeyValue> NumberValue </KeyValue>
    </NumberData>
    <DateData>
        <KeyName> DateName </KeyName>
        <KeyValue> DateValue</KeyValue>
    </DateData>
</Custom>

```

Sample JSON

```

"Custom": {
  "StringData": [
    {
      "KeyName": " StringName ",
      "KeyValue": " StringValue "
    }
  ],
  "NumberData": [
    {
      "KeyName": " NumberName ",
      "KeyValue": NumberValue
    }
  ],
  "DateData": [
    {
      "KeyName": " DateName ",
      "KeyValue": " DateValue"
    }
  ]
}

```

Below are the package details for applicationupdate web service

Wrapper Engine package:-

```
xcsxcm_ew_100_01.xcsxcm_ew_100_01 (iv_acm_rec_t IN OUT xcs_acm_axn_rec_t);
```

Main Engine package:-

```
xcsxcm_em_100_01.post_acm (iv_acm_rec_t IN OUT xcs_acm_axn_rec_t);
xcsxcm_em_100_01.get_acm(iv_acm_rec_t IN OUT xcs_acm_axn_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01.post_acm();

BEFORE:-

```
xcsxcm_ex_100_01.cv_post_acm_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsxcm_ex_100_01.post_acm_bfr(iv_acm_rec_t);
```

REPLACE:-

```
xcsxcm_ex_100_01.cv_post_acm_rep = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsxcm_ex_100_01.post_acm_rep(iv_acm_rec_t);
```

AFTER:-

```
xcsxcm_ex_100_01.cv_post_acm_afr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsxcm_ex_100_01.post_acm_afr(iv_acm_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01. get_acm();

BEFORE:-

```
xcsxcm_ex_100_01.cv_get_acm_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsxcm_ex_100_01.get_acm_bfr(iv_acm_rec_t);
```

REPLACE:-

```
xcsxcm_ex_100_01.cv_get_acm_rep = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsxcm_ex_100_01.get_acm_rep(iv_acm_rec_t);
```

AFTER:-

```
xcsxcm_ex_100_01.cv_get_acm_afr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsxcm_ex_100_01.get_acm_afr(iv_acm_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE   VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE   NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE   DATE );
```

12.35 Customer Preference Service (GET/POST/PUT)

Below mentioned are the custom fields. These fields will be part of response for GET, PUT and POST service and will be part of request for POST and PUT service.

Table 12-28 Customer Preference Service

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  < CustomUserDefinedStringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </ CustomUserDefinedStringData>
  < CustomUserDefinedNumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </ CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </ CustomUserDefinedDateData>
</Custom>
```

Sample JSON

```
"Custom": {
  " CustomUserDefinedStringData": [
    {
      "KeyName": " StringName ",
      "KeyValue": " StringValue "
    }
  ],
  " CustomUserDefinedNumberData": [
    {
      "KeyName": " NumberName ",
      "KeyValue": NumberValue
    }
  ],
  " CustomUserDefinedDateData": [
    {
      "KeyName": " DateName ",
      "KeyValue": " DateValue"
    }
  ]
}
```

Below are the package details for applicationupdate web service

Wrapper Engine package:-

```
xcscpr_ew_100_01.xcscpr_ew_100_01 (iv_xcs_cpr_rec_t IN OUT xcs_cpr_rec_t);
```

Main Engine package:-

```
xcscpr_em_100_01.post_cpr (iv_xcs_cpr_rec_t IN OUT xcs_cpr_rec_t);
xcscpr_em_100_01.get_cpr(iv_xcs_cpr_rec_t IN OUT xcs_cpr_rec_t);
xcscpr_em_100_01.put_cpr(iv_xcs_cpr_rec_t IN OUT xcs_cpr_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01. post_cpr();

BEFORE:-

```
xcscpr_ex_100_01.cv_post_cpr_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcscpr_ex_100_01.post_cpr_bfr(iv_acm_rec_t);
```

REPLACE:-

```
xcscpr_ex_100_01.cv_post_cpr_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcscpr_ex_100_01.post_cpr_rep(iv_acm_rec_t);
```

AFTER:-

```
xcscpr_ex_100_01.cv_post_cpr_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcscpr_ex_100_01.post_cpr_afr(iv_acm_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01. get_cpr();

BEFORE:-

```
xcscpr_ex_100_01.cv_get_cpr_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcscpr_ex_100_01.get_cpr_bfr(iv_acm_rec_t);
```

REPLACE:-

```
xcscpr_ex_100_01.cv_get_cpr_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcscpr_ex_100_01.get_cpr_rep(iv_acm_rec_t);
```

AFTER:-

```
xcscpr_ex_100_01.cv_get_cpr_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcscpr_ex_100_01.get_cpr_afr(iv_acm_rec_t);
```

Below are the Exit point package details for xaeaac_em_100_01. put_cpr();

BEFORE:-

```
xcscpr_ex_100_01.cv_put_cpr_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcscpr_ex_100_01.put_cpr_bfr(iv_acm_rec_t);
```

REPLACE:-

```
xcscpr_ex_100_01.cv_put_cpr_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcscpr_ex_100_01.put_cpr_rep(iv_acm_rec_t);
```

AFTER:-

```
xcscpr_ex_100_01.cv_put_cpr_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
    xcscpr_ex_100_01.put_cpr_afr(iv_acm_rec_t);
```

Extensible parameters are Tab Type object

```

CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     DATE );

```

12.36 Scenario Analysis Service (PUT)

Below mentioned are the custom fields. These fields will be part of both request and response.

Table 12-29 Scenario Analysis Service

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  < CustomUserDefinedStringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </ CustomUserDefinedStringData>
  < CustomUserDefinedNumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </ CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </ CustomUserDefinedDateData>
</Custom>

```

Sample JSON

```

"Custom": {
  " CustomUserDefinedStringData": [
    {
      "KeyName": " StringName ",
      "KeyValue": " StringValue "
    }
  ],

```

```

    " CustomUserDefinedNumberData": [
      {
        "KeyName": " NumberName ",
        "KeyValue": NumberValue
      }
    ],
    " CustomUserDefinedDateData": [
      {
        "KeyName": " DateName ",
        "KeyValue": " DateValue"
      }
    ]
  }
}

```

12.37 Transaction Parameters Service (GET)

Below mentioned are the custom fields. These fields will be part of response.

Table 12-30 Transaction Parameters Service

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyValue	String
CustomUserDefinedNumberData	KeyName	String
	KeyValue	Number (Decimal)
CustomUserDefinedDateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  < CustomUserDefinedStringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </ CustomUserDefinedStringData>
  < CustomUserDefinedNumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </ CustomUserDefinedNumberData>
  <CustomUserDefinedDateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </ CustomUserDefinedDateData>
</Custom>

```

Sample JSON

```

"Custom": {
  " CustomUserDefinedStringData": [
    {
      "KeyName": " StringName ",
      "KeyValue": " StringValue "
    }
  ]
}

```

```

    }
  ],
  " CustomUserDefinedNumberData": [
    {
      "KeyName": " NumberName ",
      "KeyValue": NumberValue
    }
  ],
  " CustomUserDefinedDateData": [
    {
      "KeyName": " DateName ",
      "KeyValue": " DateValue"
    }
  ]
}

```

Below are the package details for transaction parameter web service

Wrapper Engine package:-

```

xtpprc_ew_100_01.xtpprc_ew_100_01 (iv_req_rec_t IN xws_tcd_req_rec_t,
iv_res_rec_t IN OUT xws_tcp_resp_rec_t);

```

Main Engine package:-

```

xtpprc_em_100_01.xtpprc_em_100_01 (iv_req_rec_t IN xws_tcd_req_rec_t,
iv_res_rec_t IN OUT xws_tcp_resp_rec_t);

```

Below are the Exit point package details for xtpprc_em_100_01.xtpprc_em_100_01());

BEFORE:-

```

xtpprc_ex_100_01.cv_xtpprc_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
  xtpprc_ex_100_01.xtpprc_em_100_01_bfr(iv_req_rec_t, iv_res_rec_t);

```

REPLACE:-

```

xtpprc_ex_100_01.cv_xtpprc_rep = cmncon_cl_000_01.CUSTOMIZED THEN
  xtpprc_ex_100_01.xtpprc_em_100_01_rep(iv_req_rec_t, iv_res_rec_t);

```

AFTER:-

```

xtpprc_ex_100_01.cv_xtpprc_afr = cmncon_cl_000_01.CUSTOMIZED THEN
  xtpprc_ex_100_01.xtpprc_em_100_01_afr(iv_req_rec_t, iv_res_rec_t);

```

Extensible parameters are Tab Type object

```

CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     NUMBER );

```

```
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
  ATT_NAME      VARCHAR2(30),
  ATT_VALUE     DATE );
```

12.38 Asset Tracking Attribute Service(PUT)

Below mentioned are the custom fields. These fields will be part of both request and response.

Table 12-31 Asset Tracking Attribute Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  < StringData>
    <KeyName> StringName </KeyName>
    <KeyValue> StringValue </KeyValue>
  </ StringData>
  < NumberData>
    <KeyName> NumberName </KeyName>
    <KeyValue> NumberValue </KeyValue>
  </ NumberData>
  <DateData>
    <KeyName> DateName </KeyName>
    <KeyValue> DateValue</KeyValue>
  </ DateData>
</Custom>
```

Sample JSON

```
"Custom": {
  "StringData": [
    {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    }
  ],
  "NumberData": [
    {
      "KeyName": "NumberName",
      "KeyValue": NumberValue
    }
  ],
  "DateData": [
    {
```



```

        "KeyName": "DateName",
        "KeyValue": "DateValue"
    }
]
}

```

Below are the package details for transaction parameter web service

Wrapper Engine package:-

```
xcsata_ew_100_01 .xcsata_ew_100_01 (iv_xcs_ata_rec IN OUT xcs_ata_atr_rec_t);
```

Main Engine package:-

```
xcsata_em_100_01.update_asset_tacking_atr (iv_xcs_ata_rec IN OUT
xcs_ata_atr_rec_t);
```

**Below are the Exit point package details for
xcsata_em_100_01.update_asset_tacking_atr ();**

BEFORE:-

```
xcsata_ex_100_01.cv_upd_ast_tracking_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsata_ex_100_01.upd_ast_tracking_bfr(iv_xcs_ata_rec);
```

REPLACE:-

```
xcsata_ex_100_01.cv_upd_ast_tracking_rep = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsata_ex_100_01.upd_ast_tracking_rep(iv_xcs_ata_rec);
```

AFTER:-

```
xcsata_ex_100_01.cv_upd_ast_tracking_afr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsata_ex_100_01.upd_ast_tracking_afr(iv_xcs_ata_rec);
```

Extensible parameters are Tab Type object

```

CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE    VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE    NUMBER);
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE    DATE);

```

12.39 Business Tracking Attribute Service(PUT)

Below mentioned are the custom fields. These fields will be part of both request and response.

Table 12-32 Business Tracking Attribute Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  < StringData>
    <KeyName>StringName</KeyName>
    <KeyValue>StringValue</KeyValue>
  </ StringData>
  < NumberData>
    <KeyName>NumberName</KeyName>
    <KeyValue>NumberValue</KeyValue>
  </ NumberData>
  <DateData>
    <KeyName>DateName</KeyName>
    <KeyValue>DateValue</KeyValue>
  </DateData>
</Custom>
```

Sample JSON

```
"Custom": {
  "StringData": [
    {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    }
  ],
  "NumberData": [
    {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    }
  ],
  "DateData": [
    {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  ]
}
```

Below are the package details for transaction parameter web service

Wrapper Engine package:-

```
xcsbta_ew_100_01.xcsbta_ew_100_01 (iv_xcs_bta_rec IN OUT xcs_bta_atr_rec_t);
```

Main Engine package:-

```
xcsbta_em_100_01.update_business_tacking_atr (iv_xcs_bta_rec IN OUT  
xcs_bta_atr_rec_t);
```

**Below are the Exit point package details for
xcsbta_em_100_01.update_business_tacking_atr();**

BEFORE:-

```
xcsbta_ex_100_01.cv_upd_bsn_tracking_bfr = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsbta_ex_100_01.upd_bsn_tracking_bfr(iv_xcs_bta_rec);
```

REPLACE:-

```
xcsbta_ex_100_01.cv_upd_bsn_tracking_rep = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsbta_ex_100_01.upd_bsn_tracking_rep(iv_xcs_bta_rec);
```

AFTER:-

```
xcsbta_ex_100_01.cv_upd_bsn_tracking_afr = cmncon_cl_000_01.CUSTOMIZED THEN  
xcsbta_ex_100_01.upd_bsn_tracking_afr(iv_xcs_bta_rec);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   VARCHAR2(4000));  
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   NUMBER);  
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   DATE);
```

12.40 Customer Tracking Attribute Service(PUT)

Below mentioned are the custom fields. These fields will be part of both request and response.

Table 12-33 Customer Tracking Attribute Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)

Table 12-33 (Cont.) Customer Tracking Attribute Service

Element name	Sub element	Data type
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  < StringData>
    <KeyName>StringName</KeyName>
    <KeyValue>StringValue</KeyValue>
  </ StringData>
  < NumberData>
    <KeyName>NumberName</KeyName>
    <KeyValue>NumberValue</KeyValue>
  </ NumberData>
  <DateData>
    <KeyName>DateName</KeyName>
    <KeyValue>DateValue</KeyValue>
  </DateData>
</Custom>
```

Sample JSON

```
"Custom": {
  "StringData": [
    {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    }
  ],
  "NumberData": [
    {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    }
  ],
  "DateData": [
    {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  ]
}
```

Below are the package details for transaction parameter web service

Wrapper Engine package:-

```
xcscta_ew_100_01.xcscta_ew_100_01 (iv_xcs_cta_rec IN OUT xcs_cta_atr_rec_t);
```

Main Engine package:-

```
xcscta_em_100_01.update_customer_tracking_atr (iv_xcs_cta_rec IN OUT
xcs_cta_atr_rec_t);
```

**Below are the Exit point package details for
xcscta_em_100_01.update_customer_tracking_atr();**

BEFORE:-

```
xcscta_ex_100_01.cv_upd_cust_tracking_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
xcscta_ex_100_01.upd_cust_tracking_bfr(iv_xcs_cta_rec);
```

REPLACE:-

```
xcscta_ex_100_01.cv_upd_cust_tracking_rep = cmncon_cl_000_01.CUSTOMIZED THEN
xcscta_ex_100_01.upd_cust_tracking_rep(iv_xcs_cta_rec);
```

AFTER:-

```
xcscta_ex_100_01.cv_upd_cust_tracking_afr = cmncon_cl_000_01.CUSTOMIZED THEN
xcscta_ex_100_01.upd_cust_tracking_afr(iv_xcs_cta_rec);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
ATT_NAME    VARCHAR2(30),
ATT_VALUE    VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
ATT_NAME    VARCHAR2(30),
ATT_VALUE    NUMBER);
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
ATT_NAME    VARCHAR2(30),
ATT_VALUE    DATE);
```

12.41 Account Tracking Attribute Service(PUT)

Below mentioned are the custom fields. These fields will be part of both request and response.

Table 12-34 Account Tracking Attribute Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  < StringData>
    <KeyName>StringName</KeyName>
    <KeyValue>StringValue</KeyValue>
  </ StringData>
  < NumberData>
    <KeyName>NumberName</KeyName>
    <KeyValue>NumberValue</KeyValue>
  </ NumberData>
  <DateData>
    <KeyName>DateName</KeyName>
    <KeyValue>DateValue</KeyValue>
  </DateData>
</Custom>

```

Sample JSON

```

"Custom": {
  "StringData": [
    {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    }
  ],
  "NumberData": [
    {
      "KeyName": "NumberName",
      "KeyValue": "NumberValue"
    }
  ],
  "DateData": [
    {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  ]
}

```

Below are the package details for transaction parameter web service

Wrapper Engine package:-

```
xcsact_ew_100_01.xcsact_ew_100_01 (iv_xcs_act_rec IN OUT xcs_act_atr_rec_t);
```

Main Engine package:-

```
xcsact_em_100_01.update_account_tacking_atr (iv_xcs_act_rec IN OUT
xcs_act_atr_rec_t);
```

**Below are the Exit point package details for
xcsact_em_100_01.update_account_tacking_atr();**

BEFORE:-

```
xcsact_ex_100_01.cv_upd_acnt_tracking_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsact_ex_100_01.upd_acnt_tracking_bfr(iv_xcs_act_rec);
```

REPLACE:-

```
xcsact_ex_100_01.cv_upd_acnt_tracking_rep = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsact_ex_100_01.upd_acnt_tracking_rep(iv_xcs_act_rec);
```

AFTER:-

```
xcsact_ex_100_01.cv_upd_acnt_tracking_afr = cmncon_cl_000_01.CUSTOMIZED THEN
    xcsact_ex_100_01.upd_acnt_tracking_afr(iv_xcs_act_rec);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE    VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE    NUMBER);
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE    DATE);
```

12.42 Credit Bureau Web Service(PUT)

Below mentioned are the custom fields. These fields will be part of both request and response.

Table 12-35 Credit Bureau Web Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  < StringData>
    <KeyName>StringName</KeyName>
    <KeyValue>StringValue</KeyValue>
  </ StringData>
  < NumberData>
    <KeyName>NumberName</KeyName>
    <KeyValue>NumberValue</KeyValue>
```

```

    </ NumberData>
    <DateData>
        <KeyName>DateName</KeyName>
        <KeyValue>DateValue</KeyValue>
    </DateData>
</Custom>

```

Sample JSON

```

"Custom": {
  "StringData": [
    {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    }
  ],
  "NumberData": [
    {
      "KeyName": "NumberName",
      "KeyValue": NumberValue
    }
  ],
  "DateData": [
    {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  ]
}

```

Below are the package details for credit bureau web service

Wrapper Engine packages :-

```

xaecrb_ew_100_03.xaecrb_ew_100_03 (iv_crq_rec IN OUT NOCOPY
xcs_crb_crq_res_rec_t);
xaecrb_em_100_03. xaecrb_em_100_03 (iv_crq_rec IN OUT NOCOPY
xcs_crb_crq_res_rec_t);

```

Main Engine package:-

```

xaecrb_en_100_03.put_credit_request_detail (iv_crq_rec IN OUT NOCOPY
xcs_crb_crq_res_rec_t);

```

**Below are the Exit point package details for
xaecrb_en_100_03.put_credit_request_detail ();**

BEFORE:-

```

IF xaecrb_ex_100_03.cv_put_cr_req_dtl_bfr = cmncon_cl_000_01.customized THEN
    xaecrb_ex_100_03.put_cr_req_dtl_bfr (iv_crq_rec);

```


REPLACE:-

```
IF xaecrb_ex_100_03.cv_put_cr_req_dtl_rep = cmncon_cl_000_01.customized THEN
    xaecrb_ex_100_03.put_cr_req_dtl_rep (iv_crq_rec);
```

AFTER:-

```
IF xaecrb_ex_100_03.cv_put_cr_req_dtl_afr = cmncon_cl_000_01.customized THEN
    xaecrb_ex_100_03.put_cr_req_dtl_afr (iv_crq_rec);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE   VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE   NUMBER);
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
    ATT_NAME    VARCHAR2(30),
    ATT_VALUE   DATE);
```

12.43 Delete Account Web Service(DELETE)

Below mentioned are the custom fields. These fields will be part of both request and response.

Table 12-36 Delete Account Web Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  < StringData>
    <KeyName>StringName</KeyName>
    <KeyValue>StringValue</KeyValue>
  </ StringData>
  < NumberData>
    <KeyName>NumberName</KeyName>
    <KeyValue>NumberValue</KeyValue>
  </ NumberData>
  <DateData>
    <KeyName>DateName</KeyName>
    <KeyValue>DateValue</KeyValue>
```

```
</DateData>
</Custom>
```

Sample JSON

```
"Custom": {
  "StringData": [
    {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    }
  ],
  "NumberData": [
    {
      "KeyName": "NumberName",
      "KeyValue": NumberValue
    }
  ],
  "DateData": [
    {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  ]
}
```

Below are the package details for delete account web service

Wrapper Engine packages :-

```
xcsdac_ew_100_01.xcsdac_ew_100_01 (iv_xcs_dac_rec IN OUT xcs_dac_req_rec_t)
```

Main Engine package:-

```
xcsdac_em_100_01.xcsdac_em_100_01(iv_xcs_dac_rec IN OUT xcs_dac_req_rec_t));
```

Below are the Exit point package details for xcsdac_em_100_01.xcsdac_em_100_01 ();

BEFORE:-

```
IF xcsdac_ex_100_01.cv_xcsdac_em_100_01_bfr =cmncon_cl_000_01.CUSTOMIZED THEN
  xcsdac_ex_100_01.xcsdac_em_100_01_bfr(iv_xcs_dac_rec);
```

REPLACE:-

```
IF xcsdac_ex_100_01.cv_xcsdac_em_100_01_rep =cmncon_cl_000_01.CUSTOMIZED THEN
  xcsdac_ex_100_01.xcsdac_em_100_01_rep(iv_xcs_dac_rec);
```

AFTER:-

```
IF xcsdac_ex_100_01.cv_xcsdac_em_100_01_afr =cmncon_cl_000_01.CUSTOMIZED THEN
  xcsdac_ex_100_01.xcsdac_em_100_01_afr(iv_xcs_dac_rec);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   VARCHAR2(4000));  
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   NUMBER);  
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (  
    ATT_NAME    VARCHAR2(30),  
    ATT_VALUE   DATE);
```

12.44 New Customization for RESTful Web Service

Below are the list of services introduced and it follows new naming standard for custom elements in request/response.

- Advance Disbursement Service (GET)
- Advance Disbursement Service (POST)
- Usage and Rental Service (GET)
- Collateral Usage History (POST)
- Asset Service (GET)
- Asset Service (POST)
- Application Contract Service (POST)
- User Create Service (POST)
- Rental Application Entry Service (POST)
- Vendor Create Service (POST)
- Vendor Update Service (PUT)
- Vendor Detail Fetch Service (GET)
- Vendor Comment Create service (POST)
- Vendor Comment Fetch Service (GET)
- Asset Valuation Create Service (POST)
- Asset Valuation Update Service (PUT)
- Asset Valuation Fetch Service (GET)
- Producer Comment Create Service (POST)
- Producer Comment Fetch Service (GET)
- Producer Contact Fetch Service (GET)
- Producer Contact Create Service (POST)
- Producer Compansaction Fetch Service (GET)
- Producer Holdback Fetch Service (GET)
- Producer Statement Fetch Service (GET)
- Subvention Detail Service (GET)

- Account Monitory Transaction History Fetch Service (GET)
- Webhook get maintenance details (GET)
- Webhook update event status details (PUT)
- Webhook Maintenance Post Service (POST)
- Webhook Maintenance Fetch Service (GET)
- Webhook Maintenance Update Service (PUT)
- User Update Service (PUT)
- Master Account Rolledup Balance Detail Service (GET)
- Application Decision Fetch Service (GET)
- Application Decision Update Service (PUT)
- Work Order Fetch Service (GET)
- Work Order Update Service (PUT)
- Work Order Post Service (POST)

Below mentioned are the custom fields. Custom fields will be supported in request/response for POST/PUT services and will be supported only in response for GET services.

Table 12-37 New Customization for RESTful Web Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <StringData>
    <KeyName>OrgName</KeyName>
    <KeyValue>Oracle</KeyValue>
  </StringData>
  <NumberData>
    <KeyName>BusinessPhoneNumber</KeyName>
    <KeyValue>1234.01</KeyValue>
  </NumberData>
  <DateData>
    <KeyName>CreationDate</KeyName>
    <KeyValue>2017-12-18T00:00:00</KeyValue>
  </DateData>
</Custom>
```

Sample JSON

```
"Custom": {
  "StringData": [
```

```

    {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    }
  ],
  "NumberData": [
    {
      "KeyName": "NumberName",
      "KeyValue": NumberValue
    }
  ],
  "DateData": [
    {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  ]
}

```

Below is the list of services which uses the Tracking Attribute in request/response similar to Custom Elements for extensibility.

- Producer Create Service (POST)
- Producer Update Service (PUT)
- Producer Fetch Service (GET)
- Account Tracking Attribute Fetch Service (GET)

Below mentioned are the Tracking Attribute fields. Tracking Attribute fields will be supported in request/response for POST/PUT services and will be supported only in response for GET services.

Table 12-38 Tracking Attribute fields

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<TrackingAttributes>
  <StringData>
    <KeyName>OrgName</KeyName>
    <KeyValue>Oracle</KeyValue>
  </StringData>
  <NumberData>
    <KeyName>BusinessPhoneNumber</KeyName>
    <KeyValue>1234.01</KeyValue>
  </NumberData>
  <DateData>

```

```

        <KeyName>CreationDate</KeyName>
        <KeyValue>2017-12-18T00:00:00</KeyValue>
    </DateData>
</TrackingAttributes>

```

Sample JSON

```

" TrackingAttributes ": {
  " StringData": [
    {
      "KeyName": " StringName ",
      "KeyValue": " StringValue "
    }
  ],
  " NumberData": [
    {
      "KeyName": " NumberName ",
      "KeyValue": NumberValue
    }
  ],
  " DateData": [
    {
      "KeyName": " DateName ",
      "KeyValue": " DateValue"
    }
  ]
}

```

Below mentioned are the package details and exit points for the above listed services.

Table 12-39 we

Service name	Wrapper Package	Main Package	Exit Point Packages			
			Before	Replace	After	
Advance Disbursement Service (GET)	XBTADV_EW_1 00_01.XBTADV_EW_100_0 1(iv_adv_axn_rec IN OUT xbt_adv_axn_rec_t)	XBTADV_EM_1 00_01.GET_A DV	XBTADV_EX_100_01	GET_ADV_BFR	GET_ADV_REP	GET_ADV_AFR
Advance Disbursement Service (POST)	XBTADV_EW_1 00_01.XBTADV_EW_100_0 1(iv_adv_axn_rec IN OUT	XBTADV_EM_1 00_01.POST_ ADV	XBTADV_EX_100_01			

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
	xbt_adv_axn_rec_t)		POST_ADV_BFR	POST_ADV_REP	POST_ADV_AFR
Application Contract Service (POST)					
	XAECON_EW_100_01.XAECON_EW_100_01(iv_xae_app_con_rec_t IN OUT xae_app_con_rec_t	XAECON_EM_100_01.XAECON_EM_100_01	XAECON_EX_100_01		
			XAECON_EM_100_01_BFR	XAECON_EM_100_01_REP	XAECON_EM_100_01_AFR
Rental Application Entry Service (POST)					
	xaernt_ew_121_01.xaernt_ew_121_01(iv_xae_rnt_req_rec_t IN xae_rnt_req_rec_t ,iv_xae_rnt_res_rec_t OUT xae_error_rec_t)	XAERNT_EM_121_01.XAERNT_EM_121_01	XAERNT_EX_121_01		
			XAERNT_EM_121_01_BFR	XAERNT_EM_121_01_REP	XAERNT_EM_121_01_AFR
		XAERNT_EN_121_01.SET_RENTAL_DETAILS	XAERNT_EX_121_02		
			SET_RENTAL_DETAILS_BFR	SET_RENTAL_DETAILS_REP	SET_RENTAL_DETAILS_AFR
		XAERNT_EN_121_01.INITIALIZE_RNT_DETAILS	XAERNT_EX_121_02		
			INITIALIZE_RNT_DETAILS_BFR	INITIALIZE_RNT_DETAILS_REP	INITIALIZE_RNT_DETAILS_AFR
	xaernt_en_121_01.ins		XAEINS_EX_121_01		
			INS_BFR	INS_REP	INS_AFR

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
Usage and Rental Service (GET)	XCMPRC_EW_1 21_01.XCMPRC_EW_121_0 1(iv_xcm_rnt_rec IN OUT xcm_rnt_req_rec_t)	XCMPRC_EM_1 21_01.XCMPRC_EM_121_01	XCMPRC_EX_121_01		
			XCMPRC_EM_1 21_01_BFR	XCMPRC_EM_1 21_01_REP	XCMPRC_EM_1 21_01_AFR
Asset Service (GET)	XCSASE_EW_1 00_02.XCSASE_EW_100_0 2(iv_xcs_ase_rec IN OUT xcs_get_ase_tab_t)	XCSASE_EM_1 00_02.XCSASE_EM_100_02	XCSASE_EX_100_02		
			XCSASE_EX_1 00_02_BFR	XCSASE_EX_1 00_02_REP	XCSASE_EX_1 00_02_AFR
Asset Service (POST)	XCSASE_EW_1 00_01.XCSASE_EW_100_0 1(iv_xcs_ase_rec IN OUT xcs_ase_req_rec_t)	XCSASE_EM_1 00_01.XCSASE_EM_100_01	XCSASE_EX_100_01		
			XCSASE_EX_1 00_01_BFR	XCSASE_EX_1 00_01_REP	XCSASE_EX_1 00_01_AFR
			XCSASE_EM_1 00_01.POST_ASSET	XCSASE_EX_100_01	
			POST_ASSET_BFR	POST_ASSET_REP	POST_ASSET_AFR
Asset Service (PUT)	XCSASE_EW_1 00_01.XCSASE_EW_100_0 1(iv_xcs_ase_rec IN OUT xcs_ase_req_rec_t)	XCSASE_EM_1 00_01.PUT_ASSET	XCSASE_EX_100_01		
			PUT_ASSET_BFR	PUT_ASSET_REP	PUT_ASSET_AFR

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
Asset Valuations Service (PUT)	XCSASE_EW_1 00_01.XCSAS E_EW_100_0 1(iv_xcs_as e_rec IN OUT xcs_ase_req _rec_t)	XCSASE_EM_1 00_01.PUT_A SE_AVL	XCSASE_EX_100_01		
				PUT_ASE_AVL _BFR	PUT_ASE_AVL _REP
Asset Valuations Service (POST)	XCSASE_EW_1 00_01.XCSAS E_EW_100_0 1(iv_xcs_as e_rec IN OUT xcs_ase_req _rec_t)	XCSASE_EM_1 00_01.POST_ ASE_AVL	XCSASE_EX_100_01		
				POST_ASE_AVL L_BFR	POST_ASE_AVL L_REP
Asset Valuations Service (GET)	XCSASE_EW_1 00_01.XCSAS E_EW_100_0 1(iv_xcs_as e_rec IN OUT xcs_ase_req _rec_t)	XCSASE_EM_1 00_01.GET_A SE_AVL	XCSASE_EX_100_01		
				GET_ASE_AVL _BFR	GET_ASE_AVL _REP
User Service (POST)	XUSUSR_EW_1 00_01.XUSUS R_EW_100_01 (iv_xus_usr _rec_t IN	XUSUSR_EM_1 00_01.XUSUS R_EM_100_01	XUSUSR_EX_100_01		

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
	OUT xus_usr_rec _t)		XUSUSR_EM_1 00_01_BFR	XUSUSR_EM_1 00_01_REP	XUSUSR_EM_1 00_01_AFR
Account					
Monitory	XCSTXN_EW_1	XCSTXN_EM_1	XCSTXN_EX_100_01		
Transaction	00_01.XCSTX	00_01.XCSTX			
History Fetch	N_EW_100_0	N_EM_100_01			
Service (GET)	1(iv_xcs_tx n_hist_rec IN OUT xcs_txn_his t_rec_t)		XCSTXN_EX_1 00_01_BFR	XCSTXN_EX_1 00_01_BFR	XCSTXN_EX_1 00_01_BFR
		XCSTXN_EM_1 00_01.GET_T XN_HIST	XCSTXN_EX_100_01		
			GET_TXN_HIS T_BFR	GET_TXN_HIS T_REP	GET_TXN_HIS T_AFR
Vendor details					
fetch service	XCSVEN_EW_1	XCSVEN_EM_1	XCSVEN_EX_100_01		
(GET)	00_01.XCSVE	00_01.GET_V			
	N_EW_100_0	EN			
	1(iv_ven_re q_t IN OUT xcs_ven_req _t)		GET_VEN_BFR	GET_VEN_REP	GET_VEN_AFR
Vendor					
Comment fetch	XCSVEN_EW_1	XCSVEN_EM_1	XCSVEN_EX_100_01		
service (GET)	00_01.COMME	00_01.GET_C			
	NT(iv_comme nt IN OUT xcs_ven_vcm _rec_t)	OMMENT			
			GET_COMMENT _BFR	GET_COMMENT _REP	GET_COMMENT _AFR

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
Vendor Create Service (POST)	XCSVEN_EW_1 00_01.XCSVE N_EW_100_0 1(iv_ven_re q_t IN OUT xcs_ven_req _t)	XCSVEN_EM_1 00_01.XCSVE N_EM_100_01	XCSVEN_EX_100_01		
			XCSVEN_EX_1 00_01_BFR	XCSVEN_EX_1 00_01_REP	XCSVEN_EX_1 00_01_AFR
		XCSVEN_EM_1 00_01.POST_ VEN	XCSVEN_EX_100_01		
			POST_VEN_BF R	POST_VEN_RE P	POST_VEN_AF R
Vendor Comment Create Service (POST)	XCSVEN_EW_1 00_01.COMME NT(iv_comme nt IN OUT xcs_ven_vcm _rec_t)	XCSVEN_EM_1 00_01.POST_ COMMENT	XCSVEN_EX_100_01		
			POST_COMMEN T_BFR	POST_COMMEN T_REP	POST_COMMEN T_AFR
Vendor Update Service (PUT)	XCSVEN_EW_1 00_01.XCSVE N_EW_100_0 1(iv_ven_re q_t IN OUT xcs_ven_req _t)	XCSVEN_EM_1 00_01.UPD_V EN	XCSVEN_EX_100_01		
			UPD_VEN_BFR	UPD_VEN_REP	UPD_VEN_AFR
Producer Create Service (POST)	XPRPRC_EW_1 00_02.XPRPR C_EW_100_0 2(iv_produc er IN OUT xpr_pro_dtl _rec_t)	XPRPRC_EM_1 00_02. XPRPRC_EM_1 00_02	XPRPRC_EX_100_02		
			XPRPRC_EX_1 00_02_BFR	XPRPRC_EX_1 00_02_REP	XPRPRC_EX_1 00_02_AFR
		XPRPRC _EM_100_02.	XPRPRC_EX_100_02		

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages			
			Before	Replace	After	
		POST_PRODUCER	POST_PRODUCER_BFR	POST_PRODUCER_REP	POST_PRODUCER_AFR	
Producer Fetch Service (GET)	XPRPRC_EW_100_02.XPRPRC_EW_100_02(iv_producer IN OUT xpr_pro_dt1_rec_t)	XPRPRC_EM_100_02.XPRPRC_EM_100_02	XPRPRC_EX_100_02	XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR
		XPRPRC_EM_100_02.GET_PRODUCER	XPRPRC_EX_100_02	XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR
Producer Update Service (PUT)	XPRPRC_EW_100_02.XPRPRC_EW_100_02(iv_producer IN OUT xpr_pro_dt1_rec_t)	XPRPRC_EM_100_02.XPRPRC_EM_100_02	XPRPRC_EX_100_02	XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR
		XPRPRC_EM_100_02.PUT_PRODUCER	XPRPRC_EX_100_02	XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR
Producer Contact Service (POST)	XPRPRC_EW_100_02.XPRPRC_EW_100_02(iv_contact IN OUT xpr_pro_pcn_rec_t)	XPRPRC_EM_100_02.XPRPRC_EM_100_02	XPRPRC_EX_100_02	XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
		XPRPRC _EM_100_02. POST_CONTACT	XPRPRC_EX_100_02		
			POST_COMMENT_BFR	POST_COMMENT_REPLACE	POST_COMMENT_AFR
Producer Contact Service (GET)	XPRPRC_EW_100_02.XPRPRC_EW_100_02(iv_contact IN OUT xpr_pro_pcn_rec_t)	XPRPRC_EM_100_02. XPRPRC_EM_100_02	XPRPRC_EX_100_02		
			XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REPLACE	XPRPRC_EX_100_02_AFR
		XPRPRC _EM_100_02. GET_CONTACT	XPRPRC_EX_100_02		
			GET_CONTACT_BFR	GET_CONTACT_REPLACE	GET_CONTACT_AFR
Producer Comment Service (POST)	XPRPRC_EW_100_02.XPRPRC_EW_100_02(iv_comment IN OUT xpr_pro_pcm_rec_t)	XPRPRC_EM_100_02. XPRPRC_EM_100_02	XPRPRC_EX_100_02		
			XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REPLACE	XPRPRC_EX_100_02_AFR
		XPRPRC _EM_100_02. POST_COMMENT	XPRPRC_EX_100_02		
			POST_COMMENT_BFR	POST_COMMENT_REPLACE	POST_COMMENT_AFR
Producer Comment Service (GET)	XPRPRC_EW_100_02.XPRPRC_EW_100_02(iv_comment IN OUT	XPRPRC_EM_100_02. XPRPRC_EM_100_02	XPRPRC_EX_100_02		

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
	xpr_pro_pcm_rec_t)		XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR
		XPRPRC_EM_100_02.	XPRPRC_EX_100_02		
		GET_COMMENT	GET_COMMENT_BFR	GET_COMMENT_REP	GET_COMMENT_AFR
Producer Subvention Service (GET)	XPRPRC_EW_100_02.XPRPRC_EW_100_02(iv_subvention IN OUT xpr_pro_sbv_rec_t)	XPRPRC_EM_100_02	XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR
		XPRPRC_EM_100_02.	XPRPRC_EX_100_02		
		GET_SUBVENTION	GET_SUBVENTION_BFR	GET_SUBVENTION_REP	GET_SUBVENTION_AFR
Producer Compensation Service (GET)	XPRPRC_EW_100_02.XPRPRC_EW_100_02(iv_compensation IN OUT xpr_pro_cmp_rec_t)	XPRPRC_EM_100_02	XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR
		XPRPRC_EM_100_02.	XPRPRC_EX_100_02		
		GET_COMPENSATION	GET_COMPENSATION_BFR	GET_COMPENSATION_REP	GET_COMPENSATION_AFR

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
Producer Statement Service (GET)	XPRPRC_EW_1_00_02.XPRPRC_EW_100_02(iv_statement IN OUT xpr_pro_psm_rec_t)	XPRPRC_EM_1_00_02.	XPRPRC_EX_100_02		
		XPRPRC_EM_1_00_02	XPRPRC_EX_100_02		
		XPRPRC_EM_1_00_02	XPRPRC_EX_1_00_02_BFR	XPRPRC_EX_1_00_02_REP	XPRPRC_EX_1_00_02_AFR
		XPRPRC_EM_1_00_02	XPRPRC_EX_1_00_02_BFR	XPRPRC_EX_1_00_02_REP	XPRPRC_EX_1_00_02_AFR
Producer Holdback Service (GET)	XPRPRC_EW_1_00_02.XPRPRC_EW_100_02(iv_holdback IN OUT xpr_pro_hlr_rec_t)	XPRPRC_EM_1_00_02.	XPRPRC_EX_100_02		
		XPRPRC_EM_1_00_02	XPRPRC_EX_100_02		
		XPRPRC_EM_1_00_02	XPRPRC_EX_1_00_02_BFR	XPRPRC_EX_1_00_02_REP	XPRPRC_EX_1_00_02_AFR
		XPRPRC_EM_1_00_02	XPRPRC_EX_1_00_02_BFR	XPRPRC_EX_1_00_02_REP	XPRPRC_EX_1_00_02_AFR
Get webhook maintenance (GET)	xcswhk_ew_1_00_01.xcswhk_ew_100_01(iv_evi_id IN EVE_EVI_EVW.EVI_ID%TYPE, iv_evi_jmd_id IN EVE_EVI_EVW.EVI_JMD_ID%TYPE, iv_ev	xcswhk_em_1_00_01.xcswhk_em_100_01	xcswhk_ex_100_01		
		xcswhk_em_1_00_01	xcswhk_ex_100_01		
		xcswhk_em_1_00_01	xcswhk_ex_1_00_01_afr	xcswhk_ex_1_00_01_REP	xcswhk_ex_1_00_01_afr
		xcswhk_em_1_00_01	xcswhk_ex_1_00_01_afr	xcswhk_ex_1_00_01_REP	xcswhk_ex_1_00_01_afr

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
	i_status_cd IN OUT EVE_EVI_EVW .EVI_STATUS _CD%TYPE, iv_action IN VARCHAR2,iv _err_code OUT VARCHAR2,iv _err_desc OUT VARCHAR2,iv _whk_req_re c IN OUT xcs_whk_req _rec_t);		get_whk_set up_BFR	get_whk_set up_REP	get_whk_set up_AFR
Webhooks					
update event status	xcswhk_ew_1 00_01.xcswh k_ew_100_0 1(iv_evi_id IN EVE_EVI_EVW .EVI_ID%TYP E, iv_evi_jmd_ id IN OUT EVE_EVI_EVW .EVI_JMD_ID %TYPE,iv_ev i_status_cd IN OUT EVE_EVI_EVW .EVI_STATUS _CD%TYPE, iv_action IN VARCHAR2,iv _err_code OUT VARCHAR2,iv _err_desc OUT VARCHAR2,iv	xcswhk_em_1 00_01.updat e_event_sta tus	xcswhk_ex_100_01		

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
	_whk_req_re c IN OUT xcs_whk_req _rec_t);		update_even t_status_bf r	update_even t_status_re p	update_even t_status_af r
Webhooks					
maintenance : Save the webservice details (POST)	xcswhk_ew_1 00_02.xcswh k_ew_100_0 2(iv_whk_se t_rec IN OUT xsc_whk_set _rec_t);	xcswhk_em_1 00_02.xcswh k_em_100_02 .	xcswhk_ex_1 00_02		
			post_whk_bf r	post_whk_af r	post_whk_re p
Webhook					
Maintenance : retrieve the webservice details(GET)	xcswhk_ew_1 00_02.xcswh k_ew_100_0 2(iv_whk_se t_rec IN OUT xsc_whk_set _rec_t);	xcswhk_em_1 00_02.xcswh k_em_100_02 .	xcswhk_ex_1 00_02		
			get_whk_bfr	get_whk_rep	get_whk_afr
			GET_ADV_BFR	GET_ADV_REP	GET_ADV_AFR
Work Order					
Service (GET)	XCSVNA_EW_1 00_01. XCSVNA_EW_1 00_01 (iv_xcs_vna _tab IN OUT xcs_vna_res t_tab_t)	XCSVNA_EM_1 00_01. XCSVNA_EM_1 00_01	XCSVNA_EX_100_01		
			XCSVNA_EX_1 00_01_BFR	XCSVNA_EX_1 00_01_REP	XCSVNA_EX_1 00_01_AFR
		XCSVNA_EN_1 00_01. FETCH_WORK_ ORDER	XCSVNA_EX_100_01		
			FETCH_WORK_ ORDER_BFR	FETCH_WORK_ ORDER_REP	FETCH_WORK_ ORDER_AFR

Table 12-39 (Cont.) we

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
Work Order Service (PUT)					
	XCSVNA_EW_1_00_01.	XCSVNA_EM_1_00_01.	XCSVNA_EX_100_01		
	XCSVNA_EW_1_00_01	XCSVNA_EM_1_00_01	_____		
	(iv_xcs_vna_tab IN OUT xcs_vna_res t_tab_t)		XCSVNA_EX_1_00_01_BFR	XCSVNA_EX_1_00_01_REP	XCSVNA_EX_1_00_01_AFR
		XCSVNA_EN_1_00_01.UPDATE_WORK_ORDER	_____		
			UPDATE_WORK_ORDER_BFR	UPDATE_WORK_ORDER_REP	UPDATE_WORK_ORDER_AFR
Work Order Service (POST)					
	XCSVNA_EW_1_00_01.	XCSVNA_EM_1_00_01.	XCSVNA_EX_100_01		
	XCSVNA_EW_1_00_01	XCSVNA_EM_1_00_01	_____		
	(iv_xcs_vna_tab IN OUT xcs_vna_res t_tab_t)		XCSVNA_EX_1_00_01_BFR	XCSVNA_EX_1_00_01_REP	XCSVNA_EX_1_00_01_AFR
		XCSVNA_EN_1_00_01.CREATE_WORK_ORDER	_____		
			CREATE_WORK_ORDER_BFR	CREATE_WORK_ORDER_REP	CREATE_WORK_ORDER_AFR

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE xws_att_str_rec_t AS OBJECT (
    ATT_NAME VARCHAR2(30),
    ATT_VALUE VARCHAR2(4000));
CREATE OR REPLACE TYPE xws_att_num_rec_t AS OBJECT (
    ATT_NAME VARCHAR2(30),
    ATT_VALUE NUMBER );
CREATE OR REPLACE TYPE xws_att_date_rec_t AS OBJECT (
    ATT_NAME VARCHAR2(30),
    ATT_VALUE DATE );
CREATE OR REPLACE TYPE XUS_USR_REC_T AS OBJECT (
XCS_USR_CODE VARCHAR2(30),
USR_CODE VARCHAR2(30),
USR_COM_COMPANY VARCHAR2(30),
```

```

USR_CMB_BRANCH VARCHAR2(30),
USR_CBD_DEPT VARCHAR2(30),
USR_FIRST_NAME VARCHAR2(30),
USR_MIDDLE_NAME VARCHAR2(30),
USR_LAST_NAME VARCHAR2(30),
USR_RESPONSIBILITY_CD VARCHAR2(30),
USR_RRQ_SUP_USR_CODE VARCHAR2(30),
USR_PHONE_NO1 NUMBER,
USR_PHONE_EXTN1 NUMBER,
USR_PHONE_NO2 NUMBER,
USR_PHONE_EXTN2 NUMBER,
USR_FAX_NO1 NUMBER,
USR_FAX_NO2 NUMBER,
USR_ENABLED_IND VARCHAR2(30),
USR_START_DT DATE,
USR_END_DT DATE,
USR_REPLACEMENT_USR_CODE VARCHAR2(30),
USR_REPLACEMENT_DT DATE,
USR_TYPE_CD VARCHAR2(30),
USR_TYPE_REFERENCE_ID NUMBER,
USR_EMAIL_ADDRESS VARCHAR2(160),
USR_TIME_ZONE_CD VARCHAR2(30),
USR_DEFAULT_LANGUAGE_CD VARCHAR2(30),
USR_TIME_ZONE_LEVEL_CD VARCHAR2(30),
STR_ATTR XWS_ATT_STR_TAB_T,
NUM_ATTR XWS_ATT_NUM_TAB_T,
DATE_ATTR XWS_ATT_DATE_TAB_T,
USR_RESULT XUS_RES_REC_T);

```

12.45 Securitization Web Service (POST)

Below mentioned are the custom fields. These fields will be part of both request and response.

Table 12-40 Securitization Web Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  < StringData>
    <KeyName>StringName</KeyName>
    <KeyValue>StringValue</KeyValue>
  </ StringData>
  < NumberData>
    <KeyName>NumberName</KeyName>
    <KeyValue>NumberValue</KeyValue>

```

```

</ NumberData>
<DateData>
  <KeyName>DateName</KeyName>
  <KeyValue>DateValue</KeyValue>
</DateData>
</Custom>

```

Sample JSON

```

"Custom": {
  "StringData": [
    {
      "KeyName": "StringName",
      "KeyValue": "StringValue"
    }
  ],
  "NumberData": [
    {
      "KeyName": "NumberName",
      "KeyValue": NumberValue
    }
  ],
  "DateData": [
    {
      "KeyName": "DateName",
      "KeyValue": "DateValue"
    }
  ]
}

```

Below are the package details for transaction parameter web service

Table 12-41 Package details

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
SECURITIZATI ON (POST)	XCSSEC_EW_1 00_01.	XCSSEC_EM_1 00_01.	XCSSEC_EX_100_01		
	XCSSEC_EW_1 00_01	XCSSEC_EM_1 00_01			
	(IV_SEC_POO L_REC_T IN OUT XCS_SEC_POO L_REC_T)		XCSSEC_EX_1 00_01_BFR	XCSSEC_EX_1 00_01_REP	XCSSEC_EX_1 00_01_AFR

Extensibility Parameters for Tab Types

```

CREATE OR REPLACE TYPE XCS_STATUS_REC_T AS OBJECT (
STATUS VARCHAR2(30),
DESCRIPTION VARCHAR2(2000));

```

```

CREATE OR REPLACE TYPE XCS_STATUS_TAB_T AS TABLE OF XCS_STATUS_REC_T;
CREATE OR REPLACE TYPE XCS_SEC_DTLS_REC_T AS OBJECT
(
    XCS_POOL_NAME VARCHAR2(30),
    XCS_ACC_NBR VARCHAR2(30),
    STATUS XCS_STATUS_TAB_T
);
CREATE OR REPLACE TYPE XCS_SEC_DTLS_TAB_T AS TABLE OF XCS_SEC_DTLS_REC_T;
CREATE OR REPLACE TYPE XCS_SEC_POOL_REC_T AS OBJECT
(
    USER_CODE VARCHAR2(30),
    XCS_POOL_DTLS XCS_SEC_DTLS_TAB_T,
    STRING_ATTR XWS_ATT_STR_TAB_T,
    NUMBER_ATTR XWS_ATT_NUM_TAB_T,
    DATE_ATTR XWS_ATT_DATE_TAB_T,
    STATUS XCS_STATUS_TAB_T
);

```

12.46 Calculate Parameter Update Service(PUT)

Sample XML

```

<CalculateParameterRequest>
  <UserCode>USERNAME</UserCode>
  <ModuleName>SERVICING</ModuleName>
  <EntityNumber>ACC_NBR</EntityNumber>
  <AdditionalAttributeName>ACC_UDF2_NUM</AdditionalAttributeName>
</CalculateParameterRequest>

```

Sample JSON

```

{
  "CalculateParameterRequest": {
    "UserCode": "USERCODE",
    "ModuleName": "SERVICING",
    "EntityNumber": "ACC_NBR ",
    "AdditionalAttributeName": "ACC_UDF2_NUM"
  }
}

```

Below are the package details for Update Calculate Parameter Web Service

Wrapper Engine package:-

```
xcscup_ew_100_01.xcscup_ew_100_01 (iv_xcs_cup_rec IN OUT xcs_cup_rec_t);
```

Main Engine package:-

```
xcscup_em_100_01.xcscup_em_100_01 (iv_xcs_cup_rec IN OUT xcs_cup_rec_t);
```

Below are the Exit point package details for xcscup_em_100_01.update_cup ();

BEFORE:-

```
xcscup_ex_100_01.cv_update_cup_bfr = cmncon_cl_000_01.CUSTOMIZED THEN
xcscup_ex_100_01.update_cup_bfr (iv_xcs_cup_rec IN OUT xcs_cup_rec_t);
```

REPLACE:-

```
xcscup_ex_100_01.cv_update_cup_rep = cmncon_cl_000_01.CUSTOMIZED THEN
xcscup_ex_100_01.update_cup_bfr (iv_xcs_cup_rec IN OUT xcs_cup_rec_t);
```

AFTER:-

```
xcscup_ex_100_01. cv_update_cup_afr = cmncon_cl_000_01.CUSTOMIZED THEN
xcscup_ex_100_01. update_cup_bfr (iv_xcs_cup_rec IN OUT xcs_cup_rec_t);
```

Extensible parameters are Tab Type object

```
CREATE OR REPLACE TYPE XCS_CUP_REC_T AS OBJECT (
    CUP_USERCODE          VARCHAR2(30),
    CUP_MODULE_NAME      VARCHAR2(80),
    CUP_MODULE_IDENTIFIER VARCHAR2(30),
    CUP_ADDITIONAL_ATTR  VARCHAR2(30),
    CUP_RESULT            XCS_RESULT_TAB_T)
```

12.47 User Update Service (PUT)

Sample XML

```
<Custom>
  <StringData>
    <KeyName>OrgName</KeyName>
    <KeyValue>Oracle</KeyValue>
  </StringData>
  <NumberData>
    <KeyName>BusinessPhoneNumber</KeyName>
    <KeyValue>1234.01</KeyValue>
  </NumberData>
  <DateData>
    <KeyName>CreationDate</KeyName>
    <KeyValue>2017-12-18T00:00:00</KeyValue>
  </DateData>
</Custom>
```

Sample JSON

```
"Custom": {
  "StringData": [
    {
      "KeyName": "OrgName",
      "KeyValue": "Oracle"
    }
  ],
  "NumberData": [
    {
      "KeyName": "BusinessPhoneNumber",
      "KeyValue": "1234.01"
    }
  ],
  "DateData": [
    {
      "KeyName": "CreationDate",
      "KeyValue": "2017-12-18T00:00:00"
    }
  ]
}
```

```

    "NumberData": [
    {
      "KeyName": "BusinessPhoneNumber",
      "KeyValue": 1234.01
    }
  ],
  "DateData": [
  {
    "KeyName": "CreationDate",
    "KeyValue": "2017-12-18T00:00:00"
  }
  ]
}
}
}

```

Below are the package details for Update Calculate Parameter Web Service

Wrapper Engine package:-

```
xususr_ew_100_01.xususr_ew_100_01(iv_xus_usr_rec_t IN OUT xus_usr_rec_t);
```

Main Engine package:-

```
xususr_em_100_01.xususr_em_100_01(iv_xus_usr_rec_t IN OUT xus_usr_rec_t);
```

Below are the Exit point package details for

BEFORE:-

```
xususr_ex_100_01.CV_XUS_USR_BFR = cmncon_cl_000_01.CUSTOMIZED THEN
xususr_ex_100_01.xususr_ex_100_01_bfr(iv_xus_usr_rec_t);
```

REPLACE:-

```
xususr_ex_100_01.CV_XUS_USR_REP = cmncon_cl_000_01.CUSTOMIZED THEN
xususr_ex_100_01.xususr_ex_100_01_rep(iv_xus_usr_rec_t);
```

AFTER:-

```
xususr_ex_100_01.CV_XUS_USR_AFR = cmncon_cl_000_01.CUSTOMIZED THEN
xususr_ex_100_01.xususr_ex_100_01_afr(iv_xus_usr_rec_t);
```

Extensible parameters are Tab Type object

```

xus_usr_rec_t AS OBJECT (
  xcs_usr_code      VARCHAR2(30),
  usr_code          VARCHAR2(30),
  usr_com_company   VARCHAR2(30),
  usr_cmb_branch    VARCHAR2(30),
  usr_cbd_dept      VARCHAR2(30),
  usr_first_name    VARCHAR2(30),
  usr_middle_name   VARCHAR2(30),
  usr_last_name     VARCHAR2(30),

```

```

usr_responsibility_cd VARCHAR2(30),
usr_rrq_sup_usr_code VARCHAR2(30),
usr_phone_no1        NUMBER,
usr_phone_extn1      NUMBER,
usr_phone_no2        NUMBER,
usr_phone_extn2      NUMBER,
usr_fax_no1          NUMBER,
usr_fax_no2          NUMBER,
usr_enabled_ind      VARCHAR2(30),
usr_start_dt         DATE,
usr_end_dt           DATE,
usr_replacement_usr_code VARCHAR2(30),
usr_replacement_dt   DATE,
usr_type_cd          VARCHAR2(30),
usr_type_reference_id NUMBER,
usr_email_address    VARCHAR2(160),
usr_time_zone_cd     VARCHAR2(30),
usr_default_language_cd VARCHAR2(30),
usr_time_zone_level_cd VARCHAR2(30),
str_attr             XWS_ATT_STR_TAB_T,
num_attr             XWS_ATT_NUM_TAB_T,
date_attr            XWS_ATT_DATE_TAB_T,
usr_result           XUS_RES_REC_T,
usr_req_type         VARCHAR2(30));

```

12.48 Master Account Rolledup details service (GET)

Sample XML

```

<Custom>
<StringData>
<KeyName>OrgName</KeyName>
<KeyValue>Oracle</KeyValue>
</StringData>
<NumberData>
<KeyName>BusinessPhoneNumber</KeyName>
<KeyValue>1234.01</KeyValue>
</NumberData>
<DateData>
<KeyName>CreationDate</KeyName>
<KeyValue>2017-12-18T00:00:00</KeyValue>
</DateData>
</Custom>

```

Sample JSON

```

"Custom": {
  "StringData": [
    {
      "KeyName": "OrgName",
      "KeyValue": "Oracle"
    }
  ],
  "NumberData": [

```



```

    {
      "KeyName": "BusinessPhoneNumber",
      "KeyValue": 1234.01
    }
  ],
  "DateData": [
    {
      "KeyName": "CreationDate",
      "KeyValue": "2017-12-18T00:00:00"
    }
  ]
}
}
}

```

12.49 Application Decision Fetch Service (GET/PUT)

Below mentioned are the custom fields. These fields will be part of both request/response.

Table 12-42 Application Decision Fetch Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date (YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  <StringData>
    <KeyName>OrgName</KeyName>
    <KeyValue>Oracle</KeyValue>
  </StringData>
  <NumberData>
    <KeyName>BusinessPhoneNumber</KeyName>
    <KeyValue>1234.01</KeyValue>
  </NumberData>
  <DateData>
    <KeyName>CreationDate</KeyName>
    <KeyValue>2017-12-18T00:00:00</KeyValue>
  </DateData>
</Custom>

```

Sample JSON

```

"Custom": {
  "StringData": [
    {
      "KeyName": "OrgName",
      "KeyValue": "Oracle"
    }
  ]
}

```

```

    }
  ],
  "NumberData": [
    {
      "KeyName": "BusinessPhoneNumber",
      "KeyValue": 1234.01
    }
  ],
  "DateData": [
    {
      "KeyName": "CreationDate",
      "KeyValue": "2017-12-18T00:00:00"
    }
  ]
}
}
}

```

Table 12-43 Package Details

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
Application Decision Service (GET)	XAEADC_EW_1	XAEADC_EM_1	XAEADC_EX_100_01		
	00_01.XAEAD	00_01.			
	C_EW_100_01	XAEADC_EM_1			
	(iv_auto_de c_rec_t IN OUT xae_auto_de c_rec_t)	00_01	XAEADC_EX_1 00_01_BFR	XAEADC_EX_1 00_01_REP	XAEADC_EX_1 00_01_AFR
		XAEADC_EW_1 00_01.CREATE E_ADC_DECIS ION	CREATE_ADC_ DECISION_BF R	CREATE_ADC_ DECISION_RE P	CREATE_ADC_ DECISION_AF R
		XAEADC_EW_1 00_01. CREATE_OFFE R	XAEADC_EW_1 00_01. CREATE_OFFE R_BFR	XAEADC_EW_1 00_01. CREATE_OFFE R_REP	XAEADC_EW_1 00_01. CREATE_OFFE R_AFR
		XAEADC_EW_1 00_01. SELECT_PRIC ING	XAEADC_EW_1 00_01. SELECT_PRIC ING_BFR	XAEADC_EW_1 00_01. SELECT_PRIC ING_REP	XAEADC_EW_1 00_01. SELECT_PRIC ING_AFR

Table 12-43 (Cont.) Package Details

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
		XAEADC_EW_1 00_01. CREATE_MULT IPLE_OFFERS	XAEADC_EW_1 00_01. CREATE_MULT IPLE_OFFERS _BFR	XAEADC_EW_1 00_01. CREATE_MULT IPLE_OFFERS _REP	XAEADC_EW_1 00_01. CREATE_MULT IPLE_OFFERS _AFR
Application Decision Service (PUT)	XAEADC_EW_1 00_01.XAEAD C_EW_100_01 (iv_auto_de c_rec_t IN OUT xae_auto_de c_rec_t)	XAEADC_EW_1 00_01. UPDATE_ADC_ DECISION	XAEADC_EX_100_01 UPDATE_ADC_ DECISION_BF R	UPDATE_ADC_ DECISION_RE P	UPDATE_ADC_ DECISION_AF R
		XAEADC_EW_1 00_01. POPULATE_OF F_RESPONSE	POPULATE_OF F_RESPONSE_ BFR	POPULATE_OF F_RESPONSE_ REP	POPULATE_OF F_RESPONSE_ AFR
		XAEADC_EW_1 00_01.POPUL ATE_ADC_RES PONSE	XAEADC_EW_1 00_01.POPUL ATE_ADC_RES PONSE_BFR	XAEADC_EW_1 00_01.POPUL ATE_ADC_RES PONSE_REP	XAEADC_EW_1 00_01.POPUL ATE_ADC_RES PONSE_AFR

The following types are have been created as a part of development for the service.

```
CREATE OR REPLACE TYPE xae_adc_off_rec_t FORCE AS OBJECT
(
  ade_prp_pricing    VARCHAR2(30)
  ,ade_term          NUMBER
  ,ade_rate           NUMBER
  ,ade_pmt_amt       NUMBER
  ,ade_index_type_cd VARCHAR2(30)
  ,ade_index_rate    NUMBER
  ,ade_mrate         NUMBER
  ,ade_maturity_index_type_cd VARCHAR2(30)
  ,ade_maturity_mrate NUMBER
  ,ade_accepted_ind  VARCHAR2(30)
  ,ade_adc_current_ind VARCHAR2(30)
  ,str_attr          xws_att_str_tab_t
)
```

```

, num_attr      xws_att_num_tab_t
, date_attr     xws_att_date_tab_t
);
/
CREATE OR REPLACE TYPE xae_adc_prc_rec_t FORCE AS OBJECT
(
  adc_prp_pricing      VARCHAR2(30)
, adc_current_ind     VARCHAR2(30)
, adc_app_status_cd   VARCHAR2(30)
, adc_app_sub_status_cd VARCHAR2(30)
, adc_decision_dt     DATE
, adc_underwriter_usr_code VARCHAR2(30)
, adc_term            NUMBER
, adc_pmt_amt         NUMBER
, adc_index_type_cd   VARCHAR2(30)
, adc_index_rate      NUMBER
, adc_mrate           NUMBER
, adc_rate_buy_pricing NUMBER
, adc_rate_buy        NUMBER
, adc_down_pmt_per    NUMBER
, adc_down_pmt_amt    NUMBER
, adc_adv_per         NUMBER
, adc_adv_amt         NUMBER
, adc_loan_per        NUMBER
, adc_loan_amt        NUMBER
, adc_maturity_index_type_cd VARCHAR2(30)
, adc_maturity_mrate  NUMBER
, adc_balloon_amt     NUMBER
, adc_apr             NUMBER
, adc_first_crb_source_cd VARCHAR2(30)
, adc_second_crb_source_cd VARCHAR2(30)
, adc_third_crb_source_cd VARCHAR2(30)
, adc_cr_grade_cd     VARCHAR2(30)
, adc_cr_score        NUMBER
, adc_avl_total_val   NUMBER
, adc_ltv1            NUMBER
, adc_ltv2            NUMBER
, adc_comment         VARCHAR2(4000)
, adc_cap_cost        NUMBER
, adc_residual_per    NUMBER
, adc_cap_cost_adj    NUMBER
, adc_cr_lmt          NUMBER
, adc_prm_code        VARCHAR2(30)
, adc_draw_term       NUMBER
, adc_repmt_term      NUMBER
, app_offers          xae_adc_off_tab_t
, str_attr            xws_att_str_tab_t
, num_attr            xws_att_num_tab_t
, date_attr           xws_att_date_tab_t
);
/

```

12.50 Work Order Fetch Service (GET/PUT/POST)

Below mentioned are the custom fields. These fields will be part of both request/response.

Table 12-44 Work Order Fetch Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <StringData>
    <KeyName>OrgName</KeyName>
    <KeyValue>Oracle</KeyValue>
  </StringData>
  <NumberData>
    <KeyName>BusinessPhoneNumber</KeyName>
    <KeyValue>1234.01</KeyValue>
  </NumberData>
  <DateData>
    <KeyName>CreationDate</KeyName>
    <KeyValue>2017-12-18T00:00:00</KeyValue>
  </DateData>
</Custom>
```

Sample JSON

```
"Custom": {
  "StringData": [
    {
      "KeyName": "OrgName",
      "KeyValue": "Oracle"
    }
  ],
  "NumberData": [
    {
      "KeyName": "BusinessPhoneNumber",
      "KeyValue": 1234.01
    }
  ],
  "DateData": [
    {
      "KeyName": "CreationDate",
      "KeyValue": "2017-12-18T00:00:00"
    }
  ]
}
```

Table 12-45 Package Details

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
Work Order Service (GET)	XCSVNA_EW_1_00_01.	XCSVNA_EM_1_00_01.	XCSVNA_EX_100_01		
	XCSVNA_EW_1_00_01	XCSVNA_EM_1_00_01			
	(iv_xcs_vna_tab IN OUT xcs_vna_res t_tab_t)		XCSVNA_EX_1_00_01_BFR	XCSVNA_EX_1_00_01_REP	XCSVNA_EX_1_00_01_AFR
		XCSVNA_EN_1_00_01.	XCSVNA_EX_100_01		
		FETCH_WORK_ORDER	FETCH_WORK_ORDER_BFR	FETCH_WORK_ORDER_REP	FETCH_WORK_ORDER_AFR
Work Order Service (PUT)	XCSVNA_EW_1_00_01.	XCSVNA_EM_1_00_01.	XCSVNA_EX_100_01		
	XCSVNA_EW_1_00_01	XCSVNA_EM_1_00_01			
	(iv_xcs_vna_tab IN OUT xcs_vna_res t_tab_t)		XCSVNA_EX_1_00_01_BFR	XCSVNA_EX_1_00_01_REP	XCSVNA_EX_1_00_01_AFR
		XCSVNA_EN_1_00_01.	XCSVNA_EX_100_01		
		UPDATE_WORK_ORDER			

Table 12-45 (Cont.) Package Details

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
			UPDATE_WORK _ORDER_BFR	UPDATE_WORK _ORDER_REP	UPDATE_WORK _ORDER_AFR
Work Order Service (POST)	XCSVNA_EW_1 00_01. XCSVNA_EW_1 00_01 (iv_xcs_vna _tab IN OUT xcs_vna_res t_tab_t)	XCSVNA_EM_1 00_01. XCSVNA_EM_1 00_01	XCSVNA_EX_100_01		
			XCSVNA_EX_1 00_01_BFR	XCSVNA_EX_1 00_01_REP	XCSVNA_EX_1 00_01_AFR
		XCSVNA_EN_1 00_01. CREATE_WORK _ORDER	XCSVNA_EX_100_01		
			CREATE_WORK _ORDER_BFR	CREATE_WORK _ORDER_REP	CREATE_WORK _ORDER_AFR

12.51 Pricing Fetch Service (GET)

Below mentioned are the custom fields. These fields will be part of both request/response.

Table 12-46 Pricing Fetch Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  <StringData>
    <KeyName>OrgName</KeyName>
    <KeyValue>Oracle</KeyValue>
  </StringData>
  <NumberData>
    <KeyName>BusinessPhoneNumber</KeyName>
    <KeyValue>1234.01</KeyValue>
  </NumberData>
  <DateData>
    <KeyName>CreationDate</KeyName>
    <KeyValue>2017-12-18T00:00:00</KeyValue>
  </DateData>
</Custom>

```

Sample JSON

```

"Custom": {
  "StringData": [
    {
      "KeyName": "OrgName",
      "KeyValue": "Oracle"
    }
  ],
  "NumberData": [
    {
      "KeyName": "BusinessPhoneNumber",
      "KeyValue": 1234.01
    }
  ],
  "DateData": [
    {
      "KeyName": "CreationDate",
      "KeyValue": "2017-12-18T00:00:00"
    }
  ]
}

```

12.52 Account UDF Update Service (PUT)

Below mentioned are the custom fields. These fields will be part of both request/response.

Table 12-47 Account UDF Update Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)

Table 12-47 (Cont.) Account UDF Update Service

Element name	Sub element	Data type
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```

<Custom>
  <StringData>
    <KeyName>OrgName</KeyName>
    <KeyValue>Oracle</KeyValue>
  </StringData>
  <NumberData>
    <KeyName>BusinessPhoneNumber</KeyName>
    <KeyValue>1234.01</KeyValue>
  </NumberData>
  <DateData>
    <KeyName>CreationDate</KeyName>
    <KeyValue>2017-12-18T00:00:00</KeyValue>
  </DateData>
</Custom>

```

Sample JSON

```

"Custom": {
  "StringData": [
    {
      "KeyName": "OrgName",
      "KeyValue": "Oracle"
    }
  ],
  "NumberData": [
    {
      "KeyName": "BusinessPhoneNumber",
      "KeyValue": 1234.01
    }
  ],
  "DateData": [
    {
      "KeyName": "CreationDate",
      "KeyValue": "2017-12-18T00:00:00"
    }
  ]
}

```

12.53 Stipulation Fetch Service (GET)

Below mentioned are the custom fields. These fields will be part of both request/response.

Table 12-48 Stipulation Fetch Service

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String
	KeyValue	Date(YYYY-MM-DDTHH:MM:SS)

Sample XML

```
<Custom>
  <StringData>
    <KeyName>OrgName</KeyName>
    <KeyValue>Oracle</KeyValue>
  </StringData>
  <NumberData>
    <KeyName>BusinessPhoneNumber</KeyName>
    <KeyValue>1234.01</KeyValue>
  </NumberData>
  <DateData>
    <KeyName>CreationDate</KeyName>
    <KeyValue>2017-12-18T00:00:00</KeyValue>
  </DateData>
</Custom>
```

Sample JSON

```
"Custom": {
  "StringData": [
    {
      "KeyName": "OrgName",
      "KeyValue": "Oracle"
    }
  ],
  "NumberData": [
    {
      "KeyName": "BusinessPhoneNumber",
      "KeyValue": 1234.01
    }
  ],
  "DateData": [
    {
      "KeyName": "CreationDate",
      "KeyValue": "2017-12-18T00:00:00"
    }
  ]
}
```

A

Appendix: Revision History

Table A-1 Appendix

Section	Section Name	Release Version	Description of Change
----------------	---------------------	------------------------	------------------------------