Extensibility Guide
Oracle Financial Services Lending and Leasing
Release 14.8.0.0.0

Part No. F22291-01
December 2019

ORACLE
FINANCIAL SERVICES

Table of Contents

1. Preface

This document provides an overview on extensibility capabilities supported by Oracle Financial Services Lending and Leasing Application.

1.1 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.

1.2 Conventions Used

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

1.3 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.

1.4 Assumptions

- It is assumed that the customization team is familiar with ADF and this document ‘OFSLL Extensibility 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.

2. Customizing and Extending 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.

2.1 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’.

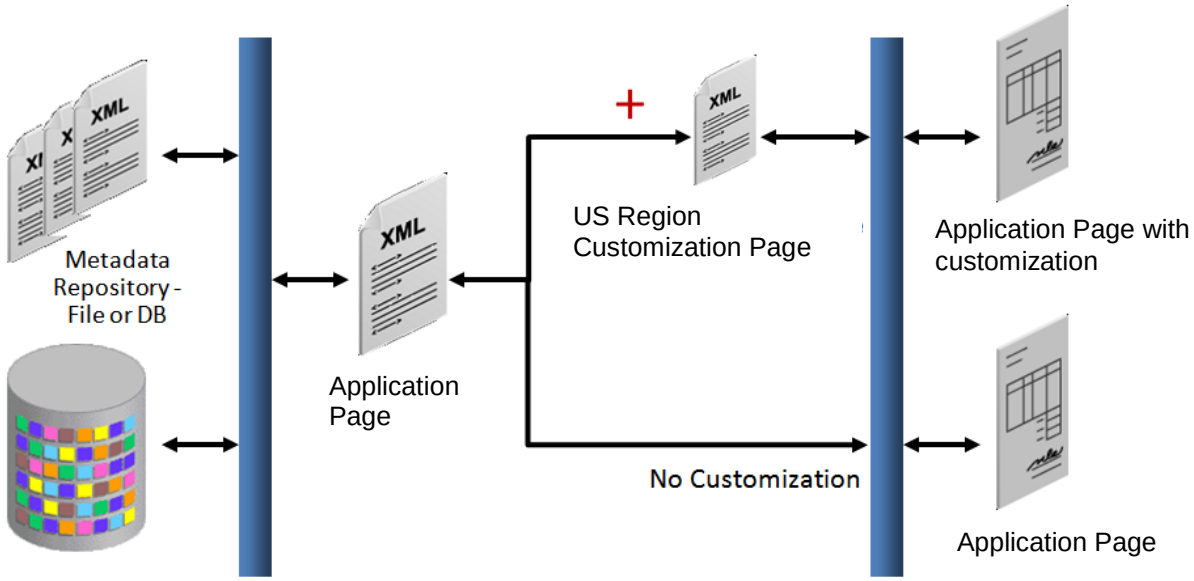
2.2 Understanding Customization Layers

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.

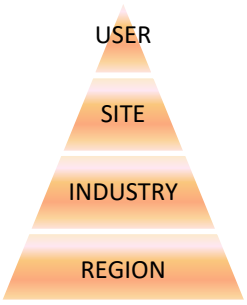


All users of the application can personalize the pages. Users can move elements around on a page, hide elements, and even add available elements to their page. When they do this personalization, the MDS Customization Engine creates an XML file specific to that user.

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.



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.

2.3 Installing Customization Tools

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

3. Using JDeveloper for Customizations

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

3.1 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

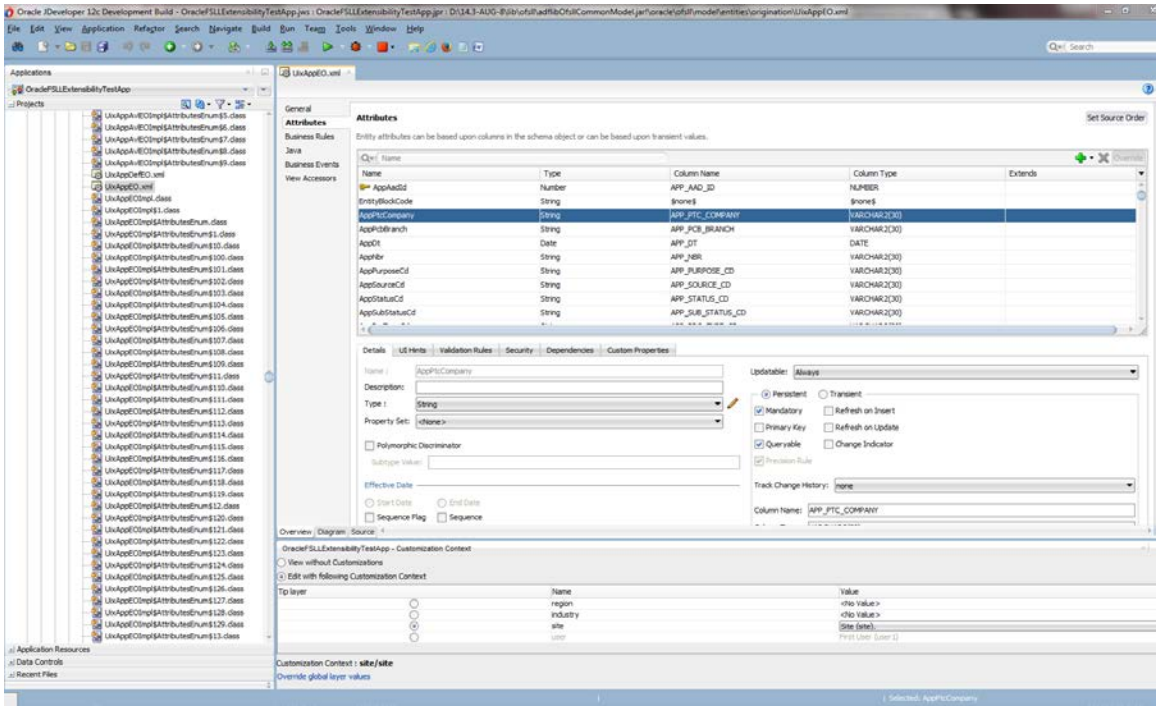
Before running JDeveloper in customization mode for the application, see Section 4.3 "Enable JDeveloper for Customization" for pre-configuration requirement.

3.2 About Customizing Oracle ADF Artifacts

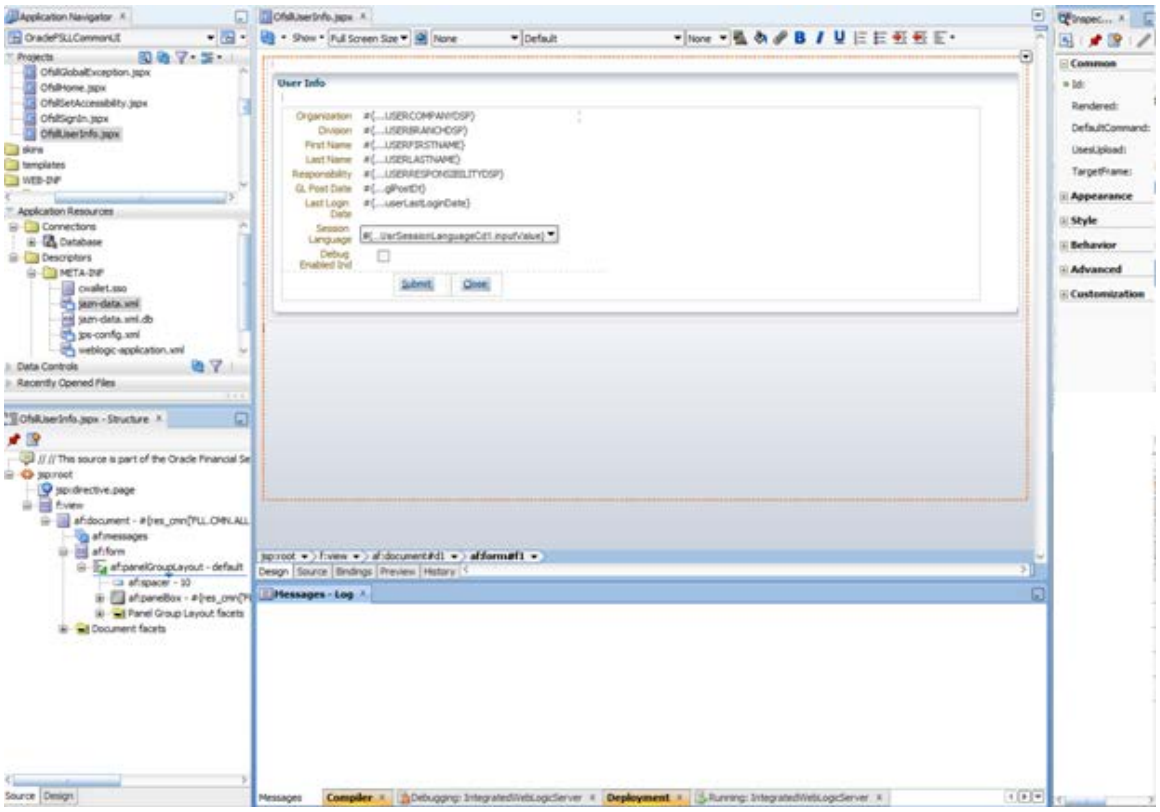
The application is built using Oracle ADF artifacts, including the following:

- 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.



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



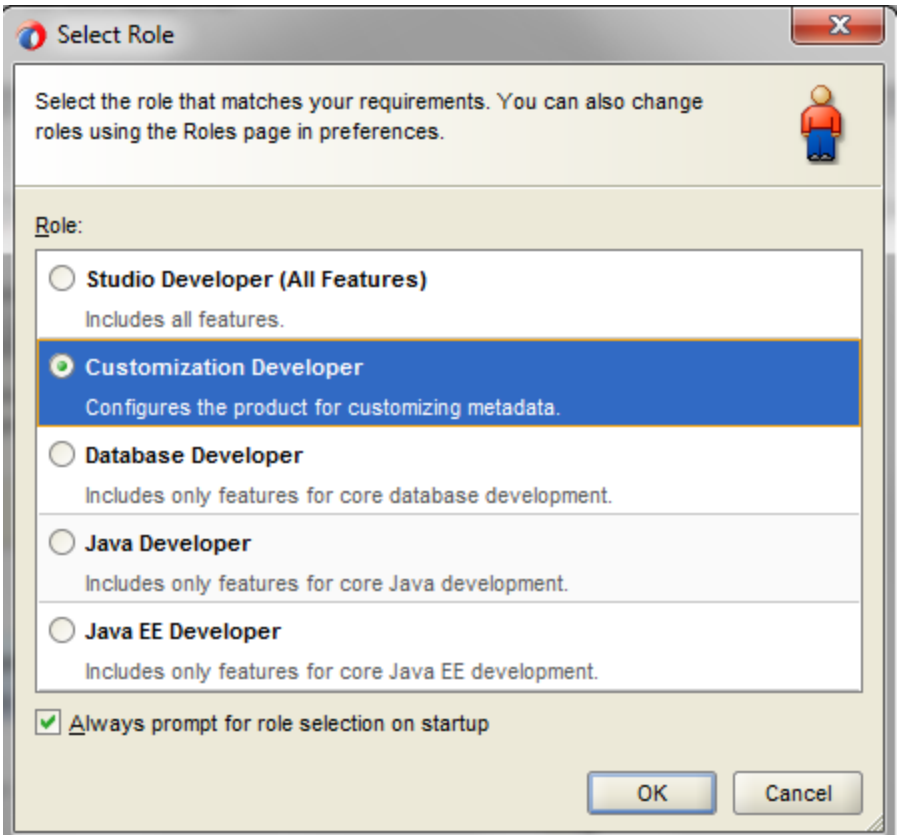
3.3 Customizing Oracle ADF Artifacts with JDeveloper

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

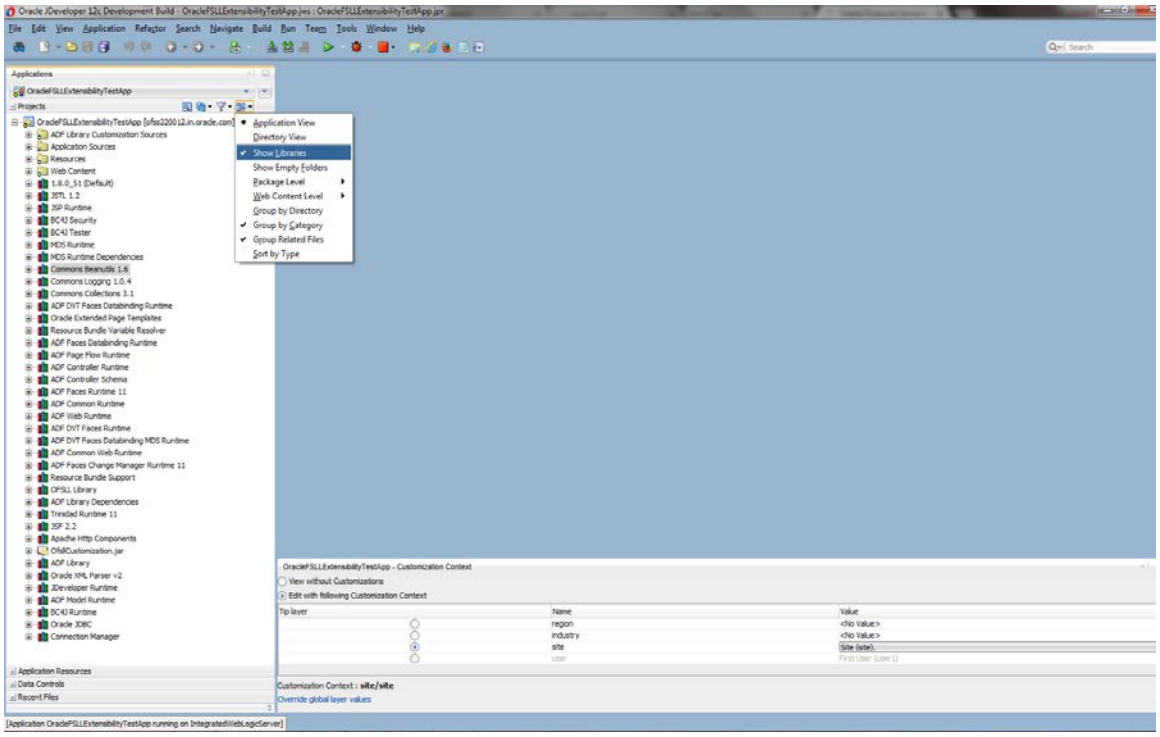
Customizing the Artifacts

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

1. Restart JDeveloper and select the **Customization Developer** role.



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.



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

3.4 ADF Customization Best Practices - Do's and Don'ts

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.

3.4.1 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 =
CommponentReference.newUIComponentReference(scraOrderRefNbr);

}
```

Example of **worng** 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.

3.4.2 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 explicitly create session object by initializing oracle.jbo.server.SessionImpl.
- 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.

4. 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.

4.1 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.

4.2 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:

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.

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.

4.3 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.

3. Rebuild the OfsllCustomization.jar using the default deployment profile.

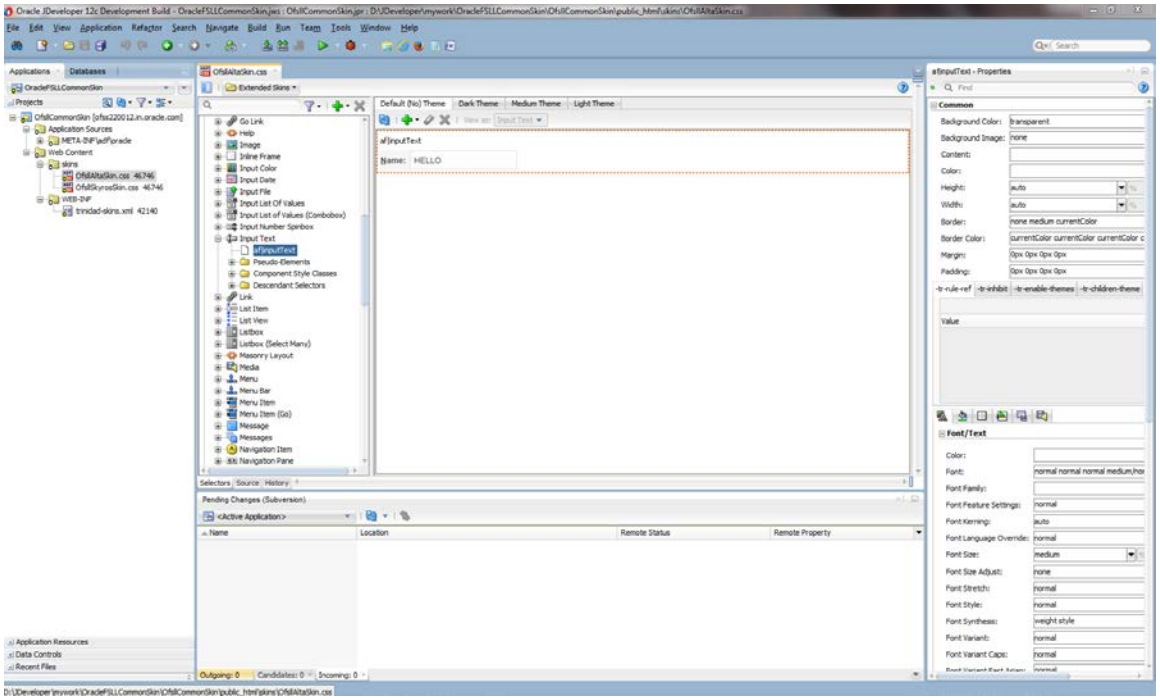
4. Edit the /cust_lib/CustomizationLayerValues.xml in Notepad and update the Region key layer, Industry key layer and Site key layer with the values added as per required customization.properties.

5. Copy the CustomizationLayerValues.xml onto JDeveloper installation location under $JDEV_HOME/jdeveloper/jdev.

4.4 Customizing the Skin

One method of customizing skin is opening the bundled *OracleFSLLCommonSkin/OracleFSLL 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.

1. Open the OracleFSLLCommonSkin 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.



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

Note

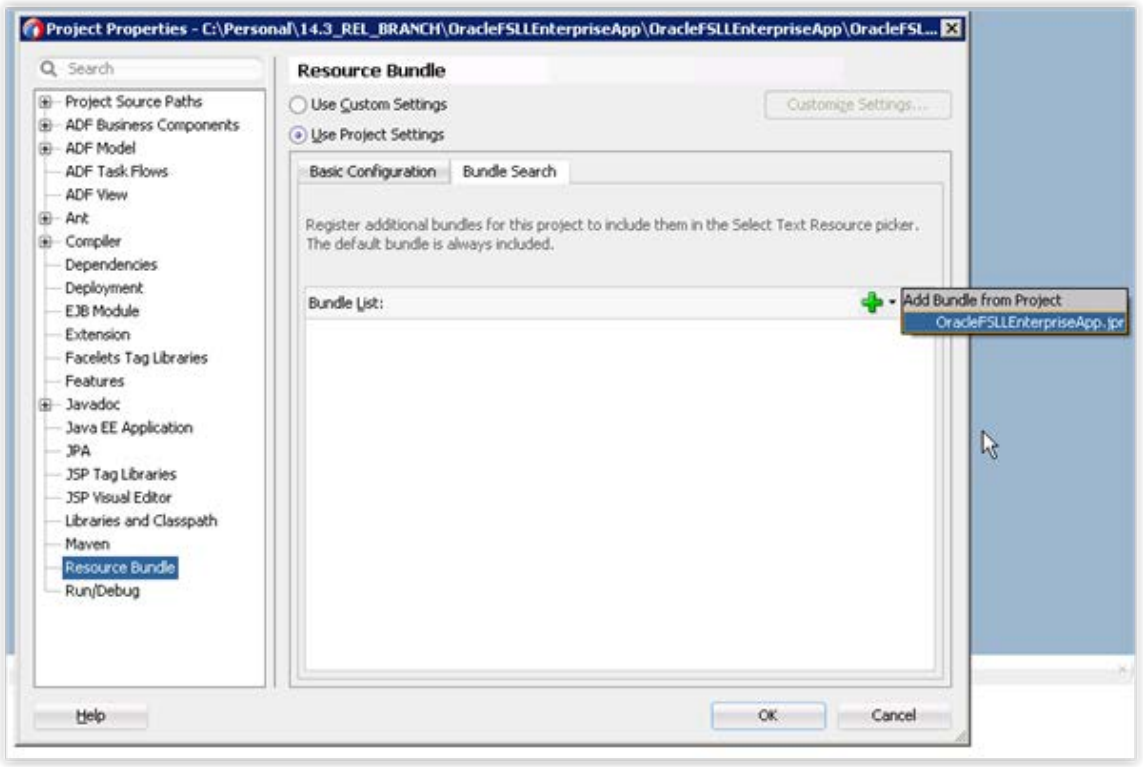
Skin can be customized using Oracle ADF Skin Editor which can be downloaded from Oracle site. If the default skin family name is changed then *trinidad-config.xml* available in *OracleFSLLEnterpriseApp* needs to be changed with new skin family name.

4.5 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.

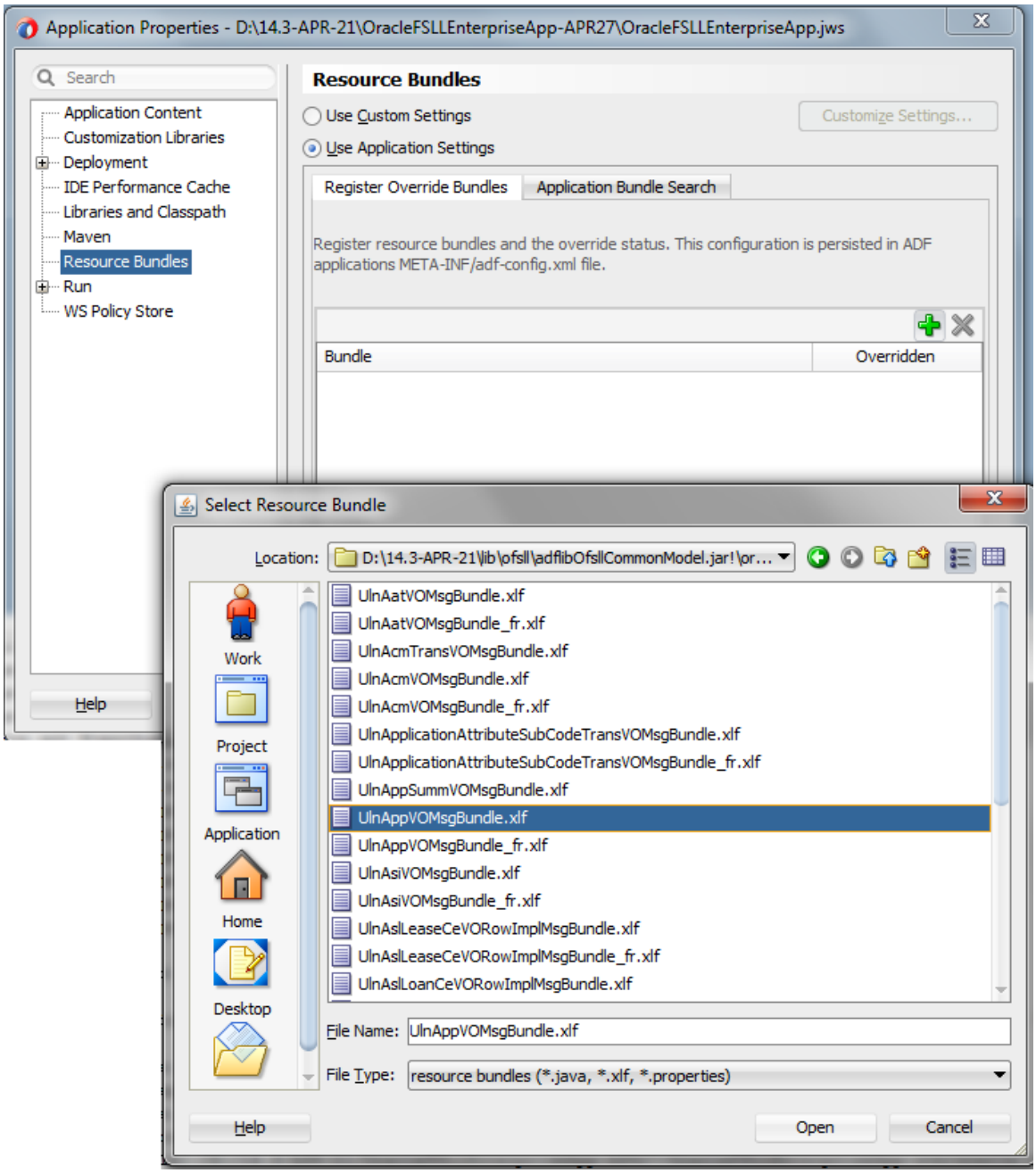
Because a new key cannot be defined in the shipped resource bundle, a new override bundle needs to be created. This can be accomplished in JDeveloper by creating an XLIFF file from the New Gallery. After the file is generated, new keys and their associated text in the XLIFF file can be entered.

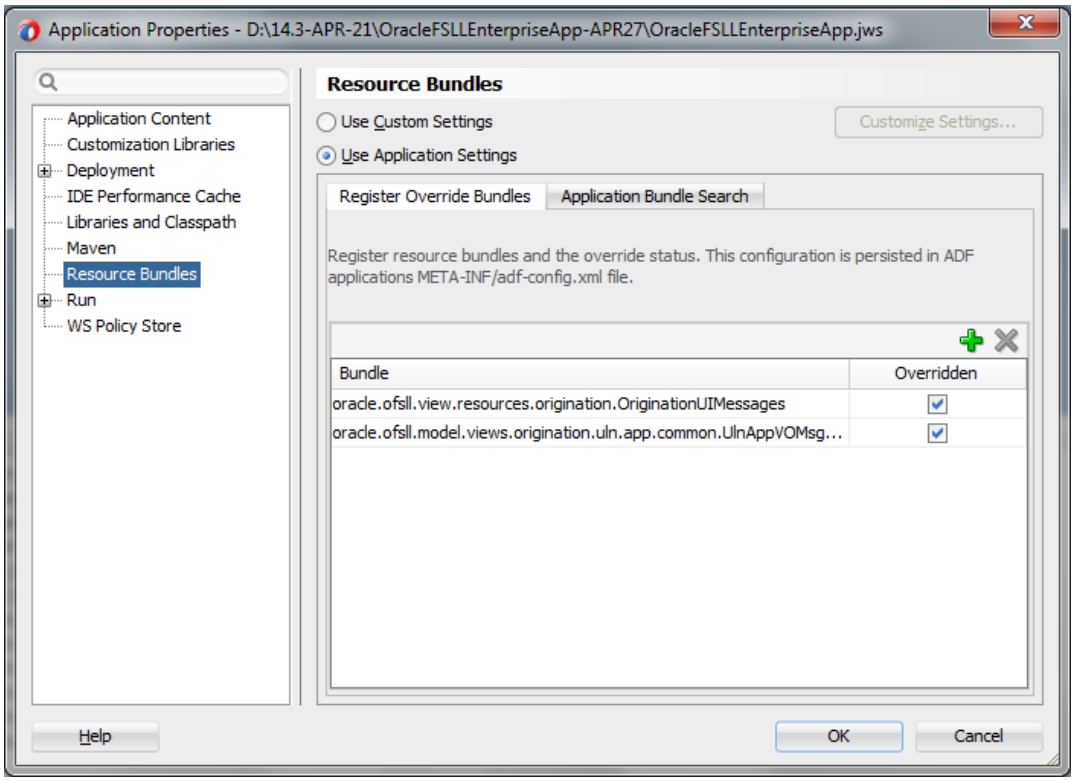
To make the newly created resource bundle available for customization, the resource bundle needs to be registered with the customization project. Newly created resource bundle can be present in customization project or as a separate project. To register the resource bundle with customization project,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



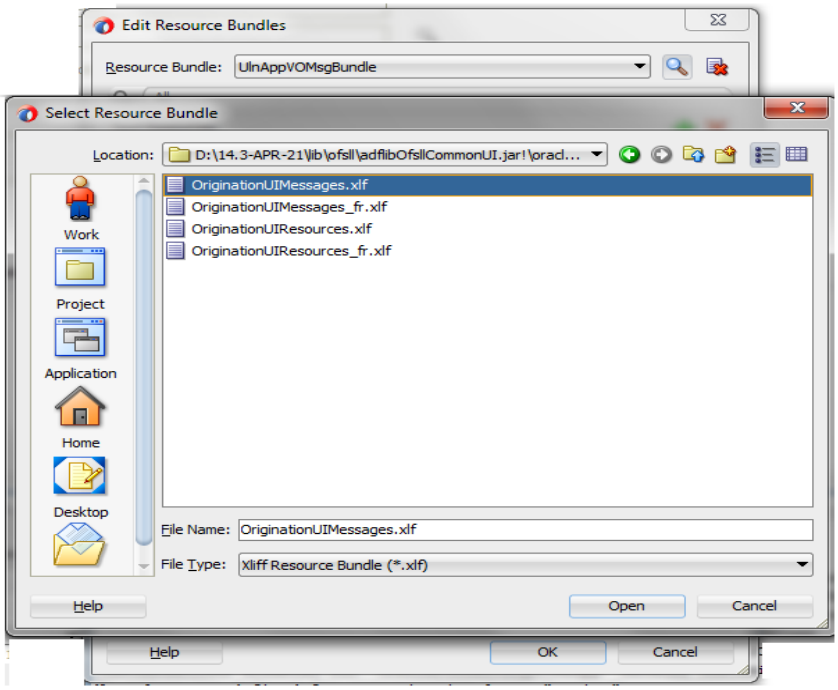
Step to override a message bundle which already exists in model or ui jar, is shown below:

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.

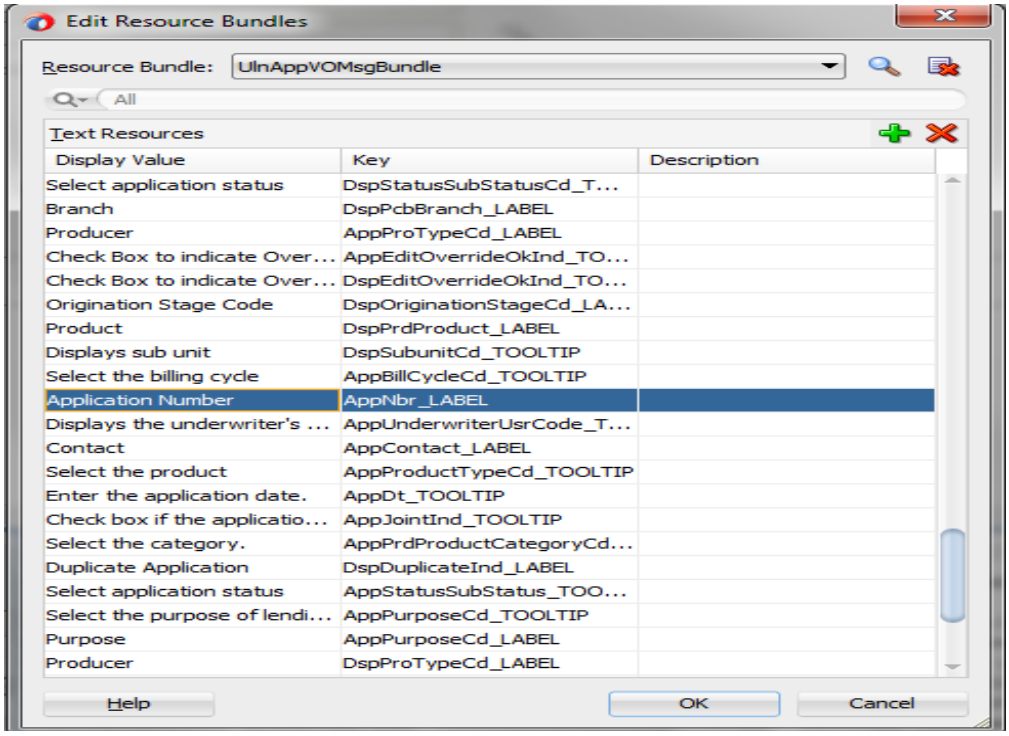

Application Properties - D:\14.3-APR-21\OracleFSLLEnterpriseApp-APR27\OracleFSLLEnterpriseApp.jws
Search
Application Content
Customization Libraries
Deployment
IDE Performance Cache
Libraries and Classpath
Maven
Resource Bundles
Run
WS Policy Store
Help
Resource Bundles
Use Custom Settings
Customize Settings...
Use Application Settings
Register Override Bundles
Application Bundle Search
Register resource bundles and the override status. This configuration is persisted in ADF applications META-INF/adf-config.xml file.
Bundle
Overridden
Select Resource Bundle
Location:
D:\14.3-APR-21\lib\ofsll\adflibOfsllCommonModel.jar!\or...
Work
Project
Application
Home
Desktop
UlnAatVOMsgBundle.xlf
UlnAatVOMsgBundle_fr.xlf
UlnAcmTransVOMsgBundle.xlf
UlnAcmVOMsgBundle.xlf
UlnAcmVOMsgBundle_fr.xlf
UlnApplicationAttributeSubCodeTransVOMsgBundle.xlf
UlnApplicationAttributeSubCodeTransVOMsgBundle_fr.xlf
UlnAppSummVOMsgBundle.xlf
UlnAppVOMsgBundle.xlf
UlnAppVOMsgBundle_fr.xlf
UlnAsiVOMsgBundle.xlf
UlnAsiVOMsgBundle_fr.xlf
UlnAslLeaseCeVORowImplMsgBundle.xlf
UlnAslLeaseCeVORowImplMsgBundle_fr.xlf
UlnAslLoanCeVORowImplMsgBundle.xlf
File Name:
UlnAppVOMsgBundle.xlf
File Type:
resource bundles (*.java, *.xlf, *.properties)
Help
Open
Cancel



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 overridden from the jar under lib folder.



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



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

4.6 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.

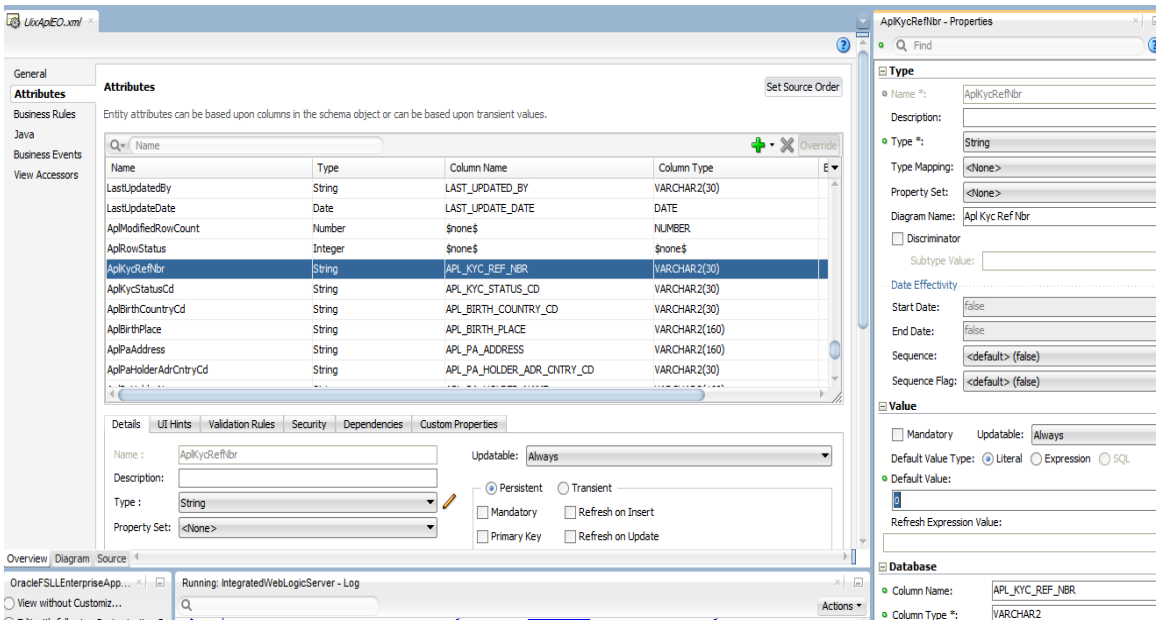
Task: Edit Attributes

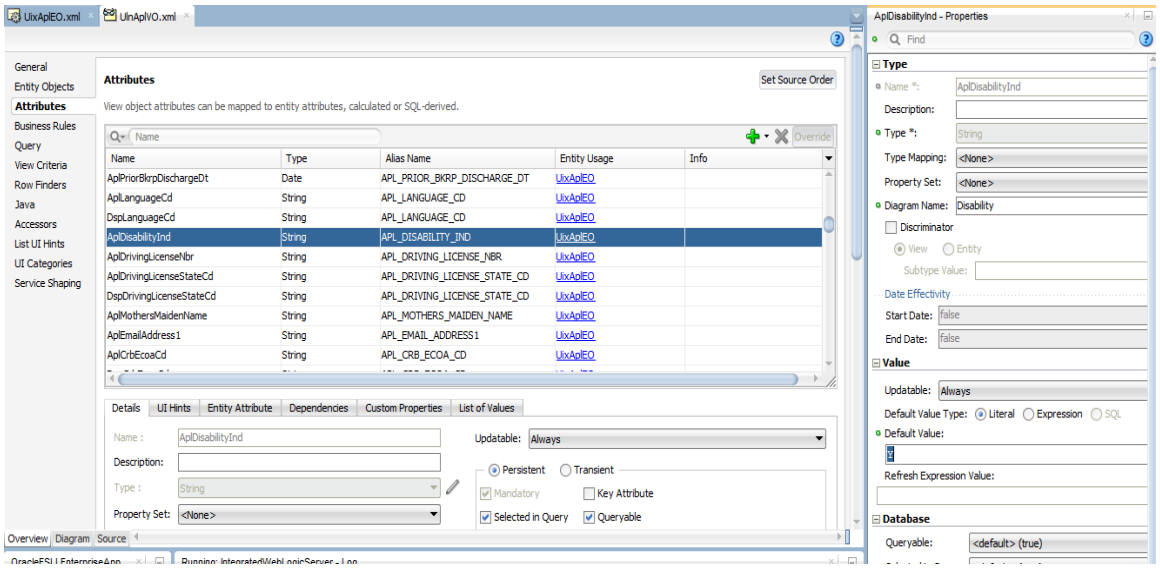
The properties of an attribute can be customized from an entity object or view object using JDeveloper. When an entity object opened or viewed in the overview editor, the attributes of the object can be seen on click in the Attributes tab. When an attribute is selected, its properties are displayed in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



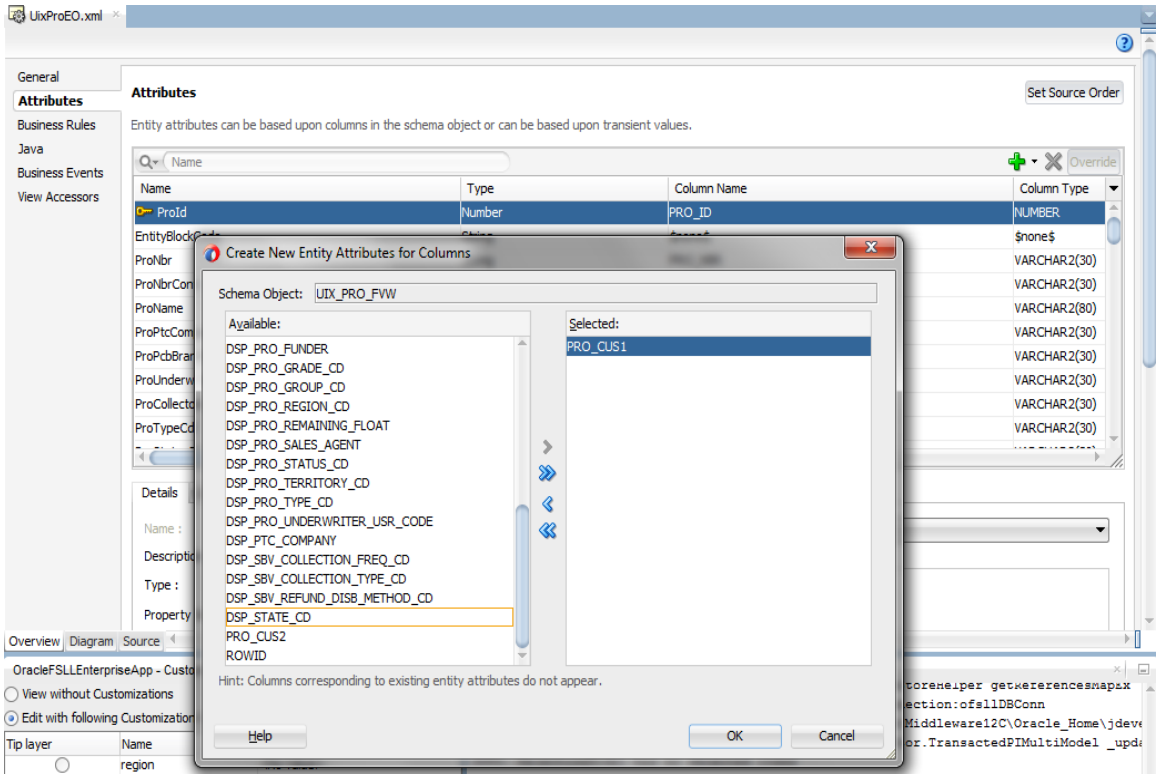


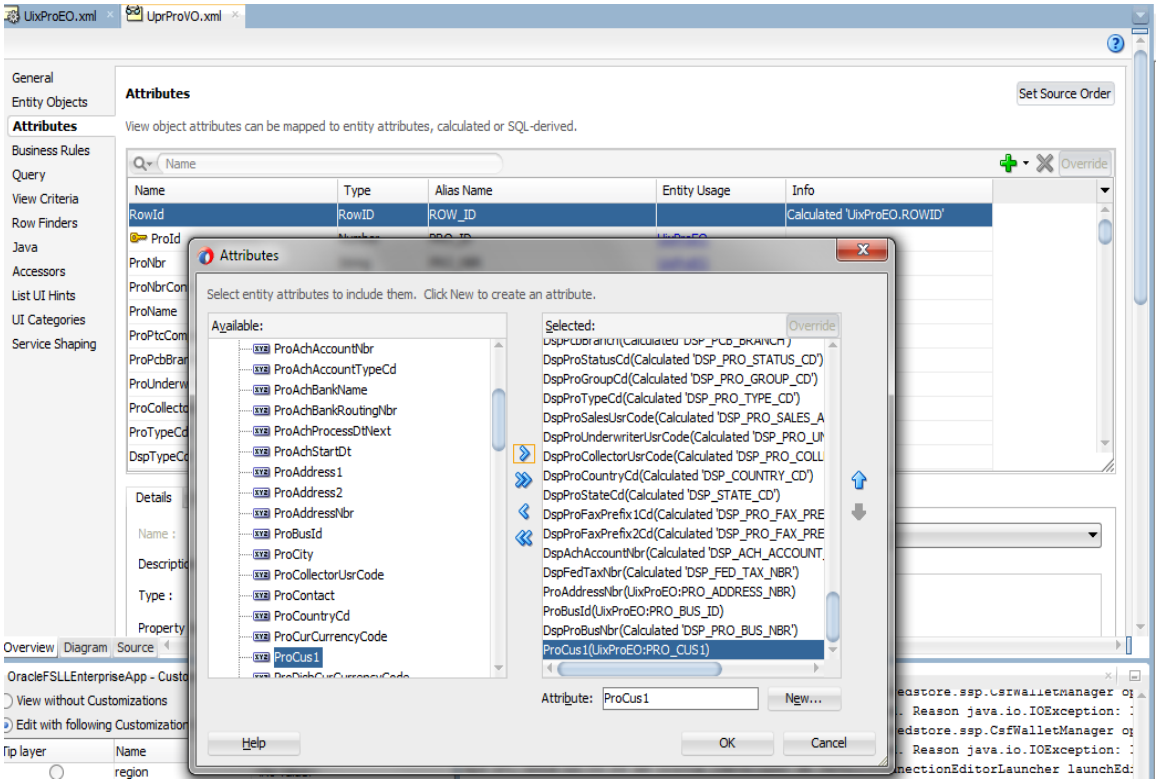
Task: Add Attributes

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.


UixProEO.xml
General
Attributes
Business Rules
Java
Business Events
View Accessors
Attributes
Set Source Order
Entity attributes can be based upon columns in the schema object or can be based upon transient values.
Name
Override
Name
Type
Column Name
Column Type
ProId
Number
PRO_ID
NUMBER
$none$
VARCHAR2(30)
VARCHAR2(80)
Details
Name :
Type :
Property
Overview
Diagram
Source
Create New Entity Attributes for Columns
Schema Object:
UIX_PRO_FVW
Available:
Selected:
PRO_CUS1
DSP_PRO_FUNDER
DSP_PRO_GRADE_CD
DSP_PRO_GROUP_CD
DSP_PRO_REGION_CD
DSP_PRO_REMAINING_FLOAT
DSP_PRO_SALES_AGENT
DSP_PRO_STATUS_CD
DSP_PRO_TERRITORY_CD
DSP_PRO_TYPE_CD
DSP_PRO_UNDERWRITER_USR_CODE
DSP_PTC_COMPANY
DSP_SBV_COLLECTION_FREQ_CD
DSP_SBV_COLLECTION_TYPE_CD
DSP_SBV_REFUND_DISB_METHOD_CD
DSP_STATE_CD
PRO_CUS2
ROWID
Hint: Columns corresponding to existing entity attributes do not appear.
Help
OK
Cancel
OracleFSLLEnterpriseApp - Custo
View without Customizations
Edit with following Customizations
Tip layer
Name
region


UixProEO.xml
UprProVO.xml
General
Entity Objects
Attributes
Business Rules
Query
View Criteria
Row Finders
Java
Accessors
List UI Hints
UI Categories
Service Shaping
Attributes
Set Source Order
View object attributes can be mapped to entity attributes, calculated or SQL-derived.
Name
Override
Name
Type
Alias Name
Entity Usage
Info
RowId
RowID
ROW_ID
Calculated 'UixProEO.ROWID'
ProId
Details
Name :
Type :
Property
Overview
Diagram
Source
Attributes
Select entity attributes to include them. Click New to create an attribute.
Available:
Selected:
ProAchAccountNbr
ProAchAccountTypeCd
ProAchBankName
ProAchBankRoutingNbr
ProAchProcessDtNext
ProAchStartDt
ProAddress1
ProAddress2
ProAddressNbr
ProBusId
ProCity
ProCollectorUsrCode
ProContact
ProCountryCd
ProCurCurrencyCode
ProCus1
DspProStatusCd(Calculated 'DSP_PRO_STATUS_CD')
DspProGroupCd(Calculated 'DSP_PRO_GROUP_CD')
DspProTypeCd(Calculated 'DSP_PRO_TYPE_CD')
DspProSalesUsrCode(Calculated 'DSP_PRO_SALES_A
DspProUnderwriterUsrCode(Calculated 'DSP_PRO_UI
DspProCollectorUsrCode(Calculated 'DSP_PRO_COLL
DspProCountryCd(Calculated 'DSP_COUNTRY_CD')
DspProStateCd(Calculated 'DSP_STATE_CD')
DspProFaxPrefix1Cd(Calculated 'DSP_PRO_FAX_PRE
DspProFaxPrefix2Cd(Calculated 'DSP_PRO_FAX_PRE
DspAchAccountNbr(Calculated 'DSP_ACH_ACCOUNT
DspFedTaxNbr(Calculated 'DSP_FED_TAX_NBR')
ProAddressNbr(UixProEO:PRO_ADDRESS_NBR)
ProBusId(UixProEO:PRO_BUS_ID)
DspProBusNbr(Calculated 'DSP_PRO_BUS_NBR')
ProCus1(UixProEO:PRO_CUS1)
Attribute:
ProCus1
New...
Help
OK
Cancel
OracleFSLLEnterpriseApp - Custo
View without Customizations
Edit with following Customizations
Tip layer
Name
region

Task: Edit Entity Objects

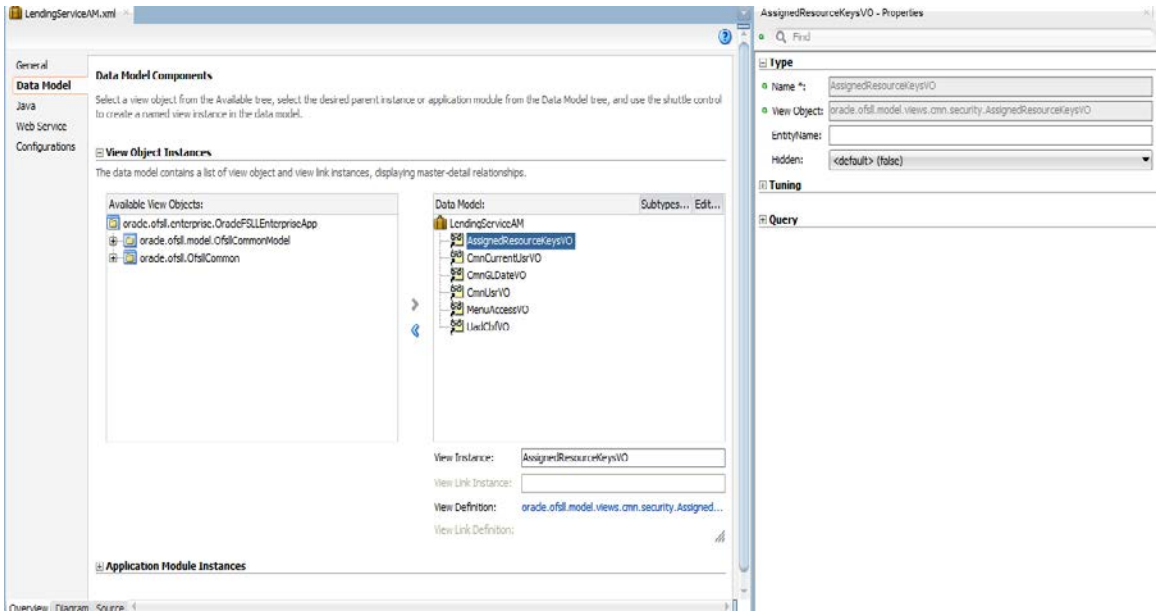
In JDeveloper, edit entity objects using the overview editor. In the Application Navigator, right-click an entity object, and choose **Open**. Then click on the navigator tabs to view and edit the various features of the entity object.

Task: Edit View Objects

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



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.

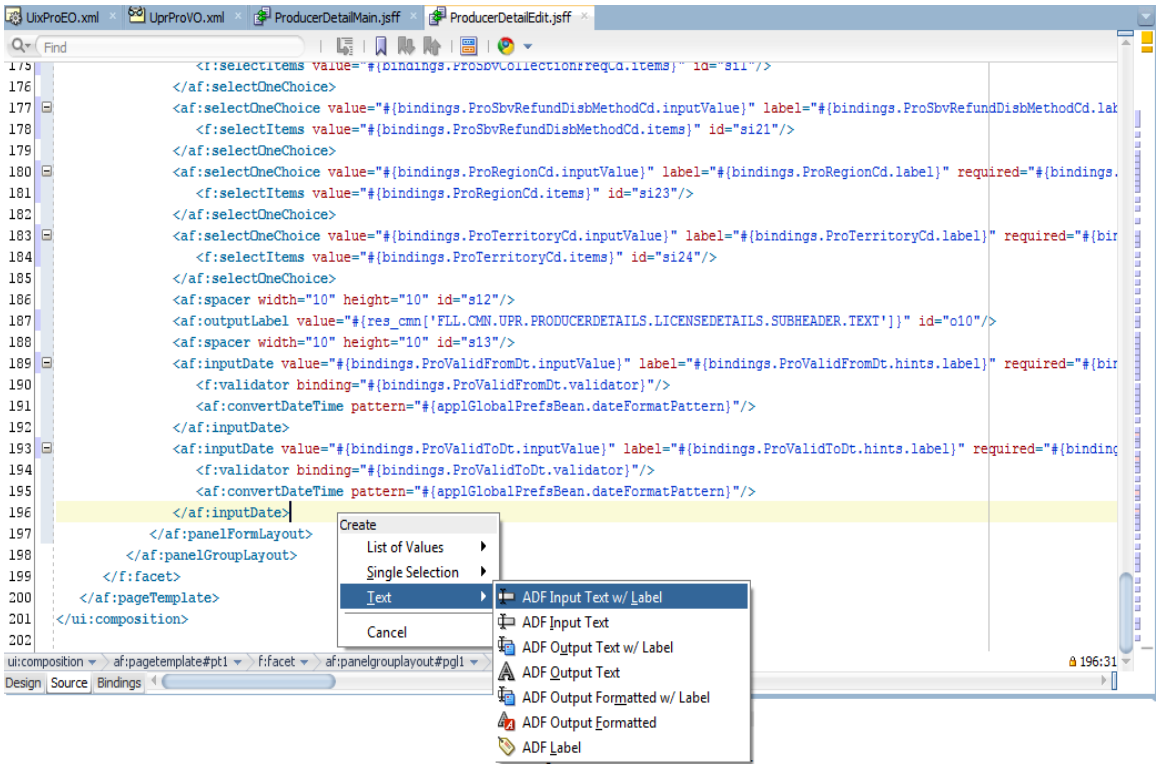
Once the changes are applied, the MDS file is created based on the customization layer value. shown below.

4.7 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.



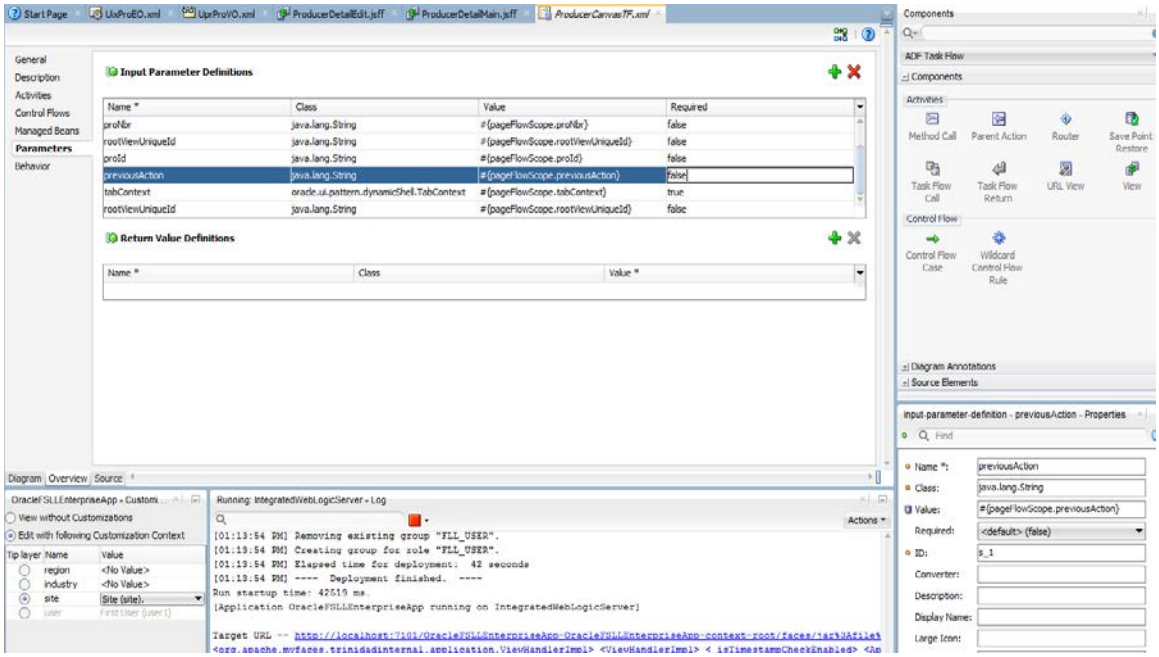
4.8 Editing Task Flows

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.



4.9 Creating 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.

4.10 Creating 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.

4.11 Creating 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

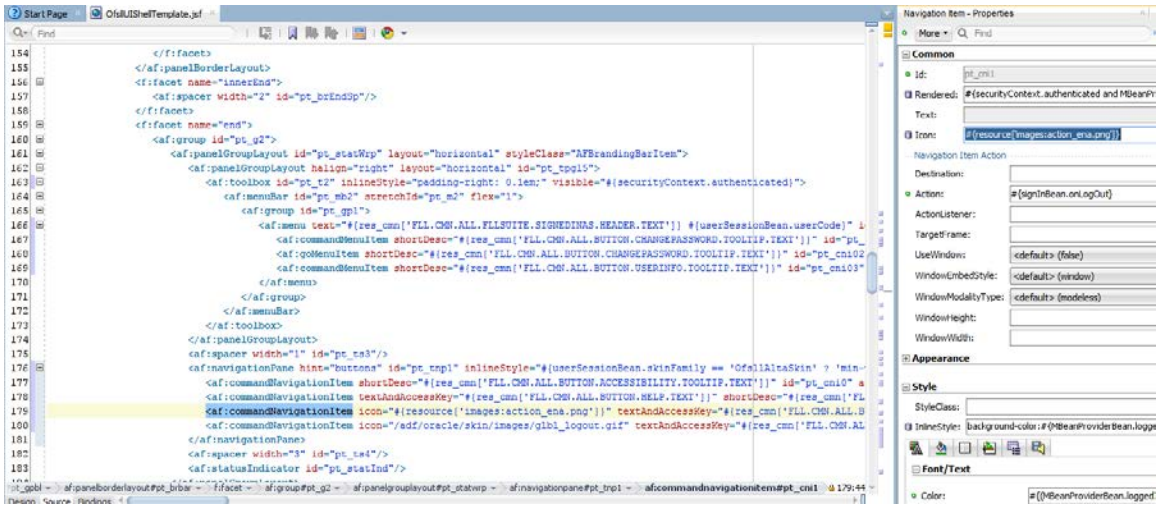
In JDeveloper, pages can be created either by double-clicking a view activity in a task flow or by using the New Gallery. In the Application Navigator, right-click the project to which the page has to be added to, and choose **New**. Then in the New Gallery, expand Web Tier, and click JSF/Facelets. Then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

4.12 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.



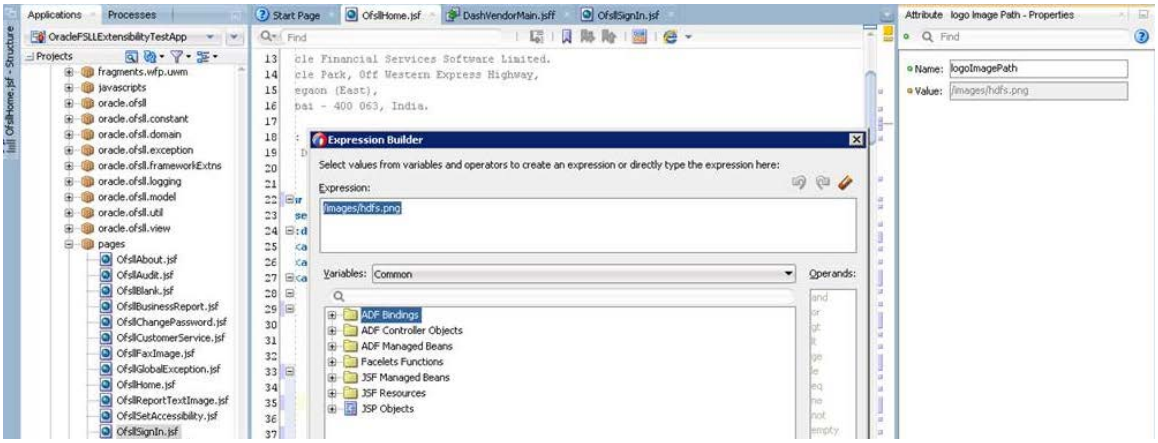
4.13 Replacing OFSLL Logo

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.

4. From property inspector's expression builder, change the value to refer the new logo file and click ok. Customization file for OfsllHome.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 OfsllSignIn.jsf.



4.14 Deploying 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

4.15 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

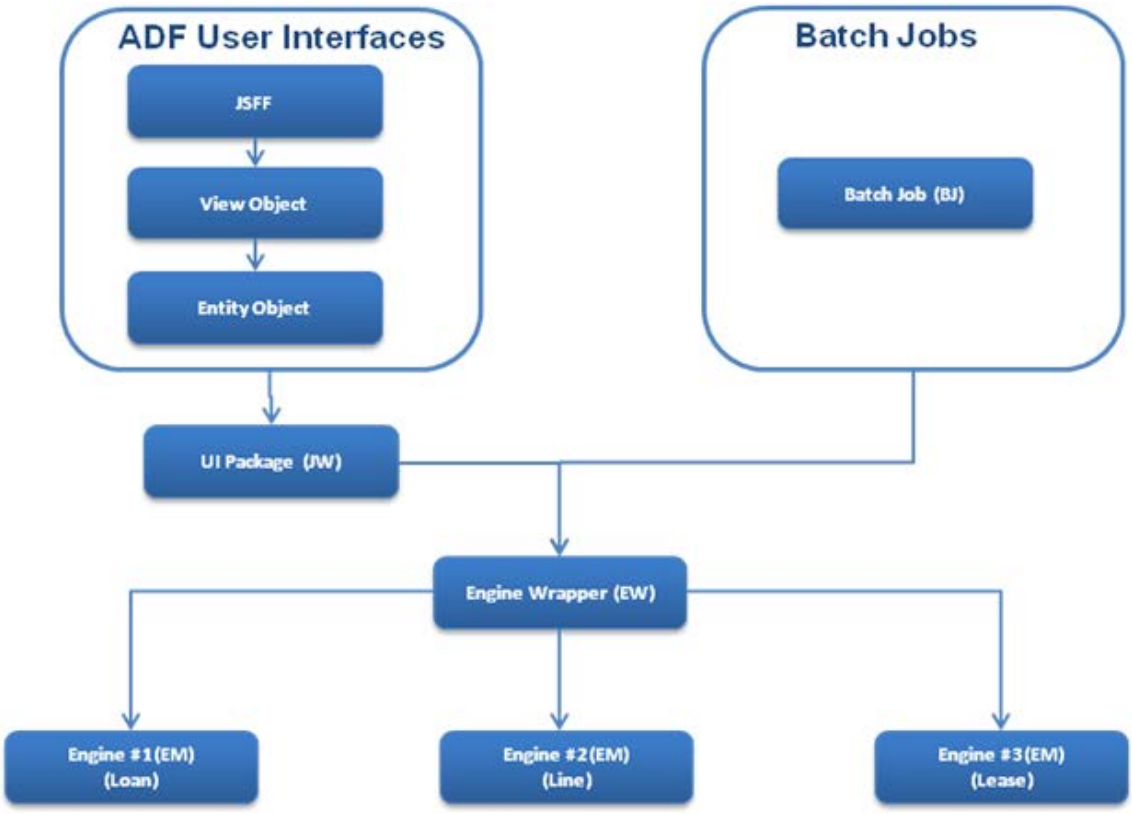
- In *Customization Developer* role, the project creates the MAR deployment profile for customization deployment.

- MAR deployment is same as EAR deployment.

5. Customizing Database Objects

5.1 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.

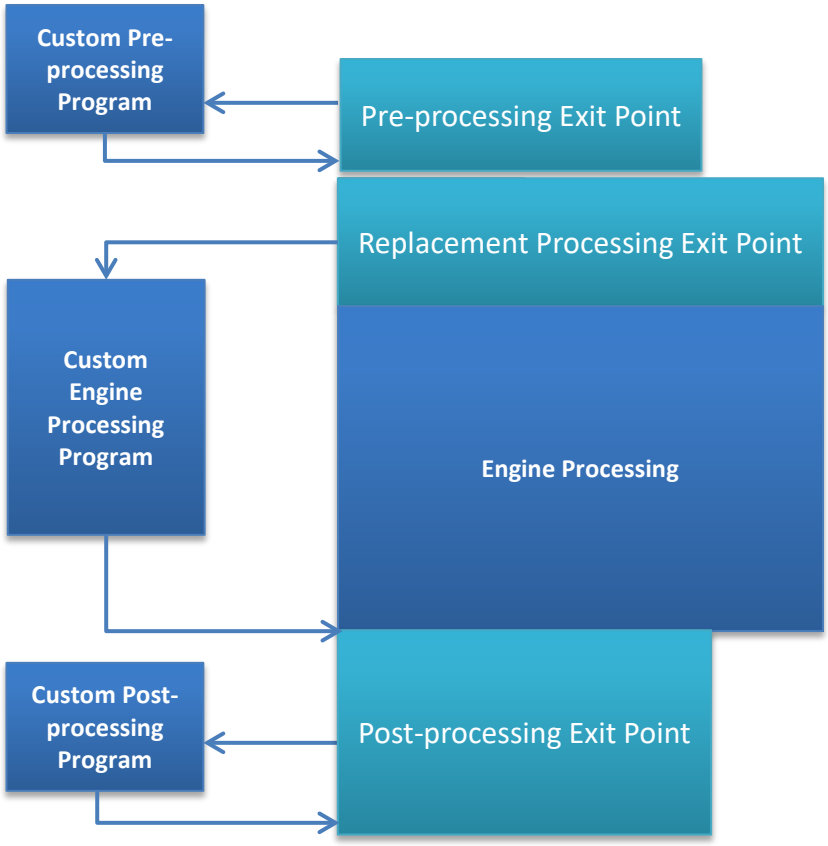


5.2 UI Java Wrapper (U*JW)

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)



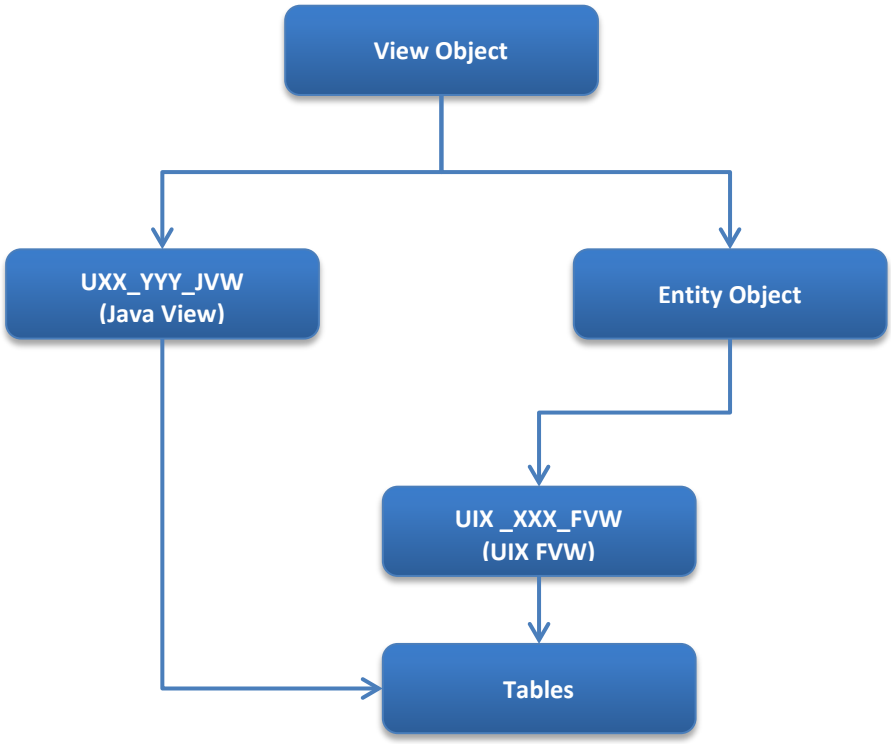
Business Logic Engine (Process)

5.3 Database Schema

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

- Table
- Table Column
- Sequence
- Index
- View

- FVW – User Interface Views
- JVW – Java Interface Views
- EVW – Engine/Wrapper Signature Views
- PL/SQL Programs



5.4 Wrapper Engine model

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

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

XXXYYY_ZZ_ABC_99		XXXYYY_ZZ_**ABC_99**	
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)
		99	Running Sequence Number
			Starting 01 to 99

5.5 Batch Job (BJ)

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

5.6 Engine Wrapper (EW)

Engine Wrapper **cannot** be customized.

5.7 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).

5.8 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).

5.9 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.

5.10 Common Features

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

5.11 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 is in files and checked in the version control systems and sent as merged statements in patch for changed (added or modified) data.

5.12 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 “YYYYMMNNNNNNND” 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 **xyz**aaiacc_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 “**xyz**aaiprc_ex_111_01.pks”, change constant

  From

  CV_SET_ACC_NBR_REP CONSTANT VARCHAR2(30) := cmncon_cl_000_01.NOT_CUSTOMIZED;

  To

  CV_SET_ACC_NBR_REP CONSTANT VARCHAR2(30) := cmncon_cl_000_01.CUSTOMIZED;
- Copy “aaiprc_ex_111_01.pkb” to “xyz aaiprc_ex_111_01.pkb”
- Call new procedure from **xyz**aaiprc_ex_111_01

```
PROCEDURE set_acc_nbr_rep(
                          iv_con_rec IN aai_con_evw%ROWTYPE
                         ,iv_acc_aad_id IN OUT aai_con_evw.con_aad_id%TYPE
                         ,iv_acc_nbr    IN OUT aai_con_evw.con_acc_nbr%TYPE) IS
BEGIN
      xyzaaiprc_en_111_01.set_acc_nbr(iv_con_rec,iv_acc_aad_id, iv_acc_nbr );
END set_acc_nbr_rep;
```

The above example shows the usage with replacement exit point. Similar way “before” and “after” exit points can be used to extend the business logic functions.

- Where ‘**xyz**’ is Customer Unique Id

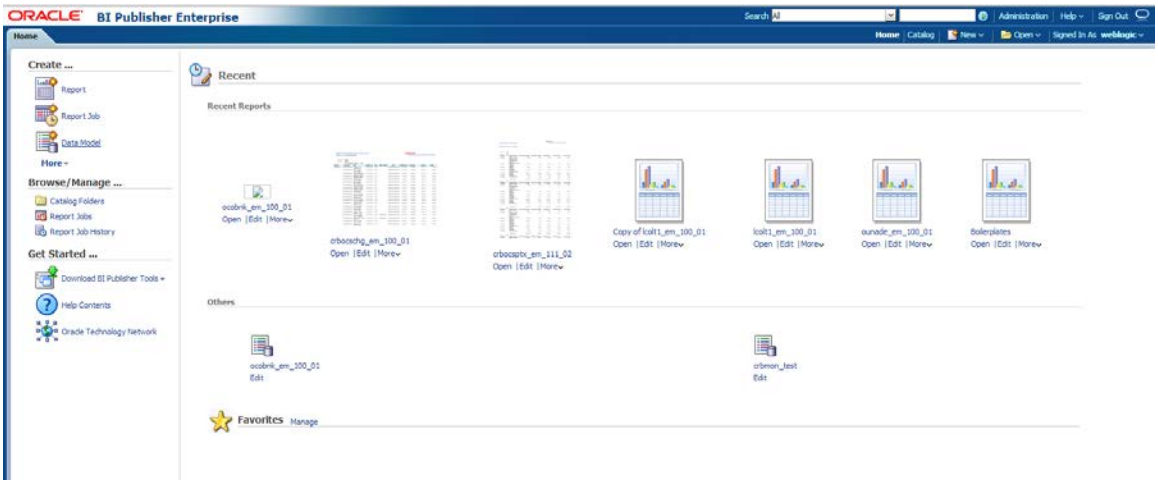
6. Creating New Custom BI Publisher Report/Letter

Pre-Requisites

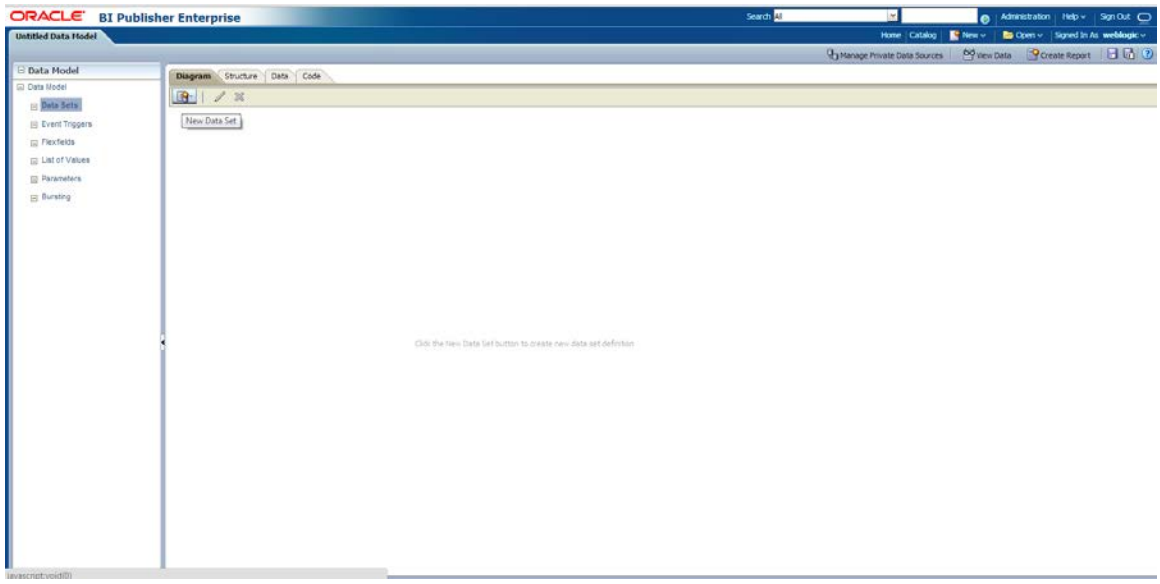
1. Changes to base reports not allowed
2. Basic knowledge on BIP and BIP client.
3. The reports should be placed into the same folder structure
   - i.e. For Reports → Shared Folders/oracle/fll/xmlp/reports
   - For Letters → Shared Folders/oracle/fll/xmlp/letters
   - For Correspondences → Shared Folders/oracle/fll/xmlp/correspondence

Creating a New Report

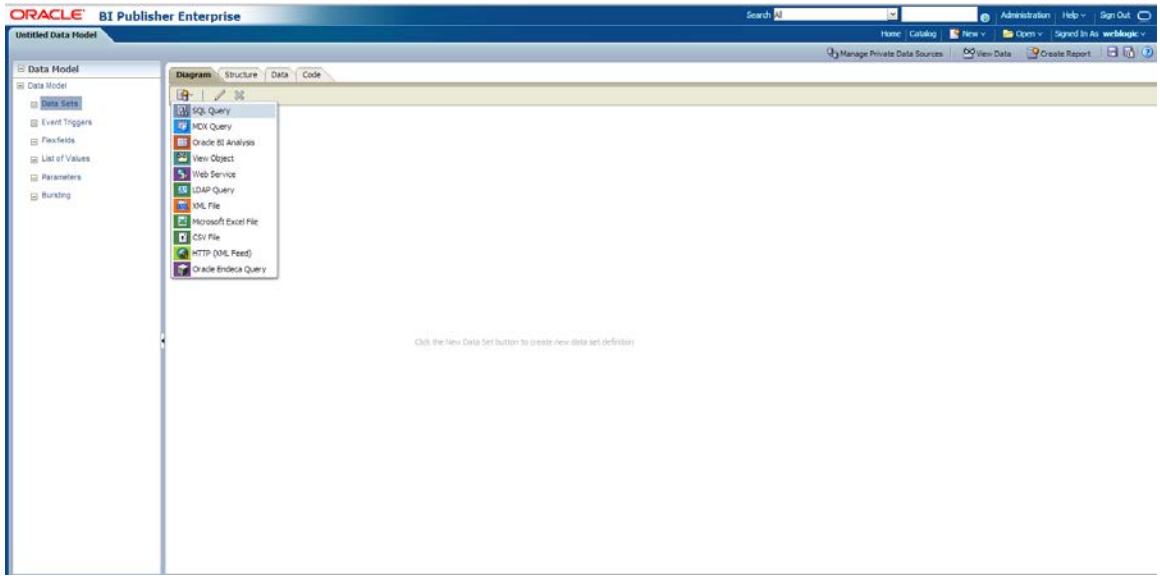
1. Login into BIP console.
2. To Create a new report first create the data model , Click on Data Model on left.



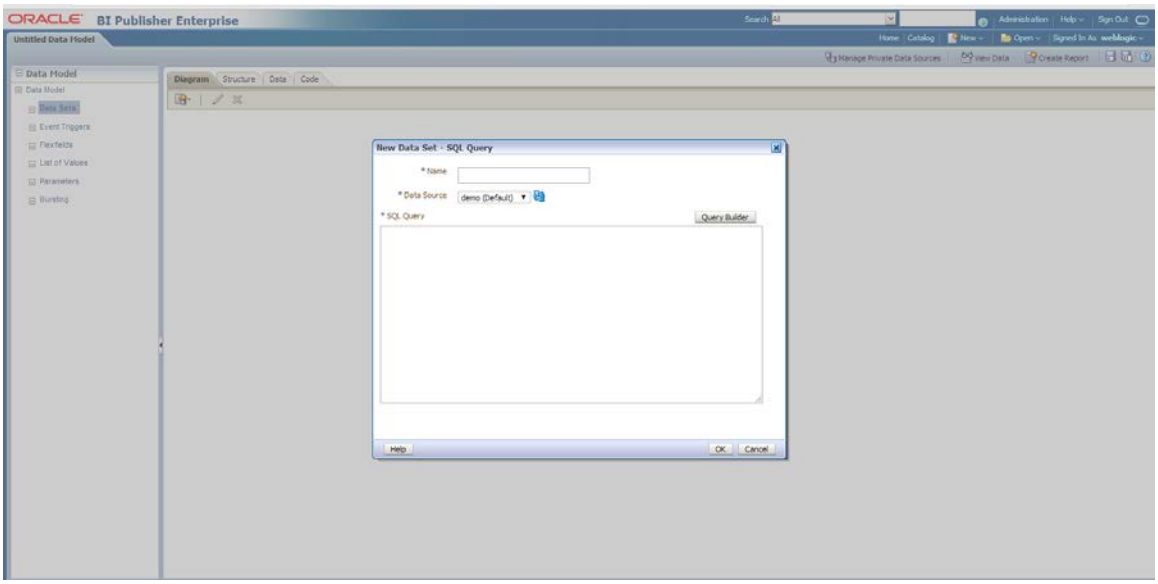
3. On click on Data Model it will open the following Screen.



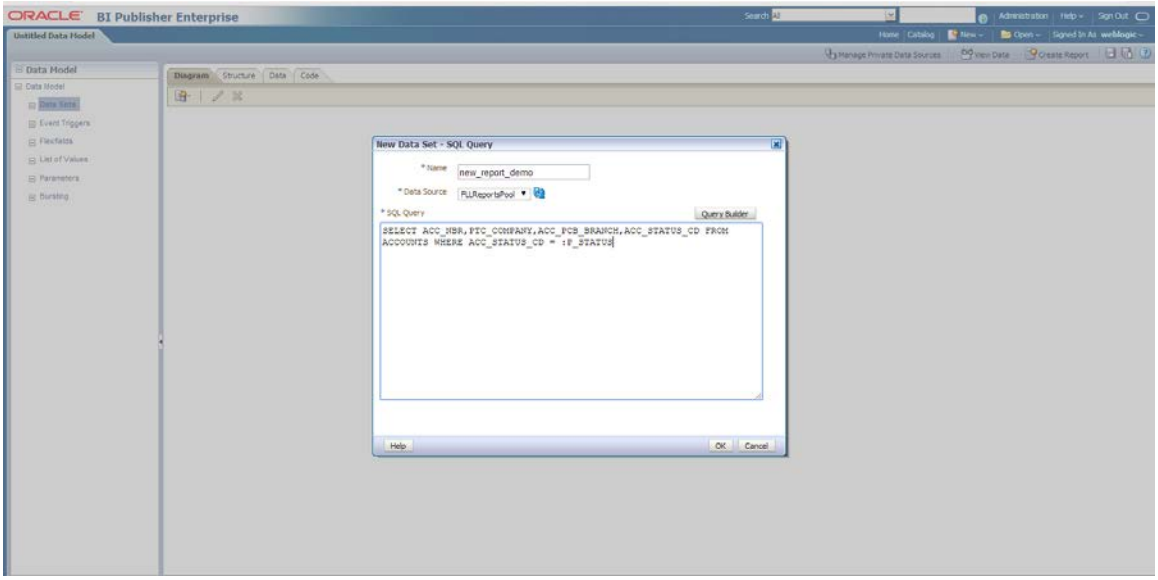
4. To create a New Data Set, Click on New Data Set and select the type “SQL Query”



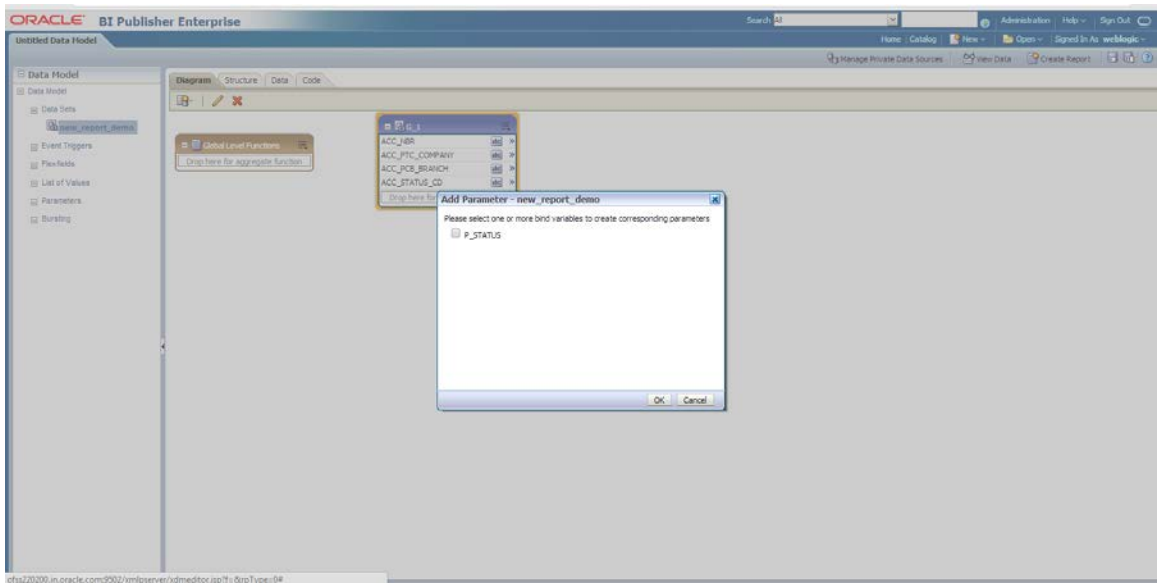
The following screen is displayed.



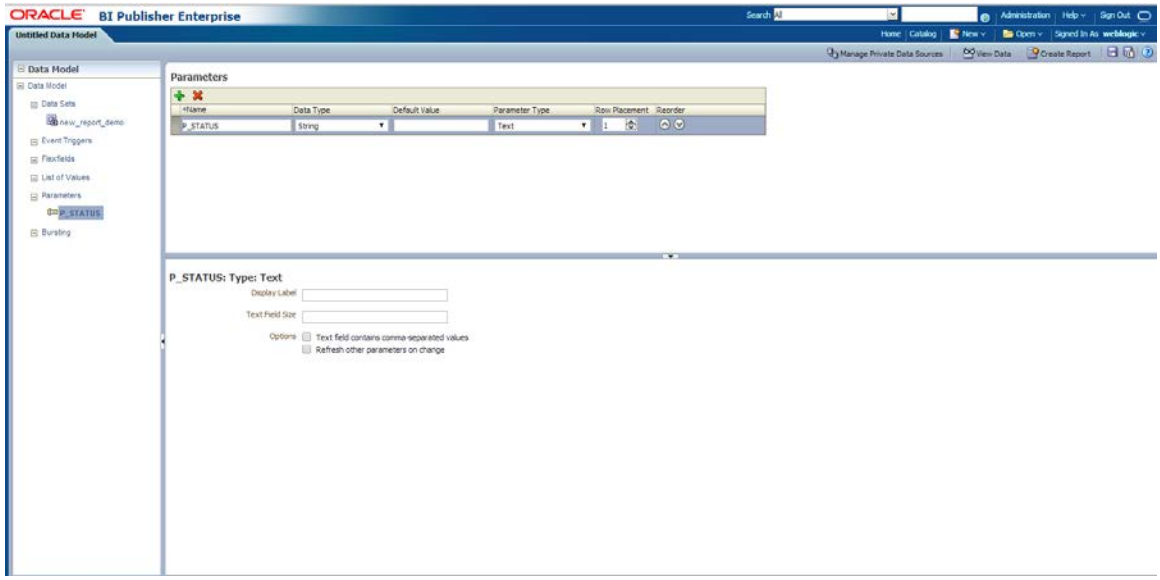
5. Enter the new data model name (Use the same name while creating the report layout)
6. Select the Data Source.
7. Add the sql query. It depends on what data needed on the report and what should be the parameters.



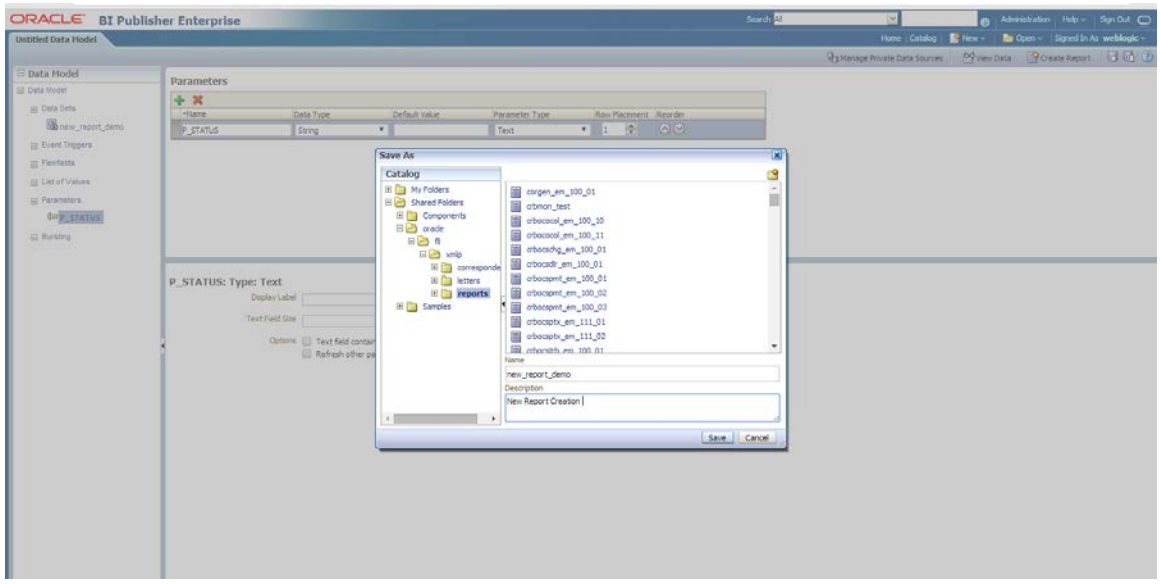
8. Click OK. A confirmation dialog is displayed to create the parameter. Select the parameter and press OK.

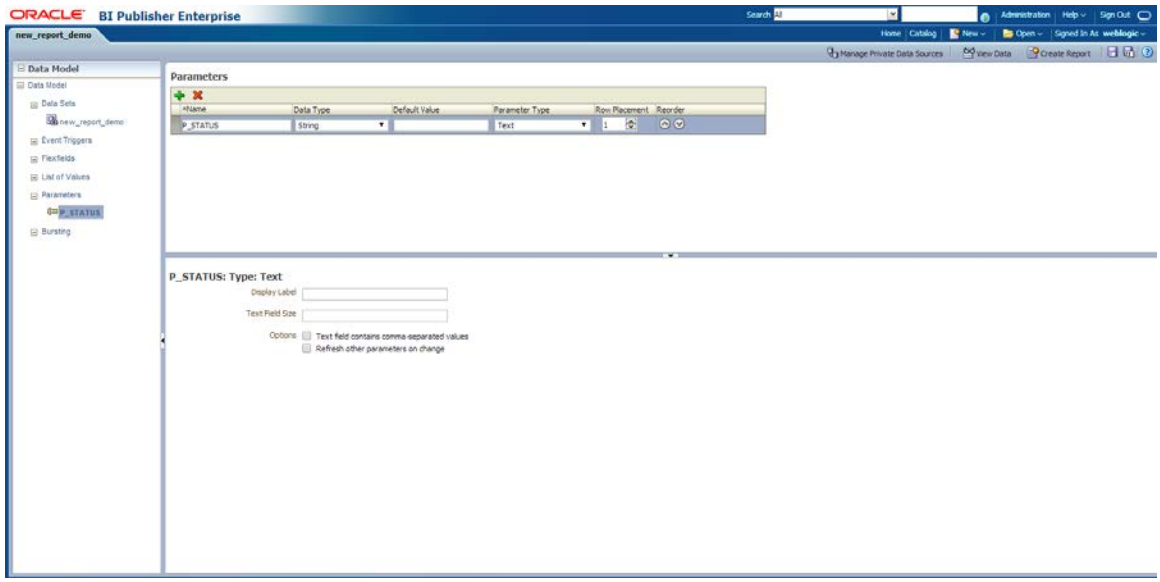


9. The parameter is created as indicated below.

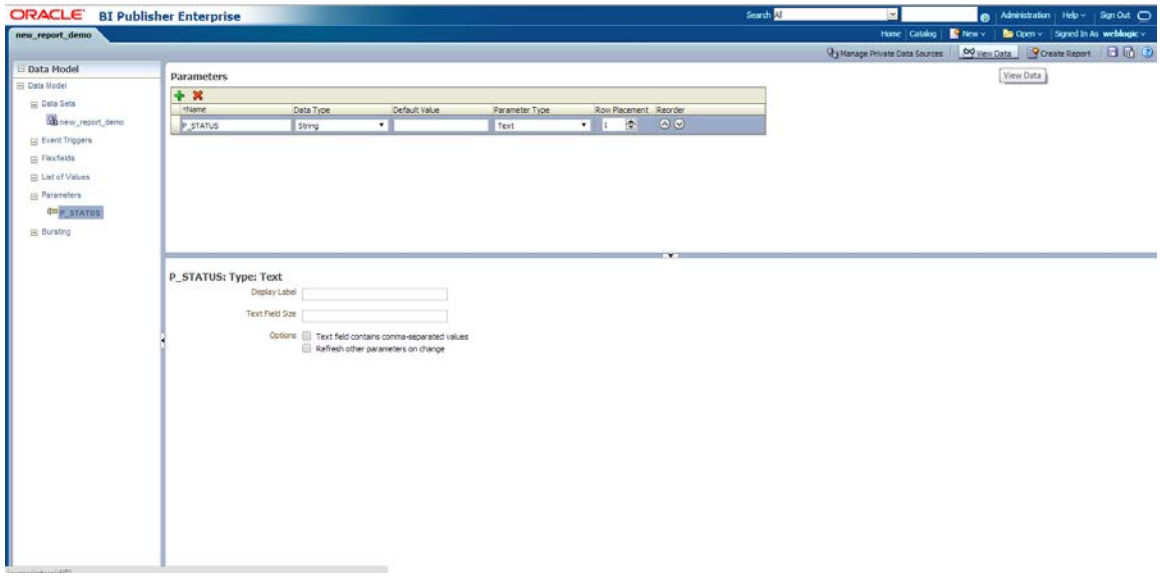


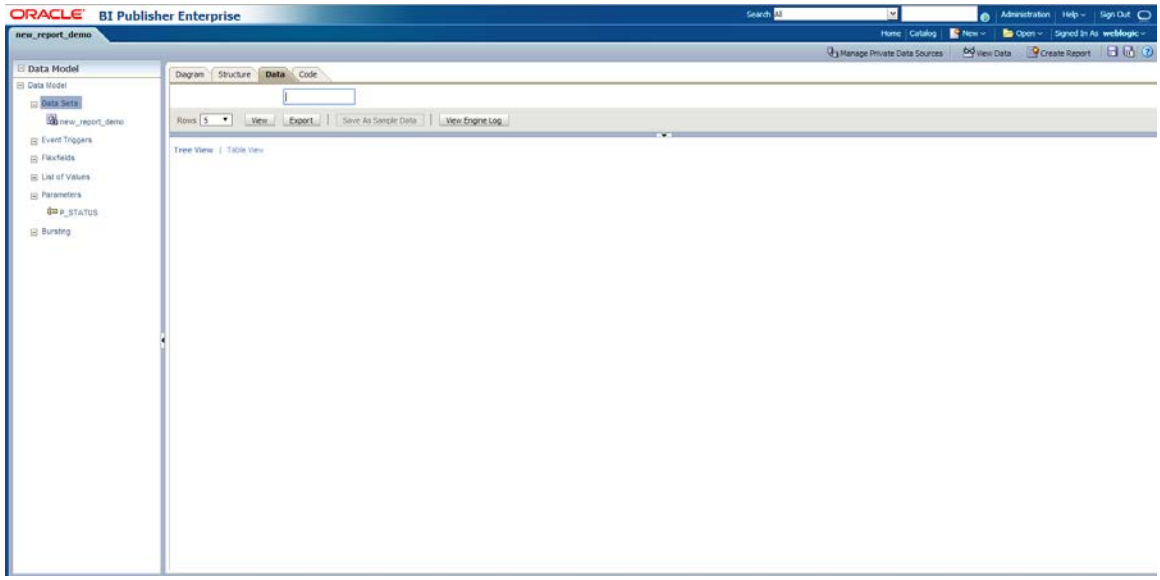
10. Save the data model. Select the directory in which you would need to save the details. For Reports save in reports directory and for Letters save it in Letters directory.
11. You can also save with the same name and specify the description of the report.



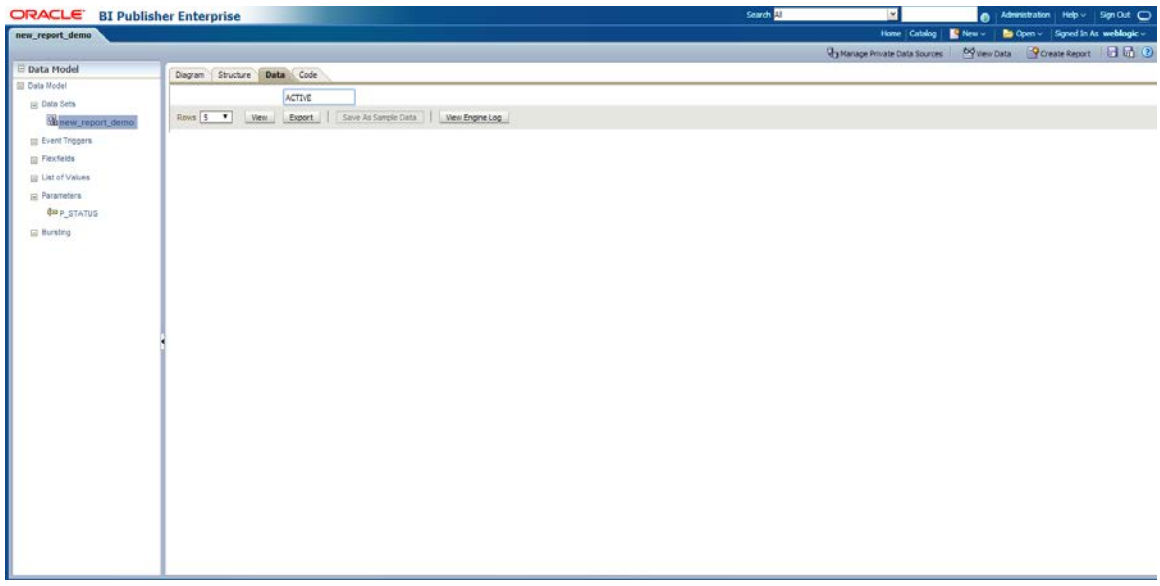


12. Once the Data Model is created, create the sample data as indicated below.
13. Click on View Data on right side of the page.

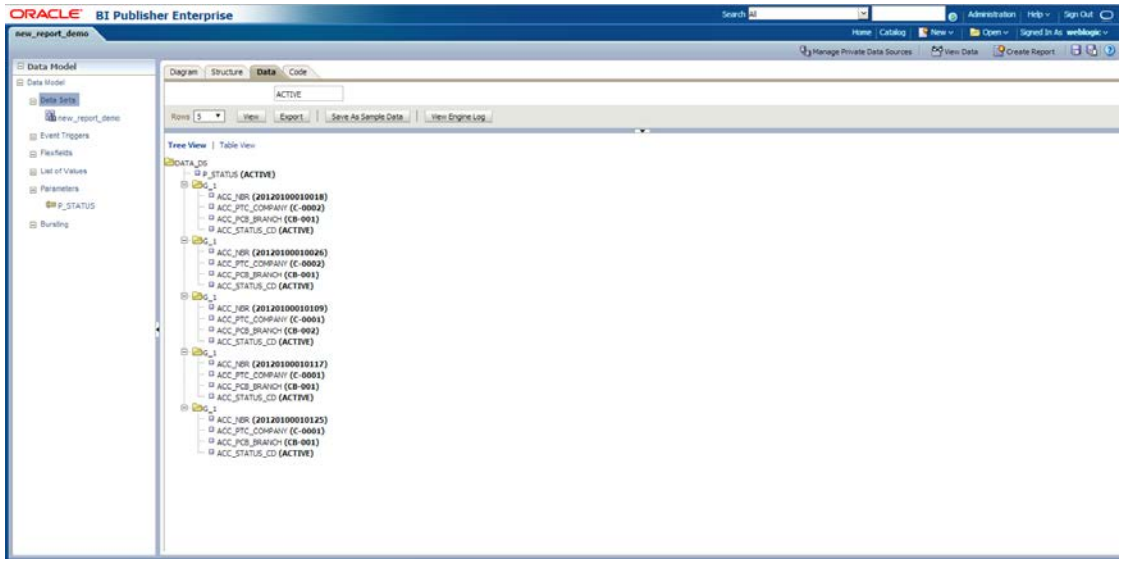




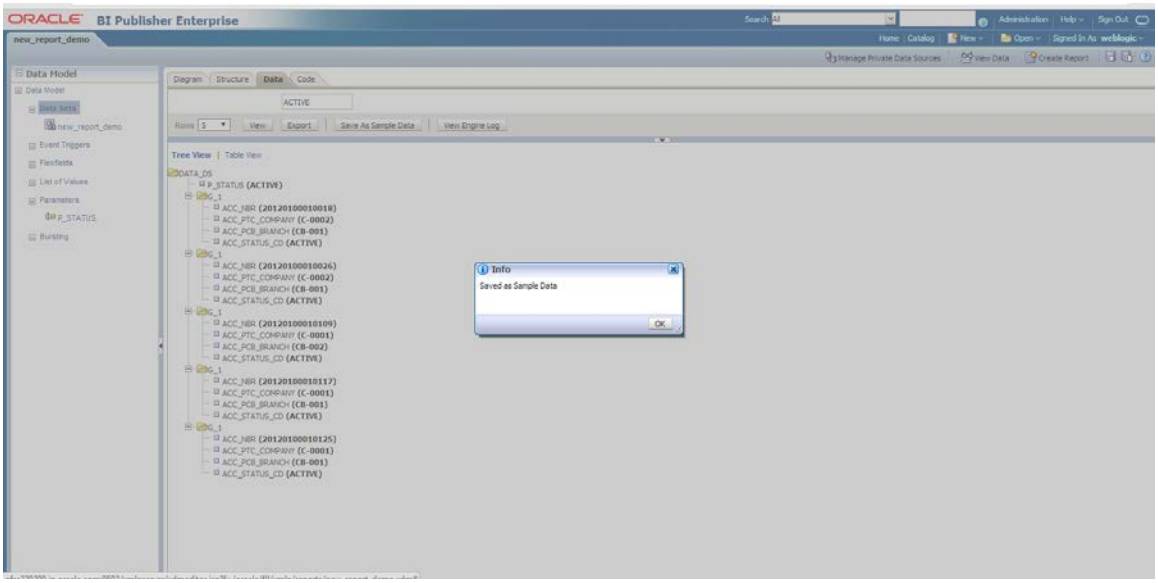
14. Specify the following value in Parameter (P_STATUS).



15. Click ‘View’. The sample data is created as indicated.



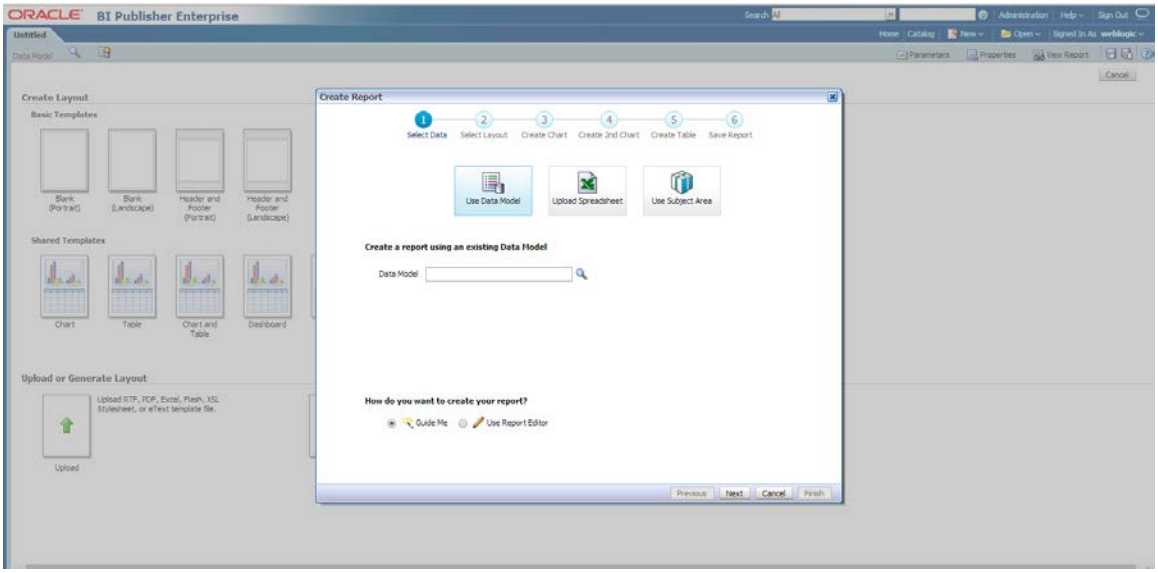
16. Click on ‘Save As Sample Data’.



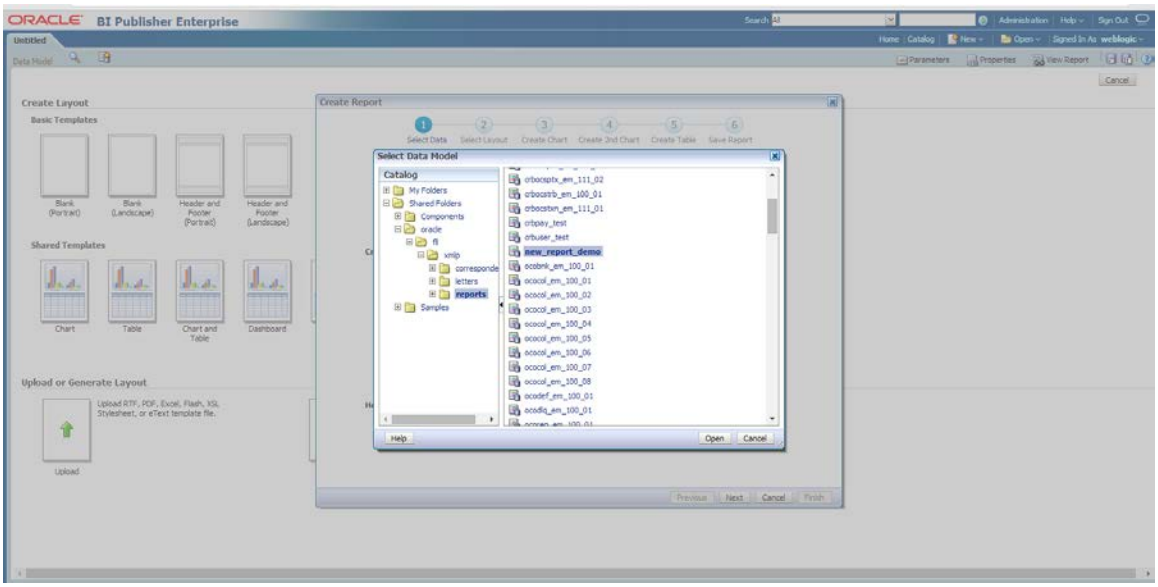
17. The Data model is created and can be use to create the report layout.

6.1 Create Report Layout

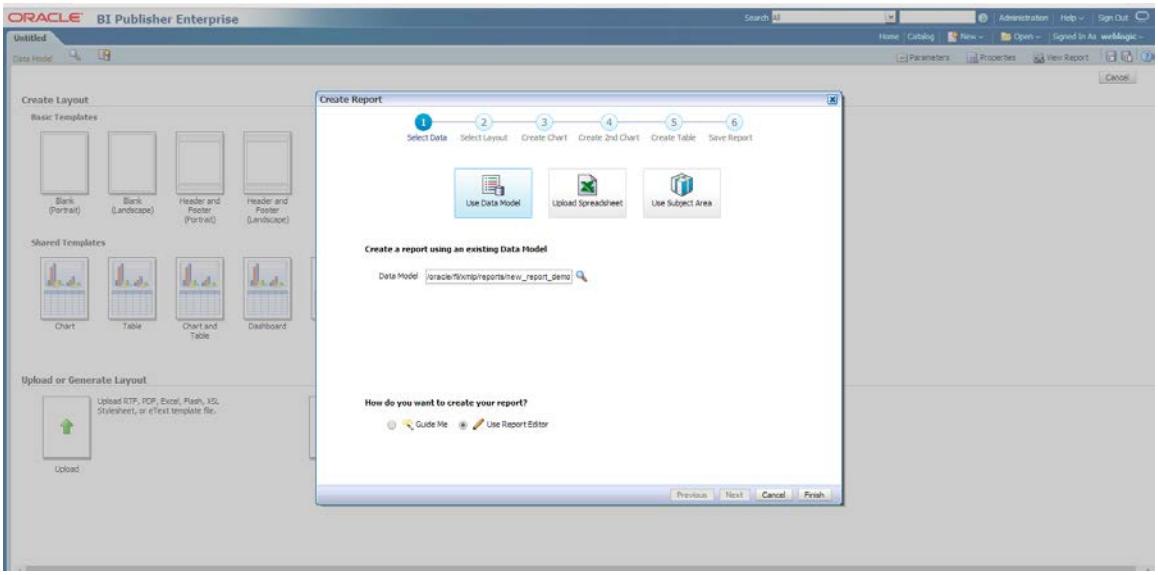
1. Navigate to Home Page and create the report layout as indicated below.
2. Click on Create > Report on LHS panel.



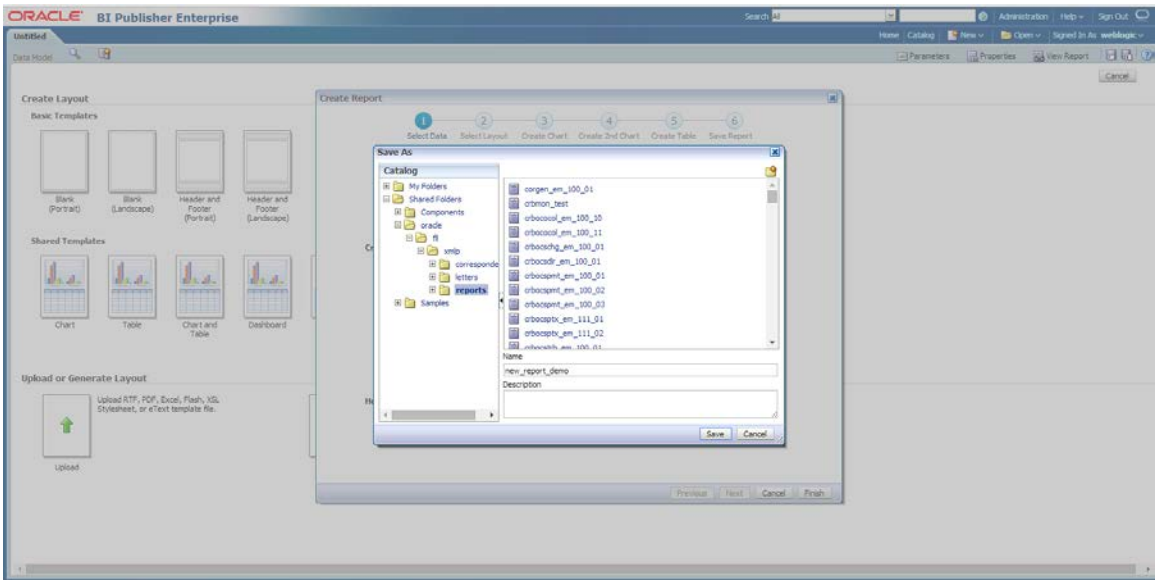
3. Select the ‘Use Data Model’ and click to browse the existing data model , and select the data model just created.



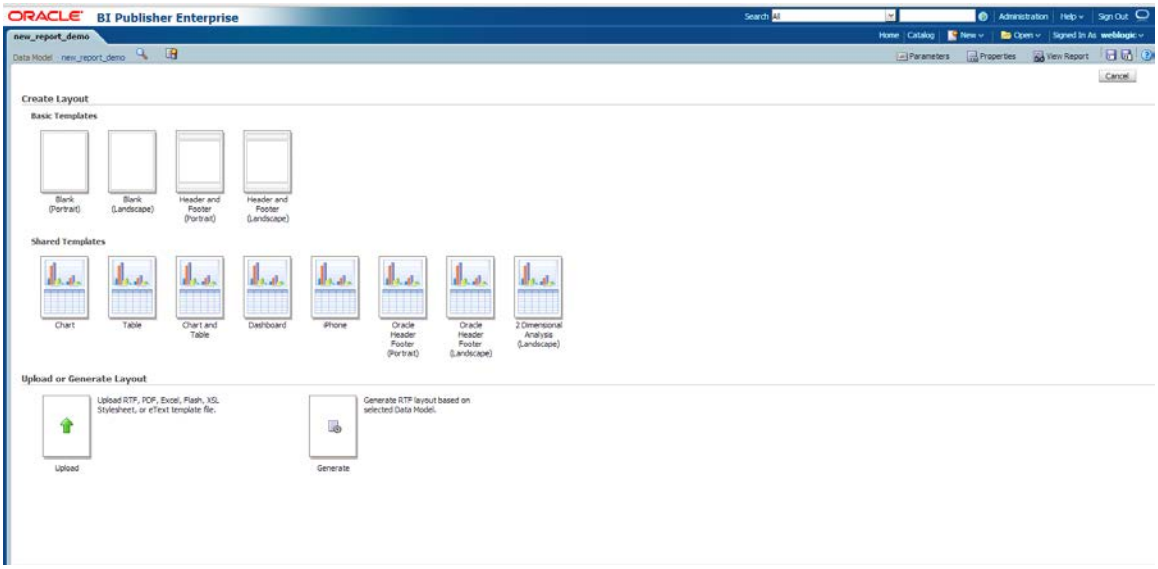
4. Click ‘Open’.



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.



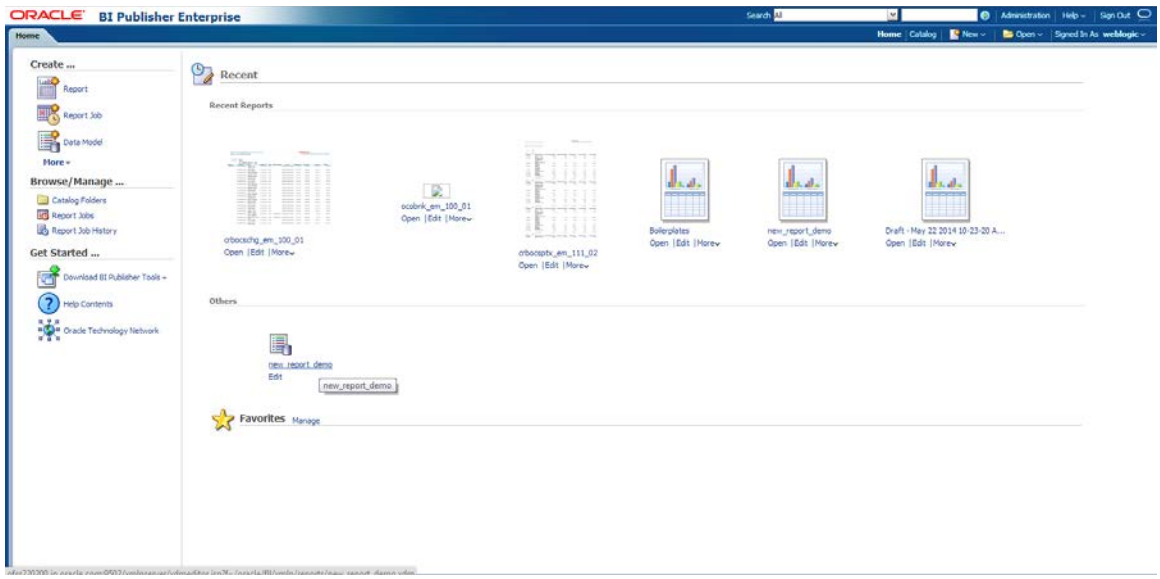
The below screen indicates the saved report.



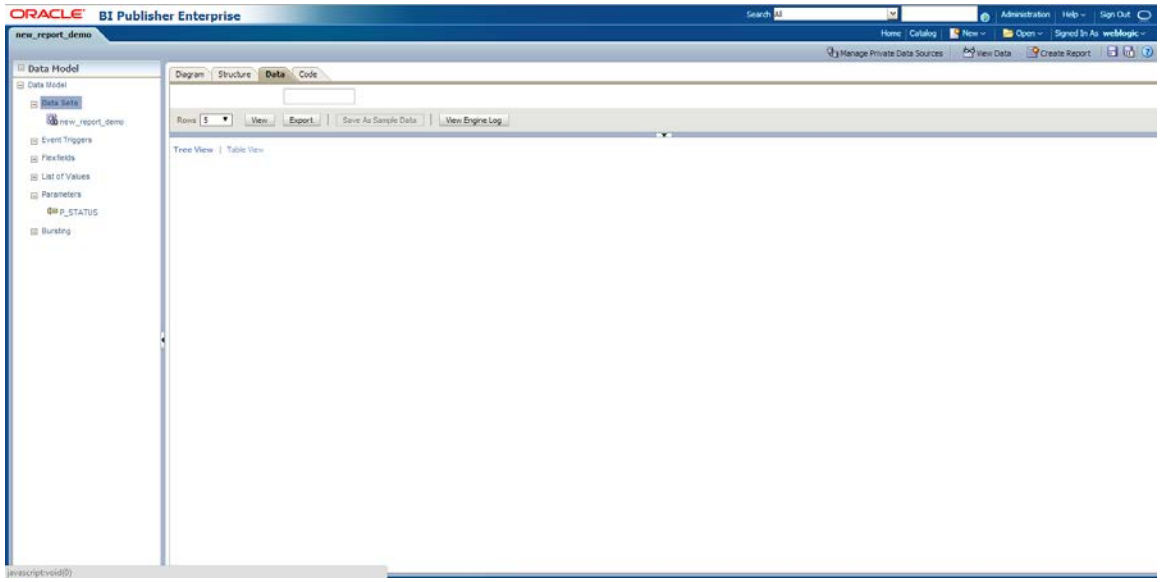
6.2 Create XML data

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

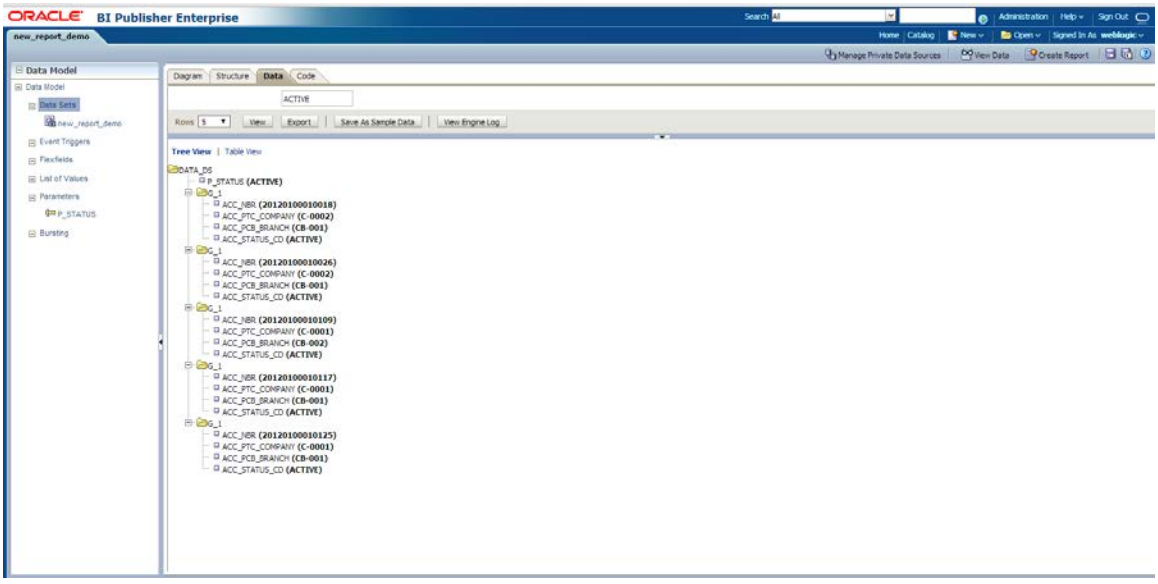
1. Navigate to Home Page and select the data model created.



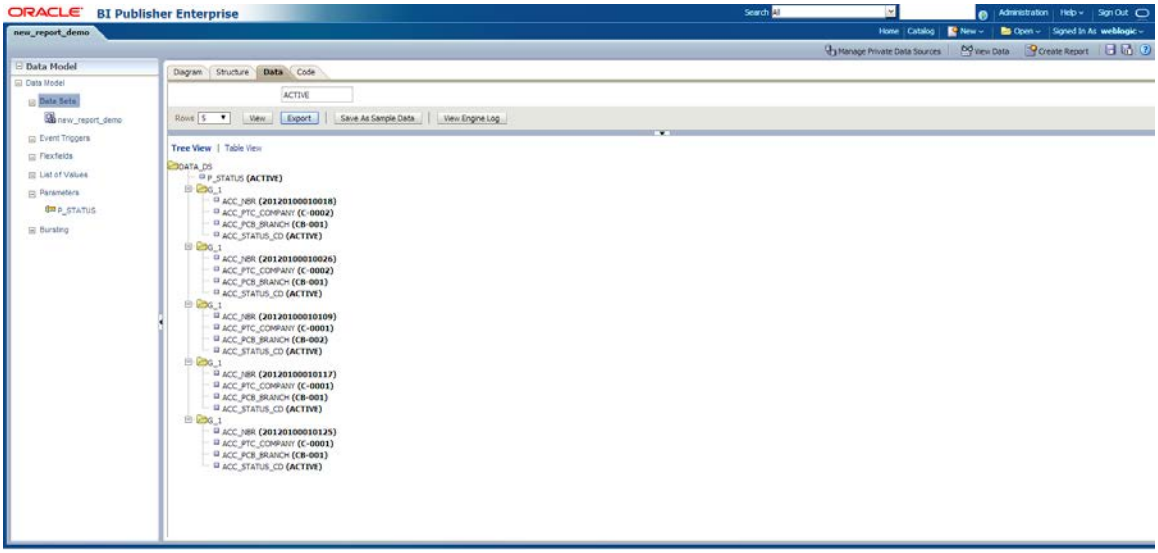
2. Select ‘Data’ Tab.



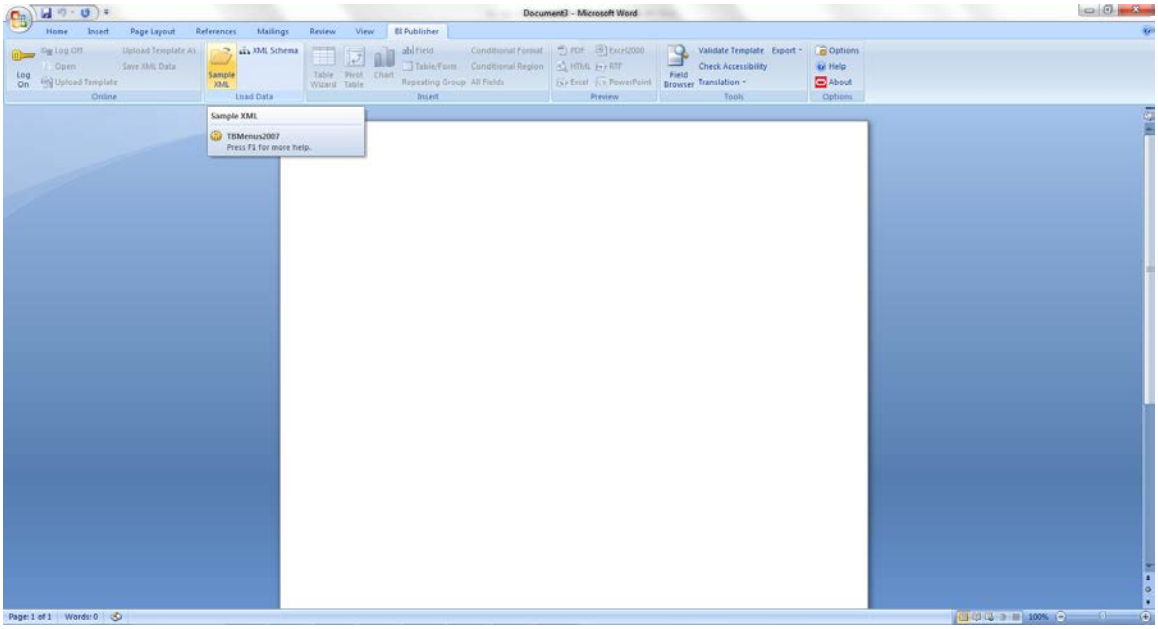
3. Enter the Parameter Value and click ‘View’.



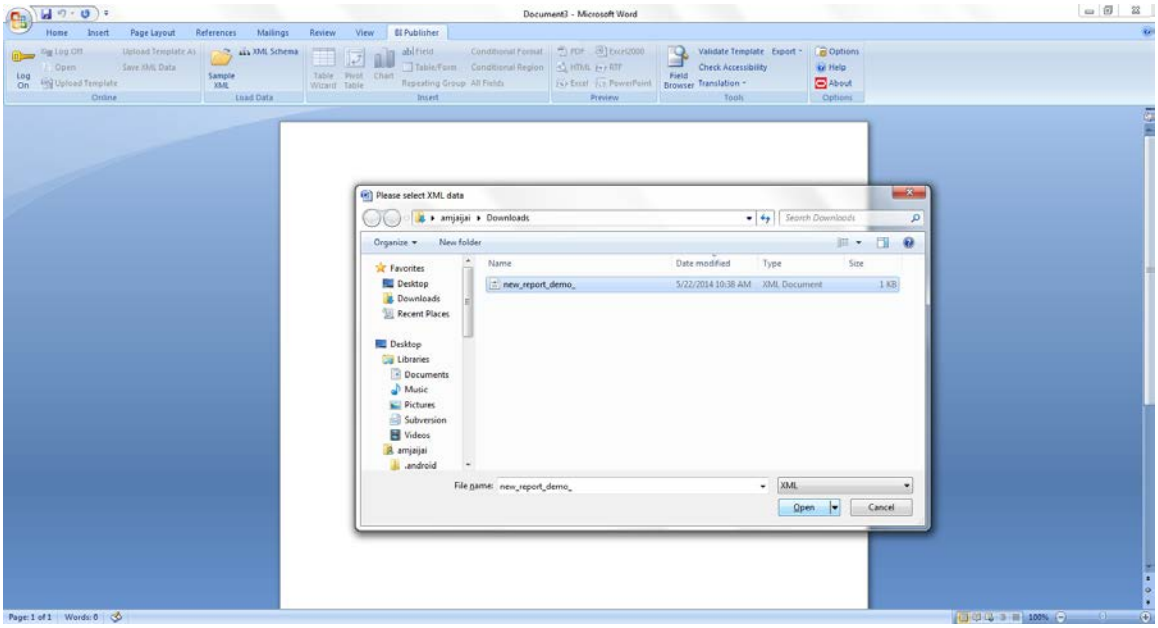
4. Click ‘Export’ (it will create the XML file locally in download folder).



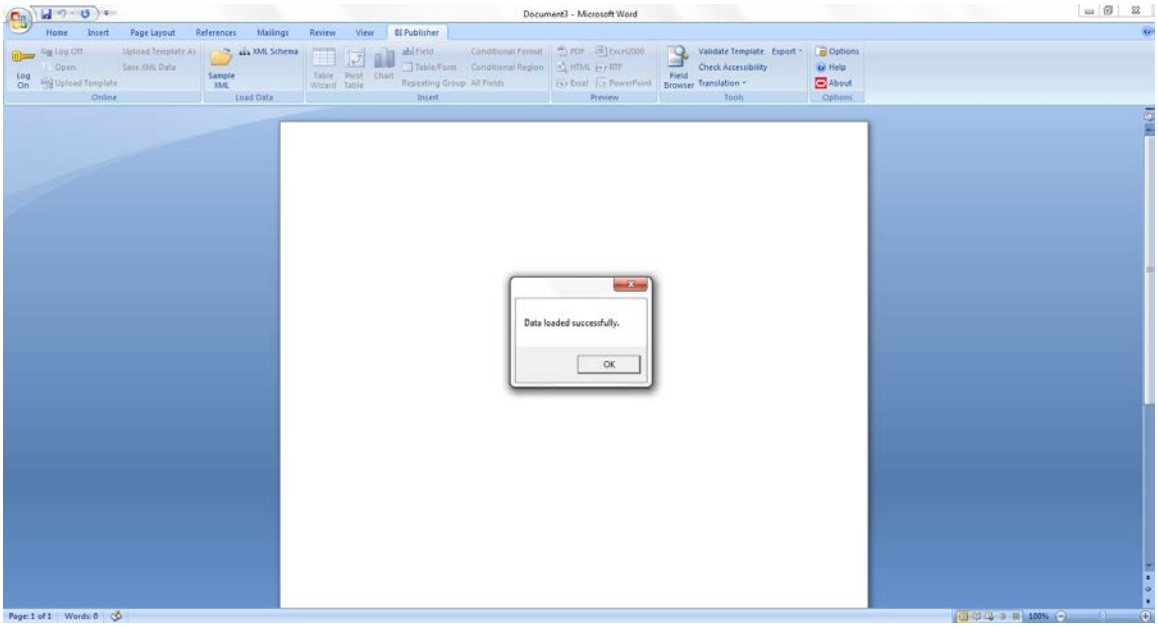
5. Open MS Word (Ensure that the BIP client is installed on User’s Machine).
6. Access BI Publisher tab and click ‘Sample data’.



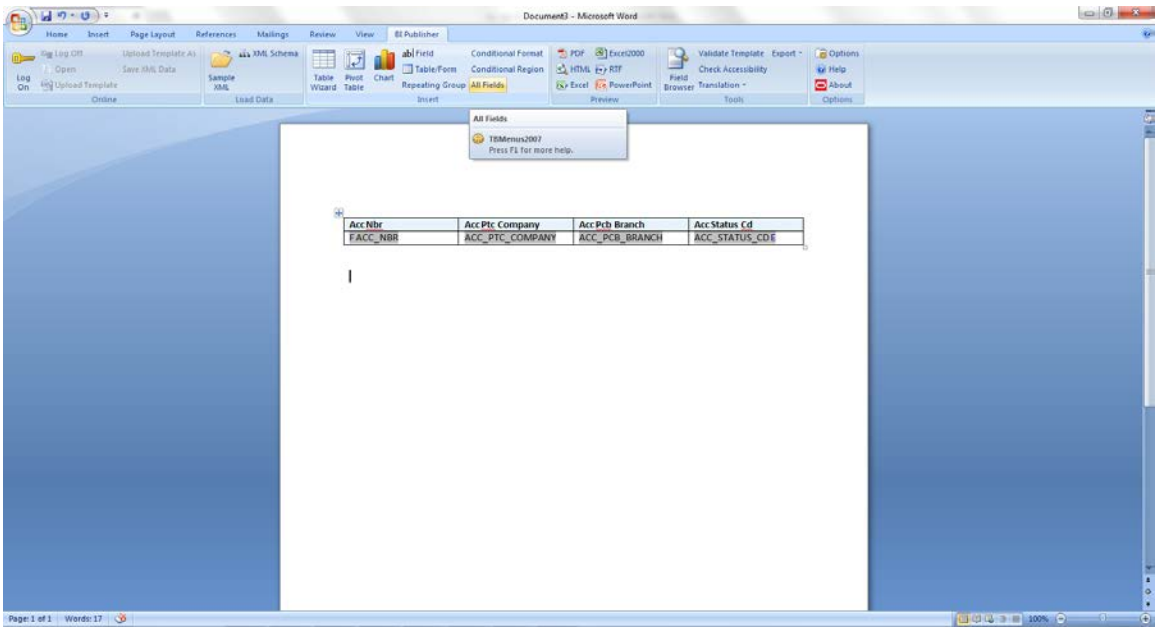
7. Select the same XML file from download.



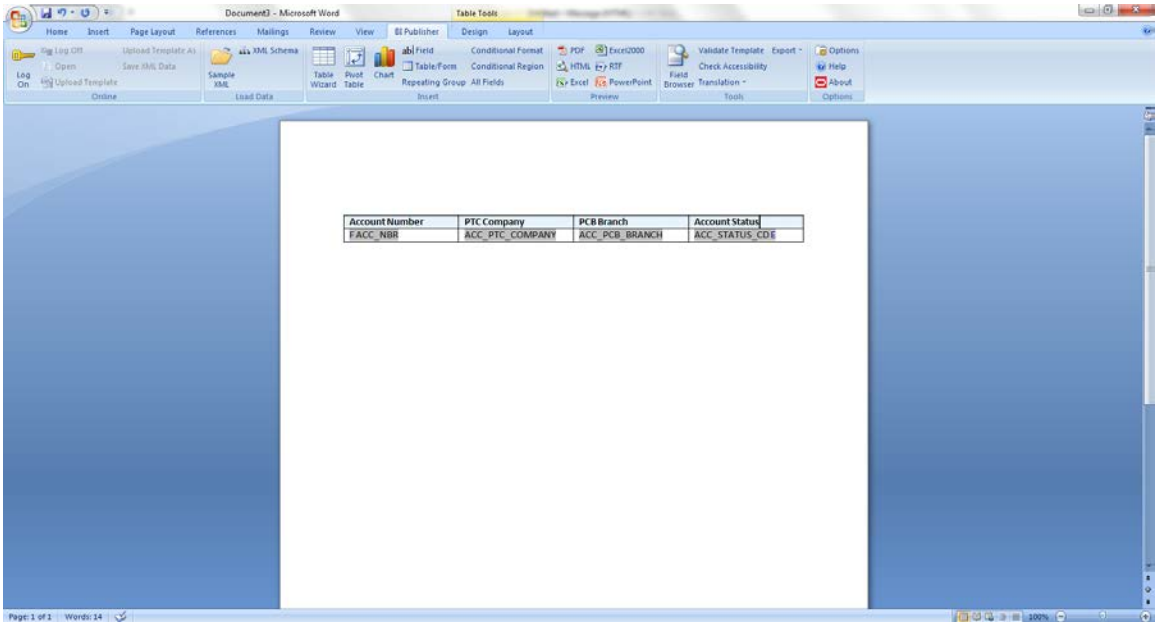
8. Select ‘Open’. Required data is loaded.



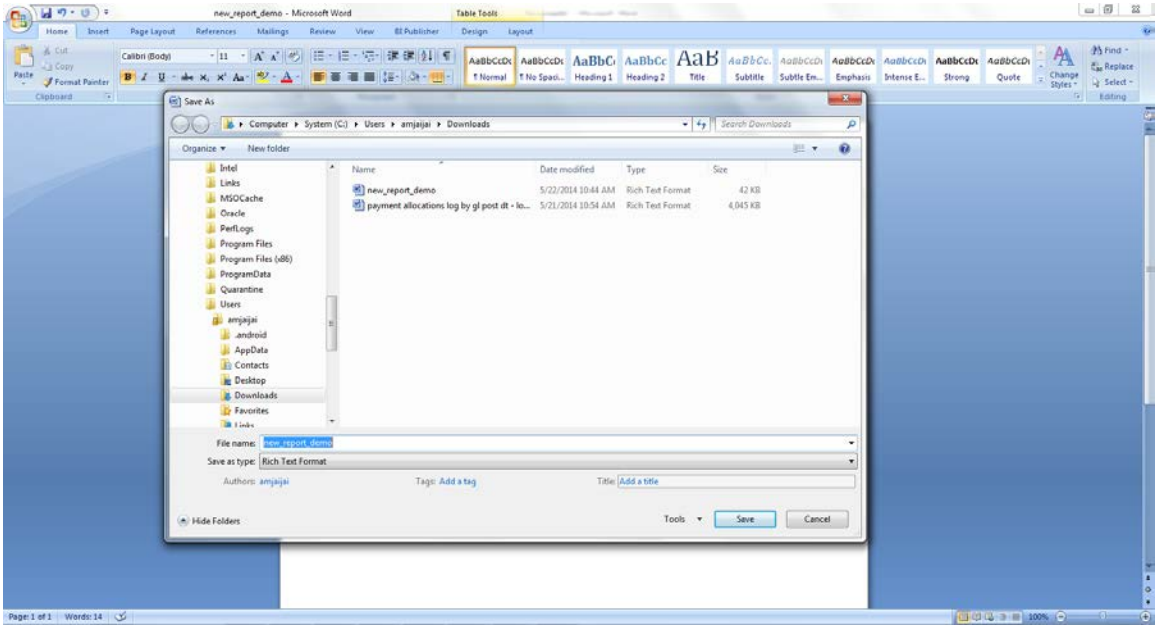
9. You can use the available option is MS word to create the desired layout.
10. First use “All Fields” option. This creates all the fields from sample xml and create the layout as follows in tabular form.



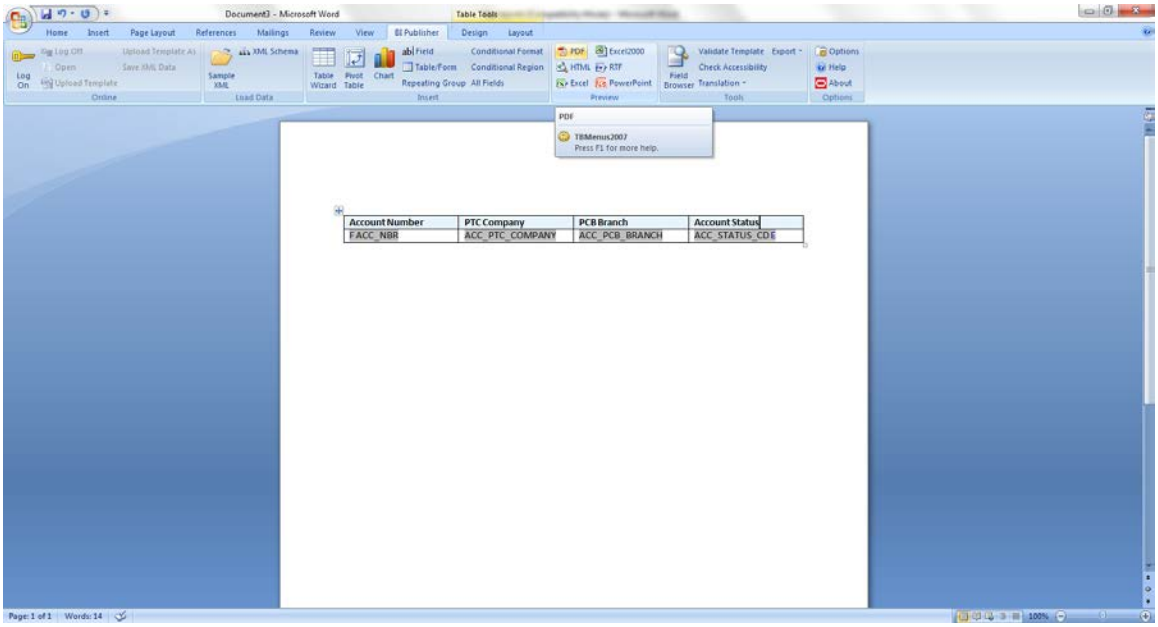
11. You can modify the headings, add new static text, logos and so on.



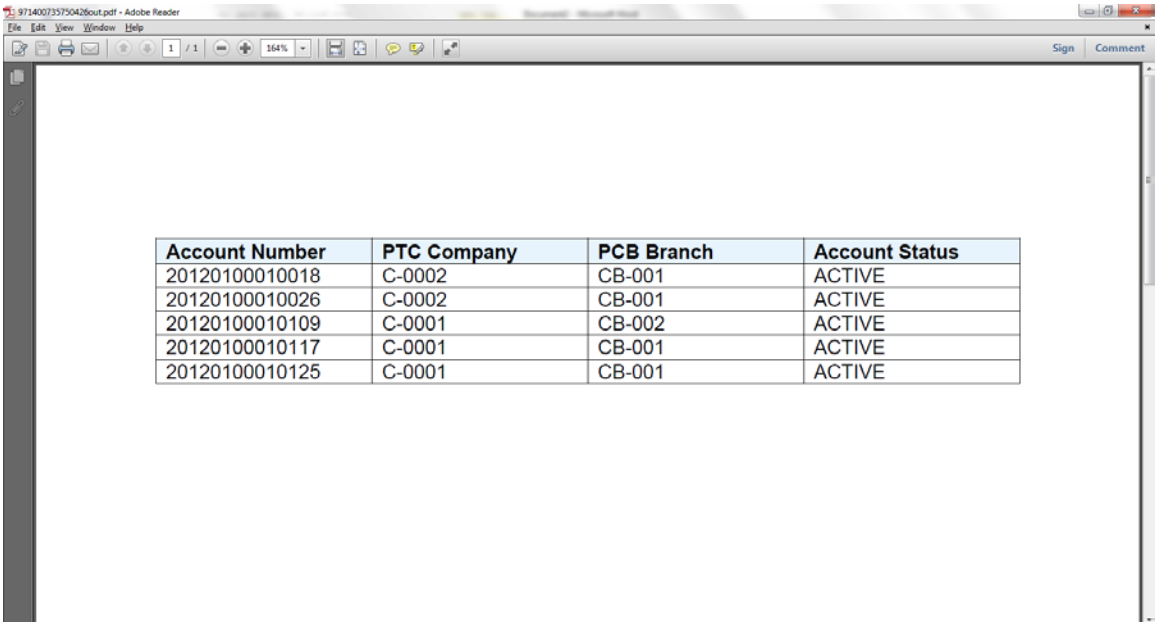
12. Save the layout with the same name as the data model and report in BIP.



13. To preview the layout, click ‘PDF’ in preview tab.



14. The report is displayed in PDF format.

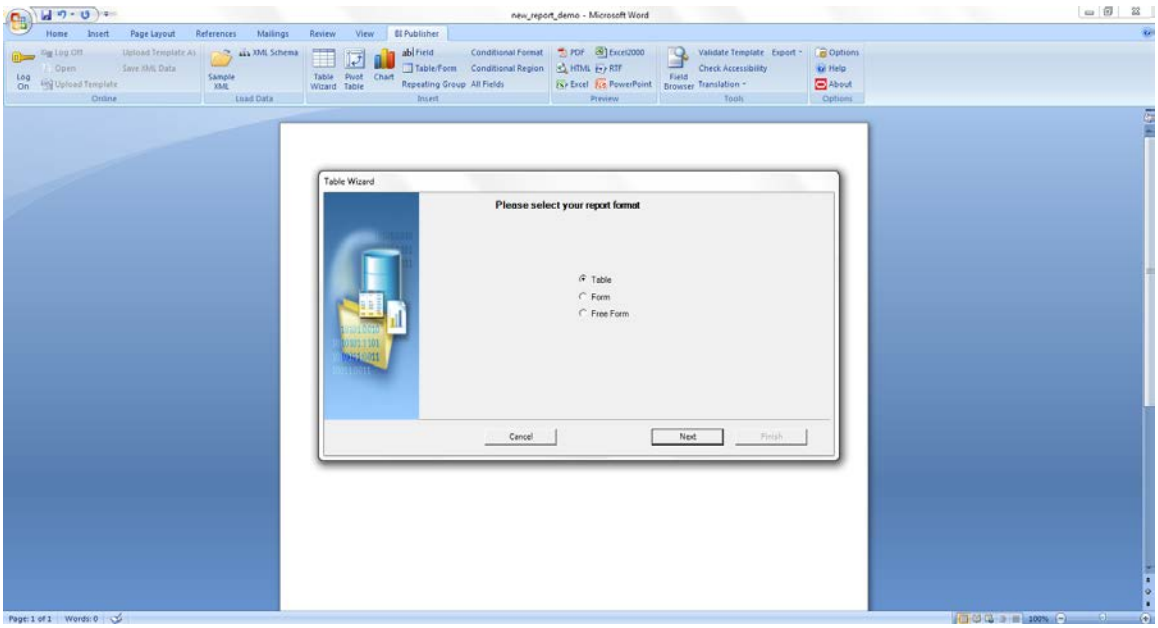


Account Number	PTC Company	PCB Branch	Account Status
20120100010018	C-0002	CB-001	ACTIVE
20120100010026	C-0002	CB-001	ACTIVE
20120100010109	C-0001	CB-002	ACTIVE
20120100010117	C-0001	CB-001	ACTIVE
20120100010125	C-0001	CB-001	ACTIVE

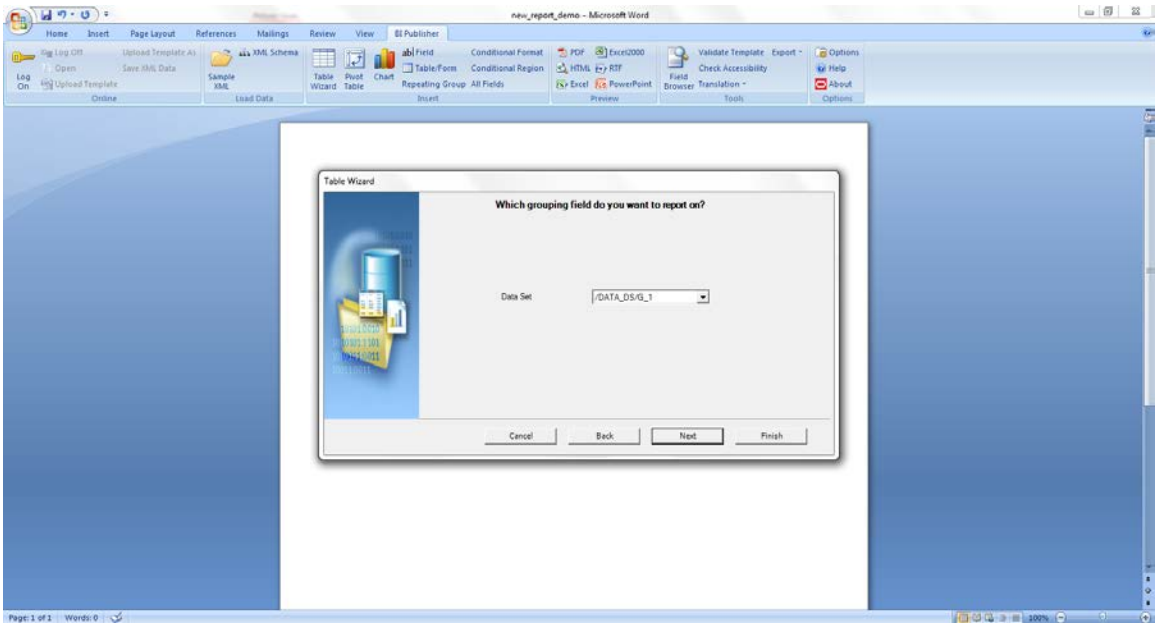
6.3 Add Dynamics to Report

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

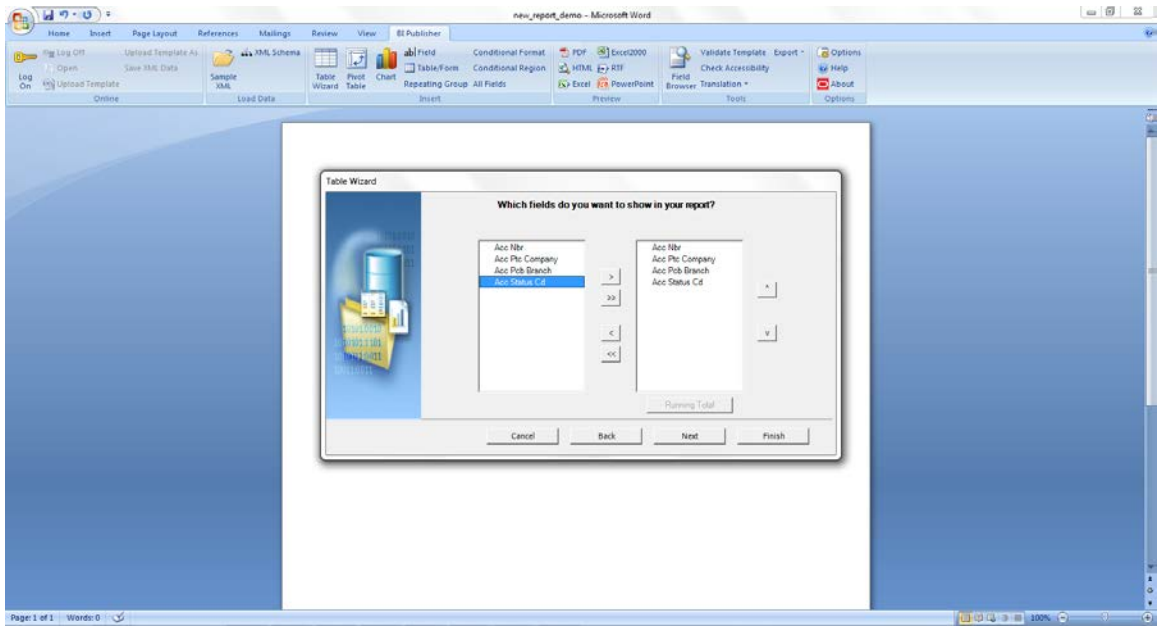
1. Select the Table Wizard from BI Publisher tab.



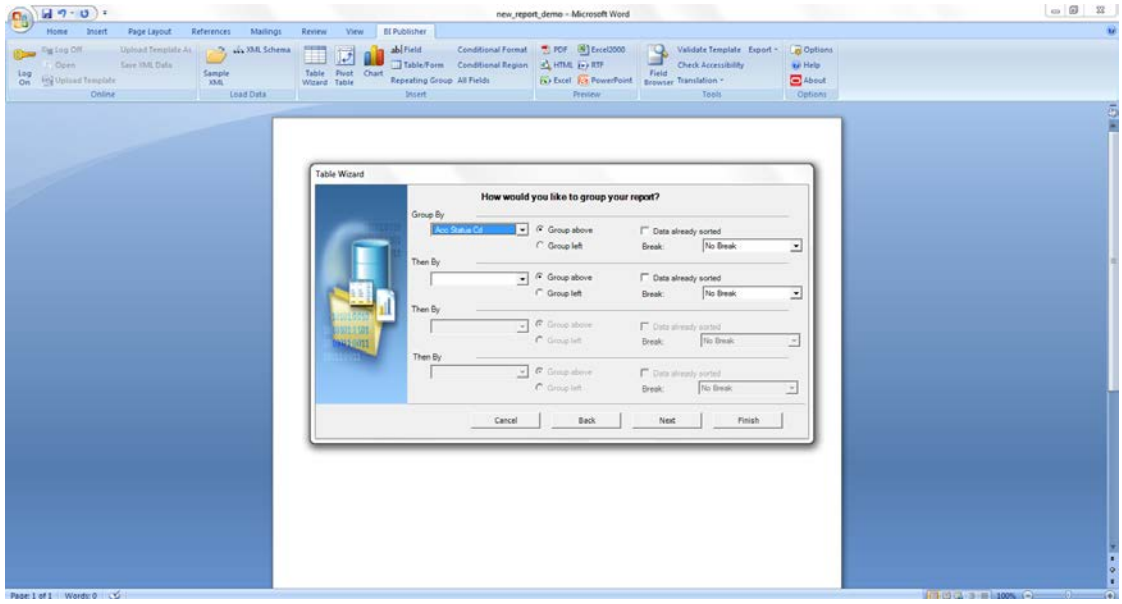
2. Select the Table and press ‘Next’.



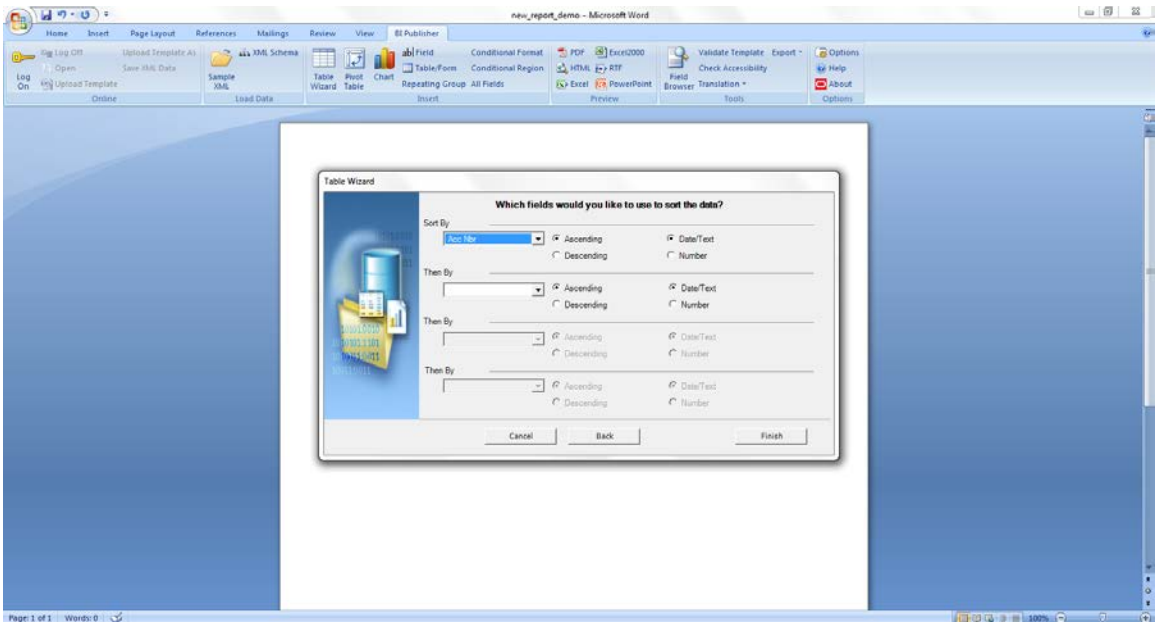
3. Use the same Data SET.



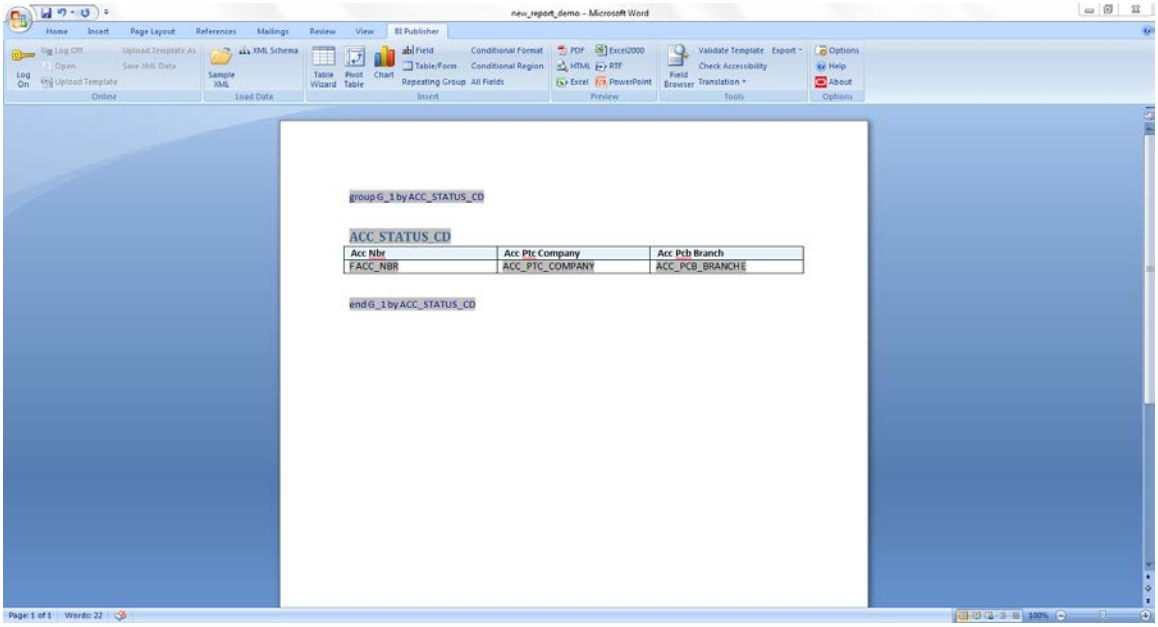
4. Select all the fields and press ‘Next’.
5. You can create the group as follows:



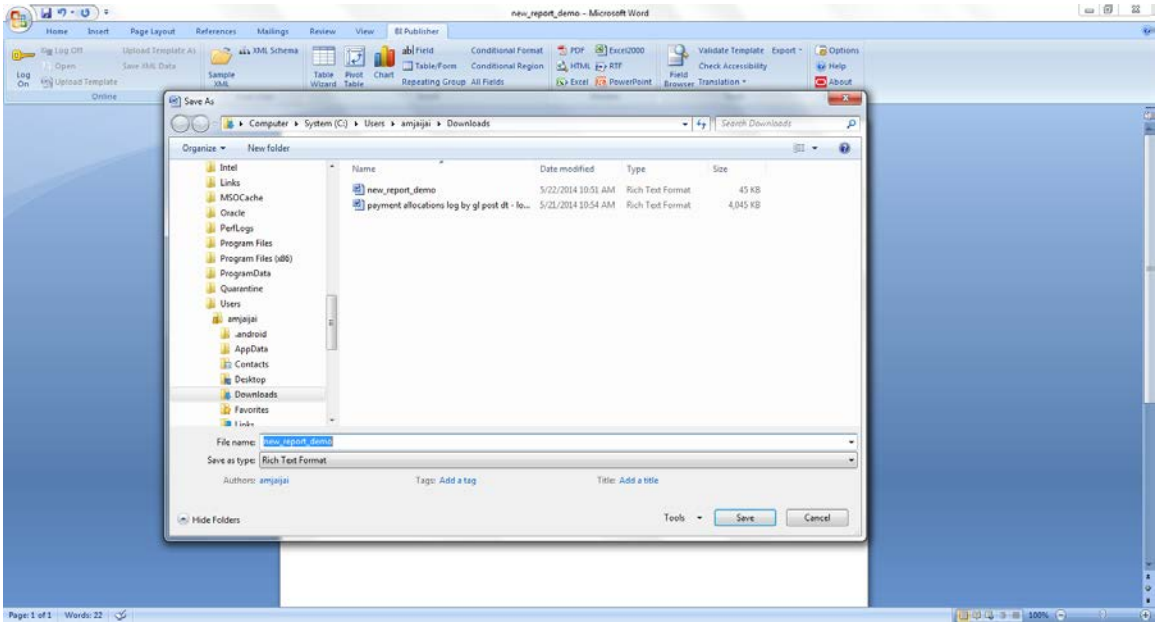
6. Click ‘Next’. You can give the sorting option as indicated.



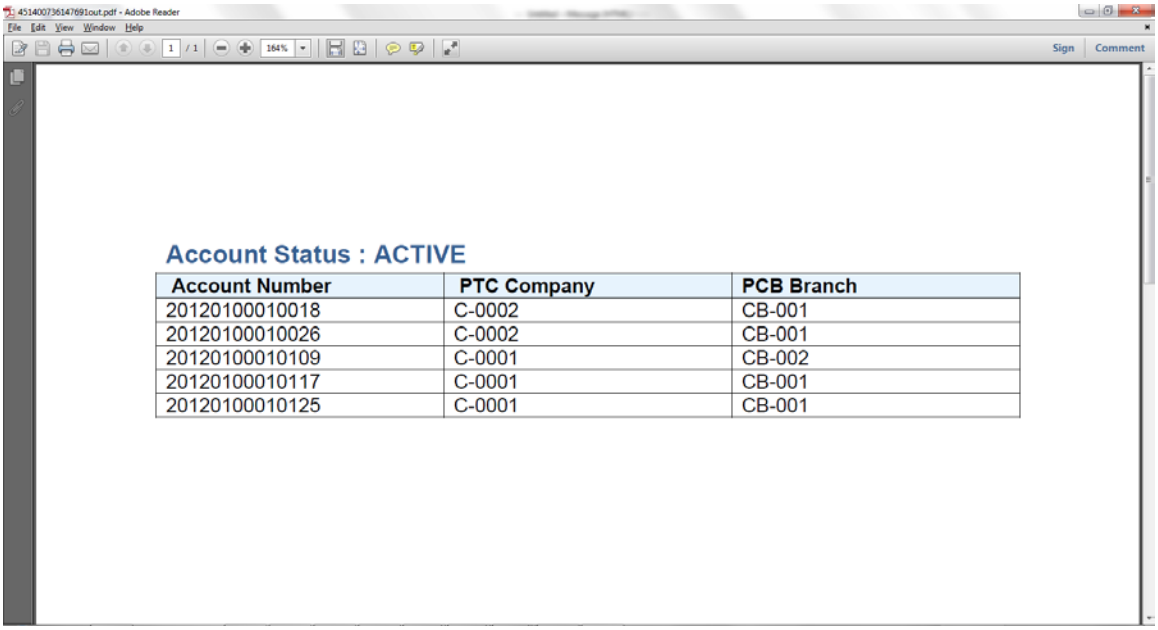
7. Click ‘Next’. The wizard creates the layout as indicated.



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



9. Preview of the report layout is as indicated.

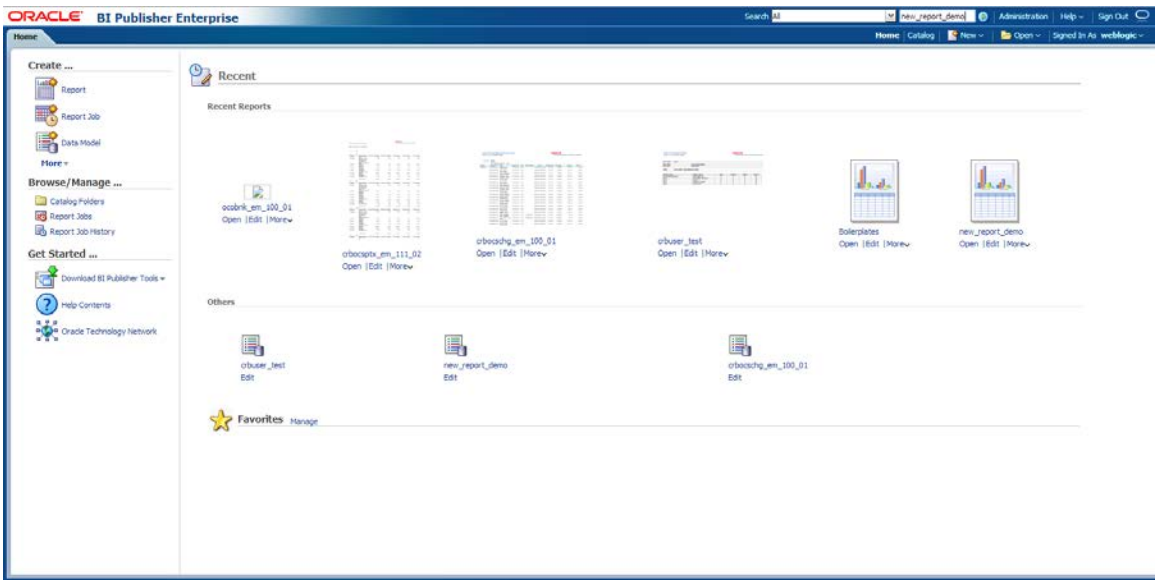


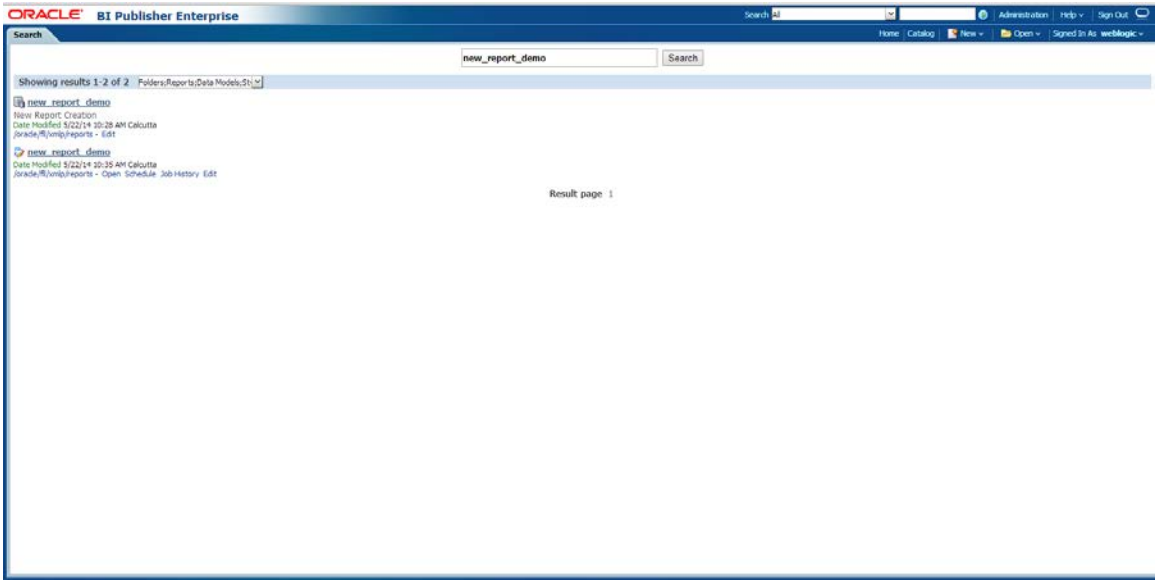
Now Layout is just created successfully and saved.

6.4 Upload Report in BIP

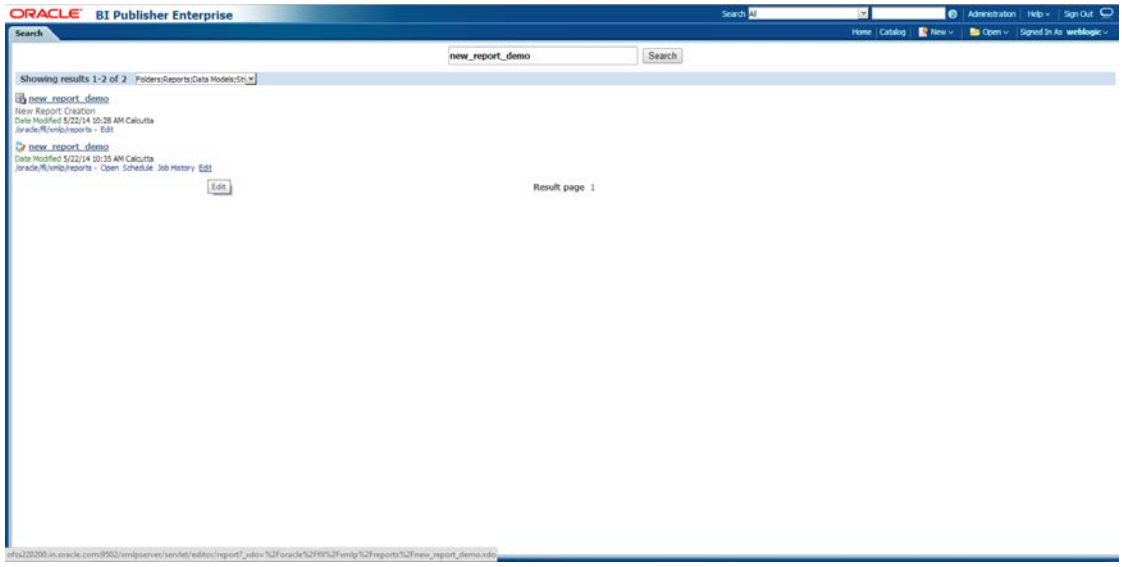
You can upload the report in BIP as indicated.

1. Navigate to BIP Console.
2. On Home page, use the search option and search for the new report created.

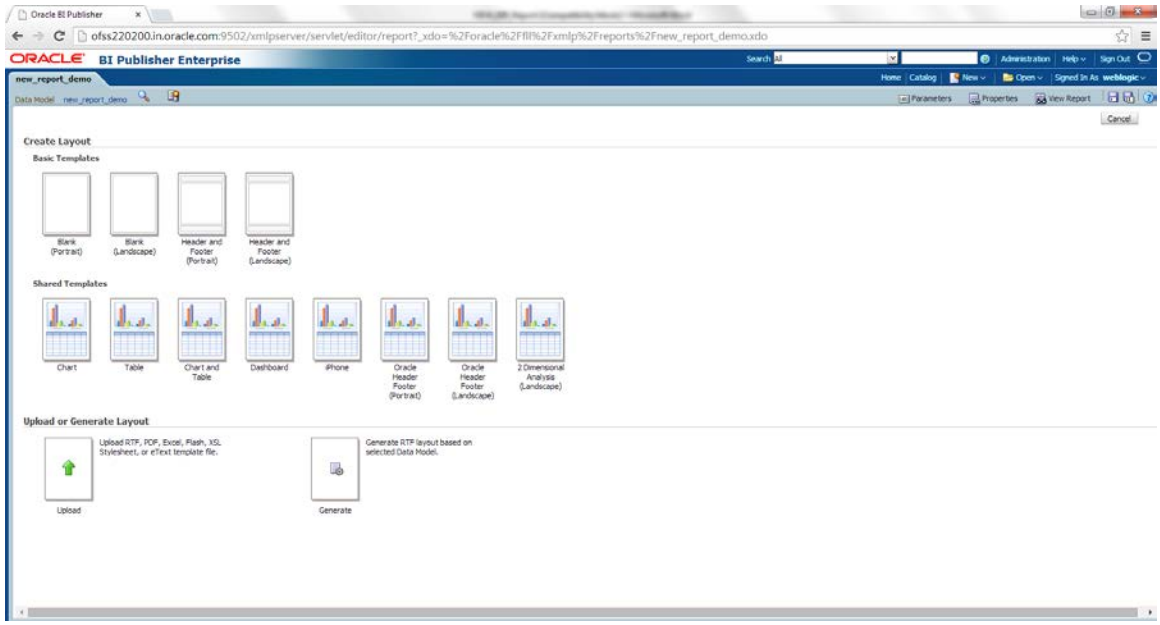




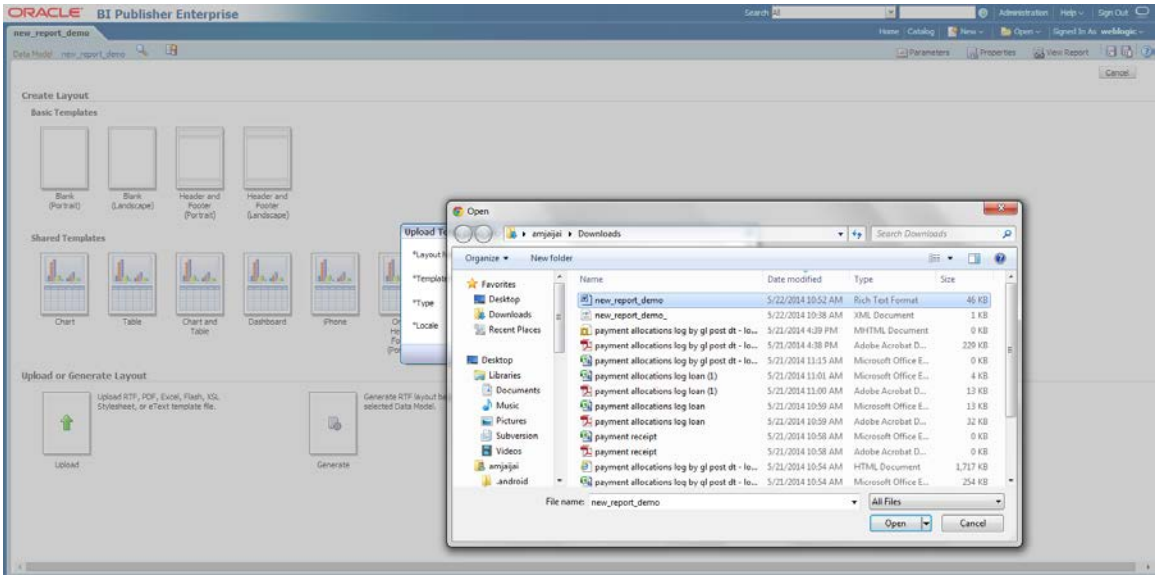
3. Click ‘Edit’ on New report layout.



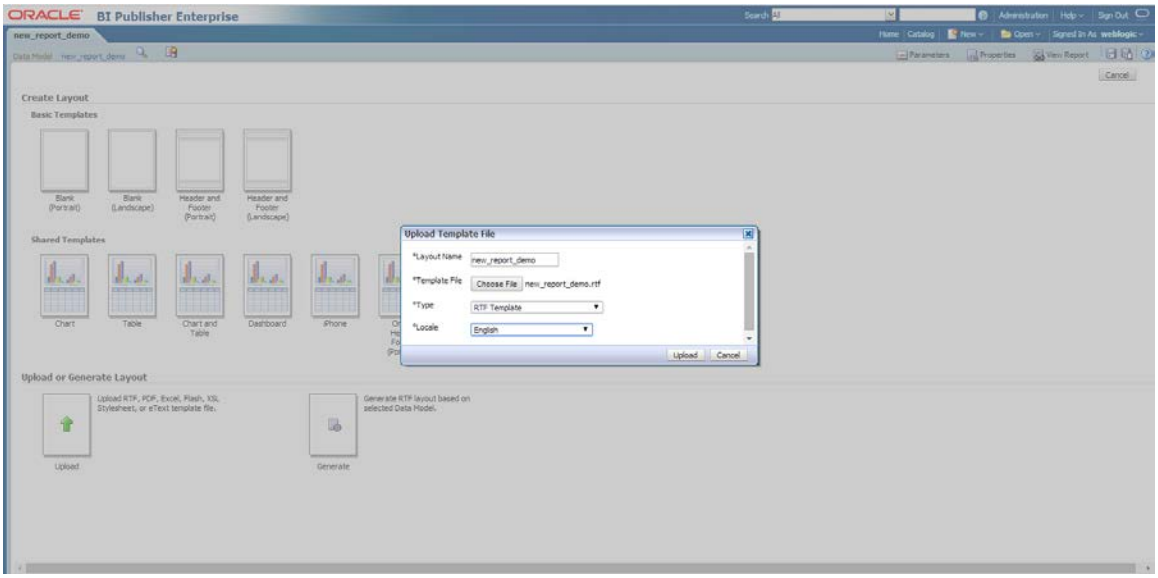
4. The following screen is displayed and allows you to upload the layout.



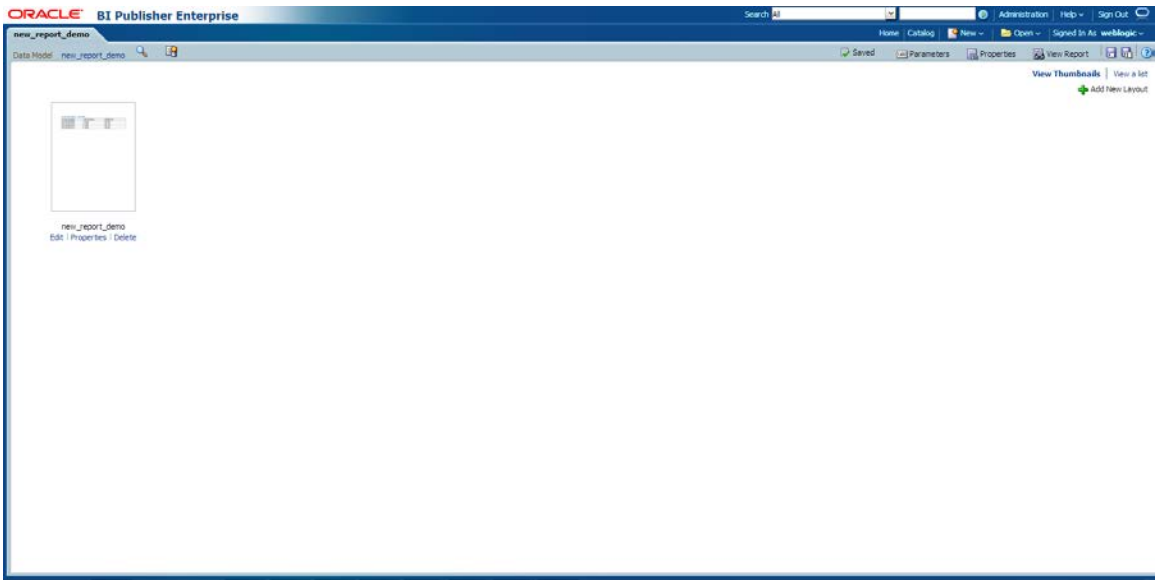
5. Click ‘Upload’.
6. Specify the same layout name (new_report_demo) and choose the file.



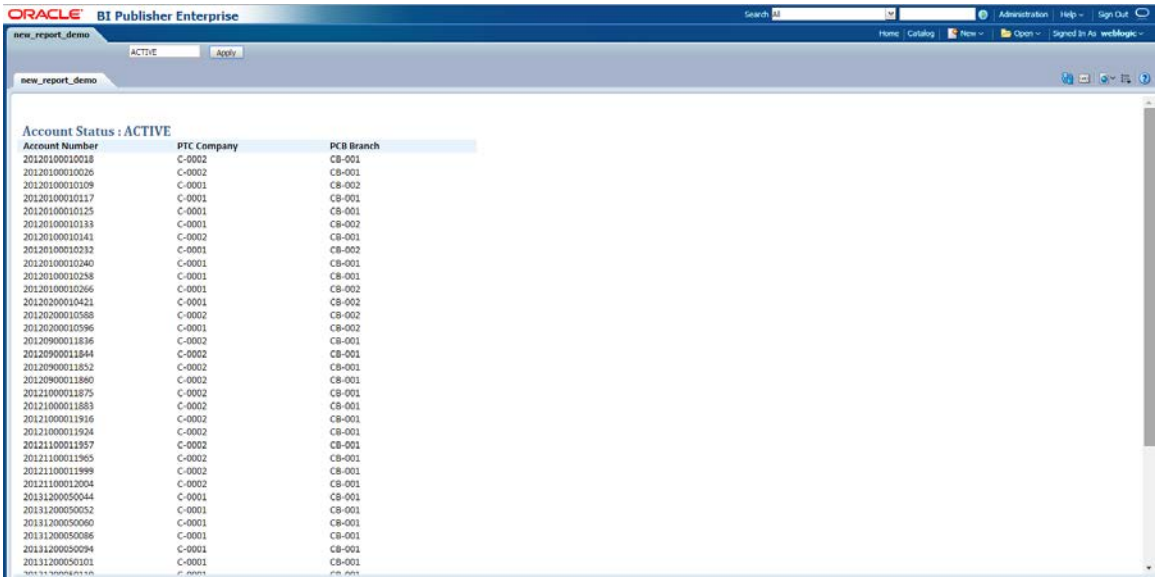
7. Select the type as ‘RTF Template’ and locale as ‘English’.



8. Click ‘Upload’.



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’.



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).

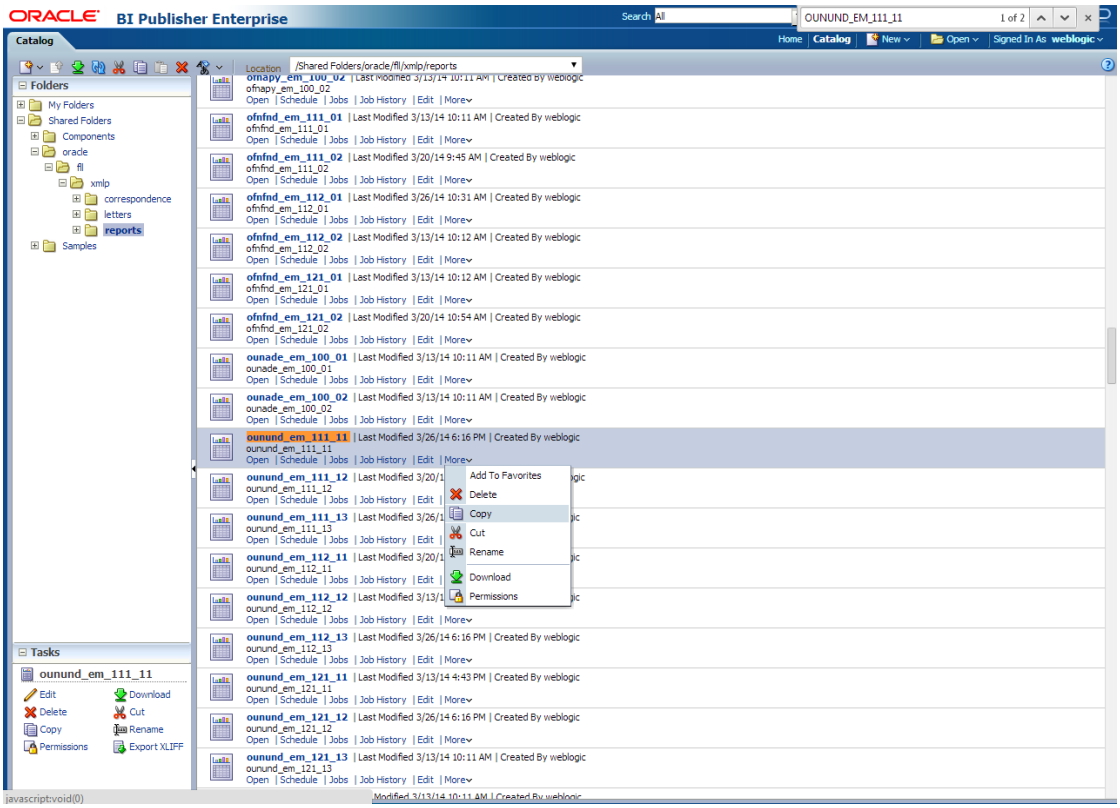
7. Customizing Existing Base BIP Reports

To customize a report, follow the steps given below:

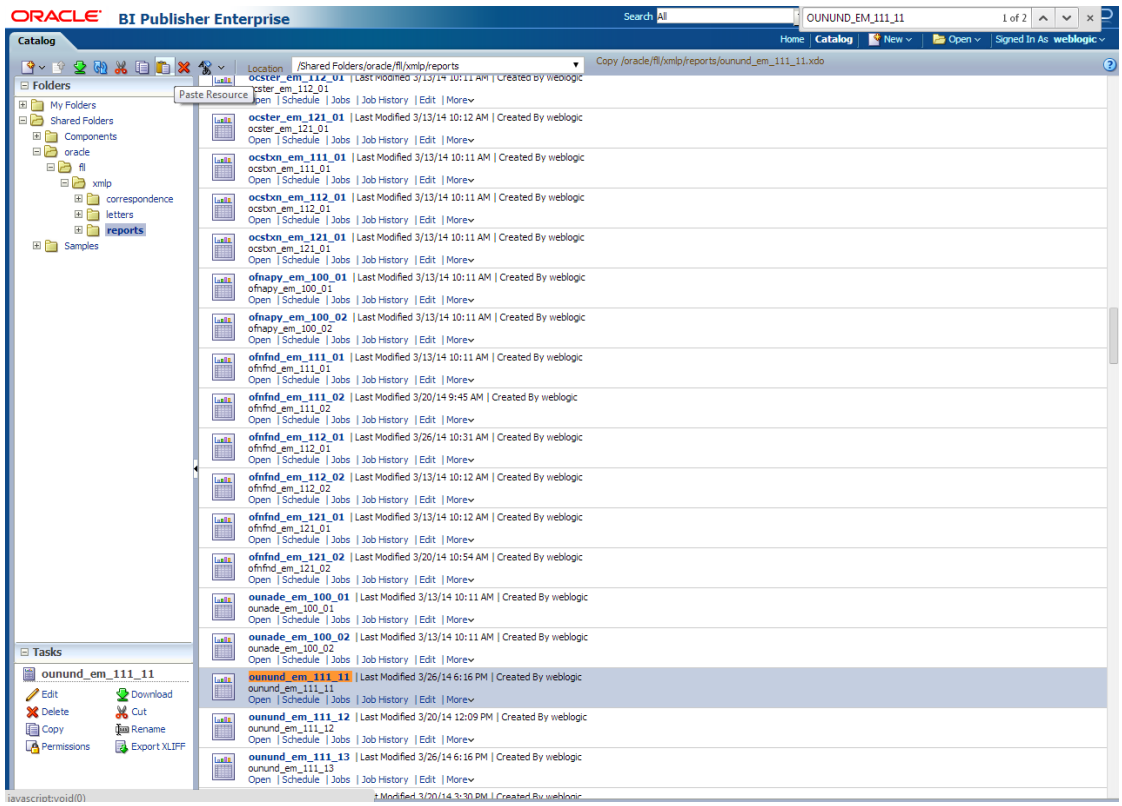
- Do not modify the OFSLL Base Report.
- Create a copy of the OFSLL Base Report, rename and modify that report. A new report also can be created. Name the report as **xyz**_<report_name>.
- Register the new report and it's parameters in OFSLL using reports setup.

Pre-Requisites

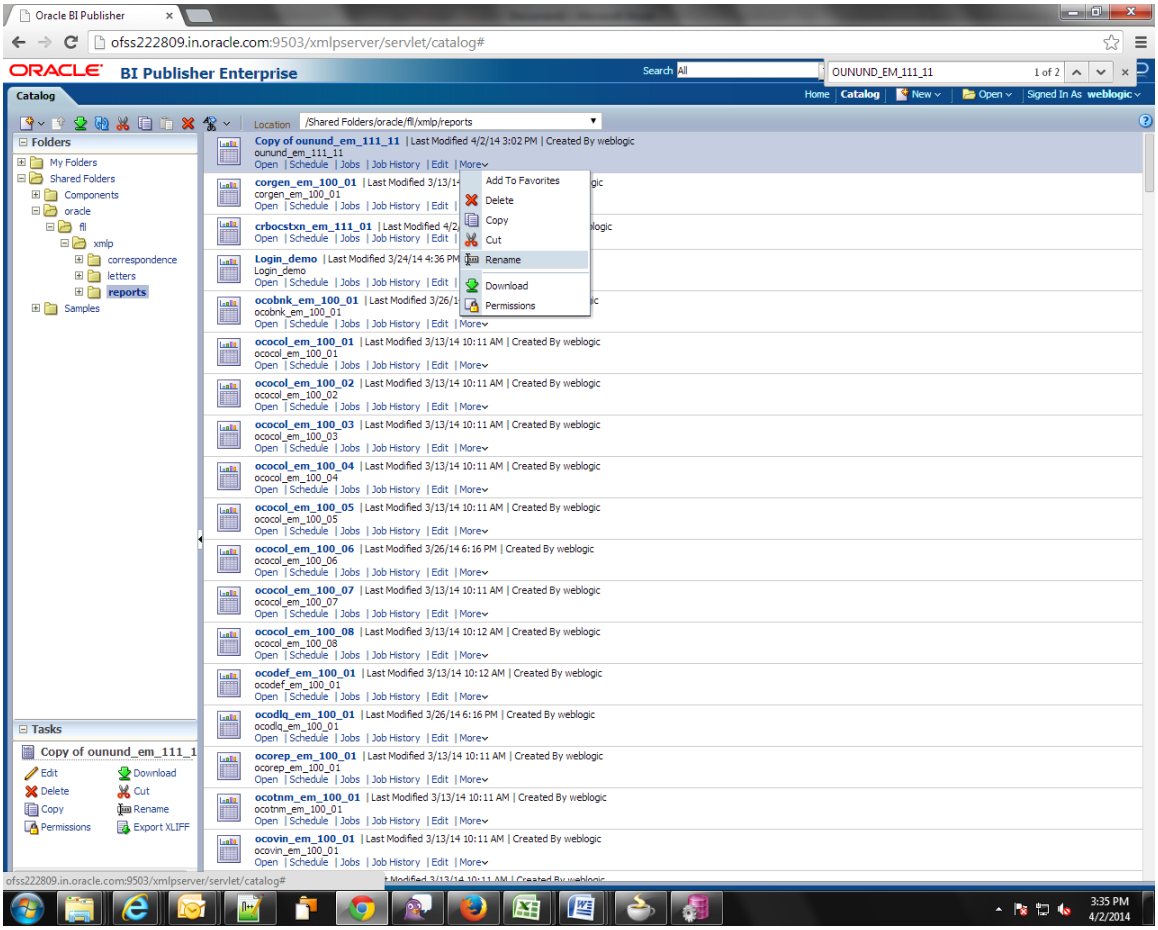
1. Please make sure that we should not change any base reports
2. The reports should be placed into the same folder structure
   - i.e. For Reports → Shared Folders/oracle/fll/xmlp/reports
   - For Letters → Shared Folders/oracle/fll/xmlp/letters
   - For Correspondences → Shared Folders/oracle/fll/xmlp/correspondence
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.



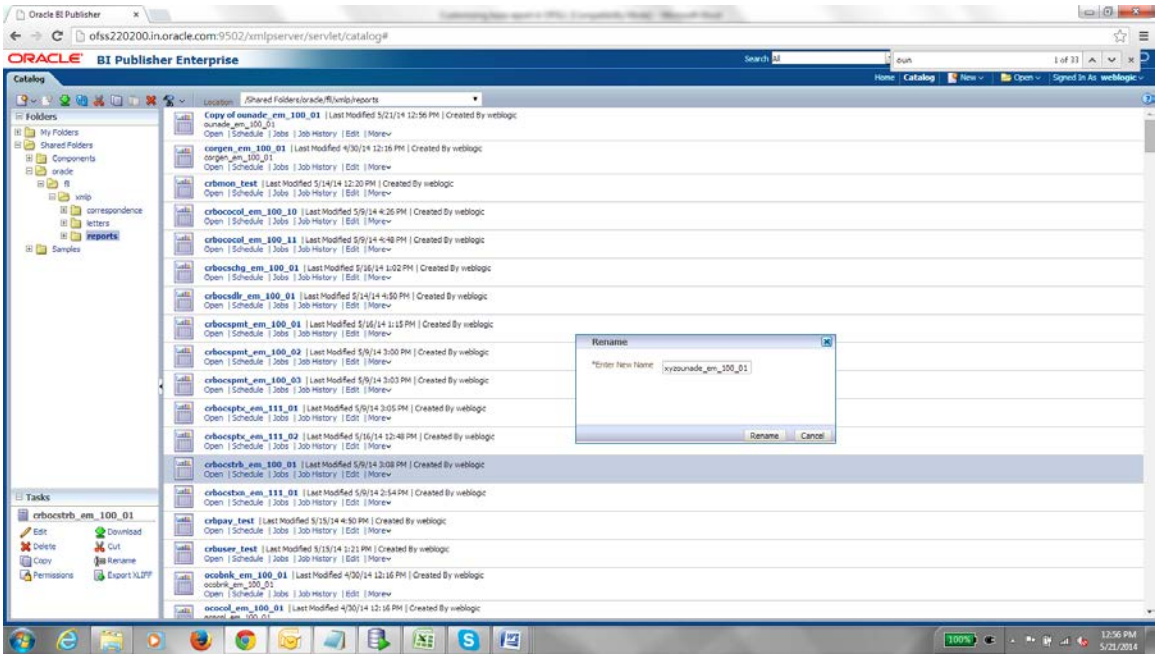
6. After pressing copy go to the folder where you want to paste the new report and press the ***Paste Resource*** button as shown in the image below



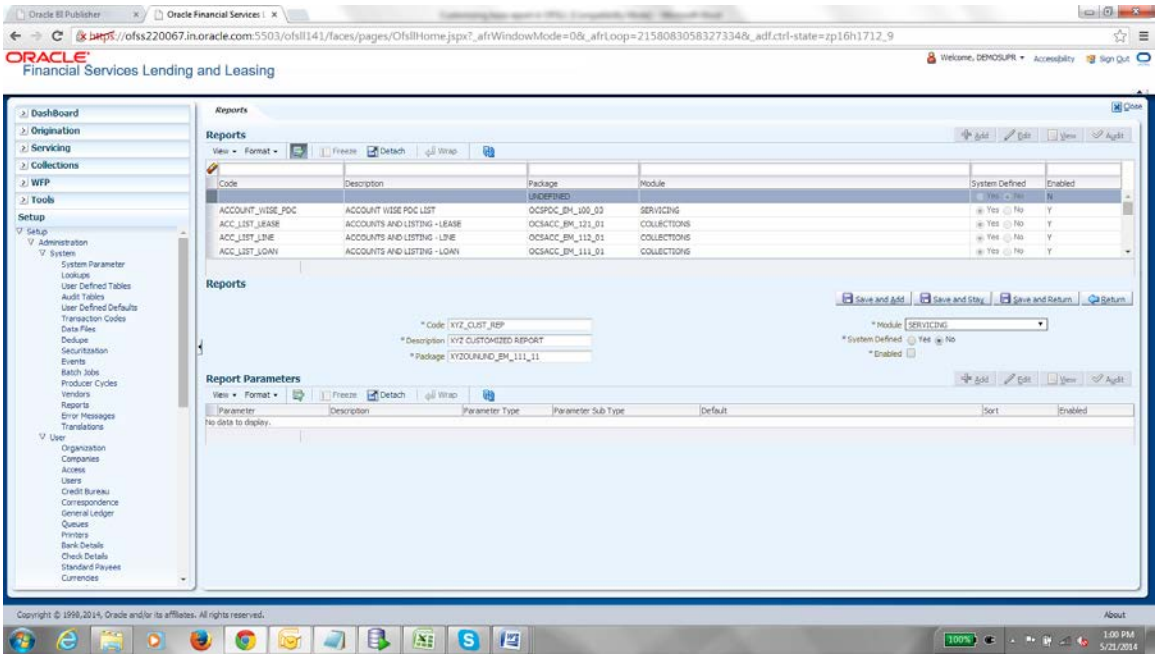
7. On pressing Paste Resource button, a new report will be created in the directory with name as Copy of ounund_em_111_11
8. Select the particular report (Copy of ounund_em_111_11) and press More→ Rename as shown in the image below



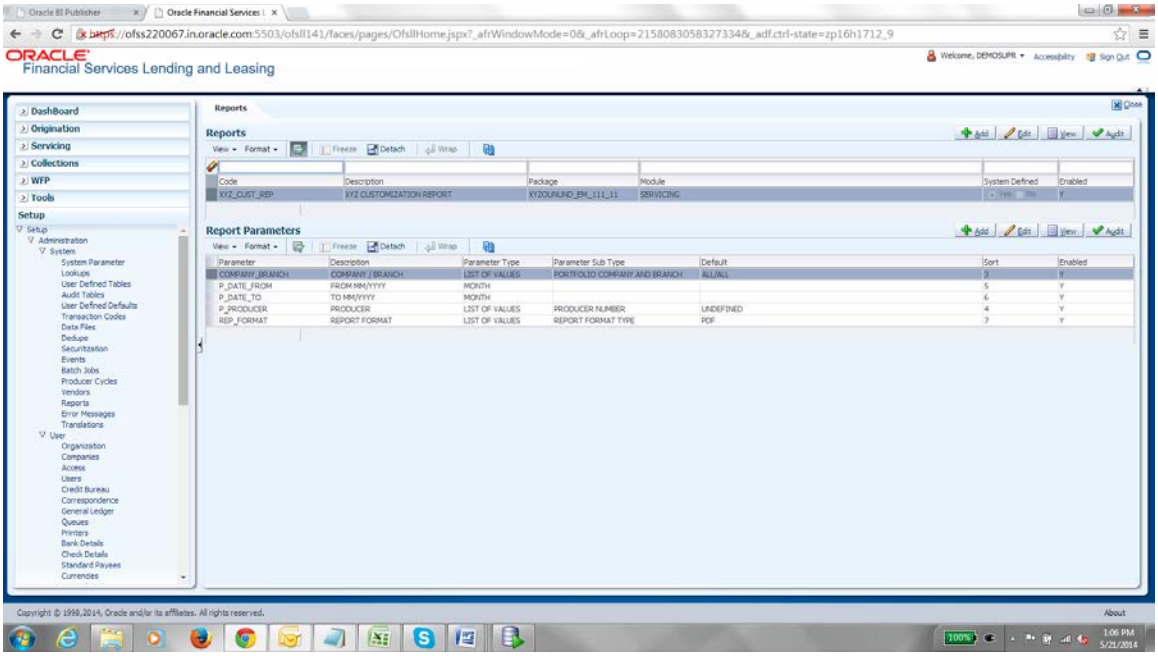
9. Enter the new name as xyz**ounund_em_111_11** and press ***Rename*** button as shown in the image below



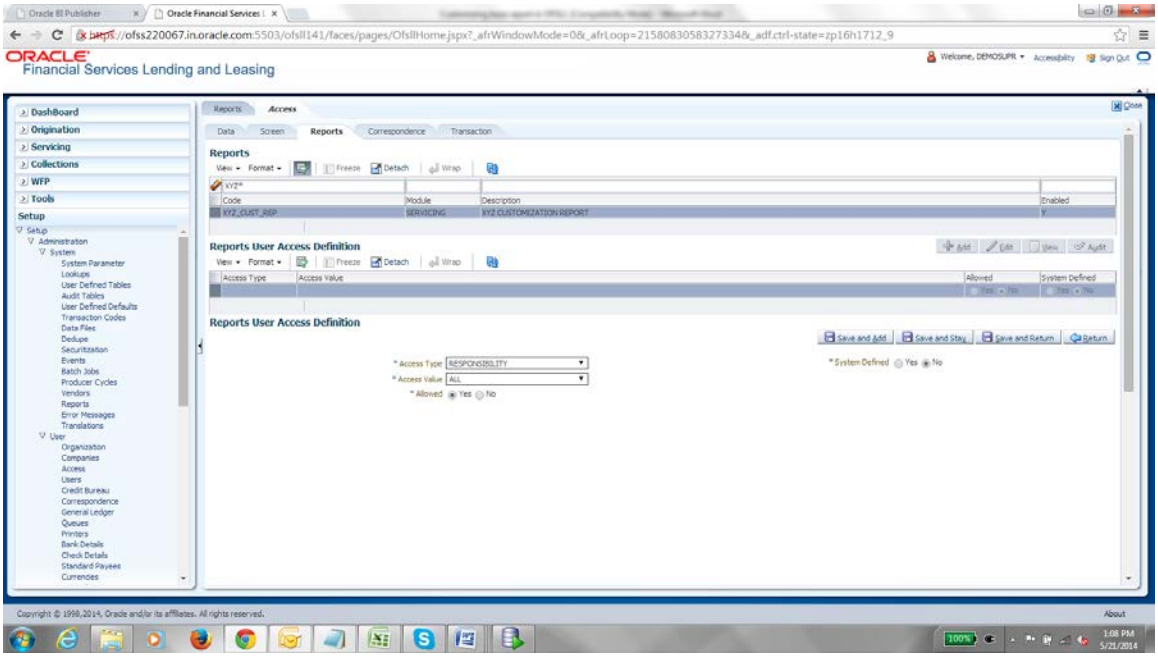
10. Do similar Copy, Paste and rename activity for the data model side also for the particular report which will be with the same name as of the report in the same directory.
11. After the new report is ready i.e. xyzounund_em_111_11. We can now do our customizations whatever is required on this new 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

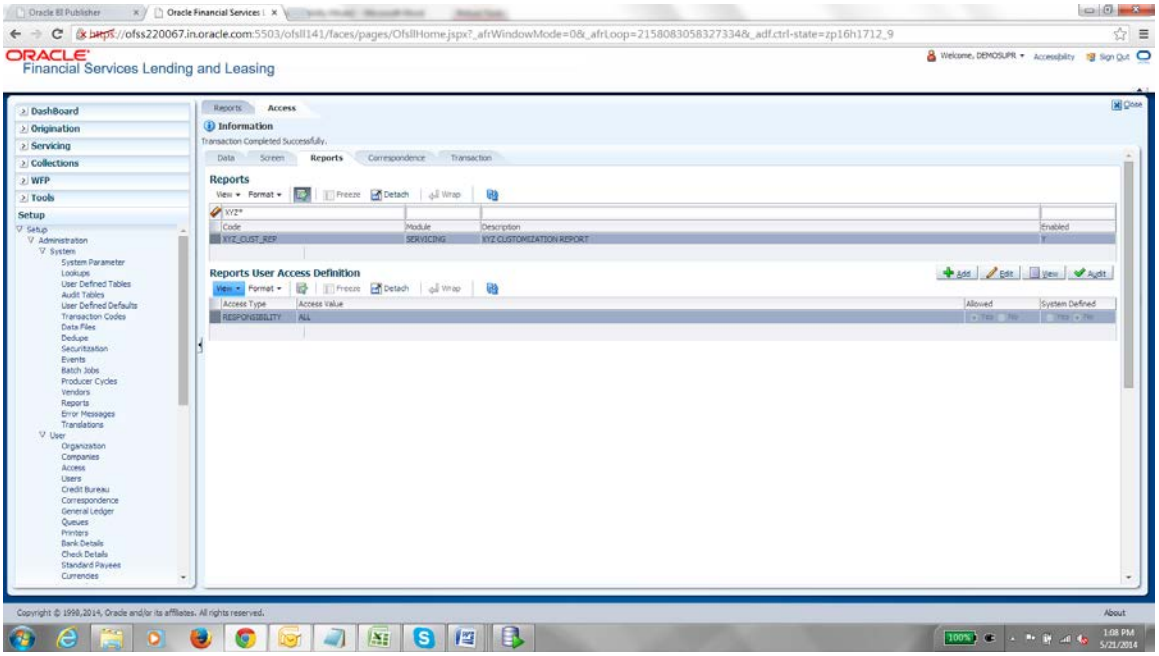


14. After Save and return please add the report parameters for the particular report as shown in the image below.

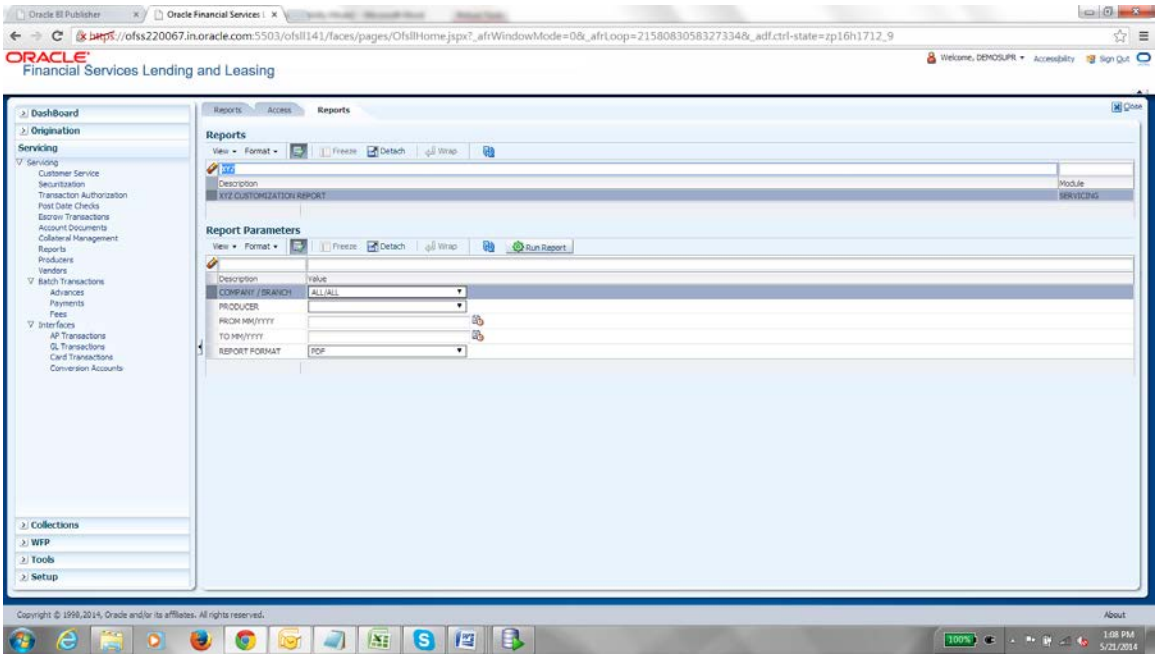


15. Now open ***User→ Access→ Reports*** Tab and select the newly created report and add the responsibility to it in Reports user access definition screen as shown in the image below





16. 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

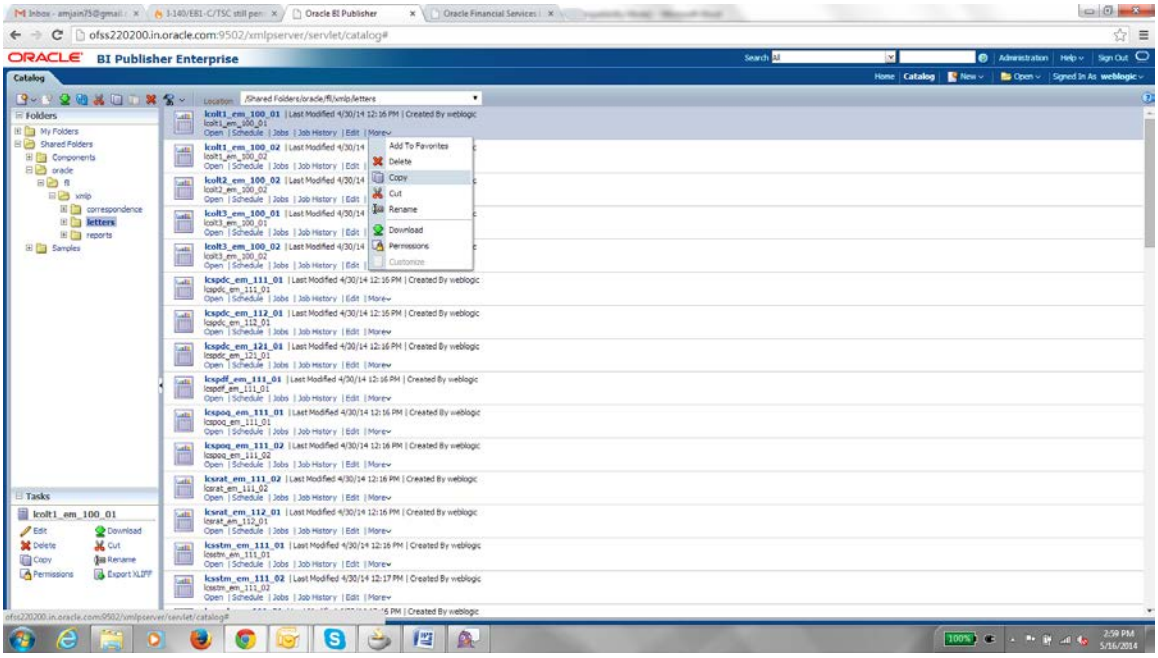


Note: There is no Impact on customization reports when a new base patch applied in the system. All customized report will not be override, removed or modified.

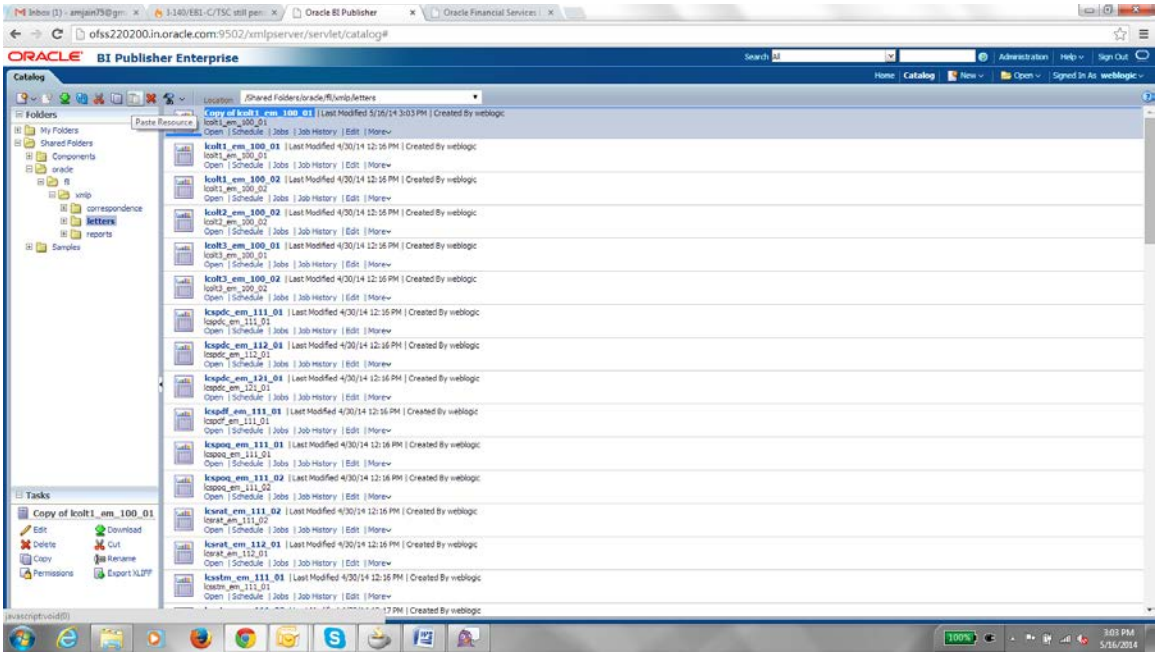
8. Customizing Existing Base BIP Letters

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/xmlp/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

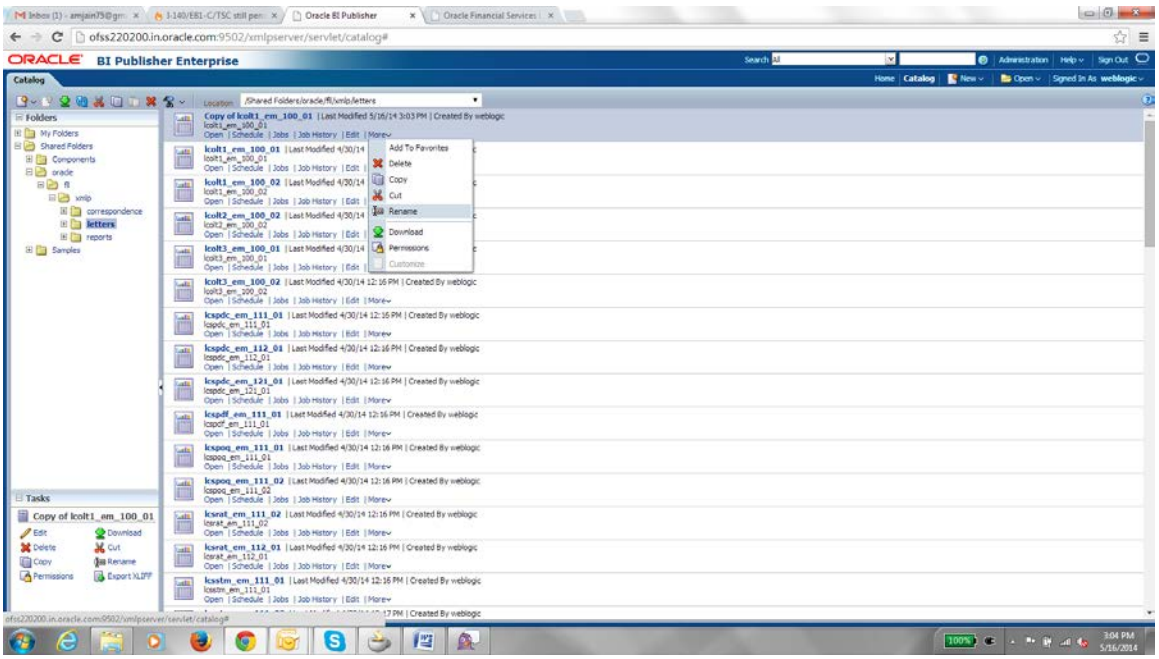


6. After pressing copy go to the folder where you want to paste the new report and press the ***Paste Resource*** button as shown in the image below.

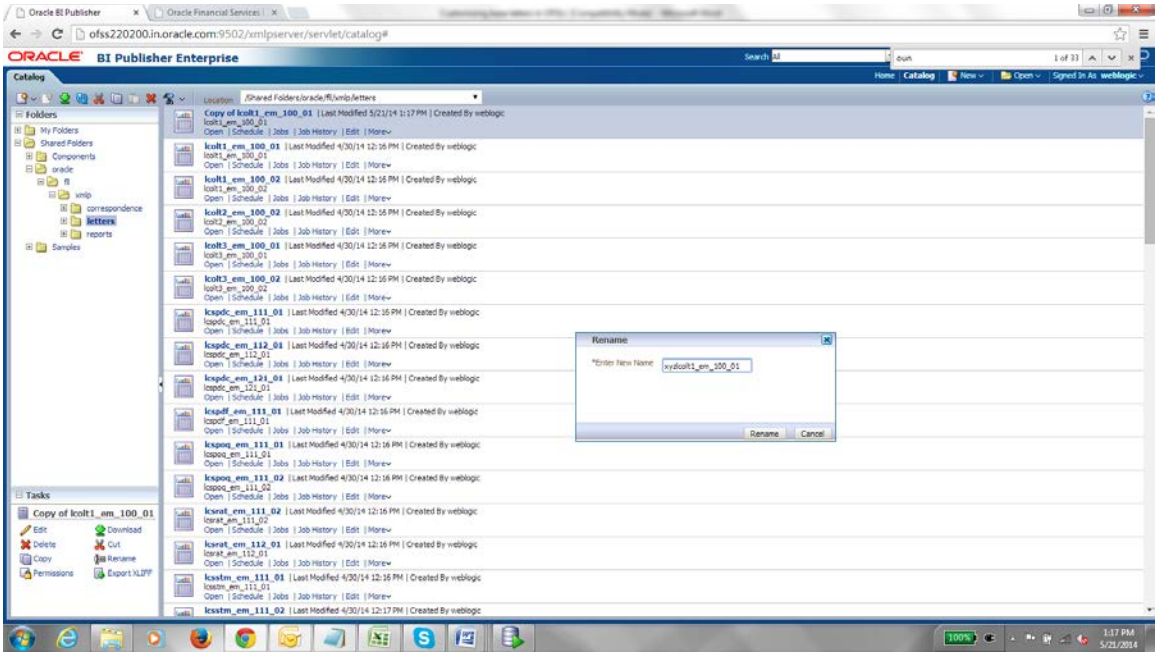


7. On pressing ***Paste Resource*** button, a new report will be created in the directory with name as ***Copy of lcolt1_em_100_01***

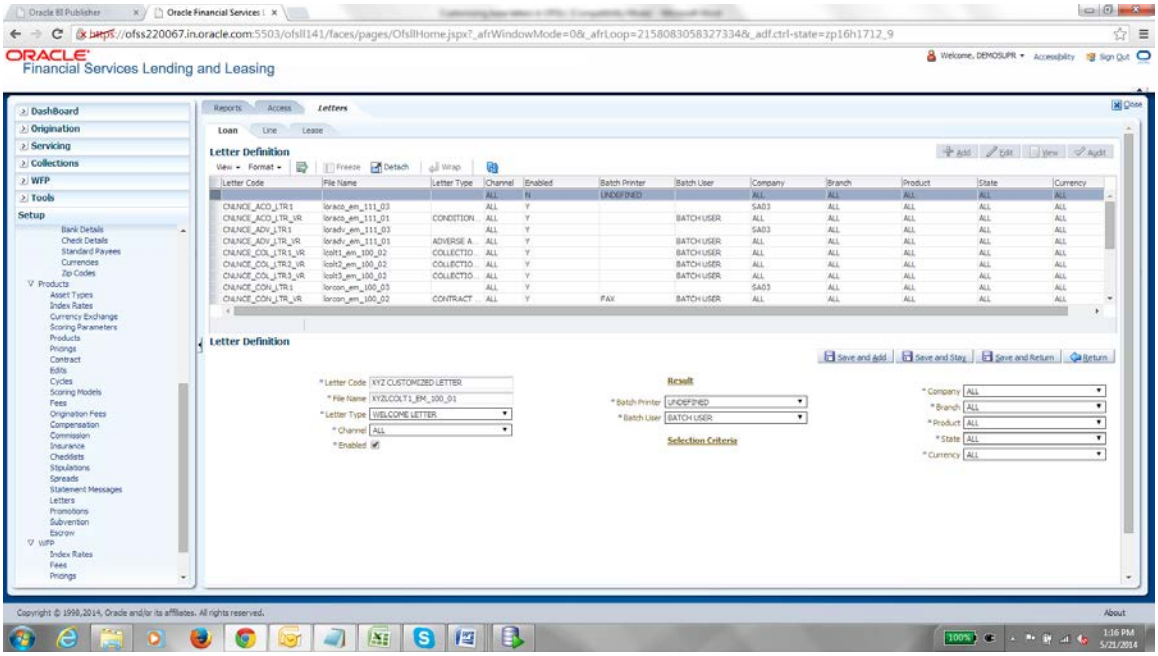
8. Select the particular report (***Copy of lcolt1_em_100_01)*** and press ***More→Rename*** as shown in the image below



9. Enter the new name as xyz**lcolt1_em_100_01** and press ***Rename*** button as shown in the image below.



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. xyzlcolt1_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 xyzlcolt1_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.

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.

9. 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

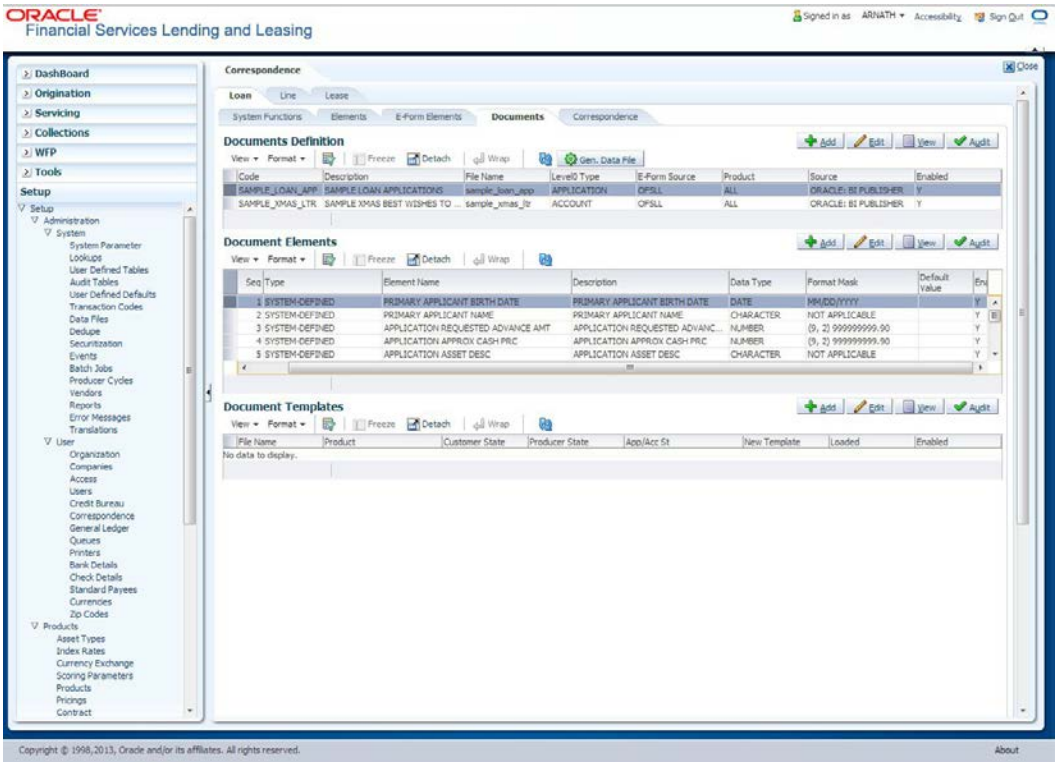
Oracle Financial Services Software assumes that the user is familiar with Word and the Merge Document command. If the user is creating e-form documents with XFDF, then Oracle Financial Services Software assumes that person is familiar with Adobe forms.

To create a 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:

Field:	Do this:
Code	Specify the document code to define the name for the new docu- ment.
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.

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

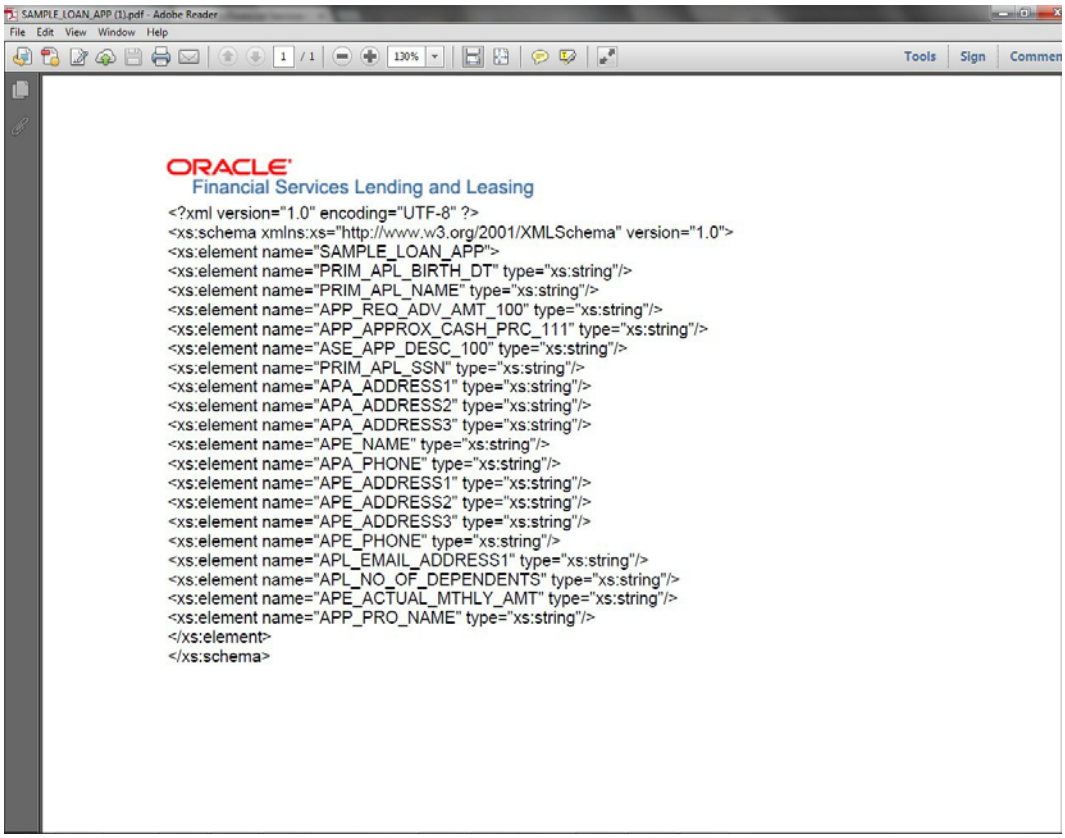


Brief description of the fields are given below:

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.
Element Name	Select the element name from the drop-down list.

Field:	Do this:
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 th 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.



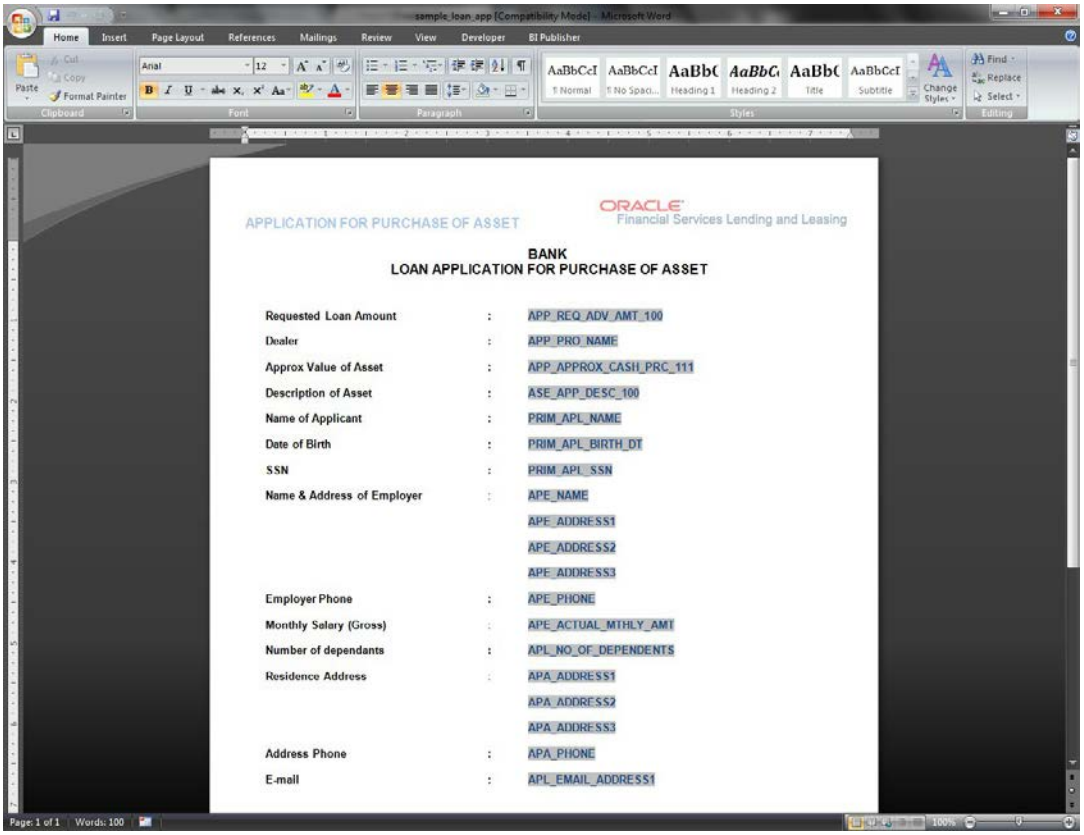
5. Copy and save the content in the pdf file as an xml file. The saved xml file should have the same name as entered in the Code column of Document Definition section. *For Example:* SAMPLE_LOAN_APP.xml.

6. 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.

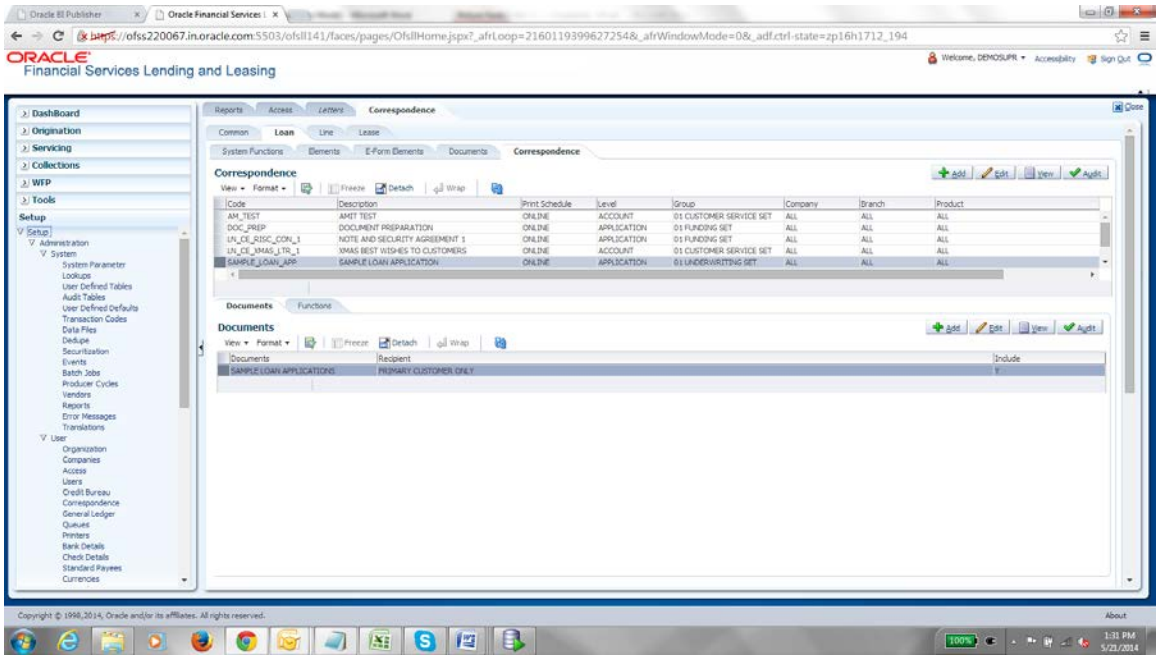
6. In BI Publisher Tab in MS Word, click on Sample XML and import the saved xml file. *For Example:* SAMPLE_LOAN_APP.xml.
7. Create the template by inserting required elements tag.



8. 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.

9. 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".
10. After the data model creation, launch the correspondence screen and click Correspondence tab.
11. You can setup a correspondence with the created doc.

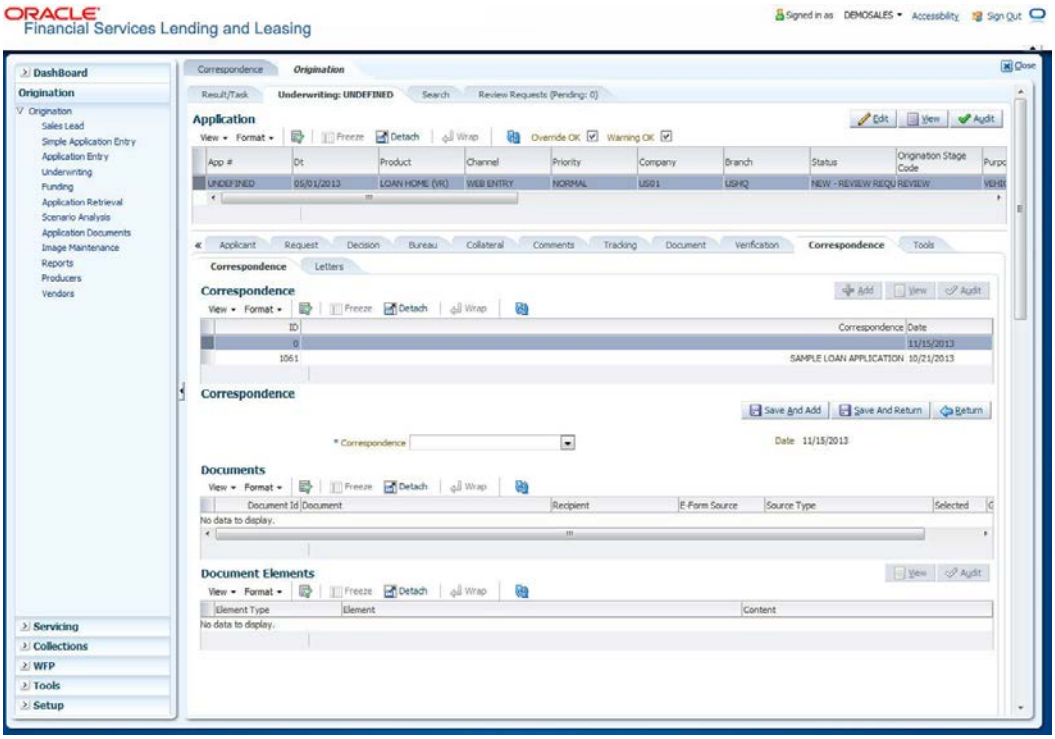

Oracle BI Publisher
Oracle Financial Services
https://ofss220067.in.oracle.com:5503/ofsll141/faces/pages/OfsllHome.jspx?_afrLoop=216011939962725 4&_afrWindowMode=0&_adf.ctrl-state=zp16h1712_194
ORACLE
Financial Services Lending and Leasing
Welcome, DEMOSUPR
Accessibility
Sign Out
DashBoard
Origination
Servicing
Collections
WFP
Tools
Setup
Setup
Administration
System
System Parameter
Lookups
User Defined Tables
Audit Tables
User Defined Defaults
Transaction Codes
Data Files
Dedupe
Securitization
Events
Batch Jobs
Producer Cycles
Vendors
Reports
Error Messages
Translations
User
Organization
Companies
Access
Users
Credit Bureau
Correspondence
General Ledger
Queues
Printers
Bank Details
Check Details
Standard Payees
Currencies
Reports
Access
Letters
Correspondence
Common
Loan
Line
Lease
System Functions
Elements
E-Form Elements
Documents
Correspondence
Correspondence
View
Format
Freeze
Detach
Wrap
Add
Edit
View
Audit
Code
Description
Print Schedule
Level
Group
Company
Branch
Product
AM_TEST
AMT TEST
ONLINE
ACCOUNT
01 CUSTOMER SERVICE SET
ALL
ALL
ALL
DOC_PREP
DOCUMENT PREPARATION
ONLINE
APPLICATION
01 FUNDING SET
ALL
ALL
ALL
LN_CE_RISC_CON_1
NOTE AND SECURITY AGREEMENT 1
ONLINE
APPLICATION
01 FUNDING SET
ALL
ALL
ALL
LN_CE_XMAS_LTR_1
XMAS BEST WISHES TO CUSTOMERS
ONLINE
ACCOUNT
01 CUSTOMER SERVICE SET
ALL
ALL
ALL
SAMPLE_LOAN_APP
SAMPLE LOAN APPLICATION
ONLINE
APPLICATION
01 UNDERWRITING SET
ALL
ALL
ALL
Documents
Functions
Documents
View
Format
Freeze
Detach
Wrap
Add
Edit
View
Audit
Documents
Recipient
Include
SAMPLE LOAN APPLICATIONS
PRIMARY CUSTOMER ONLY
Y
Copyright © 1998,2014, Oracle and/or its affiliates. All rights reserved.
About
100%
1:31 PM
5/21/2014

10. Generating Correspondence

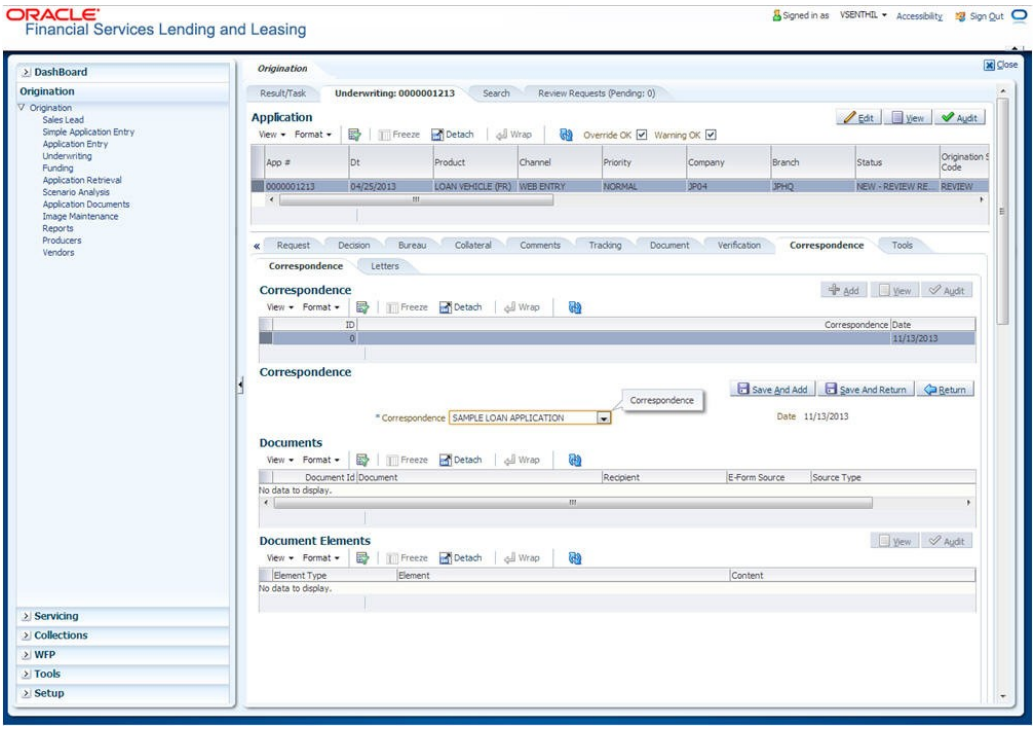
You can generate a correspondence once the respective correspondence is created in the database.

To generate a Correspondence

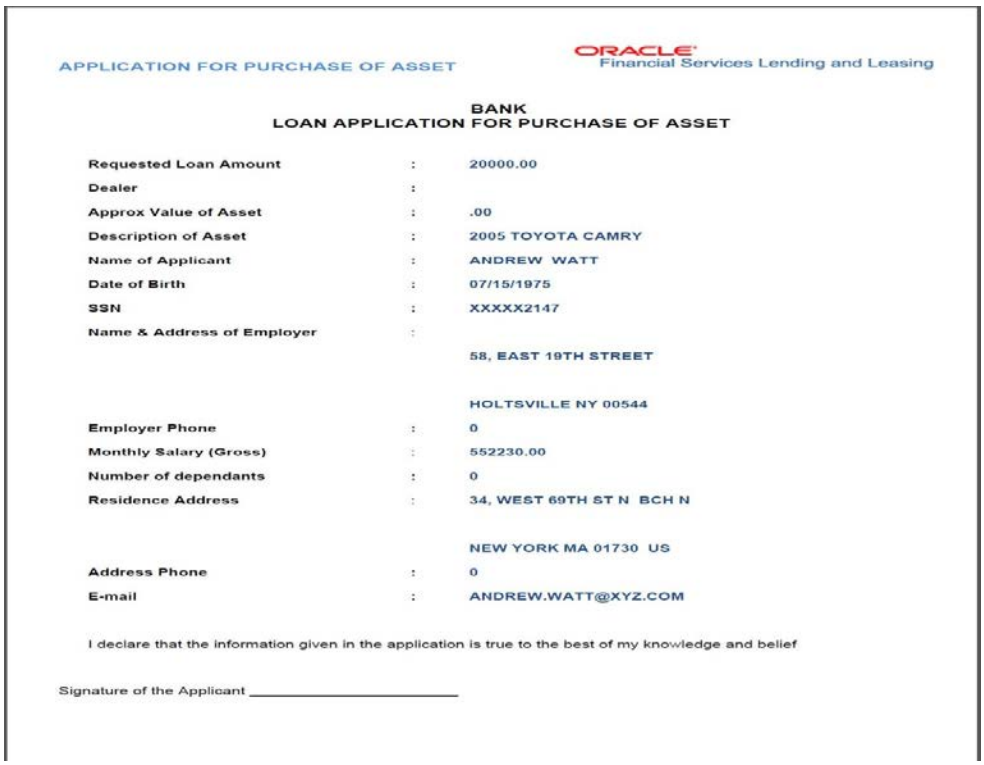
1. On the Oracle Financial Services Lending and Leasing home page, click **Origination**→ **Origination**→ **Underwriting**
2. Open the application for which the correspondence should be generated.
3. Click Corresp**ondence** tab. In the **Correspondence** section, click on **Add.**



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



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



APPLICATION FOR PURCHASE OF ASSET

ORACLE
Financial Services Lending and Leasing

BANK
LOAN APPLICATION FOR PURCHASE OF ASSET

Requested Loan Amount	:	20000.00
Dealer	:	
Approx Value of Asset	:	.00
Description of Asset	:	2005 TOYOTA CAMRY
Name of Applicant	:	ANDREW WATT
Date of Birth	:	07/15/1975
SSN	:	XXXXX2147
Name & Address of Employer	:	
		58, EAST 19TH STREET
		HOLTSVILLE NY 00544
Employer Phone	:	0
Monthly Salary (Gross)	:	552230.00
Number of dependants	:	0
Residence Address	:	34, WEST 69TH ST N BCH N
		NEW YORK MA 01730 US
Address Phone	:	0
E-mail	:	ANDREW.WATT@XYZ.COM

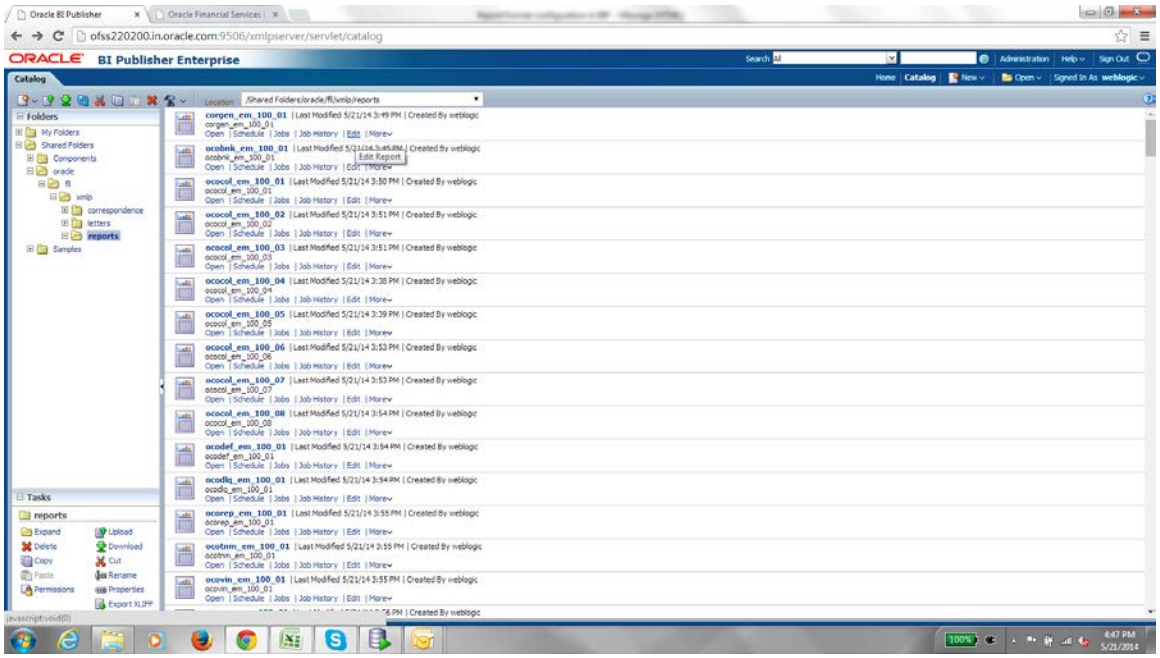
I declare that the information given in the application is true to the best of my knowledge and belief

Signature of the Applicant _____

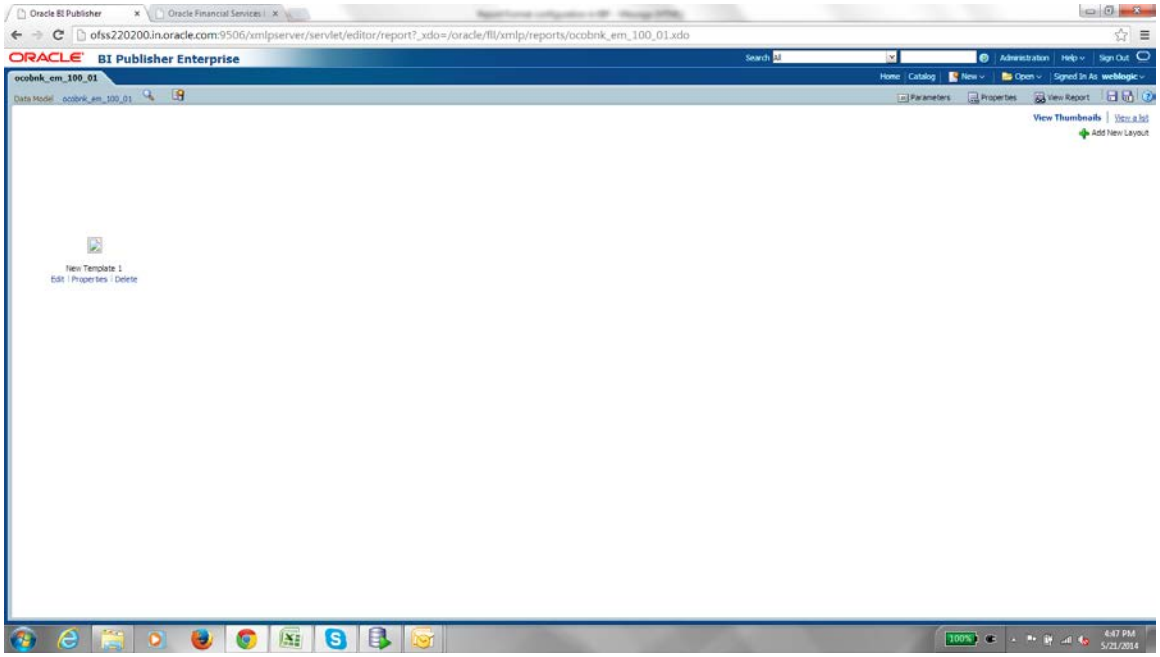
Note: There is no Impact on customization letters when a new base patch applied in the system. All customized letters will not be override , removed or modified.

11. Setting up the output Format For BIP Reports

1. Go to catalog and select the desired report (XDO) , Press the edit.

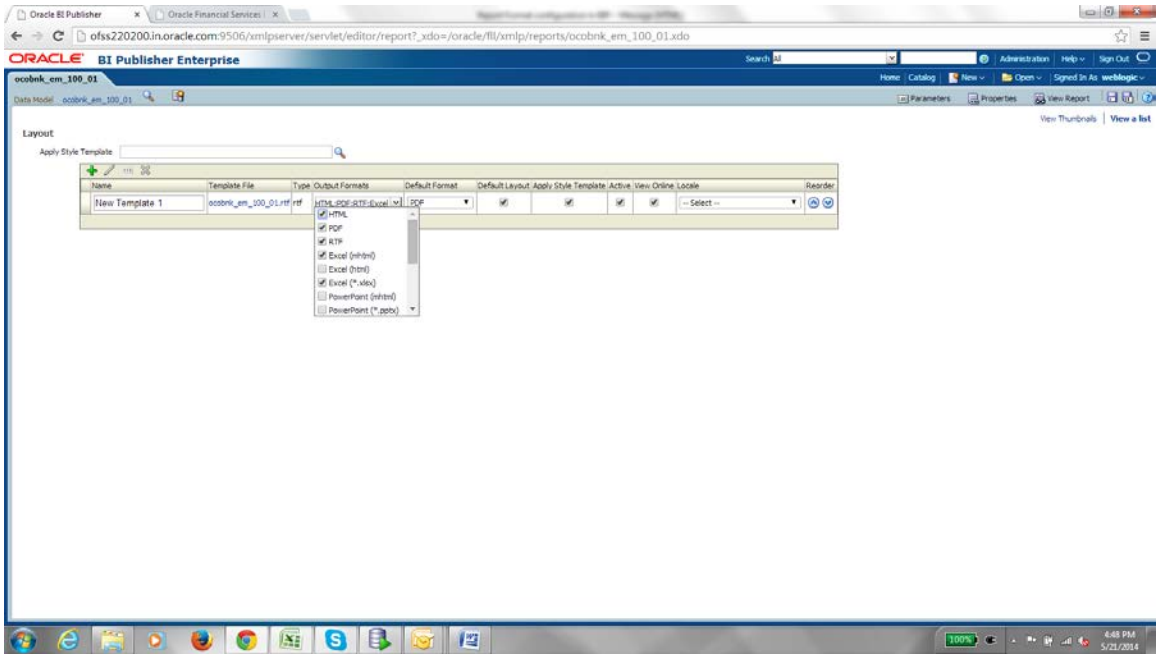


2. On Edit it will open the following page, On right corner there is “View a List “ link , click on it.

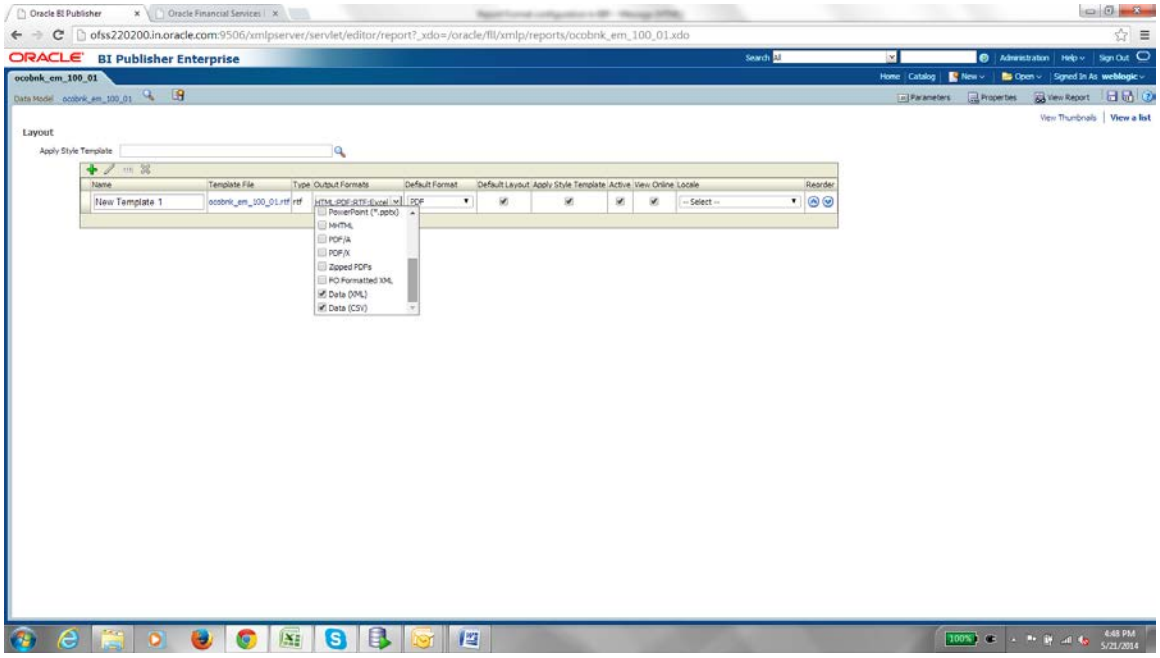


3. It will open the Layout format, click on the “Output Formats” dropdown. Select the following formats

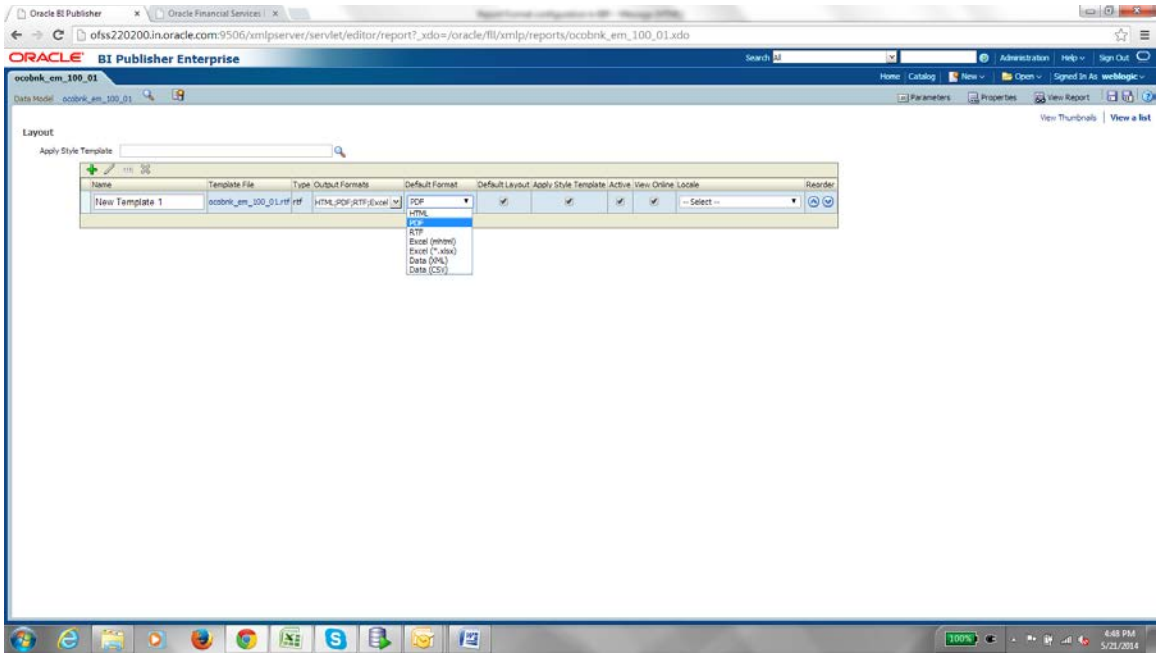
HTML,PDF ,RTF,EXCEL (.xlsx),



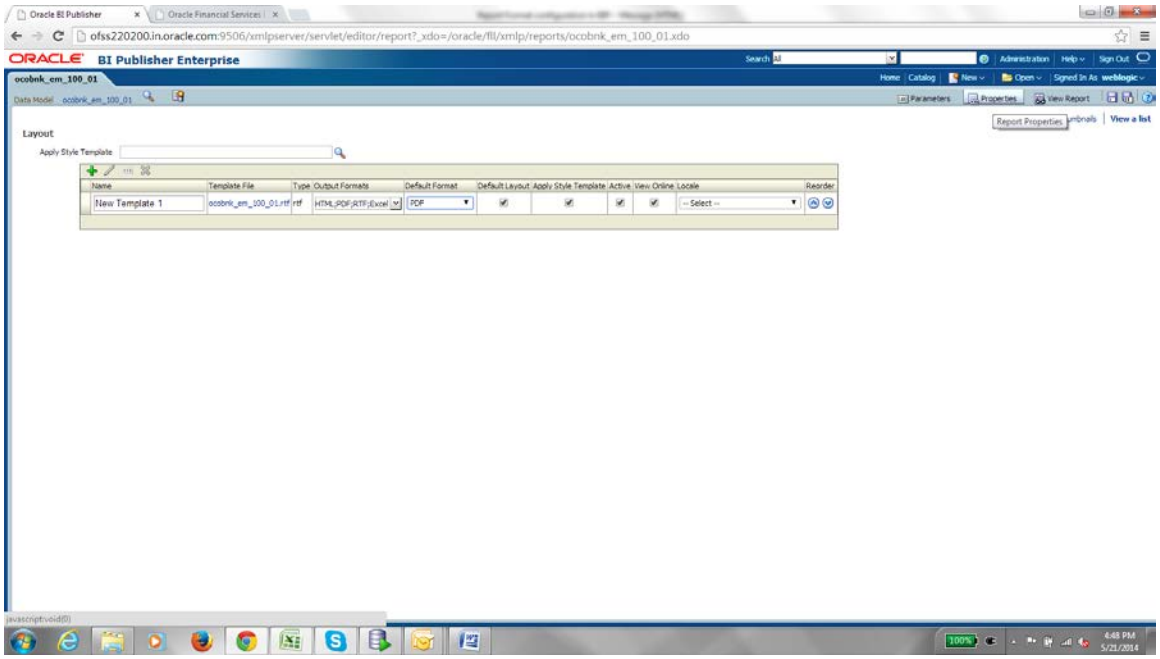
4. And in the end of drop down select DATA(XML) and DATA(CSV).



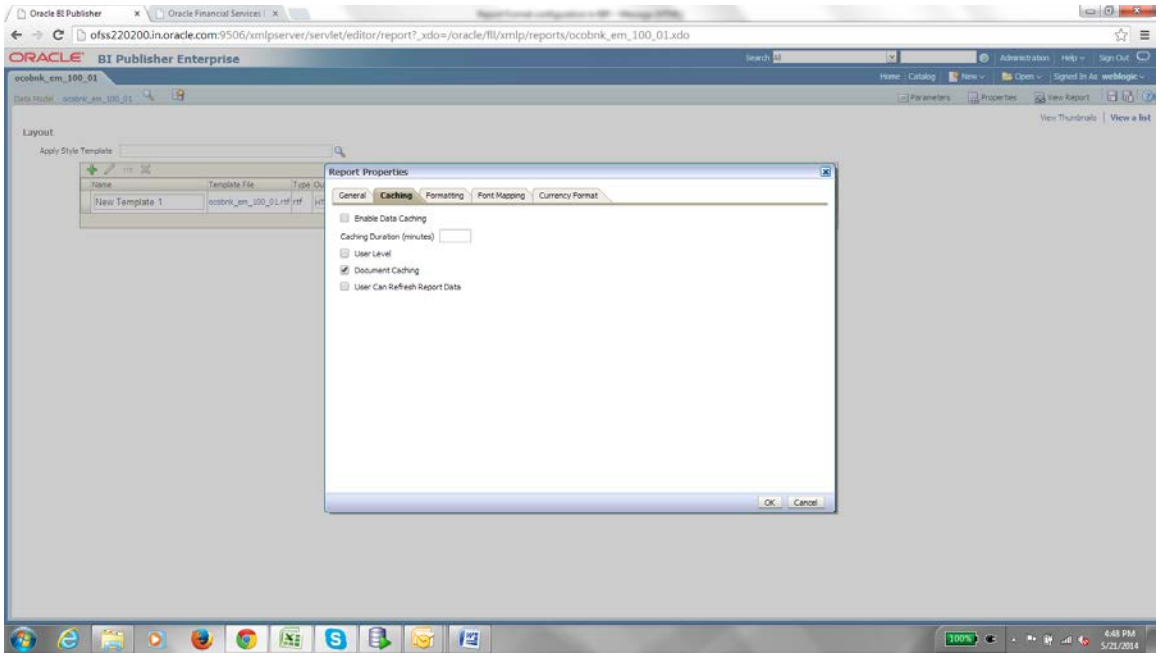
5. Also please select the Default Format as “PDF”



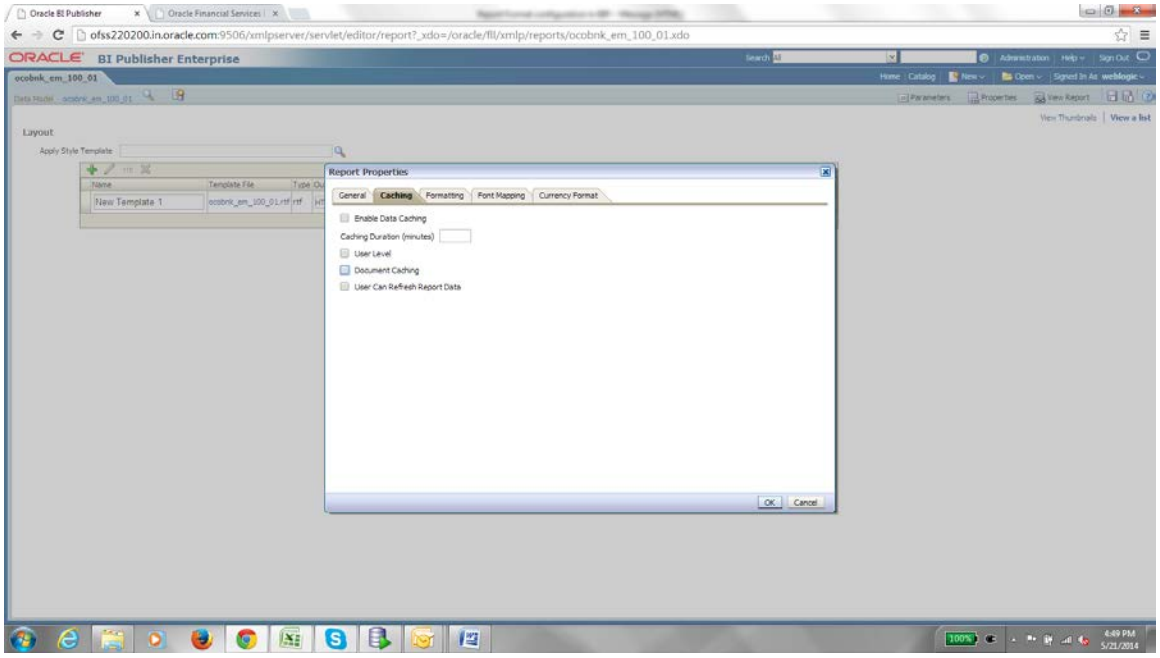
6. Once Output Format is done, I would recommend to make the Caching “False” for online reports as follows. On Right side Click on the Properties Button.



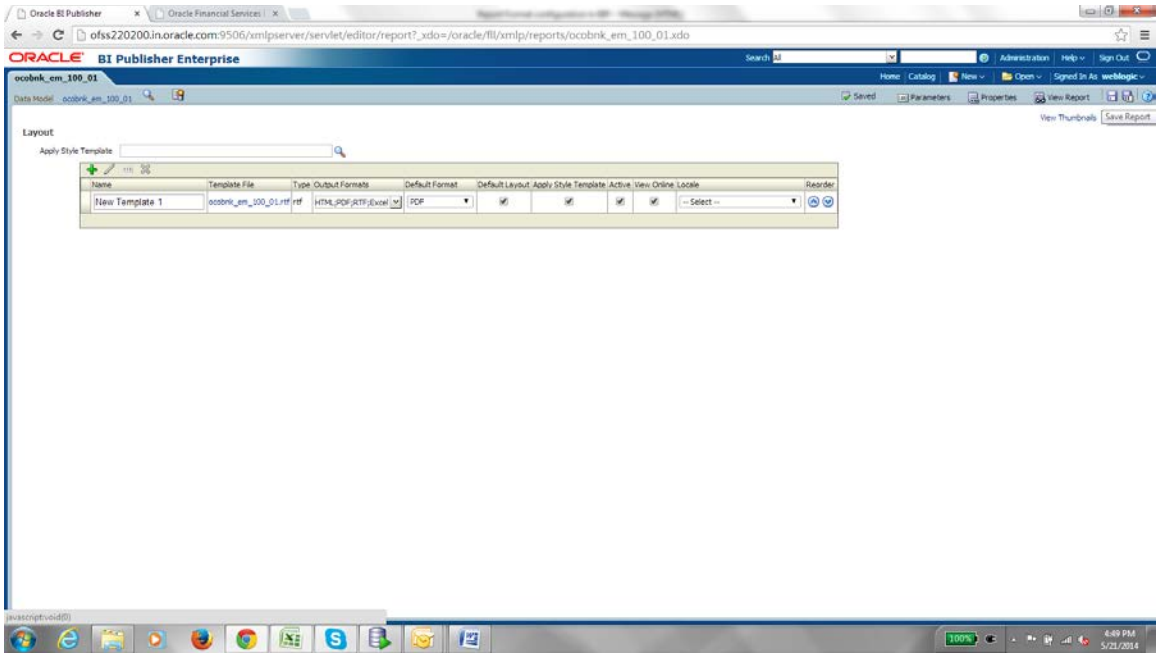
7. Select the TAB Caching, check if any of the caching is selected.



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



9. Save the changes.


Oracle BI Publisher
Oracle Financial Services
ofss220200.in.oracle.com:9506/xmlpserver/servlet/editor/report?_xdo=/oracle/fll/xmlp/reports/ocobnk_em_100_01.xdo
ORACLE BI Publisher Enterprise
ocobnk_em_100_01
Home | Catalog | New | Open | Signed In As weblogic
Data Model ocobnk_em_100_01
Saved | Parameters | Properties | View Report
View Thumbnails | Save Report
Layout
Apply Style Template
Name | Template File | Type | Output Formats | Default Format | Default Layout | Apply Style Template | Active | View Online | Locale | Reorder
New Template 1 | ocobnk_em_100_01.rtf | rtf | HTML,PDF,RTF,Excel | PDF | -- Select --
javascript:void(0)
100%
4:49 PM
5/21/2014

12. Naming Convention for Customized Objects

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>	

Object	Naming Convention	Comment
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 theguidelines 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_evw.sql will become : xyzuln_evw.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.

13. 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 Valuations(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)

13.1 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.

13.1.1 Producer related transaction

Element name	Sub Element	Child Element	Data Type	Values(Example)
TransactionDate Parameter				
	ParameterDetails			
		ParameterName	String	NA
		ParameterValue	Date(YYYY-MM-DDTHH:MM:SS)	NA
TransactionString Parameter				
	ParameterDetails			
		ParameterName	String	ACC_NBR
		ParameterValue	String	123654
	ParameterDetails			
		ParameterName	String	PTX_COMMENT
		ParameterValue	String	Test Comment
	ParameterDetails			

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

13.1.2 Other Transactions

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

13.2 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.

Element name	Sub Element	Data Type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Double)
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_apt(iv_apt_rec IN OUT NOCOPY acx_apt_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)
);
/
```

13.3 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.

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

xbtpmt_ew_100_01. xbtpmt_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)

xbtpmt_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 xbtpmt_em_100_01.post_pmt ();

Below are the exit points:-

BEFORE:

xbtpmt_ex_100_01.cv_post_pmt_bfr = cmncon_cl_000_01.CUSTOMIZED THEN

xbtpmt_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

xbtpmt_ex_100_01.cv_post_pmt_rep = cmncon_cl_000_01.CUSTOMIZED THEN

xbtpmt_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 :-

xbtpmt_ex_100_01.cv_post_pmt_afr = cmncon_cl_000_01.CUSTOMIZED THEN

xbtpmt_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));
);
/
```

13.4 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.

Element name	Sub Element	Data Type
CustomerAttributes	Name	String
	Value	String
CustomElements	Name	String
	CustomData	
CustomData	UniqueId	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

Element name	Sub Element	Data Type
BusinessAddressAttributes	Name	String
	Value	String
TransactionAttributes	Name	String
	Value	String
AllocationsAttributes	Name	String
	Value	String
CollateralAttributes	Name	String
	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));
```

13.5 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.

Element name	Sub element	Data type
CustomUserDefinedStringData	KeyName	String
	KeyName	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));
);
/
```

13.6 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.

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);
```

13.7 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.

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);
```

13.8 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.

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));
);
/
```

13.9 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.

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 -> 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 -> 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_

,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:-

xaeapp_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaeapp_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:-

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:-

xaebsa_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsa_ex_100_02.chk_afr(iv_bsa_rec ,ov_error_rec ,iv_axn_rec);

xaebsp_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:-

xaebsp_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsp_ex_100_02.chk_bfr(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);

REPLACE:-

xaebsp_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsp_ex_100_02.chk_rep(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);

AFTER:-

xaebsp_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsp_ex_100_02.chk_afr(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);

xaebsl_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:-

xaebsl_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsl_ex_100_02.chk_bfr(iv_bsl_rec ,ov_error_rec ,iv_axn_rec);

REPLACE:-

xaebsl_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsl_ex_100_02.chk_rep(iv_bsl_rec ,ov_error_rec ,iv_axn_rec);

AFTER:-

xaebsl_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsl_ex_100_02.chk_afr(iv_bsl_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);

xaebsf_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:-

xaebsf_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsf_ex_100_02.chk_bfr(iv_bsf_rec ,ov_error_rec ,iv_axn_rec);

REPLACE:-

xaebsf_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsf_ex_100_02.chk_rep(iv_bsf_rec ,ov_error_rec ,iv_axn_rec);

AFTER:-

xaebsf_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsf_ex_100_02.chk_afr(iv_bsf_rec ,ov_error_rec ,iv_axn_rec);

xaebsb_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:-

xaebsb_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsb_ex_100_02.chk_bfr(iv_bsb_rec ,ov_error_rec ,iv_axn_rec);

REPLACE:-

xaebsb_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsb_ex_100_02.chk_rep(iv_bsb_rec ,ov_error_rec ,iv_axn_rec);

AFTER:-

xaebsb_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsb_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));
);
/
```

13.10 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));
);
/
```

13.11 Call Activity (POST)

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

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 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));
);
/
```

13.12 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.

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)

13.13 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.

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)

13.14 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.

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 );
```

13.15 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.

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:-

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 );
```

13.16 Application Checklist (GET)

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

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 );
```

13.17 Outgoing File List (GET)

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

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 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 );
```

13.18 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);
```

13.19 Incoming File (GET)

Below are the package details for incoming file web service

Wrapper Engine package:-

xpfipf_ew_100_01.xpfipf_ew_100_01 (iv_req_rec_t IN OUT xws_ipf_req_rec_t);

Main Engine package:-

xpfipf_em_100_01.xpfipf_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:-

xpfipf_ex_100_01.cv_xpfipf_bfr = cmncon_cl_000_01.CUSTOMIZED THEN

xpfipf_ex_100_01.xpfipf_em_100_01_bfr(iv_req_rec_t);

REPLACE:-

xpfipf_ex_100_01.cv_xpfipf_rep = cmncon_cl_000_01.CUSTOMIZED THEN

xpfipf_ex_100_01.xpfipf_em_100_01_rep(iv_req_rec_t);

AFTER:-

xpfipf_ex_100_01.cv_xpfipf_afr = cmncon_cl_000_01.CUSTOMIZED THEN

xpfipf_ex_100_01.xpfipf_em_100_01_afr(iv_req_rec_t);

13.20 Products (GET)

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

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 );
```

13.21 Assets (GET)

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

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 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 );
```

13.22 Assets (PUT)

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

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 );
```

13.23 Asset Valuation (GET/PUT/POST)

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

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 );
```

13.24 Asset Sub Types (GET)

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

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     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);

13.25 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);

13.26 Application Update (PUT)

Below mentioned table has element name which indicates which type of custom data is passed in tha 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.

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 -> xaebsf_en_100_02

Business liabilities -> xaebsb_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:-

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:-

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:-

xaebsa_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsa_ex_100_02.chk_afr(iv_bsa_rec ,ov_error_rec ,iv_axn_rec);

xaebsp_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:-

xaebsp_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsp_ex_100_02.chk_bfr(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);

REPLACE:-

xaebsp_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsp_ex_100_02.chk_rep(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);

AFTER:-

xaebsp_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsp_ex_100_02.chk_afr(iv_bsp_rec ,ov_error_rec ,iv_axn_rec);

xaebsl_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:-

xaebsl_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsl_ex_100_02.chk_bfr(iv_bsl_rec ,ov_error_rec ,iv_axn_rec);

REPLACE:-

xaebsl_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsl_ex_100_02.chk_rep(iv_bsl_rec ,ov_error_rec ,iv_axn_rec);

AFTER:-

xaebsl_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsl_ex_100_02.chk_afr(iv_bsl_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);

xaebsf_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:-

xaebsf_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsf_ex_100_02.chk_bfr(iv_bsf_rec ,ov_error_rec ,iv_axn_rec);

REPLACE:-

xaebsf_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsf_ex_100_02.chk_rep(iv_bsf_rec ,ov_error_rec ,iv_axn_rec);

AFTER:-

xaebsf_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsf_ex_100_02.chk_afr(iv_bsf_rec ,ov_error_rec ,iv_axn_rec);

xaebsb_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:-

xaebsb_ex_100_02.CV_CHK_BFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsb_ex_100_02.chk_bfr(iv_bsb_rec ,ov_error_rec ,iv_axn_rec);

REPLACE:-

xaebsb_ex_100_02.CV_CHK_REP = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsb_ex_100_02.chk_rep(iv_bsb_rec ,ov_error_rec ,iv_axn_rec);

AFTER:-

xaebsb_ex_100_02.CV_CHK_AFR = cmncon_cl_000_01.CUSTOMIZED THEN

xaebsb_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 );
```

13.27 Application ACH (POST)

Below mentioned table has element name which indicates which type of custom data is passed in tha 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.

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);

13.28 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);

13.29 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);

13.30 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);

13.31 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);

13.32 Credit Limit Service (Customer/Business) [GET]

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

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:-

xcscrl_ew_100_01. xcscrl_ew_100_01 (iv_xcs_crl_rec IN OUT xcs_crl_rec_t);

Main Engine package:-

xcscrl_em_100_01. get_crl (iv_xcs_crl_rec IN OUT xcs_crl_rec_t);

Below are the Exit point package details for xcscrl_em_100_01. get_crl();

BEFORE:-

xcscrl_ex_100_01.cv_get_crl_bfr = cmncon_cl_000_01.CUSTOMIZED THEN

xcscrl_ex_100_01.get_crl_bfr(iv_xcs_crl_rec_t);

REPLACE:-

xcscrl_ex_100_01.cv_get_crl_rep = cmncon_cl_000_01.CUSTOMIZED THEN

xcscrl_ex_100_01.get_crl_rep(iv_xcs_crl_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);

13.33 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.

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 );
```

13.34 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.

Element name	Sub element	Data type
StringData	KeyName	String
	KeyValue	String
NumberData	KeyName	String
	KeyValue	Number (Decimal)
DateData	KeyName	String

Element name	Sub element	Data type
	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);

13.35 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.

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 );
```

13.36 Scenario Analysis Service (PUT)

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

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"
    }
   ]
 }
```

13.37 Transaction Parameters Service (GET)

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

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 );
```

13.38 Asset Tracking Attribute Service(PUT)

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

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);
```

13.39 Business Tracking Attribute Service(PUT)

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

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);
```

13.40 Customer Tracking Attribute Service(PUT)

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

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:-

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_tacking_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_tacking_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);
```

13.41 Account Tracking Attribute Service(PUT)

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

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);
```

13.42 Credit Bureau Web Service(PUT)

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

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);
```

13.43 Delete Account Web Service(DELETE)

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

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);
```

13.44 New Customization for RESTful Web Service

Below are the list of services introduced and it follows new naming statndard 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 OnBoarding 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)

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.

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 are the list of services which uses the Tracking Attribute in request/response similar to Custom Elements for extensibilty.

- 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.

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.

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
Advance Disbursement Service (GET)	XBTADV_EW_100_01. XBTADV_EW_100_01(iv_adv_axn_rec IN OUT xbt_adv_axn_rec_t)	XBTADV_EM_100_01.GET_ADV	XBTADV_EX_100_01		
			GET_ADV_BFR	GET_ADV_REP	GET_ADV_AFR
Advance	XBTADV_EW_100_01.	XBTADV_	XBTADV_EX_100_01		

Service name	Wrapper Package	Main Package	Exit Point Packages		
Disbursement Service (POST)	XBTADV_EW_100_01(iv_adv_axn_rec IN OUT xbt_adv_axn_rec_t)	EM_100_01.POST_ADV	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_BFR
		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
Usage and Rental Service (GET)	XCMPRC_EW_121_01.XCMPRC_EW_121_01(iv_xcm_rnt_rec IN OUT xcm_rnt_req_rec_t)	XCMPRC_EM_121_01.XCMPRC_EM_121_01	XCMPRC_EX_121_01		
			XCMPRC_EM_121_01_BFR	XCMPRC_EM_121_01_REP	XCMPRC_EM_121_01_AFR
Asset Service (GET)	XCSASE_EW_100_02.XCSASE_EW_100_02(iv_xcs_ase_rec IN OUT xcs_get_ase_tab_t)	XCSASE_EM_100_02.XCSASE_EM_100_02	XCSASE_EX_100_02		
			XCSASE_EX_100_02_BFR	XCSASE_EX_100_02_REP	XCSASE_EX_100_02_AFR
Asset	XCSASE_EW_100_01.	XCSASE_	XCSASE_EX_100_01		

Service name	Wrapper Package	Main Package	Exit Point Packages		
Service (POST)	XCSASE_EW_100_01(iv_xcs_ase_rec IN OUT xcs_ase_req_rec_t)	EM_100_01.XCSASE_EM_100_01	XCSASE_EX_100_01_BFR	XCSASE_EX_100_01_REP	XCSASE_EX_100_01_AFR
		XCSASE_EM_100_01.POST_ASSET	XCSASE_EX_100_01		
			POST_ASSET_BFR	POST_ASSET_REP	POST_ASSET_AFR
Asset Service (PUT)	XCSASE_EW_100_01.XCSASE_EW_100_01(iv_xcs_ase_rec IN OUT xcs_ase_req_rec_t)	XCSASE_EM_100_01.PUT_ASSET	XCSASE_EX_100_01		
			PUT_ASSET_BFR	PUT_ASSET_REP	PUT_ASSET_AFR
Asset Valuations Service (PUT)	XCSASE_EW_100_01.XCSASE_EW_100_01(iv_xcs_ase_rec IN OUT xcs_ase_req_rec_t)	XCSASE_EM_100_01.PUT_ASE_AVL	XCSASE_EX_100_01		
			PUT_ASE_AVL_BFR	PUT_ASE_AVL_REP	PUT_ASE_AVL_AFR
Asset Valuations Service (POST)	XCSASE_EW_100_01.XCSASE_EW_100_01(iv_xcs_ase_rec IN OUT xcs_ase_req_rec_t)	XCSASE_EM_100_01.POST_ASE_AVL	XCSASE_EX_100_01		
			POST_ASE_AVL_BFR	POST_ASE_AVL_REP	POST_ASE_AVL_AFR
Asset Valuations Service (GET)	XCSASE_EW_100_01.XCSASE_EW_100_01(iv_xcs_ase_rec IN OUT xcs_ase_req_rec_t)	XCSASE_EM_100_01.GET_ASE_AVL	XCSASE_EX_100_01		
			GET_ASE_AVL_BFR	GET_ASE_AVL_REP	GET_ASE_AVL_AFR
User Service (POST)	XUSUSR_EW_100_01.XUSUSR_EW_100_01 (iv_xus_usr_rec_t IN OUT xus_usr_rec_t)	XUSUSR_EM_100_01.XUSUSR_EM_100_01	XUSUSR_EX_100_01		
			XUSUSR_EM_100_01_BFR	XUSUSR_EM_100_01_REP	XUSUSR_EM_100_01_AFR

Service name	Wrapper Package	Main Package	Exit Point Packages		
Account Monitory Transaction History Fetch Service (GET)	XCSTXN_EW_100_01.XCSTXN_EW_100_01(iv_xcs_txn_hist_rec IN OUT xcs_txn_hist_rec_t)	XCSTXN_EM_100_01.XCSTXN_EM_100_01	XCSTXN_EX_100_01		
			XCSTXN_EX_100_01_BFR	XCSTXN_EX_100_01_BFR	XCSTXN_EX_100_01_BFR
		XCSTXN_EM_100_01.GET_TXN_HIST	XCSTXN_EX_100_01		
			GET_TXN_HIST_BFR	GET_TXN_HIST_REP	GET_TXN_HIST_AFR
Vendor details fetch service (GET)	XCSVEN_EW_100_01.XCSVEN_EW_100_01(iv_ven_req_t IN OUT xcs_ven_req_t)	XCSVEN_EM_100_01.GET_VEN	XCSVEN_EX_100_01		
			GET_VEN_BFR	GET_VEN_REP	GET_VEN_AFR
Vendor Comment fetch service (GET)	XCSVEN_EW_100_01.COMMENT(iv_comment IN OUT xcs_ven_vcm_rec_t)	XCSVEN_EM_100_01.GET_COMMENT	XCSVEN_EX_100_01		
			GET_COMMENT_BFR	GET_COMMENT_REP	GET_COMMENT_AFR
Vendor Create Service (POST)	XCSVEN_EW_100_01.XCSVEN_EW_100_01(iv_ven_req_t IN OUT xcs_ven_req_t)	XCSVEN_EM_100_01.XCSVEN_EM_100_01	XCSVEN_EX_100_01		
			XCSVEN_EX_100_01_BFR	XCSVEN_EX_100_01_REP	XCSVEN_EX_100_01_AFR
		XCSVEN_EM_100_01.POST_VEN	XCSVEN_EX_100_01		
			POST_VEN_BFR	POST_VEN_REP	POST_VEN_AFR
Vendor Comment Create Service (POST)	XCSVEN_EW_100_01.COMMENT(iv_comment IN OUT xcs_ven_vcm_rec_t)	XCSVEN_EM_100_01.POST_COMMENT	XCSVEN_EX_100_01		
			POST_COMMENT_BFR	POST_COMMENT_REP	POST_COMMENT_AFR
Vendor Update Service (PUT)	XCSVEN_EW_100_01.XCSVEN_EW_100_01(iv_ven_req_t IN OUT xcs_ven_req_t)	XCSVEN_EM_100_01.UPD_VEN	XCSVEN_EX_100_01		
			UPD_VEN_BFR	UPD_VEN_REP	UPD_VEN_AFR
Producer	XPRPRC_EW_100_02.	XPRPRC_	XPRPRC_EX_100_02		

Service name	Wrapper Package	Main Package	Exit Point Packages		
Create Service (POST)	XPRPRC_EW_100_02(iv_producer IN OUT xpr_pro_dtl_rec_t)	EM_100_02. XPRPRC_EM_100_02	XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR
		XPRPRC_EM_100_02.POST_PRODUCER	XPRPRC_EX_100_02		
			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_dtl_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		
			GET_PRODUCER_BFR	GET_PRODUCER_REP	GET_PRODUCER_AFR
Producer Update Service (PUT)	XPRPRC_EW_100_02. XPRPRC_EW_100_02(iv_producer IN OUT xpr_pro_dtl_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		
			PUT_PRODUCER_BFR	PUT_PRODUCER_REP	PUT_PRODUCER_AFR
Producer Contact Service (POST)	XPRPRC_EW_100_02. XPRPRC_EW_100_02(iv_contactIN 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
		XPRPRC_EM_100_02.POST_CONTACT	XPRPRC_EX_100_02		
			POST_COMMENT_BFR	POST_COMMENT_REP	POST_COMMENT_AFR
Producer	XPRPRC_EW_100_02.	XPRPRC_	XPRPRC_EX_100_02		

Service name	Wrapper Package	Main Package	Exit Point Packages		
Contact Service (GET)	XPRPRC_EW_100_02(iv_contactIN OUT xpr_pro_pcn_rec_t)	EM_100_02. XPRPRC_EM_100_02	XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR
		XPRPRC_EM_100_02.GET_CONTACT	XPRPRC_EX_100_02		
			GET_CONTACT_BFR	GET_CONTACT_REP	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_REP	XPRPRC_EX_100_02_AFR
		XPRPRC_EM_100_02. POST_COMMENT	XPRPRC_EX_100_02		
			POST_COMMENT_BFR	POST_COMMENT_REP	POST_COMMENT_AFR
Producer Comment Service (GET)	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_REP	XPRPRC_EX_100_02_AFR
		XPRPRC_EM_100_02. GET_COMMENT	XPRPRC_EX_100_02		
			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_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_SUBVENTION	XPRPRC_EX_100_02		
			GET_SUBVENTION_BFR	GET_SUBVENTION_REP	GET_SUBVENTION_AFR
Producer	XPRPRC_EW_100_02.	XPRPRC_	XPRPRC_EX_100_02		

Service name	Wrapper Package	Main Package	Exit Point Packages		
Compensation Service (GET)	XPRPRC_EW_100_02(iv_compensation IN OUT xpr_pro_cmp_rec_t)	EM_100_02. XPRPRC_EM_100_02	XPRPRC_EX_100_02_BFR	XPRPRC_EX_100_02_REP	XPRPRC_EX_100_02_AFR
		XPRPRC_EM_100_02. GET_COMPENSATION	XPRPRC_EX_100_02		
			GET_COMPENSATION_BFR	GET_COMPENSATION_REP	GET_COMPENSATION_AFR
Producer Statement Service (GET)	XPRPRC_EW_100_02. XPRPRC_EW_100_02(iv_statements IN OUT xpr_pro_psm_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_STATEMNTS	XPRPRC_EX_100_02		
			GET_STATEMENTS_BFR	GET_STATEMENTS_REP	GET_STATEMENTS_AFR
Producer Holdback Service (GET)	XPRPRC_EW_100_02. XPRPRC_EW_100_02(iv_holdback IN OUT xpr_pro_hlr_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_HOLDBACK_AMOUNT	XPRPRC_EX_100_02		
			GET_HOLDBACK_AMOUNT_BFR	GET_HOLDBACK_AMOUNT_REP	GET_HOLDBACK_AMOUNT_AFR
Get webhook maintenance (GET)	xcswhk_ew_100_01. xcswhk_ew_100_01 (iv_evi_id IN EVE_EVI_EVW.EVI_ID%TYPE, iv_evi_jmd_id IN OUT EVE_EVI_EVW.EVI_JMD_ID%TYPE, iv_evi_status_cd IN OUT EVE_EVI_EVW.EVI_S	xcswhk_em_100_01. xcswhk_em_100_01	xcswhk_ex_100_01		
			xcswhk_ex_100_01_afr	xcswhk_ex_100_01_REP	xcswhk_ex_100_01_afr
		xcswhk_em_100_01.	xcswhk_ex_100_01		

Service name	Wrapper Package	Main Package	Exit Point Packages		
	TATUS_CD%TYPE, iv_action IN VARCHAR2, iv_err_code OUT VARCHAR2, iv_err_desc OUT VARCHAR2, iv_whk_req_rec IN OUT xcs_whk_req_rec_t);	get_whk_setup	get_whk_setup_BFR	get_whk_setup_REP	get_whk_setup_AFR
Service name	**Wrapper Package**	**Main Package**	**Exit Point Packages**		
Web hooks update event status	xcswhk_ew_100_01.xcswhk_ew_100_01 (iv_evi_id IN EVE_EVI_EVW.EVI_ID%TYPE, iv_evi_jmd_id IN OUT EVE_EVI_EVW.EVI_JMD_ID%TYPE, iv_evi_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_rec IN OUT xcs_whk_req_rec_t);	xcswhk_em_100_01.update_event_status	xcswhk_ex_100_01		
			update_event_status_bfr	update_event_status_rep	update_event_status_afr
Webhooks maintenance : Save the webservice details (POST)	xcswhk_ew_100_02.xcswhk_ew_100_02(iv_whk_set_rec IN OUT xsc_whk_set_rec_t);	xcswhk_em_100_02.xcswhk_em_100_02.	xcswhk_ex_100_02		
			post_whk_bfr	post_whk_afr	post_whk_rep
Webhook Maintenance : retrieve the webservice details(GET)	xcswhk_ew_100_02.xcswhk_ew_100_02(iv_whk_set_rec IN OUT xsc_whk_set_rec_t);	xcswhk_em_100_02.xcswhk_em_100_02.	xcswhk_ex_100_02		
			get_whk_bfr	get_whk_rep	get_whk_afr

Service name	Wrapper Package	Main Package	Exit Point Packages		
Webhook Maintenance : update the webhook maintenance(PUT)	xcswhk_ew_100_02.xcswhk_ew_100_02(iv_whk_set_rec IN OUT xsc_whk_set_rec_t);	xcswhk_em_100_02.xcswhk_em_100_02.	xcswhk_ex_100_02		
			put_whk_bfr	put_whk_rep	put_whk_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);

13.45 Securitization Web Service(POST)

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

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

Service name	Wrapper Package	Main Package	Exit Point Packages		
			Before	Replace	After
SECURITI	XCSSEC_EW_100_01.	XCSSEC_	XCSSEC_EX_100_01		

Service name	Wrapper Package	Main Package	Exit Point Packages		
ZATION (POST)	XCSSEC_EW_100_01 (IV_SEC_POOL_REC_T IN OUT XCS_SEC_POOL_REC_T)	EM_100_01. XCSSEC_EM_100_01	XCSSEC_EX_100_01_BFR	XCSSEC_EX_100_01_REP	XCSSEC_EX_100_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

);
```

13.46 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)
```

14. Appendix: Revision History

Section	Section Name	Release Version	Description of Change
13	RESTful Web Services Extensibility	1.0	Added 13.38, 13.39, 13.40, 13.41 sub sections for Asset, Business, Customer and Account Tracking Attributes respectively.

Extensibility Guide
December 2019
Version 14.8.0.0.0

Oracle Financial Services Software Limited
Oracle Park
Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India

Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax:+91 22 6718 3001
https://www.oracle.com/industries/financial-services/index.html