Oracle® Revenue Management and Billing

Version 2.9.0.0.0

Reporting SDK Guide

Revision 2.1

F34133-01

August, 2020

Oracle Revenue Management and Billing Reporting SDK Guide

F34133-01

Preface

About This Document

This document will help you to understand ORMB reporting platform and assist you to configure ORMB Reporting system for generating different reports.

Intended Audience

This document is intended for the following audience:

- Implementation Team
- Consulting Team
- Development Team

Organization of the Document

The information in this document is organized into the following sections:

Section No.	Section Name	Description
Section 1	Introduction	Provides an overview of ORMB web reporting platform. It also lists benefits of the platform.
Section 2	Web Reporting	Explains the web-reporting platform. It lists the necessary components and how to the utility for using Groovy to generate reports.
Section 3	ORMB Document Generation	Explains the steps to be follow to generate reports. It also provides information about the various functions, algorithms defined for generating reports in PDF.
Section 4	Future Direction	Lists future enhancements to be made in the platform.

Related Documents

You can refer to the following documents for more information:

Document	Description
Oracle Revenue Management and Billing Bill and Letter Reports Configuration Guide	Explains the ORMB web reporting platform and also lists the benefits of the platform. It also assists in configuring the ORMB Reporting system for generating different reports.
Oracle Revenue Management and Billing Reporting Installation Guide	Lists the prerequisites, supported platforms, and hardware and software requirements for installing the Oracle Revenue Management and Billing (ORMB) reporting framework.

Document	Description
Oracle Revenue Management and Billing Reporting User Guide	Explains the ORMB reporting process and the various types of frameworks provided by ORMB. It also explains how to use Groovy to generate different reports.

Conventions

The following conventions are used across the document:

Convention	Meaning
boldface	Boldface indicates graphical user interface elements associated with an action, or terms defined in the text.
italic	Italic indicates a document or book title.
`monospace`	Monospace indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or information that an end-user needs to enter in the application.

Change Log

Revision	Last Update	Updated Section	Comments
2.1	19-Jan-2022	-	Added Bookmarks in PDF

Contents

1. Introduction

ORMB application provides a unique web-reporting platform. This platform offers best-in-class enterprise reporting feature that enables customers to uncover new insights and take faster, more informed business decisions.

ORMB Reporting feature easily integrates with existing infrastructure, provides reusable design and data elements, and enables an Easy API Access.

The ORMB reporting platform provides following benefits:

- It allows you generate and deliver customized reports to colleagues in different departments and customers on time.
- It allows implementers to efficiently create and update document templates.
- It supports web based reporting, and charts data representation with existing or custom groovy services.
- It allows data representation using query zones with SQL in tabular formats.
- It supports output in PDF, CSV, XML, JSON formats using Custom Formatting Objects Processor (FOP) Templates.

2. Web Reporting

ORMB Reporting platform provides support for tabular data representation using SQL commands across different query zone. Each of these channels can be configure as per the user's requirements.

It also supports representing data in the form of charts with both existing and customized services. Using ORMB SVG API, you can generate below mentioned charts:

- Bar Chart
- Pier Chart
- Line Chart
- XY Chart
- Bubble chart
- Time Series

The designer can control the type, size, position, and contents of the chart using the following attributes by configuring a graphical representation of an XML list. You can use following HTML attributes.

`oraChart="type:pie, stacked, cluster, line, area, combo;"`

A sample code for generating pie chart using HTML attributes are as below:

```
<html>
<head>
<title>Pie Chart</title>
</head>
<body>
<div style="width:100%; height:290px;"
    oraChart="type:pie;"
    oraChartSeries1="list:set; labelPath:date; amount:amount; "
    oraChartBroadcast="BILL_ID:billId;">
</div>
</body>
<xml>
<root>
  <set>
<date>05-02-2003</date>
<amount>163.24</amount>
<billId>592211649441</billId>
  </set>
  <set>
<date>06-02-2003</date>
```

```
<amount>97.29</amount>
<billId>592211649442</billId>
  </set>
  <set>
<date>07-02-2003</date>
<amount>54.38</amount>
<billId>592211649443</billId>
  </set>
</root>
</xml>
</html>
```

The generated pie chart is represented as:

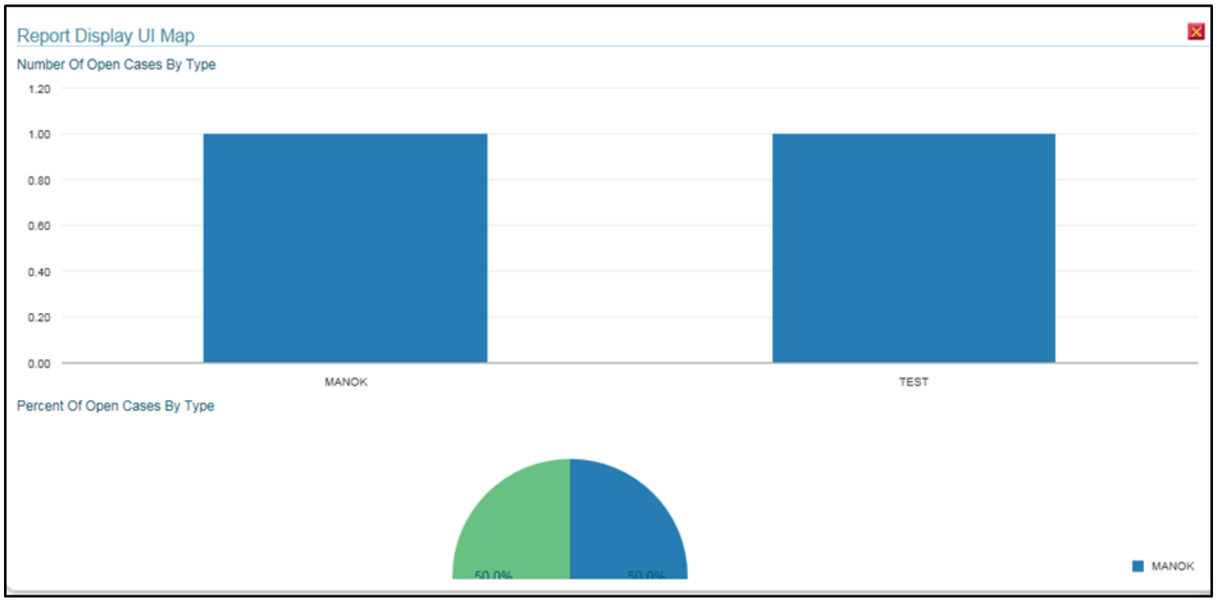


Figure 1: Pie Chart

Note: For more details on configuring a chart, refer to *ORMB Help Guide*.

2.1 Web Reporting Using Groovy

ORMB platform provides two components, ‘Service Script’ and ‘User Interface (UI) Map’ that enable you to edit existing Reports or create new reports using ORMB FOP framework. This framework allows you to use Apache Groovy utility. This utility needs configure with ‘Service Script’ and ‘UI Map’ before they can use. The Service Script consists of actual business logic and needs to create, whereas UI Map designed with the same schema as that of service script. The UI Map will call service script, perform necessary operation and show chart as an output on UI Map.

Groovy code can incorporate in scripts using the step type Groovy Members. For each script with Groovy code, there will be a single Groovy class created by concatenating all Groovy Members steps.

For security reasons, the product and third party Java classes available for scripting in Groovy will be restricted.

Note: For more information on the allowable base classes, refer Groovy Javadoc in the Application Viewer and 'View Groovy Javadoc' link in the context sensitive Script Tips zone of ORMB Help Section.

To view the list of allowable third party classes, refer 'View third party Groovy whitelist' link in the Script Tips zone.

If the script has configured to use a scripting engine version, it can include a combination of regular and Groovy Members step types. The script step types will define the process executed. The Groovy Members steps will contain code that can be called from Edit Data step types within the script using the 'Invoke Groovy' command. This command supports only Groovy methods that receive no arguments and return void. Refer to the section on edit data steps for more details.

For scripts using Groovy Members, the framework provides a superclass containing methods that support common scripting actions such as move commands, string evaluation, and methods to invoke business objects, business services and service scripts. For details of the supported methods, refer the Groovy specific JavaDocs.

Below examples, explain how you can implement web reporting using Groovy.

2.1.1 Service Script

To implement web reporting using service scripts, you need to follow the below sequence

1. Create Service Script
2. Define step type for the created service script
3. Define schema for the created service script

2.1.1.1 Defining a Service Script

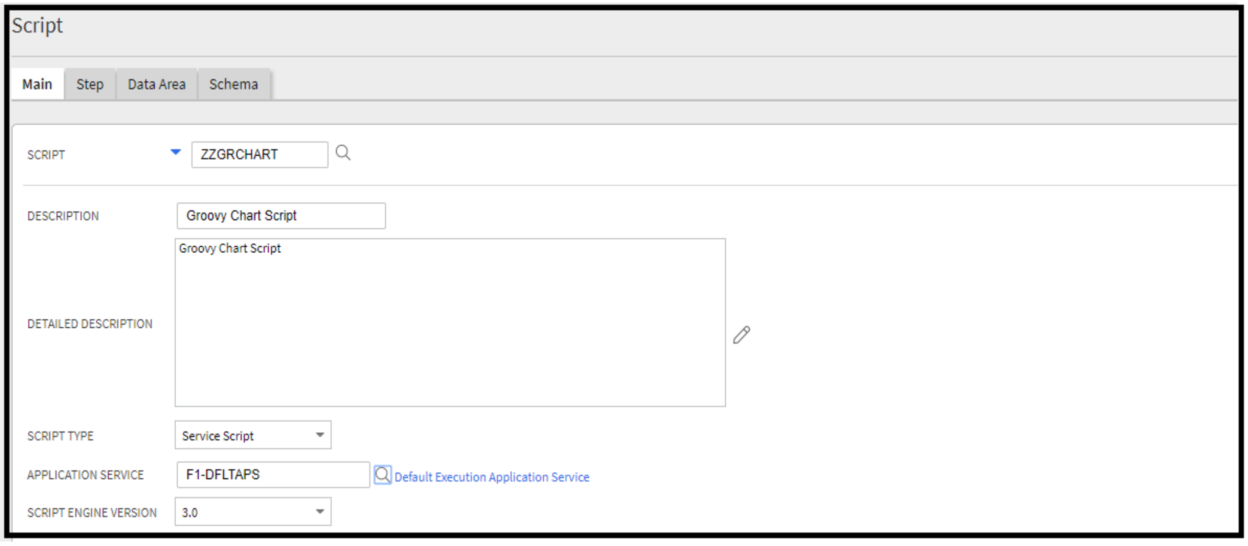
To define Service Script, follow the below steps:

1. From the Admin menu, select S and then click Script.
2. Click Add option from the Script sub-menu. The Script screen appears.
3. Enter values in the following fields within Main tab:

Field Name	Description	Value
Name	Used to define name of the script.	For example, ZZGRCHART
Script Type	Used to define respective server-based script.	Service Script Note: It is mandatory to select 'Service Script' as type.
Application Service	Used to indicate if the execution of the script should secure	For example, F1-DFLTAPS
Script Engine Version	Used to define the version of the XML Path Language (XPath) used for the script	3.0

		Note: It is mandatory to use 3.0 as version.

4. Click Save.



Script

SCRIPT: ZZGRCHART

DESCRIPTION: Groovy Chart Script

DETAILED DESCRIPTION: Groovy Chart Script

SCRIPT TYPE: Service Script

APPLICATION SERVICE: F1-DFLTAPS Default Execution Application Service

SCRIPT ENGINE VERSION: 3.0

Figure 2: Service Script

After creating a script, you must define Steps.

2.1.1.2 Defining a Step

You can implement web reporting using Step Types as:

- Edit Data
- Groovy Members

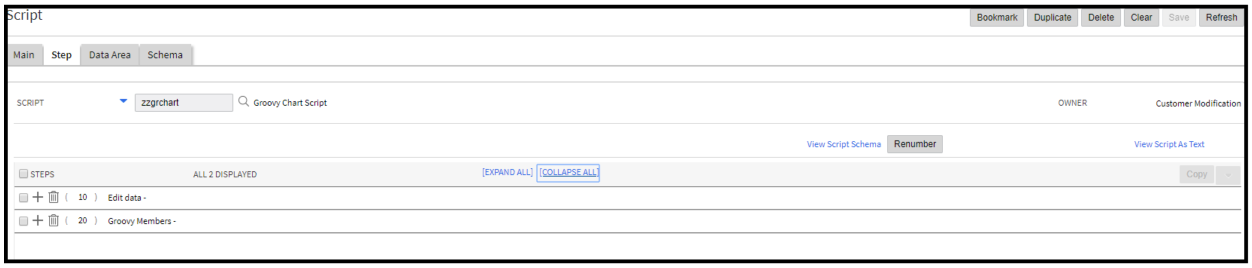
To define Step Type, follow the below steps:

1. From the Admin menu, select S and then click Script.
2. Click Add option from the Script sub-menu. The Script screen appears.
3. Click Step tab from the Script screen.
4. Enter values in the following fields:

Field Name	**Description**	**Value**
Step Sequence	Used to define the relative position of this step in respect of the other steps.	For example, • 10 • 20 Note: This value can be any unique integer value.
Step Type	Used to define the **Step Type** that corresponds with the step	• Edit Data • Groovy Members

Text	**Used to** specify the **information that displays in the script area when the step executes.**	
Edit Data Text	Used to specify commands to control the script processing.	

5. Click Save.



SCRIPT zzgrchart Groovy Chart Script OWNER Customer Modification

STEPS ALL 2 DISPLAYED

(10) Edit data -

(20) Groovy Members -

Figure 3: Script – Step

A sample sequence for Step Type ‘Edit Data’ is given below:

- **Step Sequence**: 10. This value can be any unique integer value.
- **Step Type**: Edit Data
- **Edit Data Text**: invokeGroovy 'function';

A sample sequence for Step Type ‘Groovy Members’ is listed below

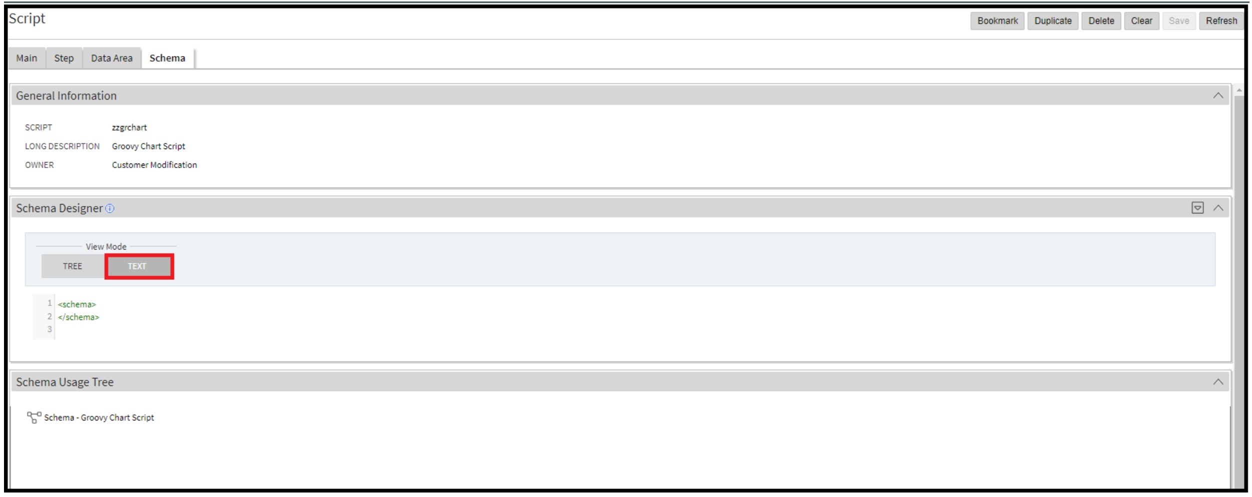
- **Step Sequence**: 20. This value can be any unique integer value.
- **Step Type**: Groovy Members
- **Edit Data Text**: A sample code is listed below:

```
public void function() {
logger.info("Groovy Test  : ");
String tdTypeCd = "C1-ADJUP";
String languageCode = "ENG";
String tdStatusDesc = "O";
String userId = "AHTRF";


def  query  =  createPreparedStatement("SELECT  TT.TD_TYPE_CD,  COUNT
(TE.TD_ENTRY_ID)  COUNT,  LO.DESCR  STATUS_DESCR  FROM  CI_TD_TYPE  TT,
CI_TD_TYPE_L  TTL,  CI_TD_ENTRY  TE,  CI_TD_MSG_PARM  TP,  CI_MSG_L  ML,
CI_LOOKUP LO, SC_USER UR WHERE TT.TD_TYPE_CD       = TTL.TD_TYPE_CD AND
( TRIM(TT.TD_TYPE_CD)      =TRIM(:tdTypeCd)) AND TRIM(TTL.LANGUAGE_CD)
=TRIM(:languageCode)  AND  TE.TD_TYPE_CD            = TT.TD_TYPE_CD  AND
TE.ENTRY_STATUS_FLG     <>   'C'   AND   (   TRIM(TE.ENTRY_STATUS_FLG)
=TRIM(:tdStatusDesc)) AND LO.FIELD_NAME         = 'ENTRY_STATUS_FLG' AND
LO.FIELD_VALUE          = TE.ENTRY_STATUS_FLG AND TRIM(LO.LANGUAGE_CD)
=TRIM(:languageCode) AND TE.TD_ENTRY_ID        = TP.TD_ENTRY_ID(+)  AND
TE.MESSAGE_CAT_NBR    = ML.MESSAGE_CAT_NBR AND TE.MESSAGE_NBR          =
ML.MESSAGE_NBR AND TRIM(ML.LANGUAGE_CD)       =TRIM(:languageCode)  AND
TRIM(UR.USER_ID)                  =TRIM(:userId)  GROUP  BY  LO.DESCR,
TT.TD_TYPE_CD","");
  query.bindString("tdTypeCd", tdTypeCd, "TD_TYPE_CD");
  query.bindString("tdStatusDesc", tdStatusDesc, "ENTRY_STATUS_FLG");
  query.bindString("languageCode", languageCode, "LANGUAGE_CD");
  query.bindString("userId", userId, "USER_ID");
  def list = query.list();
  for(def row : list){

move(row.getString("COUNT"),"parm/+caseTypeChartReportDtlsList/count")
     move(row.getString("STATUS_DESCR"),"parm/caseTypeChartReportDtlsL
ist[last()]/tdStatusDesc")
     move(row.getString("TD_TYPE_CD"),"parm/caseTypeChartReportDtlsLis
t[last()]/toDoType")
}
  }
```

2.1.1.3 Defining a Schema

To define a Schema, follow the below steps:

1. From the **Admin** menu, select **S** and then click **Script**.
2. Click **Add** option from the **Script** sub-menu. The **Script** screen appears.
3. Click **Schema** tab from the Script screen.
4. Click the **Text** icon present in **Schema Designer** screen.
5. Enter the code in the pane.
6. Click **Save**.



SCRIPT	zzgrchart
LONG DESCRIPTION	Groovy Chart Script
OWNER	Customer Modification

```
<schema>
</schema>
```

Schema - Groovy Chart Script

Figure 4: Script - Schema

A sample code for defining a schema using Schema Designer is listed below:

```
<schema>
    <tdStatusDesc mapField="TD_STATUS_DESCR"/>
    <toDoType mapField="TD_TYPE_CD"/>
    <caseTypeReportDtlsList type="list" mapList="CASE_TYPE_RPRT_DTLS">
        <creationDateTime mapField="CRE_DTTM"/>
        <messageText mapField="MESSAGE_TEXT"/>
        <toDoId mapField="TD_ENTRY_ID"/>
        <age mapField="AGE_LBL"/>
    </caseTypeReportDtlsList>
    <caseTypeChartReportDtlsListtype="list"
mapList="CASE_TYPE_OPEN_CHART_LIST">
        <count mapField="COUNT"/>
        <tdStatusDesc mapField="TD_STATUS_DESCR"/>
        <toDoType mapField="TD_TYPE_CD"/>
    </caseTypeChartReportDtlsList>
</schema>
```

2.1.2 User Interface Map

To generate UI Map, you need to follow the below sequence

1. Define a UI Map
2. Define a Schema for the UI Map

2.1.2.1 Defining a UI Map

To define a UI Map, follow the below steps:

1. From the **Admin** menu, select **U** and then click **UI Map**.

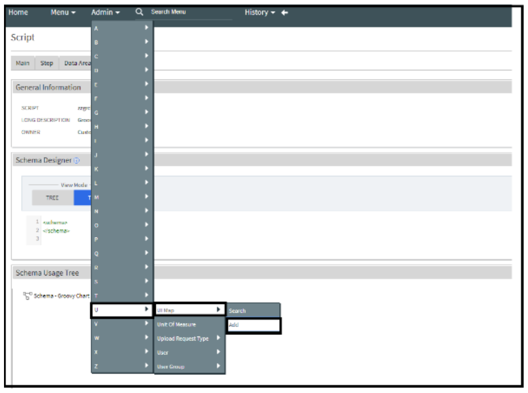
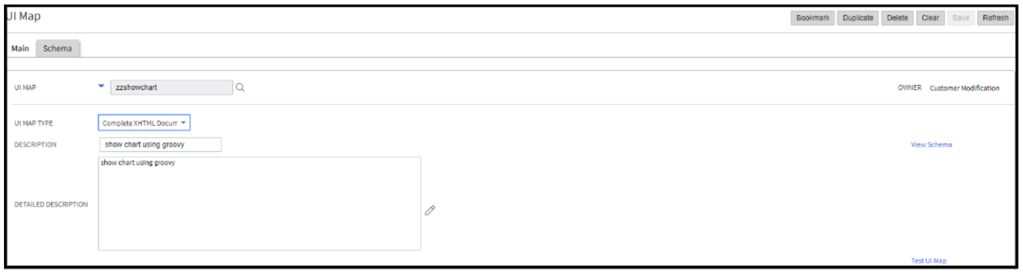


Figure 5: UI Map

2. Click **Add** option from the **UI Map** sub-menu. The **UI Map** screen appears.
3. Enter values in the following fields within Main tab:

Field Name	Description	Value	Mandatory (Yes or No)
Name	Used to specify the name of the algorithm.	ZZSHOWCHART	Yes
UI Map Type	Used to indicate the configuration type	Complete XHTML Document	Yes
Description	Used to specify description for the map.		No
Detailed Description	Used to specify detailed description for the map.		No

4. Click **Save**.



UI MAP: zzshowchart

OWNER: Customer Modification

UI MAP TYPE: Complete XHTML Docum

DESCRIPTION: show chart using groovy

DETAILED DESCRIPTION: show chart using groovy

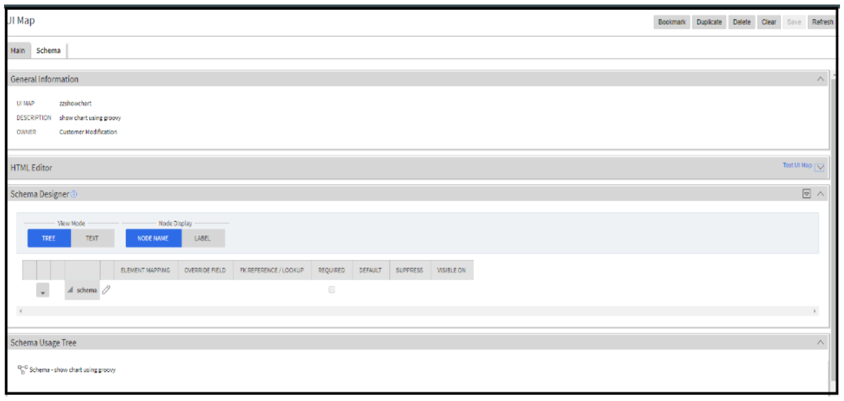
Figure 6: UI Map

2.1.2.2 Defining a Schema

To define Schema, follow the below steps:

1. From the **Admin** menu, select **U** and then click **UI Map**.
2. Click Add option from the Script sub-menu. The Script screen appears.
3. Click Schema tab from the Script screen.
4. You can define the schema in:

- HTML Editor
    i. Enter the code in editor pane.
    ii. Click **Save**.
- Schema Designer
    i. Click the **Text button** in Schema Designer screen.
    ii. Enter the code in the Source View pane.
    iii. Click Save.



General Information

UI MAP	zzshowchart
DESCRIPTION	show chart using groovy
OWNER	Customer Modification

Schema Usage Tree

Schema - show chart using groovy

Figure 7: UI Map -Schema

2.1.2.3 HTML Editor

A sample code to be used in HTML Editor is listed below:

```
<!DOCTYPE     html     PUBLIC     "-//W3C//DTD     XHTML     1.0     Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/TR/xhtml1/strict">

    <head>

        <title>  </title>

        <link href="cisDisabled.css" type="text/css" rel="stylesheet" />

 <script>

               function fnChange()

              {

                   alert('hi');

                   if (!oraInvokeSS('ZZGRCHART',null)) {

                   oraShowErrorAlert();

                   return;

                }

             }

</script>

    </head>

    <body class="oraZoneMap" onLoad="fnChange();">


<div style="width:100%; height:290px;"

    oraChart="type:stacked;"

    oraChartSeries1="list:caseTypeChartReportDtlsList;         xaxis:toDoType;
labelPath:tdStatusDesc; amount:count;script:C1_TDENTR_BP; "


oraChartBroadcast="TD_STATUS_DESCR:caseTypeChartReportDtlsList/tdStatusDesc;"
onClick="oraSubmitMap('SAVE');"></div>
```

```
<div style="width:100%; height:290px;"
    oraChart="type:pie;"
    oraChartSeries1="list:caseTypeChartReportDtlsList;     labelPath:toDoType;
amount:count;script:C1_TDENTR_BP; "

oraChartBroadcast="TD_STATUS_DESCR:caseTypeChartReportDtlsList/tdStatusDesc;"
onClick="oraSubmitMap('SAVE');"></div>


    </body>
 <xml style="display:none;"></xml>
</html>
```

2.1.2.4 Schema Designer

A sample code used in Schema Designer is listed below:

```
<schema>
    <tdStatusDesc mapField="TD_STATUS_DESCR"/>
    <toDoType mapField="TD_TYPE_CD"/>
    <caseTypeReportDtlsList type="list" mapList="CASE_TYPE_RPRT_DTLS">
        <creationDateTime mapField="CRE_DTTM"/>
        <messageText mapField="MESSAGE_TEXT"/>
        <toDoId mapField="TD_ENTRY_ID"/>
        <age mapField="AGE_LBL"/>
    </caseTypeReportDtlsList>
    <caseTypeChartReportDtlsList                                  type="list"
mapList="CASE_TYPE_OPEN_CHART_LIST">
        <count mapField="COUNT"/>
        <tdStatusDesc mapField="TD_STATUS_DESCR"/>
        <toDoType mapField="TD_TYPE_CD"/>
    </caseTypeChartReportDtlsList>
</schema>
```

2.1.3 UI Map Result

To test the scenario, click on **Test UI Map** link present in right corner of **UI Map** screen. The UI Map will call JavaScript function at '**On Load'** event. The JavaScript function calls '**Service Script**' business logic and performs necessary operations required for fetching valid data. Once resulting data fetched, control returns to UI Map with output.

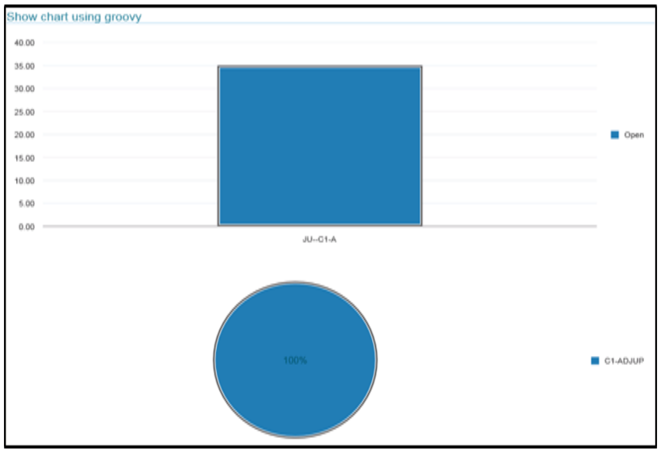


Figure 8: UI Map Result

3. ORMB Document Generation

Apache FOP (Formatting Objects Processor) library used to generate the PDF format for ORMB document. Apache FOP is a print formatter driven by XSL formatting objects. It is a Java application that reads a formatting object tree conforming to the XSL 1.1 Recommendation (05 December 2006) and then turns it into a PDF document or other output formats. It allows you to preview the outputs directly on screen.

Apache FOP is a Java application that reads a Formatting Object (FO) tree and renders it to a specified output such as PDF, PS, XML, etc. The primary output target is PDF. For more understanding, refer official Apache FOP site at https://xmlgraphics.apache.org/fop/

You can also refer GitHub at https://github.com/apache/fop

Note: The custom extracts can be configured with Groovy whereas custom templates and images can be configured for PDF.

ORMB Reporting feature supports various other document formats like CSV, XML and JSON for other types of reports. Extracts can trigger:

- Online for Ad-hoc reporting
- Batch for scheduled reporting

3.1 Install FOP In ORMB

The ORMB Reporting configuration file is an EAR file, which contains EJB Code that used for controlling and generating Reporting behavior. For further information, refer *ORMB Reporting Installation Guide*.

3.2 PDF Generation Using Apache FOP

Producing a PDF file from a XML file is a two-step process. First, you need an XSLT stylesheet that converts the **XML** to **XSL-FO**. The created XSL-FO file is also an XML file contains formatted objects. Then, ORMB Reporting framework reads the generated XSL-FO document and formats it to a PDF document.

The high-level PDF generation process comprises of different tasks and flows, each described in detail later in this document.

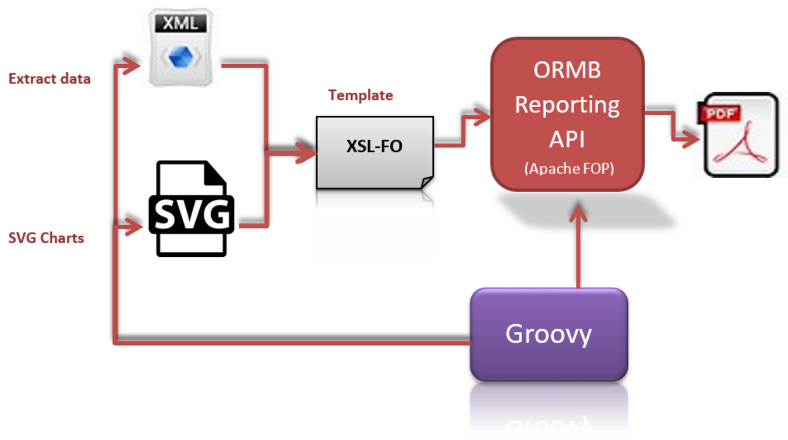


Figure 9: PDF Using Apache-FOP

Note: Currently, ORMB Reporting feature supports PDF generation for BILL and Letter and custom report.

3.2.1 Bill - PDF Generation

The framework has a 'BILL XML' which will get generated by the extract algorithm which holds data and an XSLT that creates an XML containing formatting objects by taking data from the first XML. This resultant XML de-serialized into Java objects. ORMB Reporting creates a PDF file using these Java objects.

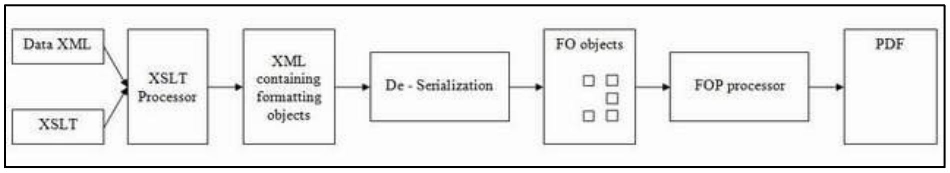


Figure 10: PDF Generation

To configure Bill for PDF generation, you need to do the following:

1. From the **Admin** menu, select **I** and then click **Installation Options – Framework**. The **Installation Options - Framework** screen appears.

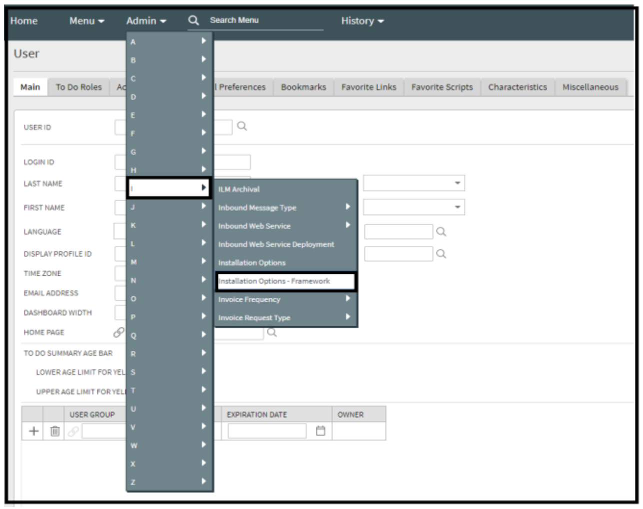
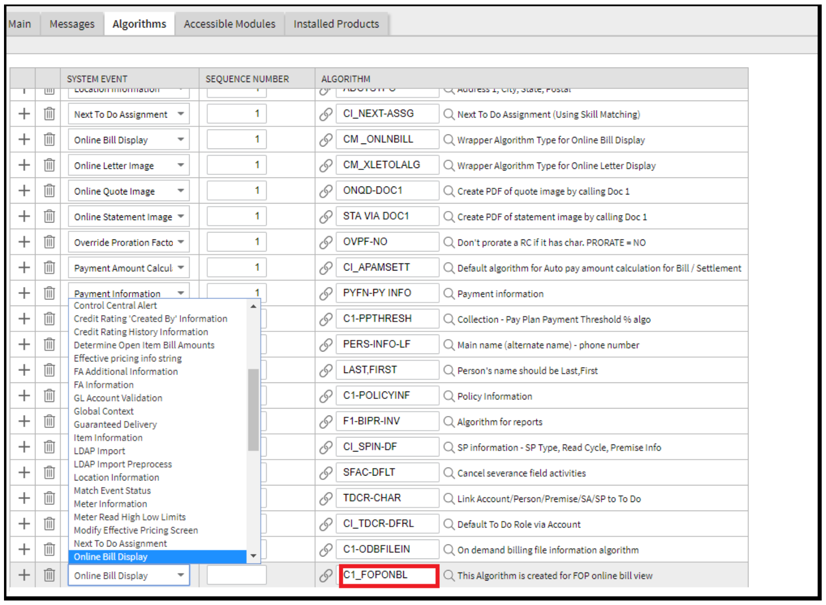


Figure 11: Installation Options - Framework Page

2. Click the **Algorithms** tab.
3. Go to 'Online Bill Display' in System Event and attach **C1_FOPONBL** to the **Online Bill Display** system event.



SYSTEM EVENT	SEQUENCE NUMBER	ALGORITHM	
Next To Do Assignment	1	CI_NEXT-ASSG	Next To Do Assignment (Using Skill Matching)
Online Bill Display	1	CM _ONLNBILL	Wrapper Algorithm Type for Online Bill Display
Online Letter Image	1	CM_XLETOLALG	Wrapper Algorithm Type for Online Letter Display
Online Quote Image	1	ONQD-DOC1	Create PDF of quote image by calling Doc 1
Online Statement Image	1	STA VIA DOC1	Create PDF of statement image by calling Doc 1
Override Proration Facto	1	OVPF-NO	Don't prorate a RC if it has char. PRORATE = NO
Payment Amount Calcul	1	CI_APAMSETT	Default algorithm for Auto pay amount calculation for Bill / Settlement
Payment Information	1	PYFN-PY INFO	Payment information
		C1-PPTHRESH	Collection - Pay Plan Payment Threshold % algo
		PERS-INFO-LF	Main name (alternate name) - phone number
		LAST,FIRST	Person's name should be Last,First
		C1-POLICYINF	Policy Information
		F1-BIPR-INV	Algorithm for reports
		CI_SPIN-DF	SP information - SP Type, Read Cycle, Premise Info
		SFAC-DFLT	Cancel severance field activities
		TDCR-CHAR	Link Account/Person/Premise/SA/SP to To Do
		CI_TDCR-DFRL	Default To Do Role via Account
		C1-ODBFILEIN	On demand billing file information algorithm
Online Bill Display		C1_FOPONBL	This Algorithm is created for FOP online bill view

Figure 12: Algorithms Page

C1-FOPONBL is the algorithm code for the algorithm used to create an online bill view. The below table gives an overview of this algorithm:

Field	**Description**
Algorithm Code	C1_ FOPONBL
Description	This Algorithm created for FOP online bill view.
Algorithm Type	C1_FOPONBL
Algorithm Type Description	This Algorithm created to view FOP online bill. It identifies respective algorithms based on the bill route type configuration corresponding to respective bills and calls FOP code to generate PDF files .To generate PDF file it will use FOP XSL template which is uploaded from Reporting template upload screen of Admin menu. It will use extract algorithm configured on 'BILL ROUTE TYPE' to generate the extract XML. Finally, by using FOP API, it will generate the PDF file, which will be stored in output folder of reporting directory.

NOTE: Configure Report Definition for the bill as given into 3.2.6

3.2.2 Letter - PDF Generation

To configure Letter templates for PDF generation, you need to do the following:

1. From the **Admin** menu, select **I** and then click **Installation Options – Framework**.

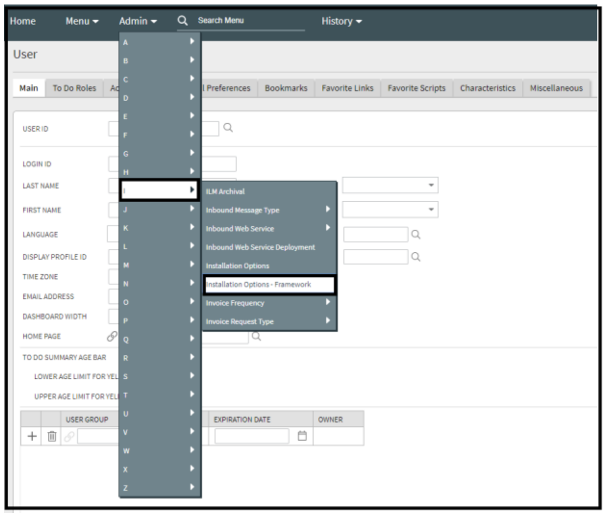
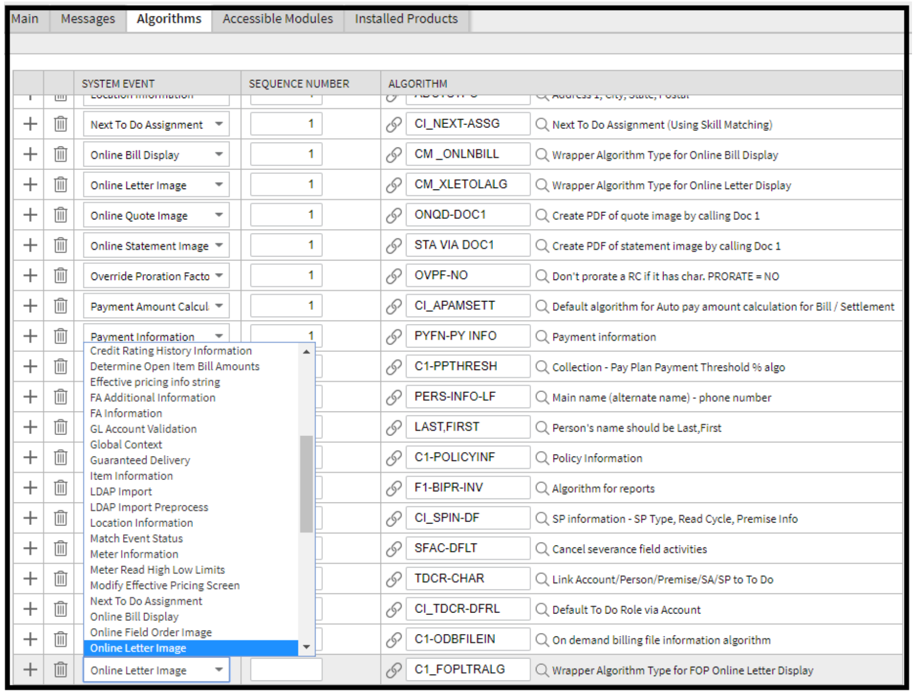


Figure 13: Installation Options - Framework Page

2. The Installation Options - Framework screen appears.
3. Click the **Algorithms** tab.
4. Go to 'Online Letter Image' in System Event and attach **C1_ FOPLTRALG** to the **Online Letter Image** system event.



SYSTEM EVENT	SEQUENCE NUMBER	ALGORITHM	
Next To Do Assignment	1	CI_NEXT-ASSG	Next To Do Assignment (Using Skill Matching)
Online Bill Display	1	CM _ONLNBILL	Wrapper Algorithm Type for Online Bill Display
Online Letter Image	1	CM_XLETOLALG	Wrapper Algorithm Type for Online Letter Display
Online Quote Image	1	ONQD-DOC1	Create PDF of quote image by calling Doc 1
Online Statement Image	1	STA VIA DOC1	Create PDF of statement image by calling Doc 1
Override Proration Facto	1	OVPF-NO	Don't prorate a RC if it has char. PRORATE = NO
Payment Amount Calcul	1	CI_APAMSETT	Default algorithm for Auto pay amount calculation for Bill / Settlement
Payment Information	1	PYFN-PY INFO	Payment information
		C1-PPTHRESH	Collection - Pay Plan Payment Threshold % algo
		PERS-INFO-LF	Main name (alternate name) - phone number
		LAST,FIRST	Person's name should be Last,First
		C1-POLICYINF	Policy Information
		F1-BIPR-INV	Algorithm for reports
		CI_SPIN-DF	SP information - SP Type, Read Cycle, Premise Info
		SFAC-DFLT	Cancel severance field activities
		TDCR-CHAR	Link Account/Person/Premise/SA/SP to To Do
		CI_TDCR-DFRL	Default To Do Role via Account
		C1-ODBFILEIN	On demand billing file information algorithm
Online Letter Image		C1_FOPLTRALG	Wrapper Algorithm Type for FOP Online Letter Display

Figure 14: Algorithms Page

The below table gives an overview of this algorithm:

Field	Description
Algorithm Code	C1_ FOPLTRALG
Description	Wrapper Algorithm Type for FOP Online Letter Display

Algorithm Type	C1_FOPLTRALG
Algorithm Type Description	Wrapper algorithm will internally call letter extract algorithm as per the Letter Template configuration. It will invoke external system interface to send request to external system. It will extract URL from the response and present bill online.

3.2.3 Custom - Report Generation

Custom PDF Report generation feature of product supports the below listed types of reporting:

- CSV Report(CSV)
- DOCX Report(DOCX)
- JSON Report(JSON)
- PDF Report(PDF)
- Web Report(WEB)
- XSL Report(XSL)

For PDF, DOCX and XSL reports, ORMB provides a Business Component which has the required methods that you can call into the Report Processing Algorithm and generate reports. For JSON, XML, CSV you can write your own logic and create your custom report.

Note: You can use product util class in which the above listed extension data can be generated using implemented methods.

Class Name: **CommonUtil_Impl**

- `public void fileWrite(String outputFilePath, String outputFileName, String fileContent, String fileExtension, String charsetName) ;`
- `public String xmlToJSONString(String xmlString)`
- `public String xmlToCSVString(String xmlString)`

To configure Custom PDF generation, follow the procedure below:

1. From the **Admin** menu, select **R** and then click **Report Details**.

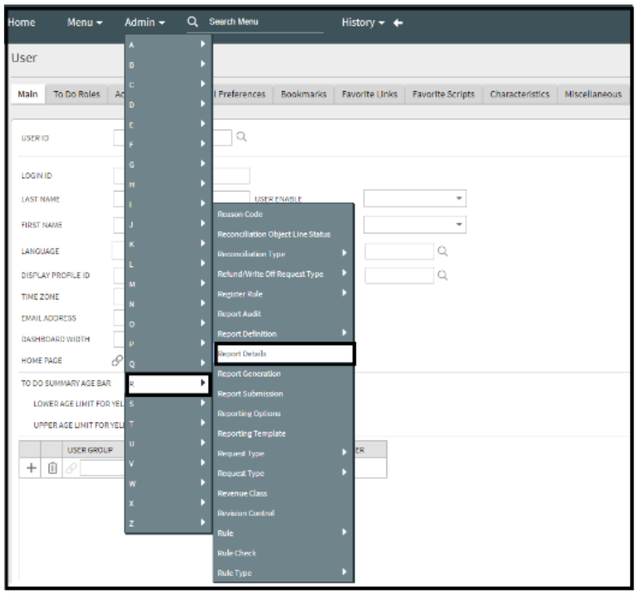


Figure 15: Report Details Page

2. Click **Add** to define a Report Code.

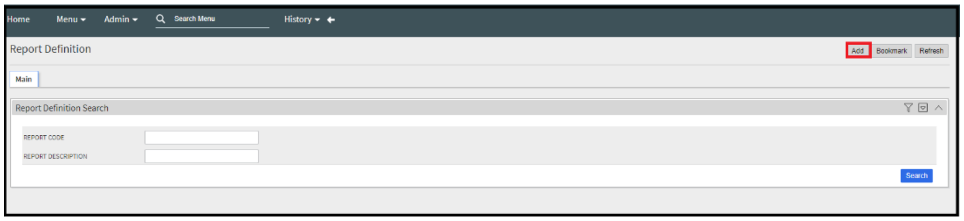
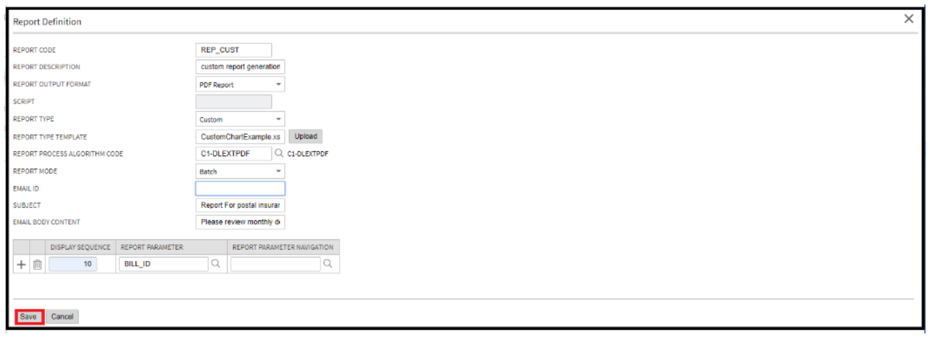


Figure 16: Report Definition Page

3. To Define Custom PDF generation Report Code Select All mandatory fields With **Report Type** as a '**Custom**'.
4. When You Select Report Type as a **Custom** then '**Report Processing Algorithm**' Fields will get enable.


Report Definition

REPORT CODE	REP_CUST
REPORT DESCRIPTION	custom report generation
REPORT OUTPUT FORMAT	PDF Report
SCRIPT	
REPORT TYPE	Custom
REPORT TYPE TEMPLATE	CustomChartExample.xs
REPORT PROCESS ALGORITHM CODE	C1-DLEXTPDF C1-DLEXTPDF
REPORT MODE	Batch
EMAIL ID	
SUBJECT	Report For postal insurar
EMAIL BODY CONTENT	Please review monthly d

DISPLAY SEQUENCE	REPORT PARAMETER	REPORT PARAMETER NAVIGATION
10	BILL_ID	

Figure 17: Report Definition - Save

5. Product has added 'ReportDataExtractAlgorithmSpot' using this implement an algorithm that will create an extract XML at below given location.

   Extract XML Location: <INSTALL_DIR /Reporting/extractxml

   Uploaded XSL(FOP template) path: <INSTALL_DIR /Reporting/xsl

   Created output File location: <INSTALL_DIR /Reporting/output

6. Select newly created Custom algorithm on **Report Processing Algorithm**.
7. Choose all required fields to proceed.

The below table gives an overview of this algorithm spot:

Field	Description
Algorithm Spot Name	ReportDataExtractAlgorithmSpot
Description	This algorithm spot allow you to create custom data extract, which will use to create PDF file. As well as using this spot, you can create XML, DOC, CSV, XLS Report its implementer responsibility to call FOP and POI API to generate respective report.

3.2.4 Develop Reporting XML

A sample XML listed below. This XML contains name and a list of friends with contact numbers.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<root>
<name>Ryan</name>
<friend>
      <name>David</name>
      <phNo>+1.603.555.0100</phNo>
</friend>
<friend>
      <name>John</name>
      <phNo>+1.603.555.0101</phNo>
</friend>
</root>
```

3.2.5 Develop Reporting Template

To produce a PDF file from XML file, follow the below steps:

1. Convert the XML to XSL-FO using an XSLT stylesheet.
2. FOP reads the generated XSL-FO document and formats it to a PDF document.

A sample **xslfo** style sheet is listed below. This first takes a name from the XML. Then this XSLT renders a table that contains details of friends available in the above XML.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.1"

      xmlns:xsl=http://www.w3.org/1999/XSL/Transform
            xmlns:fo="http://www.w3.org/1999/XSL/Format"
      exclude-result-prefixes="fo">
```

```
<xsl:template match="root">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="my-page">
      <fo:region-body margin="1in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="my-page">
    <fo:flow flow-name="xsl-region-body">
      <fo:block>Hello, <xsl:value-of select="name" />!</fo:block>
      <fo:block>
            <fo:table>
                   <fo:table-body>
                        <fo:table-row>
                  <fo:table-cell border="solid 1px black"

                  text-align="center" font-weight="bold">
                        <fo:block>
                              No.
                              </fo:block>
                  </fo:table-cell>
                  <fo:table-cell border="solid 1px black"

                  text-align="center" font-weight="bold">
                        <fo:block>
                              Name
                              </fo:block>
                  </fo:table-cell>
                  <fo:table-cell border="solid 1px black"
                  text-align="center" font-weight="bold">
                        <fo:block>
                              Phone Number
                              </fo:block>
                  </fo:table-cell>
                </fo:table-row>
                <xsl:for-each select="./friend">
```

```
                  <fo:table-row>
                  <fo:table-cell border="solid 1px black" text-align="center">
                        <fo:block>
                              <xsl:value-of select="position()" />
                              </fo:block>
                  </fo:table-cell>
                  <fo:table-cell border="solid 1px black" text-align="center">
                        <fo:block>
                              <xsl:value-of select="name" />
                              </fo:block>
                  </fo:table-cell>
                  <fo:table-cell border="solid 1px black" text-align="center">
                        <fo:block>
                              <xsl:value-of select="phNo" />
                              </fo:block>
                  </fo:table-cell>
                </fo:table-row>
                </xsl:for-each>
                   </fo:table-body>
            </fo:table>
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
</xsl:template>
</xsl:stylesheet>
```

For more details and XSLT stylesheet, refer below links:

https://netjs.blogspot.in/2015/07/how-to-create-pdf-from-xml-using-apache-fop.html

https://xmlgraphics.apache.org/fop/

The following tables list FOP templates shipped with Oracle Revenue Management and Billing application.

1. Bill Template

Sr. No	FOP Template	Description	Path
1	billExtract_Banking_Individual.xsl	FOP Template for Individual Billing	`<Installation Directory>/splapp/reporting/xsl`
2	billExtract_Comm_Banking_List.xsl	FOP Template for List Billing	`<Installation Directory>/splapp/reporting/xsl`

2. Letter Template

Sr. No	FOP Template	Description	Path
1	ExpiringCreditCard_Letter.xsl	FOP Template for Expiring Credit Card Letter	`<Installation Directory>/splapp/reporting/xsl`
2	PastDueExtract_Letter.xsl	FOP Template for Past Due Extract Letter	`<Installation Directory>/splapp/reporting/xsl`
3	NoticeOfIntent_Letter.xsl	FOP Template for Notice Of Intent Letter	`<Installation Directory>/splapp/reporting/xsl`
4	AutoPay_Letter.xsl	FOP Template for Auto Pay Letter	`<Installation Directory>/splapp/reporting/xsl`

3. Custom

Sr. No	FOP Template	Description	Path
1	CustomChartExample.xsl	FOP Template for Custom Report	`<Installation Directory>/splapp/reporting/xsl`
2	CustomReport_Ref.xsl	FOP Template for Custom Report	`<Installation Directory>/splapp/reporting/xsl`

4.

1. Others

Sr. No	FOP Template	Description	Path
1	User_Report.xsl	FOP Template for User Report	`<Installation Directory>/splapp/reporting/xsl`
2	Trial_Bill.xsl	FOP Template for Trial Bill generation	`<Installation Directory>/splapp/reporting/xsl`

3.2.6 Report Definition

ORMB reporting framework allows you to define a report definition where you can configure or define reports. ORMB provides Report definition page where you can configure report type, report generation mode or upload report templates and images for report generation. The ORMB reporting API internally uses this report definition to generate reports as explained at end of this topic.

To define a new Report Code:

1. From the **Admin** menu, select **R** and then click **Reporting Details**. **Report Definition** screen appears.

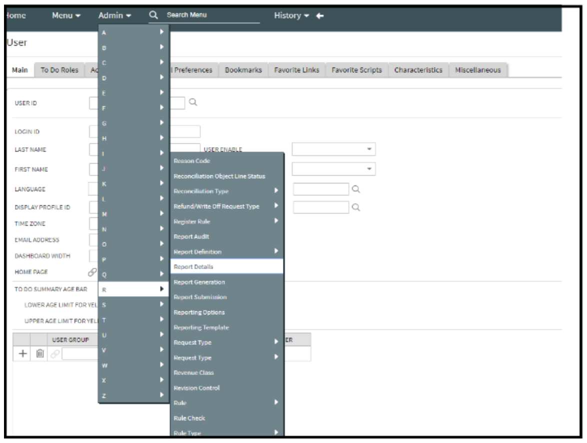


Figure 18: Reporting Template

2. Click on **Add** link present in right Top corner. **Report Definition** page appears.

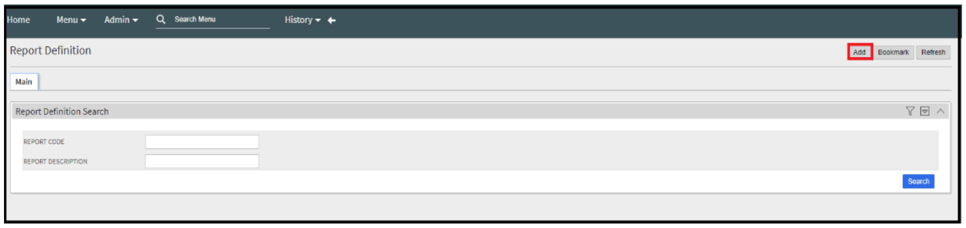
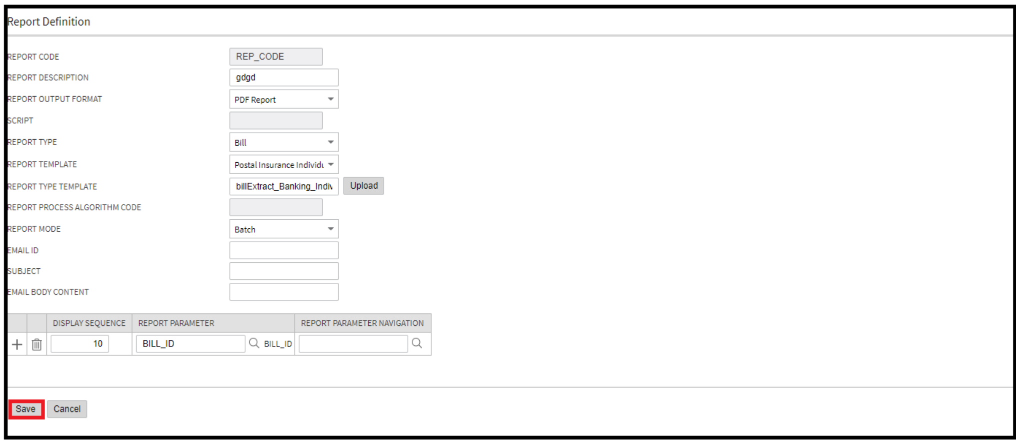


Figure 19: Reporting Template Upload



Report Definition

REPORT CODE	REP_CODE
REPORT DESCRIPTION	gdgd
REPORT OUTPUT FORMAT	PDF Report
SCRIPT	
REPORT TYPE	Bill
REPORT TEMPLATE	Postal Insurance Individu
REPORT TYPE TEMPLATE	billExtract_Banking_Indiv Upload
REPORT PROCESS ALGORITHM CODE	
REPORT MODE	Batch
EMAIL ID	
SUBJECT	
EMAIL BODY CONTENT	

DISPLAY SEQUENCE	REPORT PARAMETER	REPORT PARAMETER NAVIGATION
10	BILL_ID BILL_ID	

Save Cancel

Figure 20: Reporting Template Upload Screen

This screen contains following fields:

Fields	**Description**
Report Code	Used to define Report Code
Report Description	Used to define Report Description
Report Out Put Format	Used to define Report Output Format The values are: • CSV Report(CSV) • DOCX Report(DOCX) • JSON Report(JSON) • PDF Report(PDF) • Web Report(WEB) • XSL Report(XSL) • XML Report(XML)
Script	This field is used only when user selects Report Out Put Format as WEB.
Report Type	Used to define Report Type The values are: • Bill (BILL) • Letter(LTR) • Custom(CUST)
Report Type Template	Used to upload a Template file or images which have been used to design reports
Report Process Algorithm Code	This Field Used to define an custom algorithm for Report generation .This field will enable only if user select ‘Report Type’ as Custom
Report Mode	This field used to define report generation Mode. The values are: • Batch (BTCH) • Online(ONLN)
Email Id	Used to configure Email id (this feature is enabled only in case of Batch mode reporting generation)
Subject	Used to configure Email Subject line
Email Body Content	Used to configure Email Body Message
Sequence	Used to define sequence of parameters E.g. 10, 20, etc.
Report Parameter	Used to define report parameters
Report Parameter Navigation	Used to define Parameters navigation Option

Note: If you want to display any bill or letter in PDF format, you need to do all necessary configuration changes. Once you perform configuration changes, for each Bill Route Type Code or Letter Template Code available in the system, you need to upload an .xsl file (FOP template).

3.2.7 Develop Custom Algorithm

To generate custom reports with supported extension except pdf we need to write custom algorithm using this custom algorithm we can generate below types of reports.

- CSV Report(CSV)
- DOCX Report(DOCX)
- JSON Report(JSON)
- PDF Report(PDF)
- Web Report(WEB)
- XSL Report(XSL)
- XML Report(XML)

Using ORMB FOP Reporting, below algorithm spot has been provided with product to develop above types of reports

Algorithm Spot Name: ReportDataExtractAlgorithmSpot

Algorithm Entity Name: Report-Data Extract Algorithm (reportDataExtractor)

Business Component:: POIReportGenerationComponent_Impl

Custom Report API defined as in 'POIReportGenerationComponent' Business Component:

public void getPOIMSWordReport(MSWordReportDTO wordDto, String fileName)

public void getPOIExcelReport(String filename, Sheet sheetDto)

Custom algorithm will be used to extract XML, retrieve XSL template and generate report of all above given type. If customization team wants to change existing or add new Bill data or Letter data, they need to develop a custom algorithm on the above-mentioned respective algorithm spots.

You can write an algorithm on above given algorithm Spot.

Note: Ensure that your XML is generated only on the <application installation directory>/splapp/reporting/extractxml/file_name.xml path .

3.2.8 ORMB Reporting API

You must use ORMB provided business component 'FOPReportGenerationComponent_Impl' to call reporting API, which generates FOP reports.

The API can be defined as:

```
public String getFOPPdf(String trialBillId, String xslFIlePath, String pdfFilePath, String billExtractXmlPath, String outputHttpUrl)
```

This API has following parameters:

Name	Description

trialBillId	Bill or Letter Id used to name the PDF
xslFIlePath	Defines FOP XSL file path. The path should be `<application installation directory>/ splapp/reporting/xsl /<BILL_ROUTY TYPE_CODE/LETTER_TEMPLATE_CODE>.xsl`
pdfFilePath	This path value must be fully qualified path of PDF generation `<application installation directory>/ splapp/reporting/output/<any name/letter_id/bill_id>.pdf`
billExtractXmlPath	This path value must be fully qualified path of XML generated by algorithm `<application installation directory>/ splapp/reporting/extractxml/<generated xml name>.xml`
outputHttpUrl	The valid values are: • application_URL/billView • null

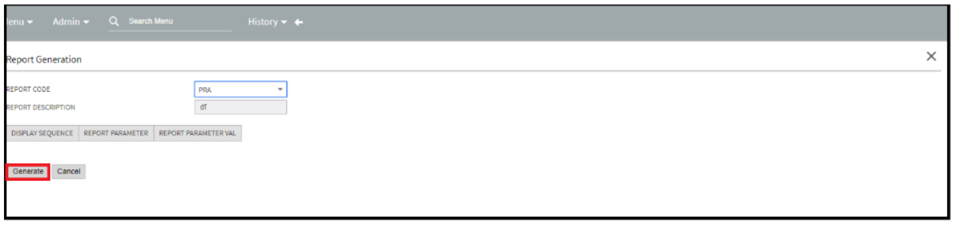
The Pseudo Code to call getFOPPdf () component method used to generate PDF file:

```
FOPReportGenerationComponent_Impl fopReportGenerationComponent = new
FOPReportGenerationComponent_Impl ();
fopReportGenerationComponent.getFOPPdf(billId, xslFIlePath,
pdfFilePath, billExtractXmlPath, outputHttpUrl);
```

3.2.9 ORMB Adhoc Report Generation

To generate adhoc reports, ORMBA introduces a new feature by which you can generate a report anytime.

1. From the **Admin** menu, select **R** and then click **Report Generation Menu**. Report Generation screen appears.
2. Select the report code of the report you wanted to create.



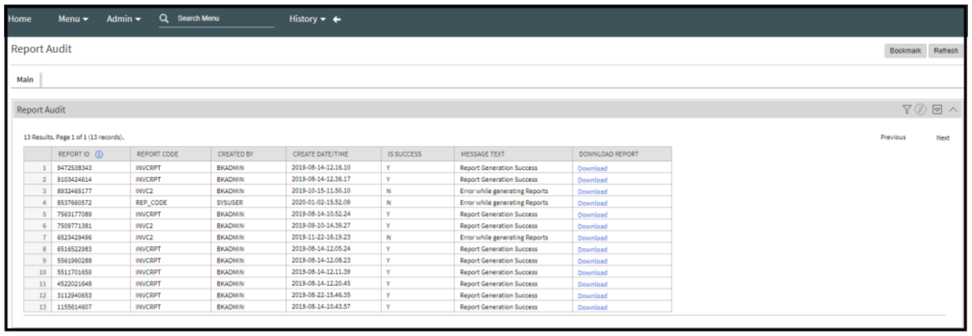
REPORT CODE: PRA
REPORT DESCRIPTION: df

Figure 21: Reporting Generation Screen

3. Provide report parameter value for which you want to generate PDF.
4. Click on **Generate button.** This generates the report and you can download it from the **Report Audit** screen.

3.2.10 ORMB Report Auditing

You can track the record of generated reports through this feature of ORMB. By using this feature of reporting, you can download the report as well as we can check the error of the specific report code record.



13 Results. Page 1 of 1 (13 records).

	REPORT ID	REPORT CODE	CREATED BY	CREATE DATE/TIME	IS SUCCESS	MESSAGE TEXT	DOWNLOAD REPORT
1	9472538343	INVCRPT	BKADMIN	2019-08-14-12.16.10	Y	Report Generation Success	Download
2	9103424614	INVCRPT	BKADMIN	2019-08-14-12.36.17	Y	Report Generation Success	Download
3	8932465177	INVC2	BKADMIN	2019-10-15-11.50.10	N	Error while generating Reports	Download
4	8537660572	REP_CODE	SYSUSER	2020-01-02-15.52.09	N	Error while generating Reports	Download
5	7563177089	INVCRPT	BKADMIN	2019-08-14-10.52.24	Y	Report Generation Success	Download
6	7509771381	INVC2	BKADMIN	2019-09-10-14.39.27	Y	Report Generation Success	Download
7	6523429496	INVC2	BKADMIN	2019-11-22-16.19.23	N	Error while generating Reports	Download
8	6516522983	INVCRPT	BKADMIN	2019-08-14-12.05.24	Y	Report Generation Success	Download
9	5561960288	INVCRPT	BKADMIN	2019-08-14-12.08.23	Y	Report Generation Success	Download
10	5511701659	INVCRPT	BKADMIN	2019-08-14-12.11.39	Y	Report Generation Success	Download
11	4522021648	INVCRPT	BKADMIN	2019-08-14-12.20.45	Y	Report Generation Success	Download
12	3112940653	INVCRPT	BKADMIN	2019-08-22-15.46.35	Y	Report Generation Success	Download
13	1155614607	INVCRPT	BKADMIN	2019-08-14-10.43.57	Y	Report Generation Success	Download

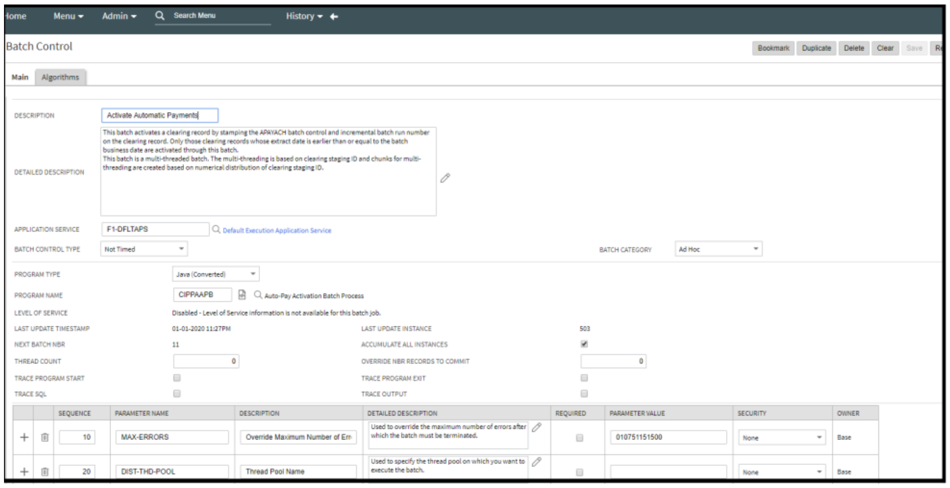
Figure 22: Report Auditing

3.2.11 ORMB Report Generation Batch

Product provide a batch, which will generate the bulk report as defined into Report definition. This batch accept the report query in which user can select the record, which is defined into the report code.

Fields	Description
Batch Code	C1-RPTGN
Description	Report Generation Batch
Detailed Description	Report Generation Batch
Program Name	com.splwg.ccb.domain.reportsubmission.ReportGenerationBatch

Batch Parameters	Description
reportQuery	This parameter of batch accept the select query without semi colon,which will select Report code defined batch parameters
reportCode	This parameter of batch accept the predefined report code on which respective reports will get generated.



Batch Control

DESCRIPTION	Activate Automatic Payments
DETAILED DESCRIPTION	This batch activates a clearing record by stamping the APAYACH batch control and incremental batch run number on the clearing record. Only those clearing records whose extract date is earlier than or equal to the batch business date are activated through this batch. This batch is a multi-threaded batch. The multi-threading is based on clearing staging ID and chunks for multi-threading are created based on numerical distribution of clearing staging ID.
APPLICATION SERVICE	F1-DFLTAPS Default Execution Application Service
BATCH CONTROL TYPE	Not Timed
BATCH CATEGORY	Ad Hoc
PROGRAM TYPE	Java (Converted)
PROGRAM NAME	CIPPAAPB Auto-Pay Activation Batch Process
LEVEL OF SERVICE	Disabled - Level of Service information is not available for this batch job.

LAST UPDATE TIMESTAMP	01-01-2020 11:27PM	LAST UPDATE INSTANCE	503
NEXT BATCH NBR	11	ACCUMULATE ALL INSTANCES	✓
THREAD COUNT	0	OVERRIDE NBR RECORDS TO COMMIT	0
TRACE PROGRAM START	☐	TRACE PROGRAM EXIT	☐
TRACE SQL	☐	TRACE OUTPUT	☐

		SEQUENCE	PARAMETER NAME	DESCRIPTION	DETAILED DESCRIPTION	REQUIRED	PARAMETER VALUE	SECURITY	OWNER
+	🗑	10	MAX-ERRORS	Override Maximum Number of Err	Used to override the maximum number of errors after which the batch must be terminated.	☐	010751151500	None	Base
+	🗑	20	DIST-THD-POOL	Thread Pool Name	Used to specify the thread pool on which you want to execute the batch.	☐		None	Base

Figure 23: Batch Control Screen

3.2.12 ORMB Reporting Using Groovy

ORMB FOP framework allows you generate reports in Apache Groovy. You can use ORMB FOP framework to edit existing bill and letter template or to create new PDF report.

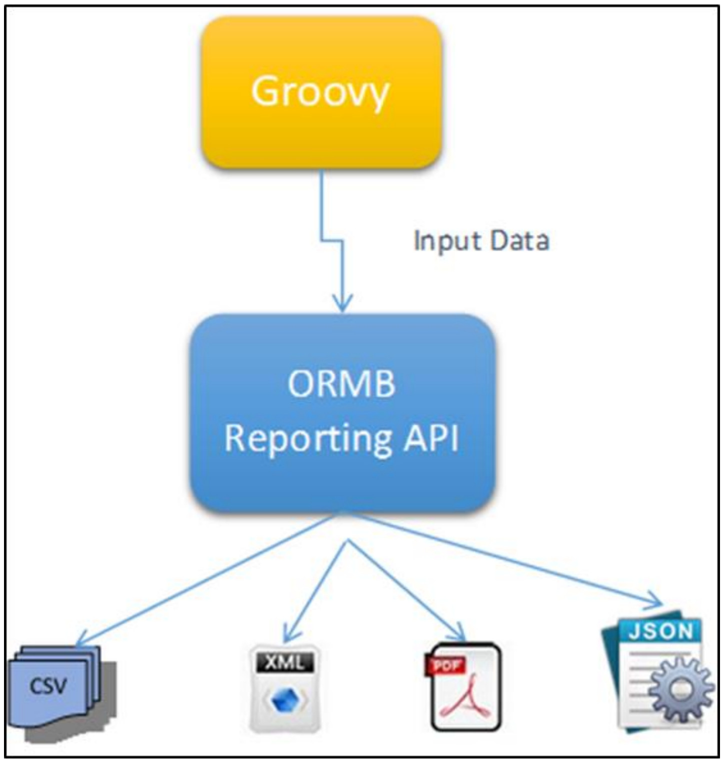


Figure 24: Reporting using Groovy

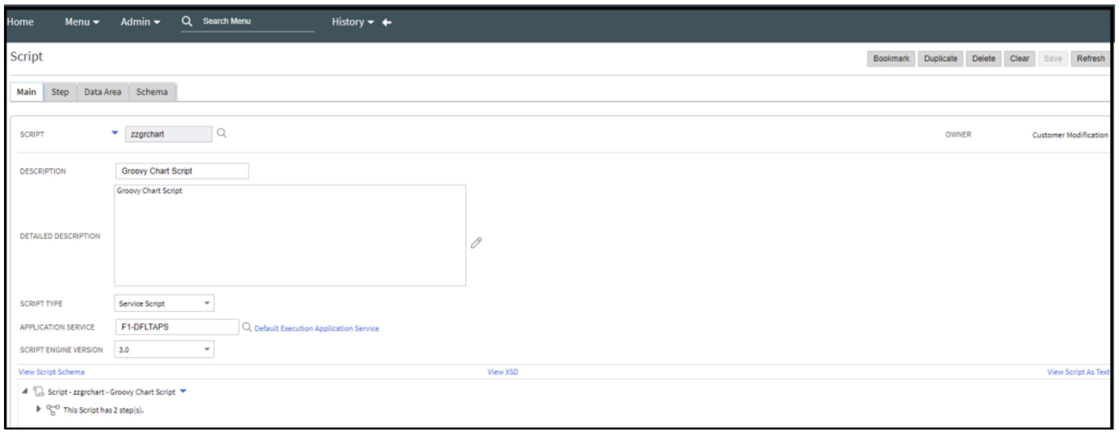
This section lists and describes the following types of reports which can be generated using Groovy

1. Report Generation On Person Context Menu
2. Custom Letter Using SS and Groovy
3. New Letter Template Creation Using Groovy Plugin Script
4. New Letter Using Custom Business Service

3.2.12.1 Report Generation on Person Context Menu

Generating Reports on Person Context Menu involves below steps:

1. **Creating Service Script** - It holds Groovy script. The Groovy Script gathers required report data and creates XML and FOP XSLT, which is used by Apache FOP to generate reports. To create new service script:
    a. From the **Admin** menu, select **S** and then click **Script**. A sub-menu appears.
    b. Click **Add** option from the Script sub-menu.
    c. Enter the details in respective fields.
    d. Select Service Script from the **Script Type** drop-down list.

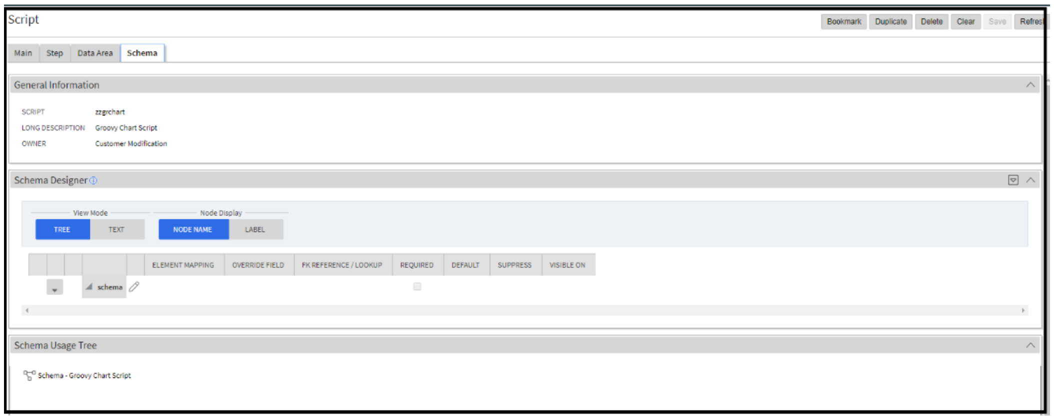


SCRIPT	zzgrchart
OWNER	Customer Modification
DESCRIPTION	Groovy Chart Script
DETAILED DESCRIPTION	Groovy Chart Script
SCRIPT TYPE	Service Script
APPLICATION SERVICE	F1-DFLTAPS Default Execution Application Service
SCRIPT ENGINE VERSION	3.0

Script - zzgrchart - Groovy Chart Script
This Script has 2 step(s).

Figure 25: Script – Main

2. **Define Schema for Service Script – Schema** defines the data elements passed to and from a service script. The purpose of the schema is to describe the input and output parameters used when invoking the script.



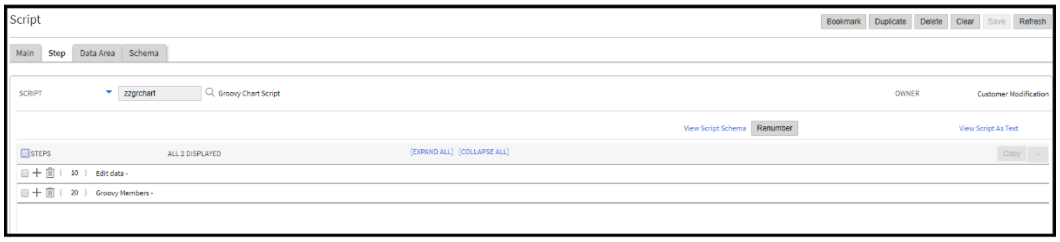
SCRIPT	zzgrchart
LONG DESCRIPTION	Groovy Chart Script
OWNER	Customer Modification

	ELEMENT MAPPING	OVERRIDE FIELD	FK REFERENCE / LOOKUP	REQUIRED	DEFAULT	SUPPRESS	VISIBLE ON
schema				☐			

Schema - Groovy Chart Script

Figure 26: Script - Schema

3. **Define Script Steps** - Steps define logic of the service script. You can use different types of steps to move data, call other scripts, execute script code or invoke business services, etc. This process involves two sub processes:
    a. **Invoking a groovy function** – To invoke a groovy function, you need to define a Step. For more information, refer to Defining a Step section.. Note that you can invoke function with no parameters.



SCRIPT zzgrchart Groovy Chart Script OWNER Customer Modification

STEPS ALL 2 DISPLAYED

(10) Edit data -

(20) Groovy Members -

Figure 27: Script – Groovy Function

    b. **Writing a groovy code** – This code performs following steps:
        - Gathers report data and generates XML file
        - Calls API, which will use XML and FOP XSLT to generate report in PDF
        - Terminates the Service Script
        - Saves the Service Script

A sample code for Groovy Script is listed below:

```
public void invoke() {
            logger.info(" ----------------------    In the DD LetterExtractAlgo
---------------------------------  ");
            def perId= evalString("string(parm/personId)");
            logger.info("Person Id:"+perId);
            String processDate = "";
            String ownerName = "";
            String ownerAddress1 = "";
            String ownerAddress2 = "";
            String ownerAddress3 = "";
            String ownerCity = "";
            String ownerPostal = "";
            String ownerState = "";
            String acctId = null;
            String emailId = null;
            String phone = null;
            java.lang.StringBuilder PIs = new java.lang.StringBuilder();
            String services = null;
            def pstmt = null;
            def row = null;
            def pstmt1 = null;
            def row1 = null;
            def pstmt2 = null;
            def rowlist = null;

            try {
                  pstmt = createPreparedStatement(
                              "select        trim(ACP.ACCT_ID)        ACCT_ID
,trim(PR.EMAILID) EMAIL,trim(PH.PHONE_TYPE_CD) || ':'||trim(PH.PHONE) PHONE,"+
                              "PR.ADDRESS1                ADDRESS1,PR.ADDRESS2
ADDRESS2,PR.ADDRESS3    ADDRESS3,PR.CITY    CITY,    PR.STATE    STATE,PR.POSTAL
POSTAL,TO_CHAR(SYSDATE,'DD/MON/YYYY') CC_DTTM " +
                              "FROM CI_PER PR, CI_PER_PHONE PH, CI_ACCT_PER ACP
" +
                              "WHERE PR.PER_ID= :perId " +
                              "AND PR.PER_ID=PH.PER_ID " +
                              "AND PR.PER_ID=ACP.PER_ID", "Person Info");
                  pstmt.bindString("perId", perId, "PER_ID");
                  row = pstmt.firstRow();
                  if (row != null) {
                        acctId = row.getString("ACCT_ID");
                        emailId = row.getString("EMAIL");
                        phone = row.getString("PHONE");
                        ownerAddress1 = row.getString("ADDRESS1");
                        ownerAddress2 = row.getString("ADDRESS2");
                        ownerAddress3 = row.getString("ADDRESS3");
                        ownerCity = row.getString("CITY");
                        ownerState = row.getString("STATE");
                        ownerPostal = row.getString("POSTAL");
                        processDate = row.getString("CC_DTTM");
                  }
                  pstmt1 = createPreparedStatement(
                              "select PN.ENTITY_NAME ENTITY_NAME " +
                              "from CI_PER PR, CI_PER_NAME PN " +
                              "WHERE PR.PER_ID=:perId " +
                              "AND PR.PER_ID=PN.PER_ID","Person Name");
                  pstmt1.bindString("perId", perId, "PER_ID");
                  row1 = pstmt1.firstRow();
                  if (row1 != null) {
                        ownerName = row1.getString("ENTITY_NAME");
                  }
                  pstmt2           =           createPreparedStatement("SELECT
DISTINCT(TRIM(PRICEITEM_CD)) PRICEITEM_CD " +
                              "FROM  CI_PARTY  PRTY,  CI_PRICELIST_ASGN  PLAS,
CI_PRICEASGN PRAS " +
                              "WHERE PRTY.PARTY_ID=:party_id " +
                              "AND PRTY.PARTY_TYPE_FLG='ACCT' " +
                              "AND PRTY.PARTY_UID=PLAS.PARTY_UID " +
                              "AND  PLAS.PRICELIST_ID=PRAS.OWNER_ID",  "Person
Services");

                  pstmt2.bindString("party_id", acctId, "PARTY_ID");
                  rowlist = pstmt2.list();
                  for (def row2 : rowlist) {
                        PIs.append(row2.getString("PRICEITEM_CD"));
                        PIs.append(",");
                  }
                  services = PIs.toString();
            } catch (Exception e) {
                  logger.error("Exception ", e);
            } finally {
                  try {
                        pstmt1.close();
                        pstmt2.close();
                  } catch (Exception e) {
                        logger.error("Error While Closing ResultSet ", e);
                  } finally {
                        pstmt1 = null;
                        pstmt2 = null;
                  }
            }
            logger.info("\n Account Id:" + acctId);
            logger.info("\n Email Id:" + emailId);
            logger.info("\n Phone:" + phone);
            logger.info("\n Services:" + services);
            logger.info("\n Name:" + ownerName);
            logger.info("\n Process Date:" + processDate);
            logger.info("\n Add 1:" + ownerAddress1);
            logger.info("\n Add 2:" + ownerAddress2);
            logger.info("\n Add 3:" + ownerAddress3);
            logger.info("\n City:" + ownerCity);
            logger.info("\n State:" + ownerState);
            logger.info("\n Postal:" + ownerPostal);

            /// Create XML Of Fields

            org.w3c.dom.Document                finalDocument                 =
com.splwg.base.support.scripting.XMLUtils.createNewDomDocument();
            org.w3c.dom.Element                         root                  =
finalDocument.createElement("DOCUMENT");
            finalDocument.appendChild(root);
            org.w3c.dom.Element                         root1                 =
com.splwg.base.support.scripting.XMLUtils.addElement(root, "DOCSET");
            org.w3c.dom.Element                   itemElement                 =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
            itemElement.setAttribute("NAME", "ProcessDt");
            itemElement.setTextContent(processDate);
            org.w3c.dom.Element                  itemElement1                 =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
            itemElement1.setAttribute("NAME", "OwnerFullNm");
            itemElement1.setTextContent(ownerName);
            org.w3c.dom.Element                  itemElement2                 =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
            itemElement2.setAttribute("NAME", "OwnerAddrLine1");
            itemElement2.setTextContent(ownerAddress1);
            org.w3c.dom.Element                  itemElement3                 =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
            itemElement3.setAttribute("NAME", "OwnerAddrLine2");
            itemElement3.setTextContent(ownerAddress2+","+ownerAddress3);
            org.w3c.dom.Element                  itemElement4                 =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
            itemElement4.setAttribute("NAME", "OwnerAddrLine3");
      itemElement4.setTextContent(ownerCity+","+ownerState+","+ownerPostal);
            org.w3c.dom.Element                  itemElement5                 =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
            itemElement5.setAttribute("NAME", "accountId");
            itemElement5.setTextContent(acctId);
            org.w3c.dom.Element                 itemElement6           =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
            itemElement6.setAttribute("NAME", "emailAddress");
            itemElement6.setTextContent(emailId);


            org.w3c.dom.Element                 itemElement7           =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
            itemElement7.setAttribute("NAME", "phone");
            itemElement7.setTextContent(phone);
            org.w3c.dom.Element                 itemElement8           =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
            itemElement8.setAttribute("NAME", "services");
            itemElement8.setTextContent(services);
            String                  letterExtractXMLFileLoc            =
"/scratch/rmbbuild/spl/DD26010/splapp/reporting/extractxml/" + perId + ".xml";
            String sourceXML = "";
            try {
      com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil
commonUtil
=com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil.Factory.new
Instance();
                  sourceXML = finalDocument.documentElement as String;

      commonUtil.fileWrite("/scratch/rmbbuild/spl/DD26010/splapp/reporting/ex
tractxml/", perId, sourceXML, "xml");
      com.splwg.ccb.domain.billing.billtransform.FOPReportGenerationComponent
fopRptComp                                                                  =
com.splwg.ccb.domain.billing.billtransform.FOPReportGenerationComponent.Facto
ry.newInstance();
                  String                     xslFileLoc                    =
"/scratch/rmbbuild/spl/DD26010/splapp/reporting/xsl/DD-WELCOME.xsl";
                  String                     pdfFilePath                   =
"/scratch/rmbbuild/spl/DD26010/splapp/reporting/output";
                  String pdfFileLoc = pdfFilePath + "/" + perId + ".pdf";
                  String   url   =   fopRptComp.getFOPPdf(perId,   xslFileLoc,
pdfFileLoc, letterExtractXMLFileLoc, " ");
            } catch (Exception e) {
                  logger.error("Exception occurred" + e.getMessage());
            }
      }
```

Groovy Plugin Script collects all the required data and creates XML. A sample code for creating Test or Sample XML or Letter Extract XML is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>

<DOCUMENT>

<DOCSET>

<FIELD NAME="ProcessDt">20/NOV/2017</FIELD>

<FIELD NAME="OwnerFullNm">Smith, Michael</FIELD>

<FIELD NAME="OwnerAddrLine1">2-4-27 Dojima, Kita-ku/FIELD>

<FIELD NAME="OwnerAddrLine2">Osaka 530-0003</FIELD>

<FIELD NAME="OwnerAddrLine3">Japan</FIELD>

<FIELD NAME="accountId">2000000000</FIELD>

<FIELD NAME="emailAddress">michael.smith@green.com</FIELD>

<FIELD NAME="phone">BUS: +1.603.555.0100</FIELD>

<FIELD
NAME="services">PERFDISCOUNT,STATETAX,PERFFLAT,PERFTHRESH,PERFSTEP,</FIELD>

</DOCSET>

</DOCUMENT>
```

APACHE FOP XSLT is required by FOP to generate PDF. It uses XML created by Groovy Script. A sample code for APACHE FOP XSLT is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet                                                   version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">

      <xsl:template match="/">

            <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

                  <fo:layout-master-set>

                        <fo:simple-page-master master-name="simpleA4"

                               page-height="29.7cm"

                  page-width="21cm"

                  margin-top="1cm"

                  margin-bottom="0.1cm"

                  margin-left="0.8cm"

                  margin-right="1.0cm" >

                              <fo:region-body    margin-top="2.5cm"    margin-
bottom="2.5cm"/>

                              <fo:region-before extent="2.0cm"/>

                              <fo:region-after extent="2.0cm"/>

                        </fo:simple-page-master>
                  </fo:layout-master-set>
                  <fo:page-sequence master-reference="simpleA4"
                initial-page-number="1">
                        <fo:static-content flow-name="xsl-region-before">
                              <fo:block font-size="7pt">
                              </fo:block>
                        </fo:static-content>

                        <fo:static-content flow-name="xsl-region-after">
                              <fo:block  font-size="9.0pt"  font-family="sans-
serif"
                        padding-after="2.0pt"    space-before="2.0pt"    text-
align="right"
                        border-top-style="ridge" border-bottom-width="0.5pt">
                              </fo:block>
                        </fo:static-content>
                        <fo:flow flow-name="xsl-region-body">
                              <xsl:apply-templates select="DOCUMENT/DOCSET" />
                              <fo:block id="last-page"> </fo:block>
                        </fo:flow>
                  </fo:page-sequence>
            </fo:root>
      </xsl:template>

      <xsl:template match="DOCUMENT/DOCSET">
            <fo:block text-align="right" font-size="9px">
                  <xsl:value-of select="//FIELD[@NAME='ProcessDt']"/>
            </fo:block>

            <fo:block text-align="right" font-size="9px" space-before="0.37in"
>Vision Corporation</fo:block>
            <fo:block text-align="right" font-size="9px" >401 Island Parkway,
Redwood Shores, CA 94065</fo:block>

            <fo:block text-align="left" font-size="9px">
                  <xsl:value-of select="//FIELD[@NAME='OwnerFullNm']"/>
            </fo:block>
            <fo:block text-align="left" font-size="9px">
                  <xsl:value-of select="//FIELD[@NAME='OwnerAddrLine1']"/>
            </fo:block>
            <fo:block text-align="left" font-size="9px">
                  <xsl:value-of select="//FIELD[@NAME='OwnerAddrLine2']"/>
            </fo:block>
            <fo:block text-align="left" font-size="9px">
                  <xsl:value-of select="//FIELD[@NAME='OwnerAddrLine3']"/>
            </fo:block>
            <fo:block      text-align="left"      font-size="9px"      space-
before="0.37in">Re: Welcome</fo:block>
            <fo:block      text-align="left"      font-size="9px"      space-
before="0.10in">Customer                                 Account:<xsl:value-of
select="//FIELD[@NAME='accountId']"/>
            </fo:block>
            <fo:block      text-align="left"      font-size="9px"      space-
before="0.37in">Dear                                             <xsl:value-of
select="//FIELD[@NAME='OwnerFullNm']"/>,</fo:block>


            <fo:block      text-align="left"      font-size="9px"      space-
before="0.10in">Thank you for Choosing Vision Products. We welcome you to our
customer loyalty program.
            </fo:block>
            <fo:block      text-align="left"      font-size="9px"      space-
before="0.10in">You are currently subscribed to below services.</fo:block>
            <fo:block text-align="left" font-size="9px" >Services:<xsl:value-
of select="//FIELD[@NAME='services']"/>
            </fo:block>
            <fo:block      text-align="left"      font-size="9px"      space-
before="0.10in">Your contact details with us are as below:</fo:block>
            <fo:block  text-align="left"  font-size="9px"  >Email:<xsl:value-of
select="//FIELD[@NAME='emailAddress']"/>
            </fo:block>
            <fo:block  text-align="left"  font-size="9px"  >Phone:<xsl:value-of
select="//FIELD[@NAME='phone']"/>
            </fo:block>
            <fo:block      text-align="left"      font-size="9px"      space-
before="0.10in">If you have any questions about our customer policy or loyalty
program, please call us at +1.650.555.0185. We appreciate your business and are
here to help you with all your business needs.</fo:block>
            <fo:block      text-align="left"      font-size="9px"      space-
before="0.37in">Sincerely,</fo:block>
            <fo:block   text-align="left"   font-size="9px">Vision   Customer
Care.</fo:block>
      </xsl:template>
      <!--Template -->
      <xsl:attribute-set name="foCellAttributeSet">
            <xsl:attribute name="padding-right">2mm</xsl:attribute>
            <xsl:attribute name="padding-left">2mm</xsl:attribute>
            <xsl:attribute name="padding-top">2mm</xsl:attribute>
            <xsl:attribute name="padding-bottom">2mm</xsl:attribute>
      </xsl:attribute-set>
</xsl:stylesheet>
```

4. **Create UI Map** - The User Interface (UI) map holds HTML to be rendered within portal zones and Business Process Assistant (BPA) scripts. UI maps allow you to create input forms and output maps that closely match your customer's business practices. In other words, the UI Map is designed to facilitate the capture and display of your business objects and business services. UI map will call Service Script created above and pass necessary input data. To create UI Map, follow the steps below:
    a. From the **Admin** menu, select **U** and then click **UI Map**. A sub-menu appears

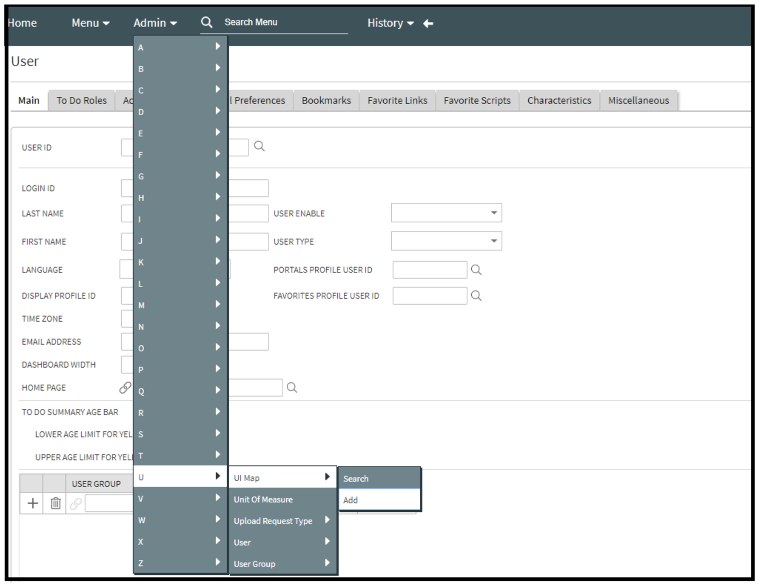
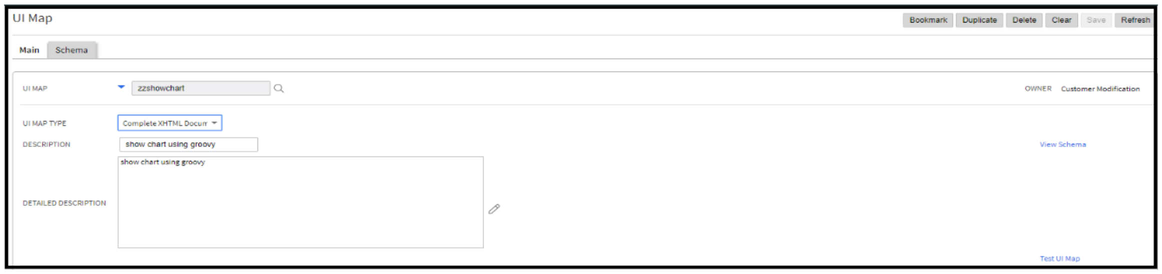


Figure 28: UI Map

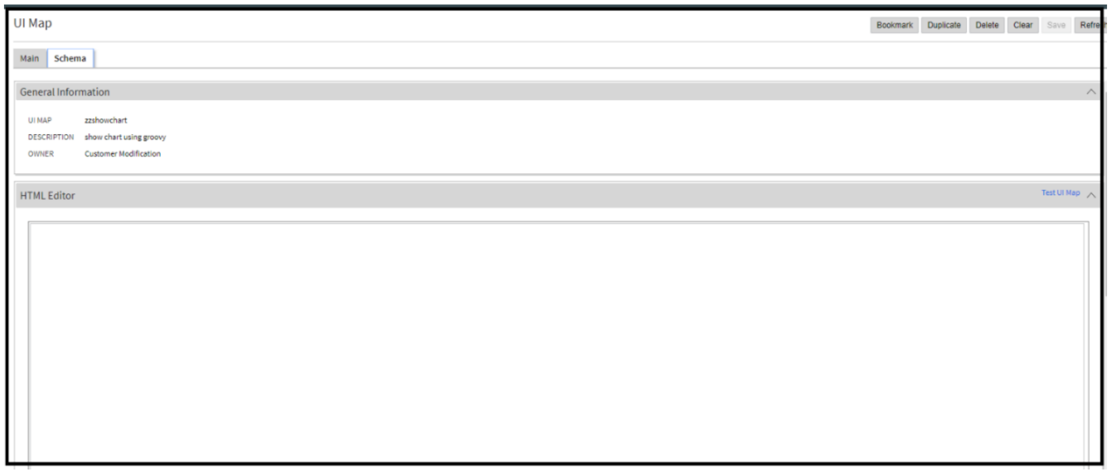
    b. Click **Add** option from the **UI Map** sub-menu.



UI MAP: zzshowchart
OWNER: Customer Modification
UI MAP TYPE: Complete XHTML Docum
DESCRIPTION: show chart using groovy
DETAILED DESCRIPTION: show chart using groovy

Figure 29: Creating UI Map

The Schema for UI map will refer to Service Script.



UI MAP zzshowchart
DESCRIPTION show chart using groovy
OWNER Customer Modification

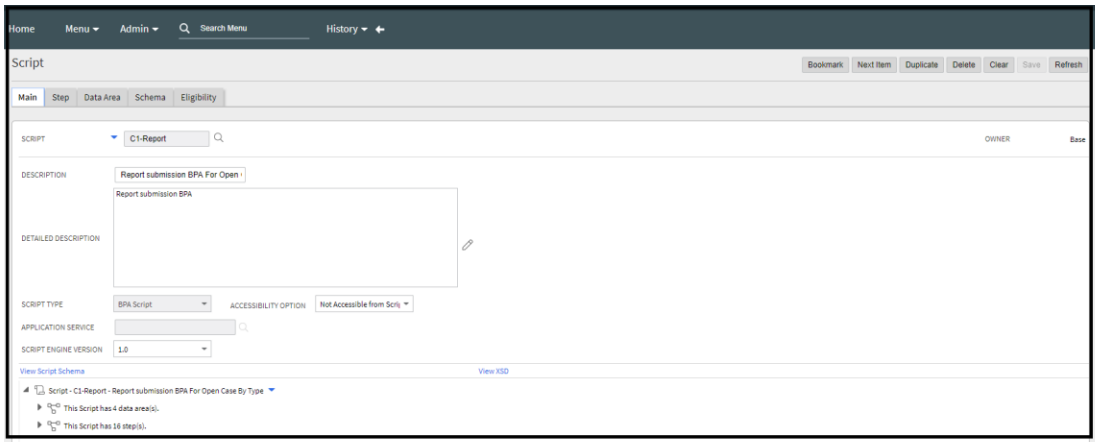
Figure 30: HTML Editor

A sample HTML code in UI Map is listed below:

```
<html
xmlns:web="http://xml.apache.org/xalan/java/com.splwg.base.web.common"
>
    <head>
        <META       http-equiv="Content-Type"       content="text/html;
charset=UTF-8">
        <title></title>
        <link href="cisDisabled.css" type="text/css" rel="stylesheet">
         <script>
              function fnSubmit()
              {
                        oraInvokeSS('ZZTESTLTRSS',null);
                        var                  vpersonId                 =
document.getElementById("personId").value;
                        var                     url                    =
location.protocol+"//"+location.hostname+":"+location.port+"/ouaf/bill
View/"+vpersonId+".pdf";
                        window.open(url,"_blank",
"toolbar=yes,scrollbars=yes,resizable=yes,top=50,left=50,width=600,hei
ght=600");
                        close();
              }
         </script>
    </head>
    <body           oraError="automate:true;           prefix:boGroup"
onload="fnSubmit();">
```

```
        <table role="presentation" oraMdLabel="EMPTY_SUMMARY" style="padding:12px;width:100%;">

            <tr class="oraErrorText">

                <td><a onclick="oraShowErrorAlert(); return false;" href=""><span oraErrorVar="ERRMSG-TEXT" class="oraErrorText" id="ERRMSG-TEXT-SPAN" aria-describedBy="ERRMSG-TEXT-SPAN" role="alert"></span></a></td>

            </tr>

        </table>

        <table role="presentation" oraMdLabel="EMPTY_SUMMARY" style="border-spacing:4px;width:100%;">

            <tr>

               <td class="oraNormal oraTableData"><input id="personId" class="oraInput" oraField="personId" type="hidden"></td>

            </tr>

        </table>

    </body>

    <xml style="display:none;"></xml>

</html>
```

5. **Create BPA Script** - BPA script is similar to Service Script but has more Step Types to be defined such as it can invoke UI maps, etc. On selecting context menu option, BPA script will invoke UI map that will in turn invoke Service Script, which generates report. To create BPA script, follow the steps below:
    a. From the **Admin** menu, select **S** and then click **Script**. A sub-menu appears.
    b. Click **Add** option from the **Script** sub-menu.
    c. Select BPA Script from the **Script Type** drop-down list.



SCRIPT C1-Report OWNER Base

DESCRIPTION Report submission BPA For Open

DETAILED DESCRIPTION Report submission BPA

SCRIPT TYPE BPA Script ACCESSIBILITY OPTION Not Accessible from Scri

APPLICATION SERVICE

SCRIPT ENGINE VERSION 1.0

View Script Schema View XSD

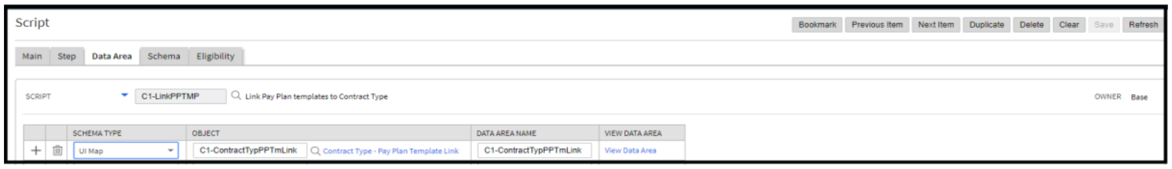
Script - C1-Report - Report submission BPA For Open Case By Type

This Script has 4 data area(s).

This Script has 16 step(s).

Figure 31: Script

6. **Define Data Area** - Add reference to UI map. This data area will be used to pass data from BPA script to UI Map. Now save the UI Map.



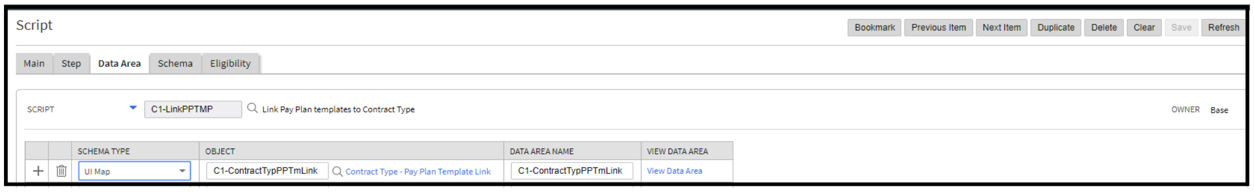
Script

SCRIPT: C1-LinkPPTMP — Link Pay Plan templates to Contract Type
OWNER: Base

SCHEMA TYPE	OBJECT	DATA AREA NAME	VIEW DATA AREA
UI Map	C1-ContractTypPPTmLink — Contract Type - Pay Plan Template Link	C1-ContractTypPPTmLink	View Data Area

Figure 32: Data Area

7. **Define Steps**
   a. From the **Admin** menu, select **S** and then click **Script**. A sub-menu appears.
   b. Click **Add** option from the **Script** sub-menu.
   c. Select BPA Script from the **Script Type** drop-down list.



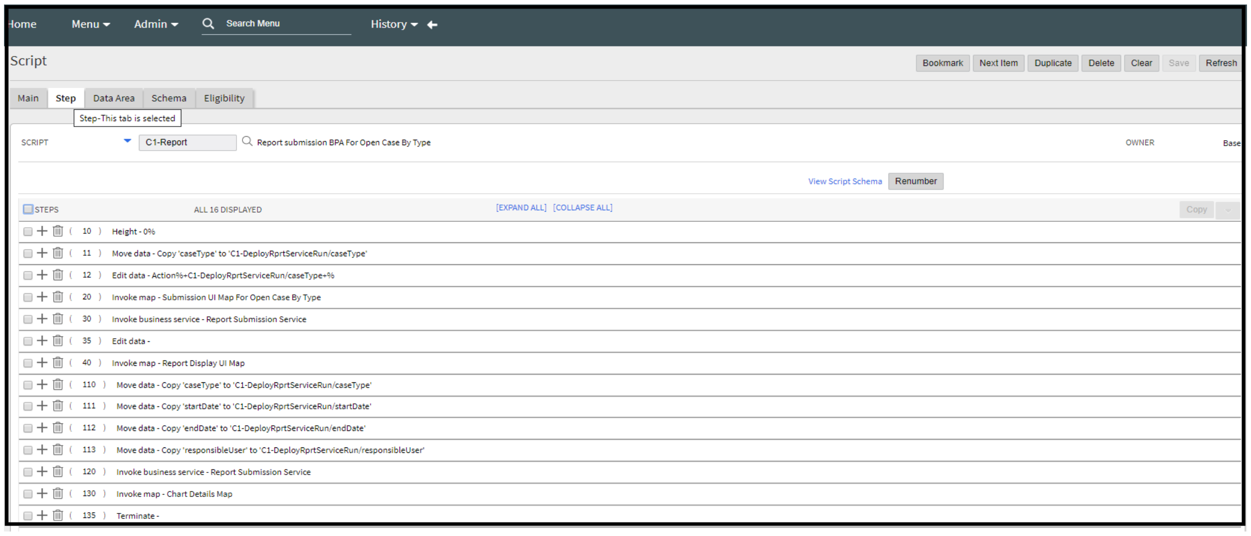
Script

SCRIPT: C1-LinkPPTMP — Link Pay Plan templates to Contract Type
OWNER: Base

SCHEMA TYPE	OBJECT	DATA AREA NAME	VIEW DATA AREA
UI Map	C1-ContractTypPPTmLink — Contract Type - Pay Plan Template Link	C1-ContractTypPPTmLink	View Data Area

Figure 33: Define Steps

   d. Move person_id from Global Context to UI Map



Script

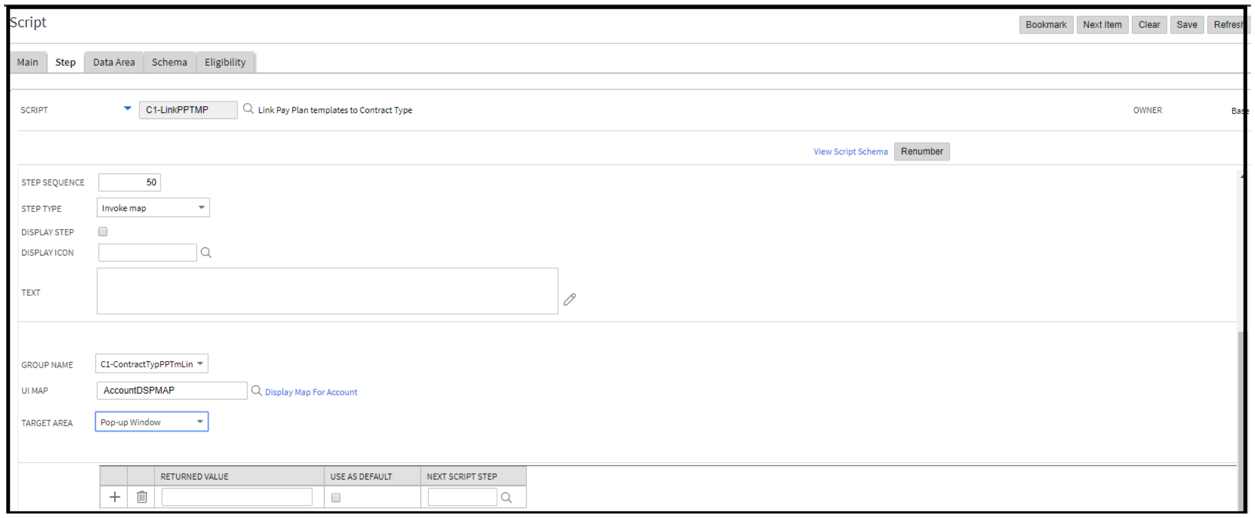
SCRIPT: C1-Report — Report submission BPA For Open Case By Type
OWNER: Base

STEPS — ALL 16 DISPLAYED

10	Height - 0%
11	Move data - Copy 'caseType' to 'C1-DeployRprtServiceRun/caseType'
12	Edit data - Action%+C1-DeployRprtServiceRun/caseType+%
20	Invoke map - Submission UI Map For Open Case By Type
30	Invoke business service - Report Submission Service
35	Edit data -
40	Invoke map - Report Display UI Map
110	Move data - Copy 'caseType' to 'C1-DeployRprtServiceRun/caseType'
111	Move data - Copy 'startDate' to 'C1-DeployRprtServiceRun/startDate'
112	Move data - Copy 'endDate' to 'C1-DeployRprtServiceRun/endDate'
113	Move data - Copy 'responsibleUser' to 'C1-DeployRprtServiceRun/responsibleUser'
120	Invoke business service - Report Submission Service
130	Invoke map - Chart Details Map
135	Terminate -

Figure 34: Define Steps

   e. Invoke UI map



SCRIPT C1-LinkPPTMP Link Pay Plan templates to Contract Type

OWNER

STEP SEQUENCE 50

STEP TYPE Invoke map

DISPLAY STEP

DISPLAY ICON

TEXT

GROUP NAME C1-ContractTypPPTmLin

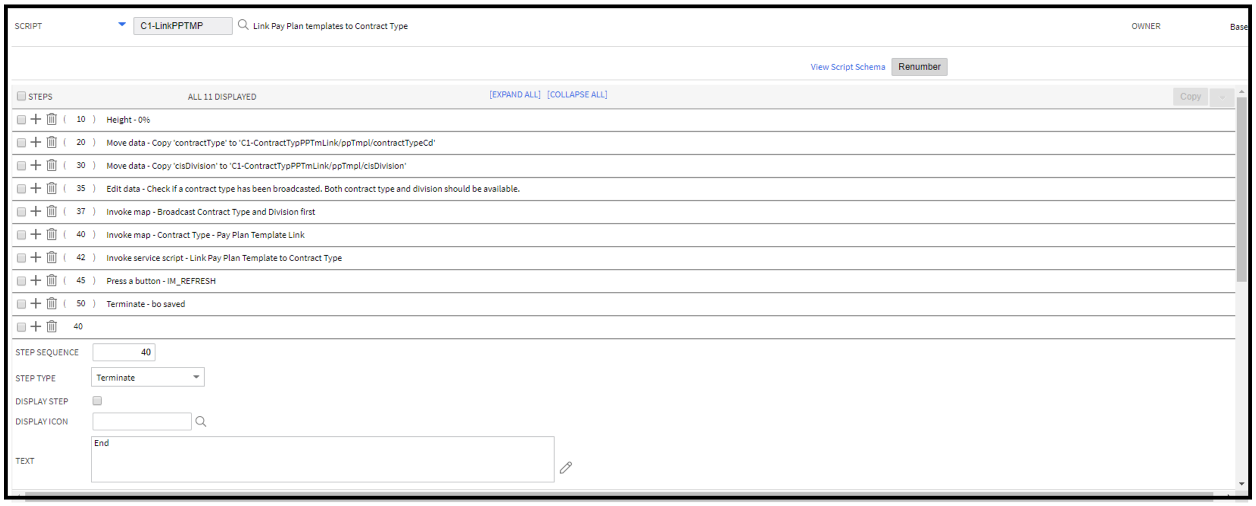
UI MAP AccountDSPMAP Display Map For Account

TARGET AREA Pop-up Window

RETURNED VALUE	USE AS DEFAULT	NEXT SCRIPT STEP

Figure 35: Invoke UI Map

f. Terminate step which ends BPA script. Now save the Script.



SCRIPT C1-LinkPPTMP Link Pay Plan templates to Contract Type

OWNER

STEPS ALL 11 DISPLAYED

(10) Height - 0%
(20) Move data - Copy 'contractType' to 'C1-ContractTypPPTmLink/ppTmpl/contractTypeCd'
(30) Move data - Copy 'cisDivision' to 'C1-ContractTypPPTmLink/ppTmpl/cisDivision'
(35) Edit data - Check if a contract type has been broadcasted. Both contract type and division should be available.
(37) Invoke map - Broadcast Contract Type and Division first
(40) Invoke map - Contract Type - Pay Plan Template Link
(42) Invoke service script - Link Pay Plan Template to Contract Type
(45) Press a button - IM_REFRESH
(50) Terminate - bo saved
40

STEP SEQUENCE 40

STEP TYPE Terminate

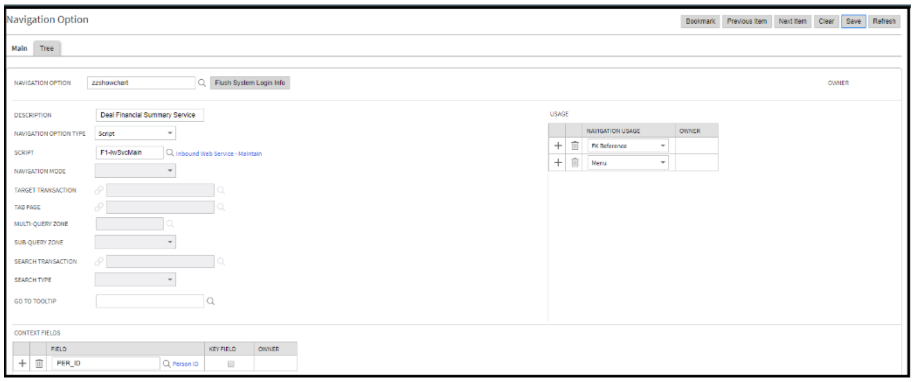
DISPLAY STEP

DISPLAY ICON

TEXT End

Figure 36: Terminate Step

8. **Define Navigation Option** - To link BPA script to context menu of Person, you need to create navigation option. To create navigation option:
    a. From the **Admin** menu, select **N** and then click **Navigation Option**. A sub-menu appears.
    b. Click **Add** option from the **Navigation Option** sub-menu.
    c. Declare 'PERSON ID' as **Context Field** in navigation option so that person id will be stored in Global context.



NAVIGATION OPTION	zzshowchart
DESCRIPTION	Deal Financial Summary Service
NAVIGATION OPTION TYPE	Script
SCRIPT	F1-IwSvcMain Inbound Web Service - Maintain
NAVIGATION MODE	
TARGET TRANSACTION	
TAB PAGE	
MULTI-QUERY ZONE	
SUB-QUERY ZONE	
SEARCH TRANSACTION	
SEARCH TYPE	
GO TO TOOLTIP	
OWNER	

USAGE

NAVIGATION USAGE	OWNER
FK Reference	
Menu	

CONTEXT FIELDS

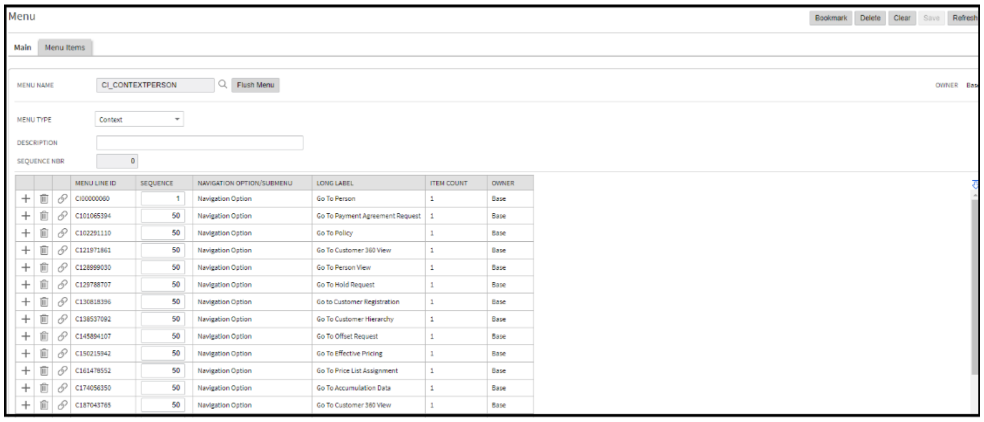
FIELD	KEY FIELD	OWNER
PER_ID Person ID		

Figure 37: Navigation

d. To add menu item on Person Context menu, which will refer ‘PERSON ID’, you need to follow the steps below:

i. Navigate to the **Admin** menu, select **M** and then click **Menu**. A sub-menu appears.

ii. Select **Search** from the menu options. Menu Search window appears.



MENU NAME	CI_CONTEXTPERSON
OWNER	Base
MENU TYPE	Context
DESCRIPTION	
SEQUENCE NBR	0

MENU LINE ID	SEQUENCE	NAVIGATION OPTION/SUBMENU	LONG LABEL	ITEM COUNT	OWNER
CI00000060	1	Navigation Option	Go To Person	1	Base
C101065394	50	Navigation Option	Go To Payment Agreement Request	1	Base
C102291110	50	Navigation Option	Go To Policy	1	Base
C121971861	50	Navigation Option	Go To Customer 360 View	1	Base
C128999030	50	Navigation Option	Go To Person View	1	Base
C129788707	50	Navigation Option	Go To Hold Request	1	Base
C130818396	50	Navigation Option	Go to Customer Registration	1	Base
C138537092	50	Navigation Option	Go To Customer Hierarchy	1	Base
C145894107	50	Navigation Option	Go To Offset Request	1	Base
C150215942	50	Navigation Option	Go To Effective Pricing	1	Base
C161478552	50	Navigation Option	Go To Price List Assignment	1	Base
C174056350	50	Navigation Option	Go To Accumulation Data	1	Base
C187043765	50	Navigation Option	Go To Customer 360 View	1	Base

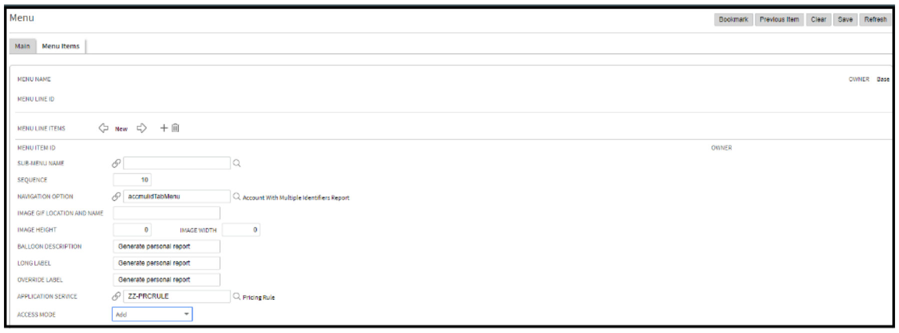
Figure 38: Person Context Menu

iii. Select ‘Context’ from the **Menu Type** drop-down list and click **Search.**

iv. Select ‘CI_CONTEXTPERSON’ Menu Name from the results. The selected menu name appears in Menu Name field.

v. Click Add (➕) icon to add new menu line at the end.

vi. Click ‘Go To Menu Items’ (➡) icon. Enter the details.



MENU NAME	
OWNER	Base
MENU LINE ID	
MENU LINE ITEMS	New
MENU ITEM ID	
SUB-MENU NAME	
SEQUENCE	10
NAVIGATION OPTION	accmulidTabMenu Account With Multiple Identifiers Report
IMAGE GIF LOCATION AND NAME	
IMAGE HEIGHT	0
IMAGE WIDTH	0
BALLOON DESCRIPTION	Generate personal report
LONG LABEL	Generate personal report
OVERRIDE LABEL	Generate personal report
APPLICATION SERVICE	ZZ-PRCRULE Pricing Rule
ACCESS MODE	Add

Figure 39: Add Menu Item

e. Provide correct Application Service for access control. Save the details.

f. Flush the cache if required.

g. Once completed ‘Go To Person’ and click on ‘Person Context Menu’. Menu item appears in context menu.

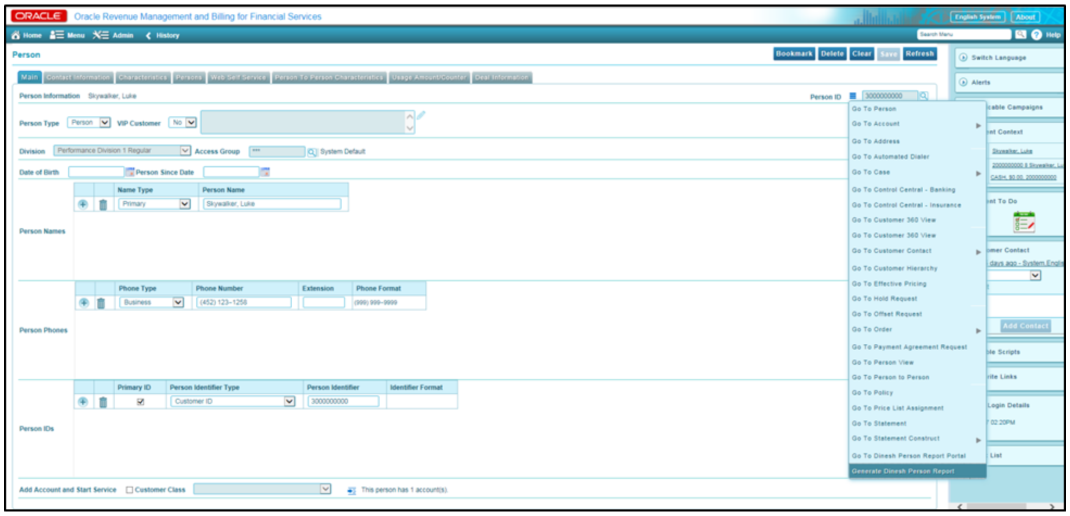
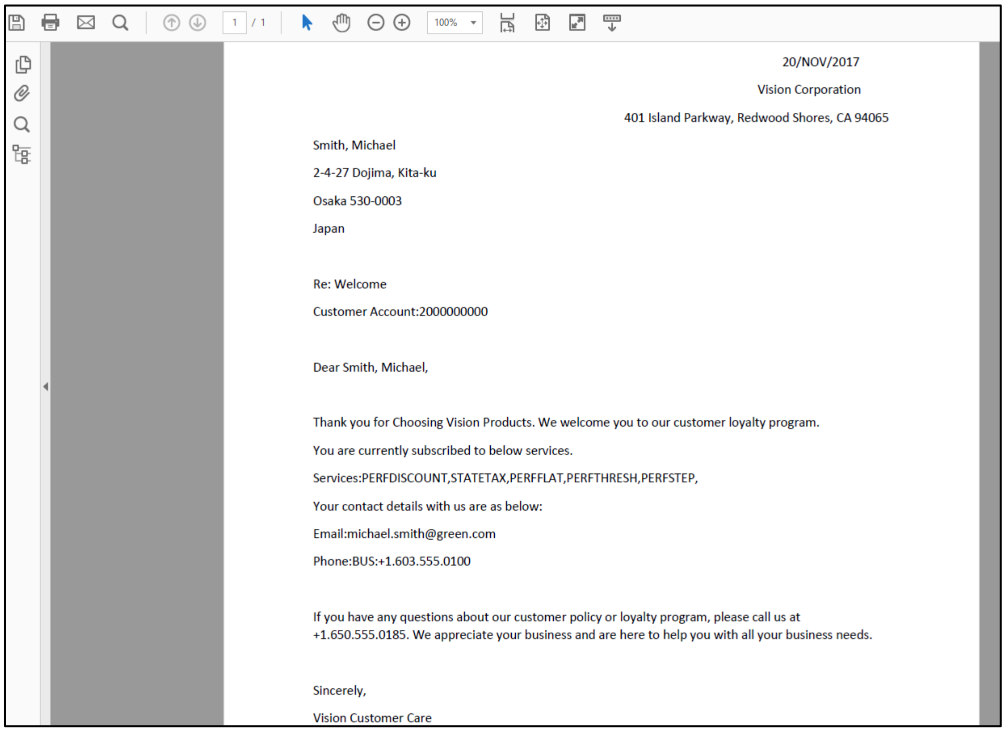


Figure 40: Person Context Menu

h. Click on Menu Item and select the report name from the list. It generates report as below.



20/NOV/2017

Vision Corporation

401 Island Parkway, Redwood Shores, CA 94065

Smith, Michael

2-4-27 Dojima, Kita-ku

Osaka 530-0003

Japan

Re: Welcome

Customer Account:2000000000

Dear Smith, Michael,

Thank you for Choosing Vision Products. We welcome you to our customer loyalty program.

You are currently subscribed to below services.

Services:PERFDISCOUNT,STATETAX,PERFFLAT,PERFTHRESH,PERFSTEP,

Your contact details with us are as below:

Email:michael.smith@green.com

Phone:BUS:+1.603.555.0100

If you have any questions about our customer policy or loyalty program, please call us at +1.650.555.0185. We appreciate your business and are here to help you with all your business needs.

Sincerely,

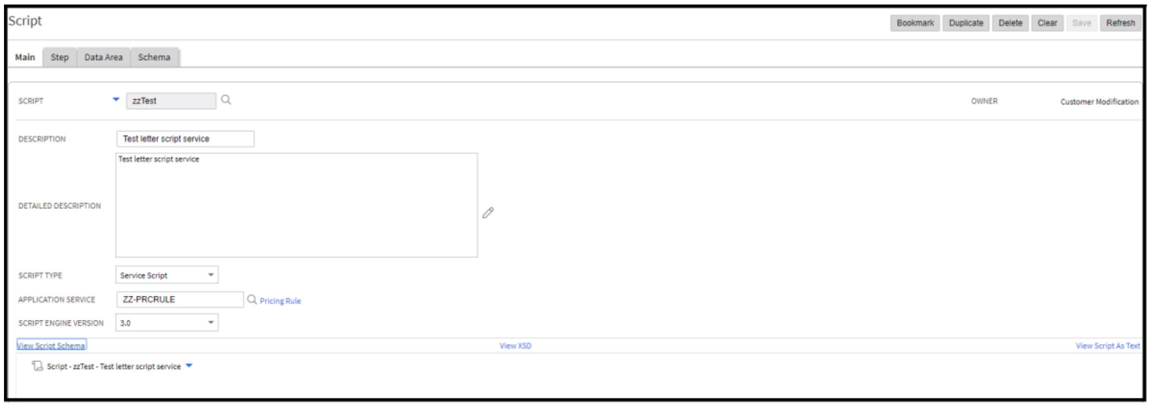
Vision Customer Care

Figure 41: Report Generation

3.2.12.2 Custom Letter Using Service Script and Groovy

Generating letter-based reports involves below steps:

1. **Creating Service Script** - It holds Groovy script. Groovy script gathers required report data and creates XML. This XML and FOP XSLT is used by Apache FOP to generate report. To create new service script:
   a. From the **Admin** menu, select **S** and then click **Script**. A sub-menu appears.
   b. Click Add option from the Script sub-menu.
   c. Enter the details in respective fields
      d. Select Service Script from the **Script Type** drop-down list.

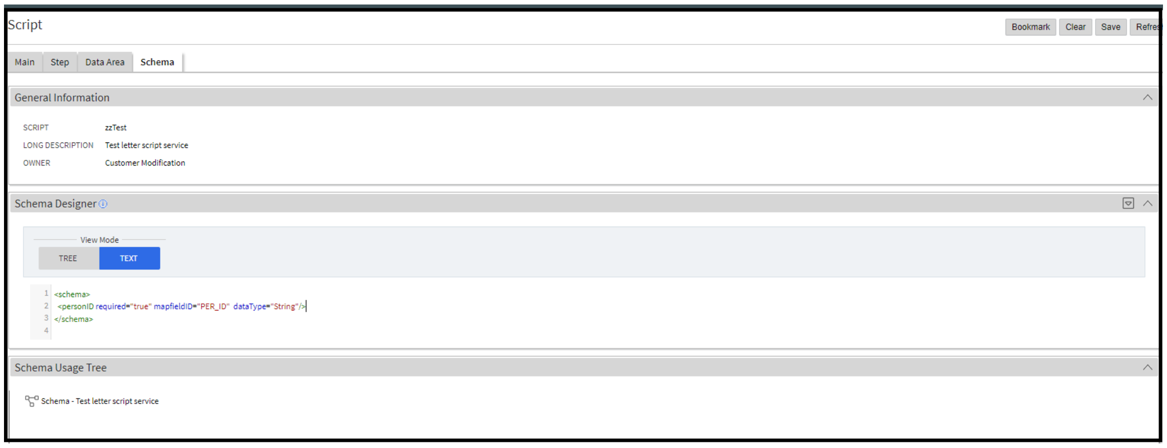


SCRIPT	zzTest
OWNER	Customer Modification
DESCRIPTION	Test letter script service
DETAILED DESCRIPTION	Test letter script service
SCRIPT TYPE	Service Script
APPLICATION SERVICE	ZZ-PRCRULE Pricing Rule
SCRIPT ENGINE VERSION	3.0

Script - zzTest - Test letter script service

Figure 42: Script - Main

2. **Defining Schema for Service Script** - It defines the data elements passed to and from a service script. The purpose of the schema is to describe the input and output parameters used when invoking the script.



SCRIPT	zzTest
LONG DESCRIPTION	Test letter script service
OWNER	Customer Modification

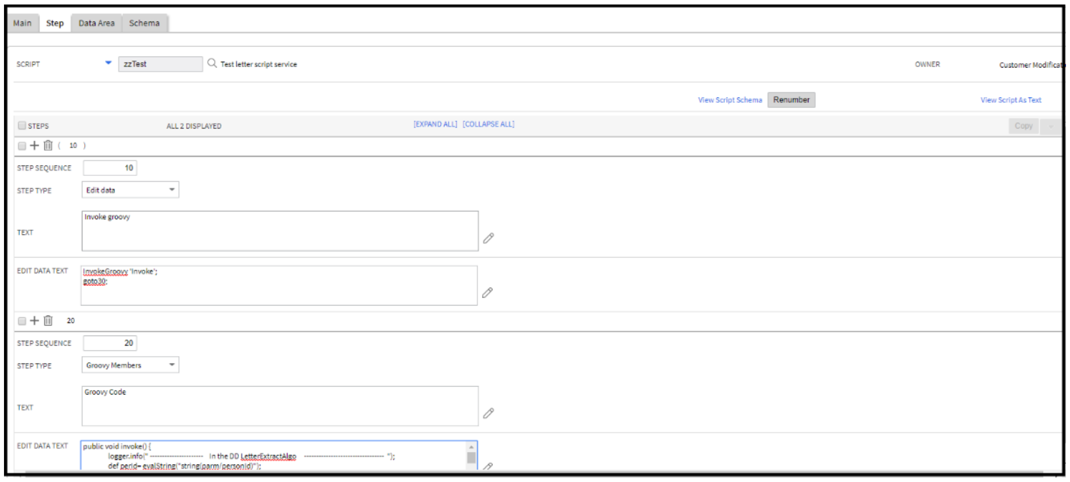
```
<schema>
 <personID required="true" mapfieldID="PER_ID"  dataType="String"/>
</schema>
```

Schema - Test letter script service

Figure 43: Script - Schema

3. **Defining Script Steps** - Steps define logic of the service script. You can use different types of steps to move data, call other Scripts, execute script code or invoke business services, etc. This process involves two sub processes:

a. **Invoking a groovy function** - To invoke a groovy function, you need to define a Step. For more information, refer to Defining a Step section. Note that you can invoke function with no parameters.


Main | Step | Data Area | Schema

SCRIPT: zzTest — Test letter script service
OWNER: Customer Modificat

View Script Schema | Renumber | View Script As Text

STEPS — ALL 2 DISPLAYED — [EXPAND ALL] [COLLAPSE ALL] — Copy

(10)

STEP SEQUENCE: 10
STEP TYPE: Edit data
TEXT: Invoke groovy
EDIT DATA TEXT:

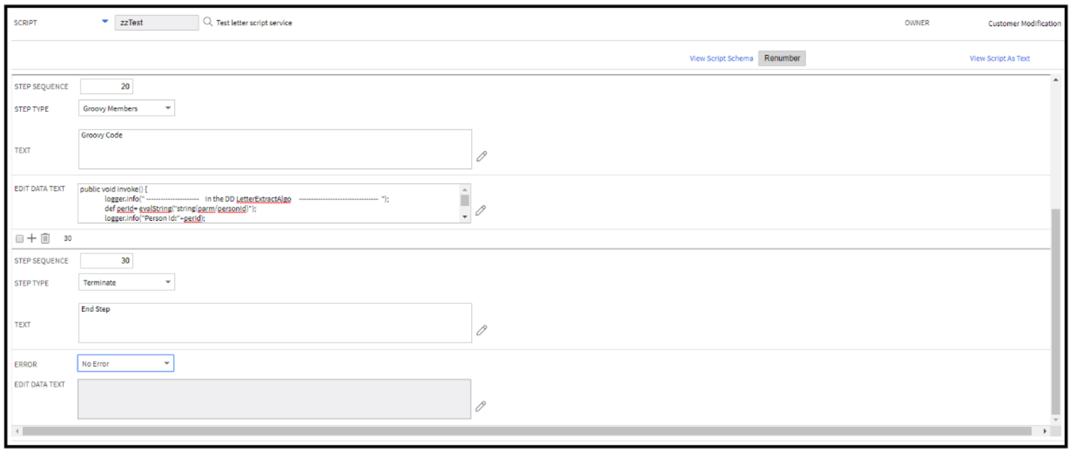
```
InvokeGroovy 'Invoke';
goto30;
```

20

STEP SEQUENCE: 20
STEP TYPE: Groovy Members
TEXT: Groovy Code
EDIT DATA TEXT:

```
public void invoke() {
        logger.info(" ---------------------   In the DD LetterExtractAlgo   ------------------------------- ");
        def perId= evalString("string(parm/personId)");
```

Figure 44: Script - Steps


SCRIPT: zzTest — Test letter script service
OWNER: Customer Modification

View Script Schema | Renumber | View Script As Text

STEP SEQUENCE: 20
STEP TYPE: Groovy Members
TEXT: Groovy Code
EDIT DATA TEXT:

```
public void invoke() {
        logger.info(" ---------------------   In the DD LetterExtractAlgo   ------------------------------- ");
        def perId= evalString("string(parm/personId)");
        logger.info("Person Id:"+perId);
```

30

STEP SEQUENCE: 30
STEP TYPE: Terminate
TEXT: End Step
ERROR: No Error
EDIT DATA TEXT:

Figure 45: Script – Groovy Function

b. **Writing a groovy code** – This code performs following steps:

- Gathers report data and generates XML file
- Calls API which will use XML and FOP XSLT to generate report in PDF
- Terminates the Service Script
- Saves the Service Script

A sample code for Groovy Script is listed below:

```
public void invoke() {
          logger.info("  ----------------------          In   the   DD
LetterExtractAlgo     ------------------------------  ");
          def perId= evalString("string(parm/personId)");
          logger.info("Person Id:"+perId);
```

```
            String processDate = "";
            String ownerName = "";
            String ownerAddress1 = "";
            String ownerAddress2 = "";
            String ownerAddress3 = "";
            String ownerCity = "";
            String ownerPostal = "";
            String ownerState = "";
            String acctId = null;
            String emailId = null;
            String phone = null;
            java.lang.StringBuilder PIs = new java.lang.StringBuilder();
            String services = null;
            def pstmt = null;
            def row = null;
            def pstmt1 = null;
            def row1 = null;
            def pstmt2 = null;
            def rowlist = null;

            try {
                  pstmt = createPreparedStatement(
                              "select      trim(ACP.ACCT_ID)      ACCT_ID
,trim(PR.EMAILID)  EMAIL,trim(PH.PHONE_TYPE_CD)  ||  ':'||trim(PH.PHONE)
PHONE,"+
                              "PR.ADDRESS1           ADDRESS1,PR.ADDRESS2
ADDRESS2,PR.ADDRESS3  ADDRESS3,PR.CITY  CITY,  PR.STATE  STATE,PR.POSTAL
POSTAL,TO_CHAR(SYSDATE,'DD/MON/YYYY') CC_DTTM " +
                              "FROM    CI_PER    PR,    CI_PER_PHONE    PH,
CI_ACCT_PER ACP " +
                              "WHERE PR.PER_ID= :perId " +
                              "AND PR.PER_ID=PH.PER_ID " +
                              "AND PR.PER_ID=ACP.PER_ID", "Person Info");
                  pstmt.bindString("perId", perId, "PER_ID");
                  row = pstmt.firstRow();
```

```
                if (row != null) {
                      acctId = row.getString("ACCT_ID");
                      emailId = row.getString("EMAIL");
                      phone = row.getString("PHONE");
                      ownerAddress1 = row.getString("ADDRESS1");
                      ownerAddress2 = row.getString("ADDRESS2");
                      ownerAddress3 = row.getString("ADDRESS3");
                      ownerCity = row.getString("CITY");
                      ownerState = row.getString("STATE");
                      ownerPostal = row.getString("POSTAL");
                      processDate = row.getString("CC_DTTM");
                }
                pstmt1 = createPreparedStatement(
                            "select PN.ENTITY_NAME ENTITY_NAME " +
                            "from CI_PER PR, CI_PER_NAME PN " +
                            "WHERE PR.PER_ID=:perId " +
                            "AND PR.PER_ID=PN.PER_ID","Person Name");
                pstmt1.bindString("perId", perId, "PER_ID");
                row1 = pstmt1.firstRow();
                if (row1 != null) {
                      ownerName = row1.getString("ENTITY_NAME");
                }
                pstmt2        =        createPreparedStatement("SELECT
DISTINCT(TRIM(PRICEITEM_CD)) PRICEITEM_CD " +
                            "FROM CI_PARTY PRTY, CI_PRICELIST_ASGN PLAS,
CI_PRICEASGN PRAS " +
                            "WHERE PRTY.PARTY_ID=:party_id " +
                            "AND PRTY.PARTY_TYPE_FLG='ACCT' " +
                            "AND PRTY.PARTY_UID=PLAS.PARTY_UID " +
                            "AND      PLAS.PRICELIST_ID=PRAS.OWNER_ID",
"Person Services");
                pstmt2.bindString("party_id", acctId, "PARTY_ID");
                rowlist = pstmt2.list();
                for (def row2 : rowlist) {
                      PIs.append(row2.getString("PRICEITEM_CD"));
                      PIs.append(",");
                }
                services = PIs.toString();
            } catch (Exception e) {
                logger.error("Exception ", e);
            } finally {
                try {
                    pstmt1.close();
                    pstmt2.close();
                } catch (Exception e) {
                    logger.error("Error While Closing ResultSet ", e);
                } finally {
                    pstmt1 = null;
                    pstmt2 = null;
                }
            }

            logger.info("\n Account Id:" + acctId);
            logger.info("\n Email Id:" + emailId);
            logger.info("\n Phone:" + phone);
            logger.info("\n Services:" + services);
            logger.info("\n Name:" + ownerName);
            logger.info("\n Process Date:" + processDate);
            logger.info("\n Add 1:" + ownerAddress1);
            logger.info("\n Add 2:" + ownerAddress2);
            logger.info("\n Add 3:" + ownerAddress3);
            logger.info("\n City:" + ownerCity);
            logger.info("\n State:" + ownerState);
            logger.info("\n Postal:" + ownerPostal);

            /// Create XML Of Fields

            org.w3c.dom.Document             finalDocument             =
com.splwg.base.support.scripting.XMLUtils.createNewDomDocument();
            org.w3c.dom.Element                  root                  =
finalDocument.createElement("DOCUMENT");
```

```
          finalDocument.appendChild(root);


          org.w3c.dom.Element                 root1                 =
com.splwg.base.support.scripting.XMLUtils.addElement(root, "DOCSET");


          org.w3c.dom.Element              itemElement              =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
          itemElement.setAttribute("NAME", "ProcessDt");
          itemElement.setTextContent(processDate);


          org.w3c.dom.Element              itemElement1             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
          itemElement1.setAttribute("NAME", "OwnerFullNm");
          itemElement1.setTextContent(ownerName);


          org.w3c.dom.Element              itemElement2             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
          itemElement2.setAttribute("NAME", "OwnerAddrLine1");
          itemElement2.setTextContent(ownerAddress1);


          org.w3c.dom.Element              itemElement3             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
          itemElement3.setAttribute("NAME", "OwnerAddrLine2");

     itemElement3.setTextContent(ownerAddress2+","+ownerAddress3);


          org.w3c.dom.Element              itemElement4             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
          itemElement4.setAttribute("NAME", "OwnerAddrLine3");

     itemElement4.setTextContent(ownerCity+","+ownerState+","+ownerPos
tal);


          org.w3c.dom.Element              itemElement5             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
          itemElement5.setAttribute("NAME", "accountId");
          itemElement5.setTextContent(acctId);
```

```
           org.w3c.dom.Element              itemElement6              =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
           itemElement6.setAttribute("NAME", "emailAddress");
           itemElement6.setTextContent(emailId);


           org.w3c.dom.Element              itemElement7              =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
           itemElement7.setAttribute("NAME", "phone");
           itemElement7.setTextContent(phone);


           org.w3c.dom.Element              itemElement8              =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
           itemElement8.setAttribute("NAME", "services");
           itemElement8.setTextContent(services);


           String              letterExtractXMLFileLoc              =
"/scratch/rmbbuild/spl/DD26010/splapp/reporting/extractxml/" + perId +
".xml";
           String sourceXML = "";
           try {

      com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil
commonUtil
=com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil.Fact
ory.newInstance();
                sourceXML = finalDocument.documentElement as String;

      commonUtil.fileWrite("/scratch/rmbbuild/spl/DD26010/splapp/report
ing/extractxml/", perId, sourceXML, "xml");

      com.splwg.ccb.domain.billing.billtransform.FOPReportGenerationCom
ponent                         fopRptComp                             =
com.splwg.ccb.domain.billing.billtransform.FOPReportGenerationComponen
t.Factory.newInstance();
                String                   xslFileLoc                   =
"/scratch/rmbbuild/spl/DD26010/splapp/reporting/xsl/DD-WELCOME.xsl";
                String                   pdfFilePath                  =
"/scratch/rmbbuild/spl/DD26010/splapp/reporting/output";
                String pdfFileLoc = pdfFilePath + "/" + perId + ".pdf";
                String  url  =  fopRptComp.getFOPPdf(perId,  xslFileLoc,
pdfFileLoc, letterExtractXMLFileLoc, " ");
```

```
          } catch (Exception e) {
               logger.error("Exception occurred" + e.getMessage());
          }
     }
```

A sample code for Test/SAMPLE/ Letter Extract XML using Groovy Plugin Script is listed below. It gets all the required data and creates XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
<DOCSET>
<FIELD NAME="ProcessDt">20/NOV/2017</FIELD>
<FIELD NAME="OwnerFullNm"> Smith, Michael</FIELD>
<FIELD NAME="OwnerAddrLine1"> 2-4-27 Dojima, Kita-ku</FIELD>
<FIELD NAME="OwnerAddrLine2"> Osaka 530-0003</FIELD>
<FIELD NAME="OwnerAddrLine3">Japan</FIELD>
<FIELD NAME="accountId"> 2000000000</FIELD>
<FIELD NAME="emailAddress"> michael.smith@green.com</FIELD>
<FIELD NAME="phone">BUS: +1.603.555.0100</FIELD>
<FIELD
NAME="services">PERFDISCOUNT,STATETAX,PERFFLAT,PERFTHRESH,PERFSTEP,</F
IELD>
</DOCSET>
</DOCUMENT>
```

A sample code for APACHE FOP XSLT is listed below. It is required by FOP to generate PDF and uses XML created by Groovy Script

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
     <xsl:template match="/">
          <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
               <fo:layout-master-set>
                    <fo:simple-page-master master-name="simpleA4"
                            page-height="29.7cm"
                page-width="21cm"
                margin-top="1cm"
                margin-bottom="0.1cm"
```

```
                 margin-left="0.8cm"
                 margin-right="1.0cm" >
                          <fo:region-body  margin-top="2.5cm"  margin-
bottom="2.5cm"/>
                          <fo:region-before extent="2.0cm"/>
                          <fo:region-after extent="2.0cm"/>
                     </fo:simple-page-master>
                </fo:layout-master-set>
                <fo:page-sequence master-reference="simpleA4"
               initial-page-number="1">

                     <fo:static-content flow-name="xsl-region-before">
                          <fo:block font-size="7pt">

                          </fo:block>
                     </fo:static-content>

                     <fo:static-content flow-name="xsl-region-after">

                          <fo:block      font-size="9.0pt"      font-
family="sans-serif"
                       padding-after="2.0pt"     space-before="2.0pt"
text-align="right"
                       border-top-style="ridge"        border-bottom-
width="0.5pt">

                          </fo:block>

                     </fo:static-content>

                     <fo:flow flow-name="xsl-region-body">
                          <xsl:apply-templates
select="DOCUMENT/DOCSET" />
                          <fo:block id="last-page"> </fo:block>
                     </fo:flow>
                </fo:page-sequence>
```

```
          </fo:root>
     </xsl:template>

     <xsl:template match="DOCUMENT/DOCSET">
          <fo:block text-align="right" font-size="9px">
               <xsl:value-of select="//FIELD[@NAME='ProcessDt']"/>
          </fo:block>

          <fo:block    text-align="right"    font-size="9px"    space-
before="0.37in" >Vision Corporation</fo:block>
          <fo:block  text-align="right"  font-size="9px"  >401  Island
Parkway, Redwood Shores, CA 94065</fo:block>

          <fo:block text-align="left" font-size="9px">
               <xsl:value-of select="//FIELD[@NAME='OwnerFullNm']"/>
          </fo:block>
          <fo:block text-align="left" font-size="9px">
               <xsl:value-of
select="//FIELD[@NAME='OwnerAddrLine1']"/>
          </fo:block>
          <fo:block text-align="left" font-size="9px">
               <xsl:value-of
select="//FIELD[@NAME='OwnerAddrLine2']"/>
          </fo:block>
          <fo:block text-align="left" font-size="9px">
               <xsl:value-of
select="//FIELD[@NAME='OwnerAddrLine3']"/>
          </fo:block>
          <fo:block    text-align="left"    font-size="9px"    space-
before="0.37in">Re: Welcome</fo:block>
          <fo:block    text-align="left"    font-size="9px"    space-
before="0.10in">Customer                        Account:<xsl:value-of
select="//FIELD[@NAME='accountId']"/>
          </fo:block>
          <fo:block    text-align="left"    font-size="9px"    space-
before="0.37in">Dear                                    <xsl:value-of
select="//FIELD[@NAME='OwnerFullNm']"/>,</fo:block>
```

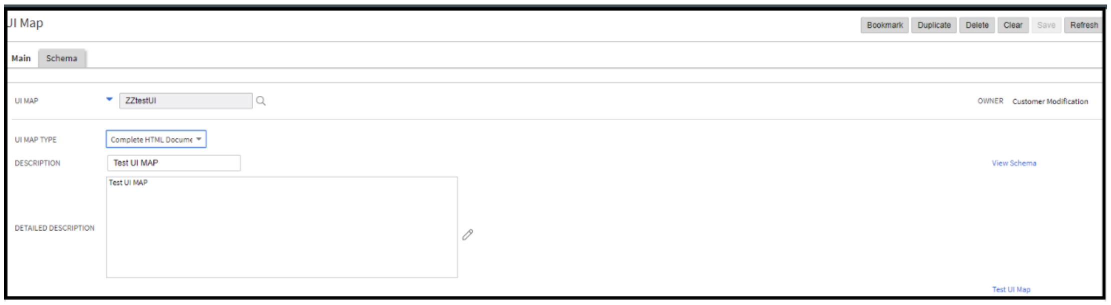
```
           <fo:block    text-align="left"    font-size="9px"    space-before="0.10in">Thank you for Choosing Vision Products. We welcome you to our customer loyalty program.
           </fo:block>
           <fo:block    text-align="left"    font-size="9px"    space-before="0.10in">You are currently subscribed to below services.</fo:block>
           <fo:block         text-align="left"         font-size="9px" >Services:<xsl:value-of select="//FIELD[@NAME='services']"/>
           </fo:block>
           <fo:block    text-align="left"    font-size="9px"    space-before="0.10in">Your contact details with us are as below:</fo:block>
           <fo:block         text-align="left"         font-size="9px" >Email:<xsl:value-of select="//FIELD[@NAME='emailAddress']"/>
           </fo:block>
           <fo:block         text-align="left"         font-size="9px" >Phone:<xsl:value-of select="//FIELD[@NAME='phone']"/>
           </fo:block>


           <fo:block    text-align="left"    font-size="9px"    space-before="0.10in">If you have any questions about our customer policy or loyalty program, please call us at +1.650.555.0185. We appreciate your business and are here to help you with all your business needs.</fo:block>


           <fo:block    text-align="left"    font-size="9px"    space-before="0.37in">Sincerely,</fo:block>
           <fo:block text-align="left" font-size="9px"> Vision  Customer Care.</fo:block>
      </xsl:template>
      <!--Template -->
      <xsl:attribute-set name="foCellAttributeSet">
           <xsl:attribute name="padding-right">2mm</xsl:attribute>
           <xsl:attribute name="padding-left">2mm</xsl:attribute>
           <xsl:attribute name="padding-top">2mm</xsl:attribute>
           <xsl:attribute name="padding-bottom">2mm</xsl:attribute>
      </xsl:attribute-set>
</xsl:stylesheet>
```

4. **Creating UI Map** - The User Interface (UI) map holds HTML to be rendered within portal zones and Business Process Assistant (BPA) scripts. UI maps allow you to create input forms and output

maps that closely match your customer's business practices. In other words, the UI Map is designed to facilitate the capture and display of your business objects and business services. To create UI Map, follow the steps below:

a. From the **Admin** menu, select **U** and then click **UI Map**. A sub-menu appears.

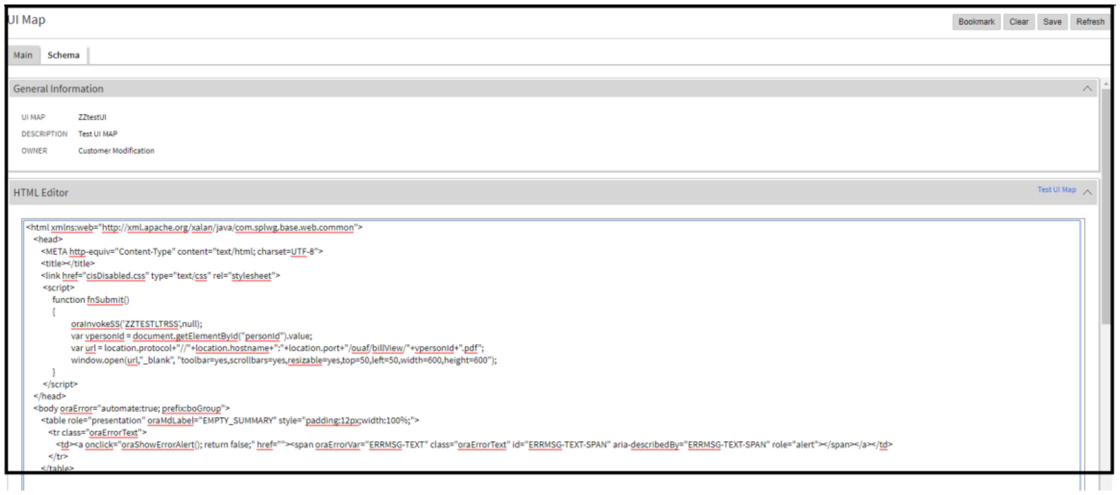
b. Click **Add** option from the UI Map sub-menu.



UI MAP: ZZtestUI
OWNER: Customer Modification
UI MAP TYPE: Complete HTML Docume
DESCRIPTION: Test UI MAP
DETAILED DESCRIPTION: Test UI MAP

Figure 46: Creating UI MAP

The UI map will call Service Script created above and pass necessary input data.



UI MAP ZZtestUI
DESCRIPTION Test UI MAP
OWNER Customer Modification

```
<html xmlns:web="http://xml.apache.org/xalan/java/com.splwg.base.web.common">
  <head>
    <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
    <link href="cisDisabled.css" type="text/css" rel="stylesheet">
     <script>
       function fnSubmit()
       {
            oraInvokeSS('ZZTESTLTRSS',null);
            var vpersonId = document.getElementById("personId").value;
            var url = location.protocol+"//"+location.hostname+":"+location.port+"/ouaf/billView/"+vpersonId+".pdf";
            window.open(url,"_blank", "toolbar=yes,scrollbars=yes,resizable=yes,top=50,left=50,width=600,height=600");
       }
     </script>
  </head>
  <body oraError="automate:true; prefix:boGroup">
    <table role="presentation" oraMdLabel="EMPTY_SUMMARY" style="padding:12px;width:100%;">
      <tr class="oraErrorText">
        <td><a onclick="oraShowErrorAlert(); return false;" href=""><span oraErrorVar="ERRMSG-TEXT" class="oraErrorText" id="ERRMSG-TEXT-SPAN" aria-describedBy="ERRMSG-TEXT-SPAN" role="alert"></span></a></td>
      </tr>
    </table>
```

Figure 47: Service Script

The Schema for UI map will refer to Service Script.

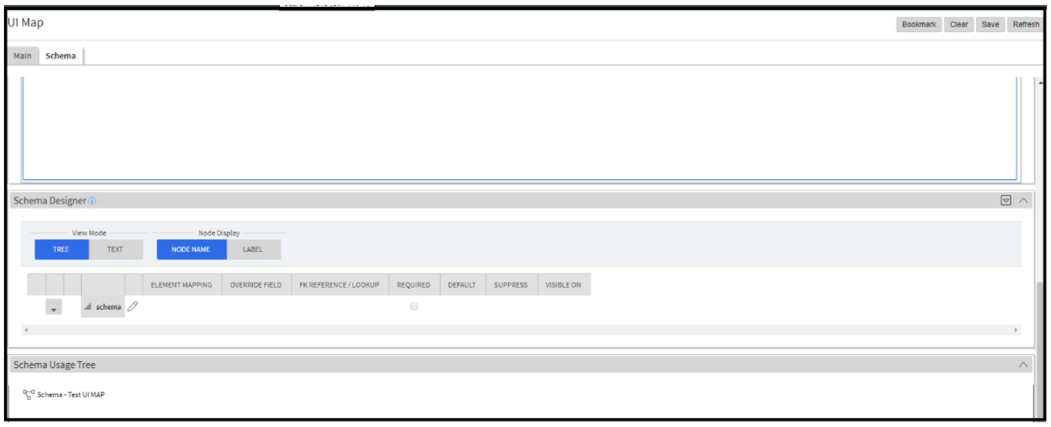


Figure 48: UI Map Schema

A sample code for HTML is listed below:

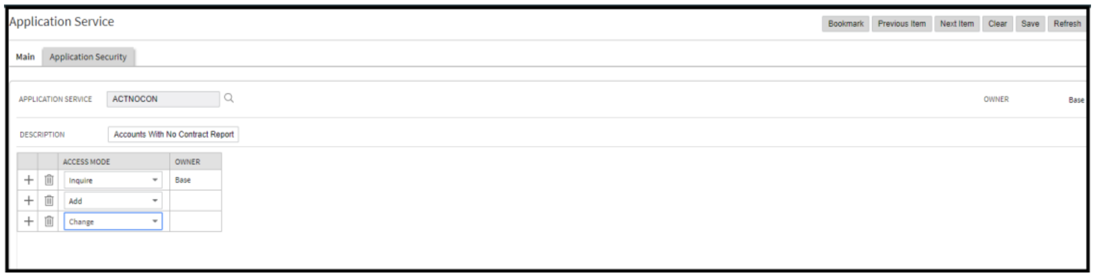
```
<html
xmlns:web="http://xml.apache.org/xalan/java/com.splwg.base.web.common"
>
    <head>
        <META       http-equiv="Content-Type"       content="text/html;
charset=UTF-8">
        <title></title>
        <link href="cisDisabled.css" type="text/css" rel="stylesheet">
         <script>
              function fnSubmit()
              {
                        oraInvokeSS('ZZTESTLTRSS',null);
                        var                 vpersonId                  =
document.getElementById("personId").value;
                        var                    url                     =
location.protocol+"//"+location.hostname+":"+location.port+"/ouaf/bill
View/"+vpersonId+".pdf";
                        window.open(url,"_blank",
"toolbar=yes,scrollbars=yes,resizable=yes,top=50,left=50,width=600,hei
ght=600");
              }
         </script>
    </head>
    <body oraError="automate:true; prefix:boGroup">
        <table      role="presentation"      oraMdLabel="EMPTY_SUMMARY"
style="padding:12px;width:100%;">
            <tr class="oraErrorText">
                <td><a   onclick="oraShowErrorAlert();   return   false;"
href=""><span      oraErrorVar="ERRMSG-TEXT"      class="oraErrorText"
id="ERRMSG-TEXT-SPAN"              aria-describedBy="ERRMSG-TEXT-SPAN"
role="alert"></span></a></td>
            </tr>
        </table>


        <table      role="presentation"      oraMdLabel="EMPTY_SUMMARY"
style="border-spacing:4px;width:100%;">
            <tr>
```

```
                <td        oraLabel="personId"        class="oraLabel
oraTableLabel"><label for="personId"></label></td><td class="oraNormal
oraTableData"><input          id="personId"          class="oraInput"
oraField="personId"></td>

            </tr>

            <tr>

                <td      class="oraSectionStart      oraEmbeddedTable"
colspan="2">

                    <table                         role="presentation"
oraMdLabel="EMPTY_SUMMARY" style="border-spacing:2px;">

                        <tr>

                            <td><input           onClick="fnSubmit();"
oraMdLabel="SUBMIT" class="oraButton" type="button"></td>

                        </tr>

                    </table>

                </td>

            </tr>

        </table>

    </body>

    <xml style="display:none;"></xml>

</html>
```

5. **Creating Application Service** - Application Service provides access control to entities created in ORMB. Using Application Service, you can control actions performed on entities like Zone, UI Maps. You can also control which user groups can access which ORMB entities. To create Application Service:

   a. From the **Admin** menu, select using Admin **A** and then click Application Service. A sub-menu appears.

   b. Click **Add** option from the Application Service sub-menu.



APPLICATION SERVICE: ACTNOCON
OWNER: Base
DESCRIPTION: Accounts With No Contract Report

ACCESS MODE	OWNER
Inquire	Base
Add	
Change	

Figure 49: Application Service

   c. Add Application Service to required USER_GROUPS.

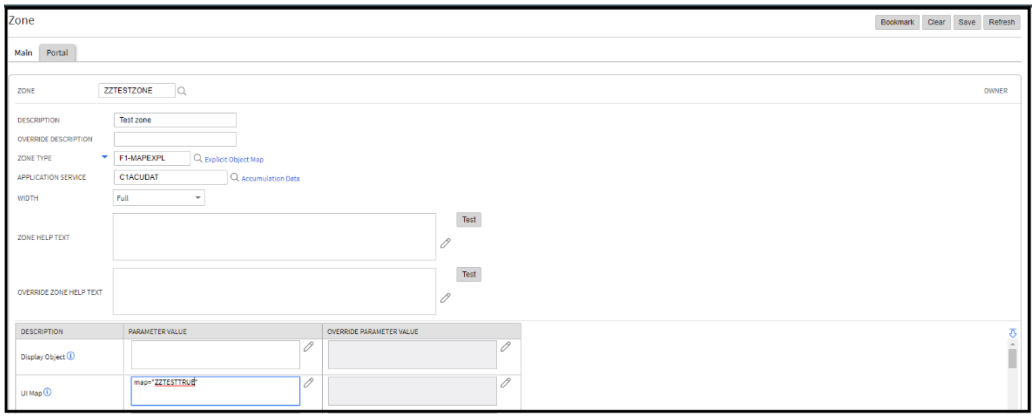
d. You can create a separate Page in ORMB which can accept custom input and generate letters or reports. Before creating the same, you need to create Portal and map zone. A Portal is a container which holds UI in ORMB.

Note: One portal can have multiple zones, and each zone can display different UI Maps.

6. **Creating Map Zone** - Map Zone is a type of zone which displays UI Maps. To create Map Zone,

a. From the **Admin** menu, select **Z** and then click **Zone**. A sub-menu appears.

b. Click **Add** option from the Zone sub-menu.

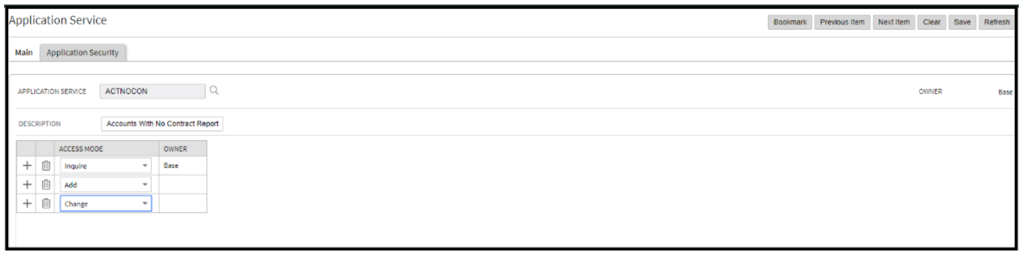


ZONE	ZZTESTZONE
OWNER	
DESCRIPTION	Test zone
OVERRIDE DESCRIPTION	
ZONE TYPE	F1-MAPEXPL Explicit Object Map
APPLICATION SERVICE	C1ACUDAT Accumulation Data
WIDTH	Full
ZONE HELP TEXT	
OVERRIDE ZONE HELP TEXT	

DESCRIPTION	PARAMETER VALUE	OVERRIDE PARAMETER VALUE
Display Object		
UI Map	map="ZZTESTTRUE"	

Figure 50: Creating Map Zone

c. Provide Application Service and UI Map. Save the changes.



APPLICATION SERVICE	ACTNOCON
OWNER	Base
DESCRIPTION	Accounts With No Contract Report

ACCESS MODE	OWNER
Inquire	Base
Add	
Change	

Figure 51: Application Service UI Map Link

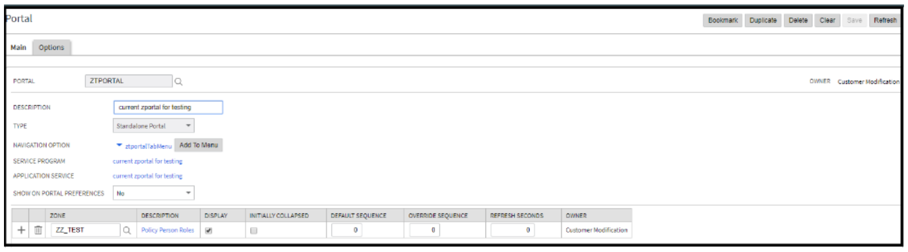
7. **Creating Portal** - To create portal, follow the steps below:

a. From the Admin menu, select **P** and then click **Portal**. A sub-menu appears.

b. Click **Add** option from the Portal sub-menu.

c. Add map zone to the zone list.

d. Save the changes.



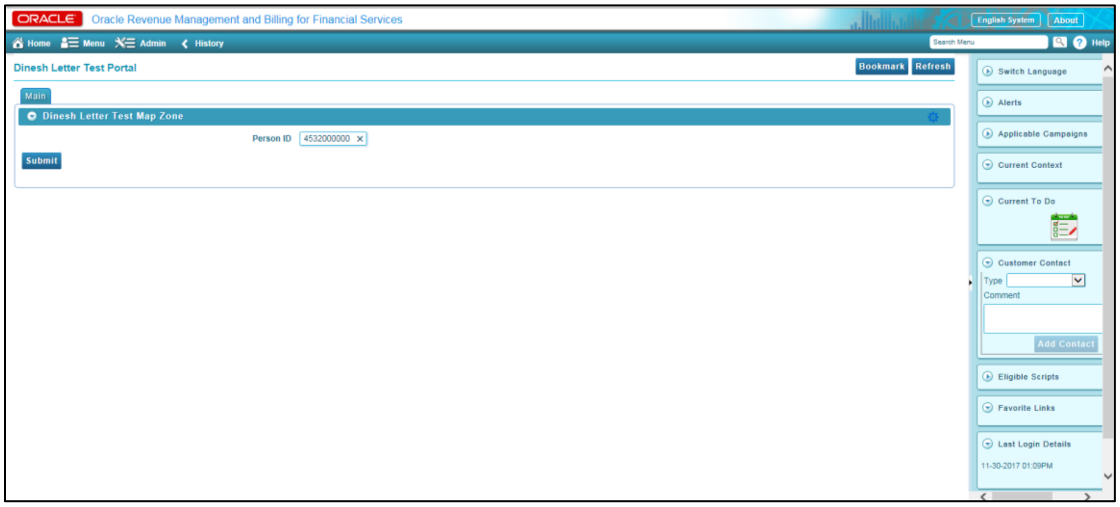
PORTAL	ZTPORTAL
OWNER	Customer Modification
DESCRIPTION	current zportal for testing
TYPE	Standalone Portal
NAVIGATION OPTION	ztportalTabMenu
SERVICE PROGRAM	current zportal for testing
APPLICATION SERVICE	current zportal for testing
SHOW ON PORTAL PREFERENCES	No

ZONE	DESCRIPTION	DISPLAY	INITIALLY COLLAPSED	DEFAULT SEQUENCE	OVERRIDE SEQUENCE	REFRESH SECONDS	OWNER
ZZ_TEST	Policy Person Roles	✓	☐	0	0	0	Customer Modification

Figure 52: Portal Menu

e. New Application Service will be created for portal. Add it to the required USER_GROUPS. Save the changes.

f. Portal will be automatically added under Main Menu. Access the portal.

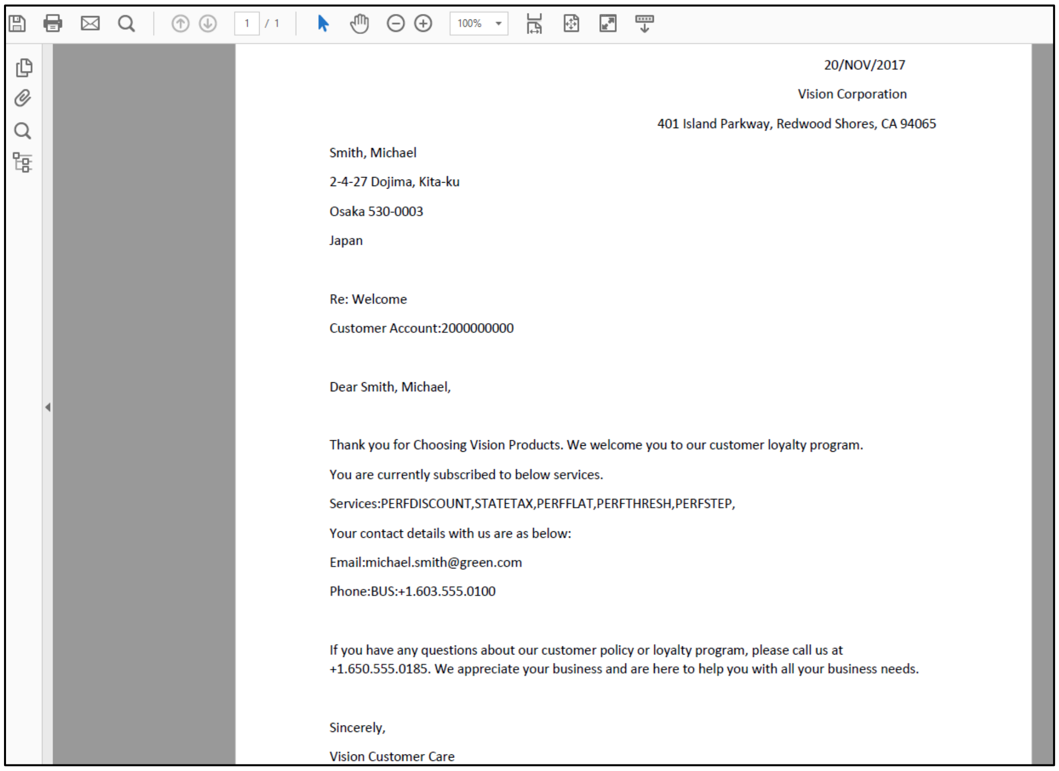


Person ID 4532000000

11-30-2017 01:09PM

Figure 53: Portal Screen

g. Provide data and click **Submit**. The report is generated.



20/NOV/2017

Vision Corporation

401 Island Parkway, Redwood Shores, CA 94065

Smith, Michael

2-4-27 Dojima, Kita-ku

Osaka 530-0003

Japan

Re: Welcome

Customer Account:2000000000

Dear Smith, Michael,

Thank you for Choosing Vision Products. We welcome you to our customer loyalty program.

You are currently subscribed to below services.

Services:PERFDISCOUNT,STATETAX,PERFFLAT,PERFTHRESH,PERFSTEP,

Your contact details with us are as below:

Email:michael.smith@green.com

Phone:BUS:+1.603.555.0100

If you have any questions about our customer policy or loyalty program, please call us at +1.650.555.0185. We appreciate your business and are here to help you with all your business needs.

Sincerely,

Vision Customer Care

Figure 54: Report

3.2.12.3 New Letter Template Creation Using Groovy Plugin Script

This section explains prerequisites required and steps to be followed to generate letter template using Groovy plugin.

Prerequisites

- Apache FOP in ORMB such as Online Letter Display Algorithm changes, etc. must be set up prior to generating Letter Template Creation Using Groovy Plugin.
- Groovy Plugin script will be used to extract required data and generate XML file.
- Online letter display extract algorithm will use FOP XSLT along with generated XML and call Apache FOP APIs to generate PDF report. Ensure that FOP XSLT name should be same as name of the letter template defined.

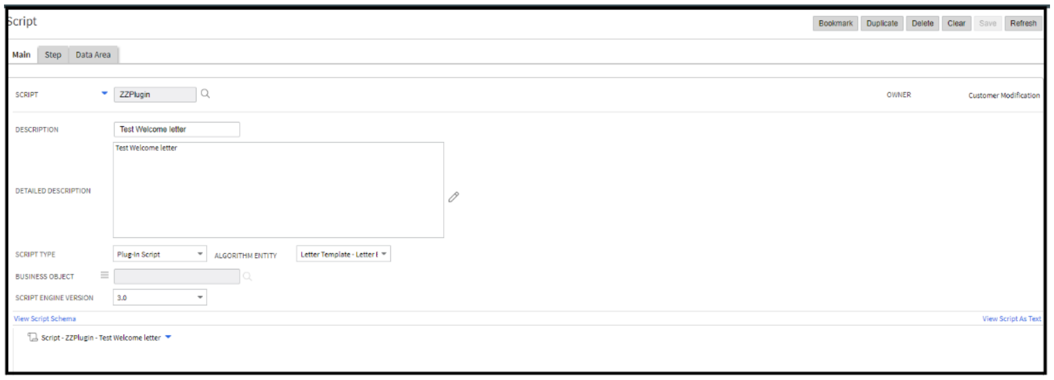
Creating new letter templates using Groovy Plugin involves below steps:

1. **Creating Groovy Plugin Script** - Creation of Plugin script is almost same as that of creating a Service Script, with some differences like:
   - There is no Schema definition for Plugin Script.
   - Plug-in script can be associated with Algorithm Spot.

   To create Groovy Plugin script:

   a. From the **Admin** menu, select **S** and then click **Script**. A sub-menu appears.

   b. Click **Add** option from the **Script** sub-menu.



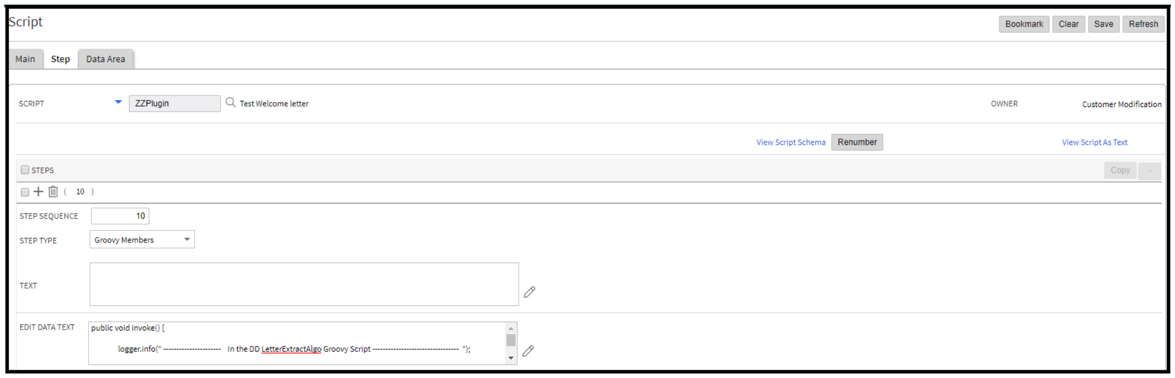
SCRIPT: ZZPlugin
OWNER: Customer Modification
DESCRIPTION: Test Welcome letter
DETAILED DESCRIPTION: Test Welcome letter
SCRIPT TYPE: Plug-In Script
ALGORITHM ENTITY: Letter Template - Letter E
BUSINESS OBJECT:
SCRIPT ENGINE VERSION: 3.0

Script - ZZPlugin - Test Welcome letter

Figure 55: Script Creation

   c. Select **Plug-in Script from Script Type** list, Letter Template - Letter Extract from the **Algorithm Entity** and Groovy from the **Script Engine Version** drop-down lists.

   d. Go to Step section, select Groovy Members from Step Type list, and paste your groovy code in **Edit Data Text** field. Ensure that you use '`public void invoke()`', as this method will be called when script is executed.



SCRIPT	ZZPlugin Test Welcome letter
OWNER	Customer Modification
STEPS	(10)
STEP SEQUENCE	10
STEP TYPE	Groovy Members
TEXT	
EDIT DATA TEXT	public void invoke() { logger.info(" -------------------- In the DD LetterExtractAlgo Groovy Script ------------------------------- ");

Figure 56: Script – Step Type

A sample code for Test Script is listed below:

```
public void invoke() {

           logger.info("  ----------------------         In  the  DD
LetterExtractAlgo Groovy Script ---------------------------------  ");

           String                        ccId                       =
this.customerContact.getId().getIdValue().toString();
           String                    letterTmplCd                   =
this.letterTemplate.getId().getIdValue().toString();

           logger.info("\n Customer Contact Id:" + ccId);
           logger.info("\n Letter Template Cd:" + letterTmplCd);

           String processDate = "";
           String ownerName = "";
           String ownerAddress1 = "";
           String ownerAddress2 = "";
           String ownerAddress3 = "";
           String ownerCity = "";
           String ownerPostal = "";
           String ownerState = "";
           String acctId = null;
           String emailId = null;
           String phone = null;
           java.lang.StringBuilder PIs = new java.lang.StringBuilder();
           String services = null;
           def pstmt = null;
           def row = null;
           def pstmt1 = null;
           def row1 = null;
           def pstmt2 = null;
           def rowlist = null;

           try {
                 pstmt = createPreparedStatement(
                           "select      trim(ACP.ACCT_ID)      ACCT_ID
,trim(PR.EMAILID) EMAIL,trim(PH.PHONE_TYPE_CD) || ':'||trim(PH.PHONE)
PHONE,"+
                           "PR.ADDRESS1           ADDRESS1,PR.ADDRESS2
ADDRESS2,PR.ADDRESS3 ADDRESS3,PR.CITY CITY, PR.STATE STATE,PR.POSTAL
POSTAL,TO_CHAR(CC.CC_DTTM,'DD/MON/YYYY') CC_DTTM " +
                           "FROM CI_CC CC, CI_PER PR, CI_PER_PHONE PH,
CI_ACCT_PER ACP " +
                           "WHERE CC.CC_ID=:cc_id " +
                           "AND CC.LTR_TMPL_CD=:ltr_tmpl_id " +
                           "AND CC.PER_ID=PR.PER_ID " +
                           "AND CC.PER_ID=PH.PER_ID " +
                           "AND CC.PER_ID=ACP.PER_ID", "Person Info");
                pstmt.bindString("cc_id", ccId, "CC_ID");
                pstmt.bindString("ltr_tmpl_id",          letterTmplCd,
"LTR_TMPL_CD");
                row = pstmt.firstRow();
                if (row != null) {
                      acctId = row.getString("ACCT_ID");
                      emailId = row.getString("EMAIL");
                      phone = row.getString("PHONE");
                      ownerAddress1 = row.getString("ADDRESS1");
                      ownerAddress2 = row.getString("ADDRESS2");
                      ownerAddress3 = row.getString("ADDRESS3");
                      ownerCity = row.getString("CITY");
                      ownerState = row.getString("STATE");
                      ownerPostal = row.getString("POSTAL");
                      processDate = row.getString("CC_DTTM");
                }
                pstmt1 = createPreparedStatement(
                           "select PN.ENTITY_NAME ENTITY_NAME " +
                           "from CI_CC CC, CI_PER PR, CI_PER_NAME PN "
+
                           "WHERE CC.CC_ID=:cc_id " +
                           "AND CC.LTR_TMPL_CD=:ltr_tmpl_id " +
                           "AND CC.PER_ID=PR.PER_ID " +
                           "AND CC.PER_ID=PN.PER_ID","Person Name");
                pstmt1.bindString("cc_id", ccId, "CC_ID");
                pstmt1.bindString("ltr_tmpl_id",         letterTmplCd,
"LTR_TMPL_CD");
                row1 = pstmt1.firstRow();

                if (row1 != null) {
                     ownerName = row1.getString("ENTITY_NAME");
                }
                pstmt2        =        createPreparedStatement("SELECT
DISTINCT(TRIM(PRICEITEM_CD)) PRICEITEM_CD " +
                           "FROM CI_PARTY PRTY, CI_PRICELIST_ASGN PLAS,
CI_PRICEASGN PRAS " +
                           "WHERE PRTY.PARTY_ID=:party_id " +
                           "AND PRTY.PARTY_TYPE_FLG='ACCT' " +
                           "AND PRTY.PARTY_UID=PLAS.PARTY_UID " +
                           "AND      PLAS.PRICELIST_ID=PRAS.OWNER_ID",
"Person Services");

                pstmt2.bindString("party_id", acctId, "PARTY_ID");

                 rowlist = pstmt2.list();

                for (def row2 : rowlist) {

                     PIs.append(row2.getString("PRICEITEM_CD"));
                     PIs.append(",");
                }
```

```
                    services = PIs.toString();
               } catch (Exception e) {
                    logger.error("Exception ", e);

               } finally {
                    try {


                         pstmt1.close();
                         pstmt2.close();
                    } catch (Exception e) {
                         logger.error("Error While Closing ResultSet ", e);
                    } finally {
                         pstmt1 = null;
                         pstmt2 = null;
                    }
               }

               logger.info("\n Account Id:" + acctId);
               logger.info("\n Email Id:" + emailId);
               logger.info("\n Phone:" + phone);
               logger.info("\n Services:" + services);
               logger.info("\n Name:" + ownerName);
               logger.info("\n Process Date:" + processDate);
               logger.info("\n Add 1:" + ownerAddress1);
               logger.info("\n Add 2:" + ownerAddress2);
               logger.info("\n Add 3:" + ownerAddress3);
               logger.info("\n City:" + ownerCity);
               logger.info("\n State:" + ownerState);
               logger.info("\n Postal:" + ownerPostal);

               /// Create XML Of Fields

               org.w3c.dom.Document            finalDocument            =
com.splwg.base.support.scripting.XMLUtils.createNewDomDocument();
```

```
           org.w3c.dom.Element                  root                  =
finalDocument.createElement("DOCUMENT");
           finalDocument.appendChild(root);


         org.w3c.dom.Element                 root1                  =
com.splwg.base.support.scripting.XMLUtils.addElement(root, "DOCSET");


           org.w3c.dom.Element             itemElement              =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
           itemElement.setAttribute("NAME", "ProcessDt");
           itemElement.setTextContent(processDate);


           org.w3c.dom.Element             itemElement1             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
           itemElement1.setAttribute("NAME", "OwnerFullNm");
           itemElement1.setTextContent(ownerName);
           org.w3c.dom.Element             itemElement2             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
           itemElement2.setAttribute("NAME", "OwnerAddrLine1");
           itemElement2.setTextContent(ownerAddress1);
           org.w3c.dom.Element             itemElement3             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
           itemElement3.setAttribute("NAME", "OwnerAddrLine2");

     itemElement3.setTextContent(ownerAddress2+","+ownerAddress3);
           org.w3c.dom.Element             itemElement4             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
           itemElement4.setAttribute("NAME", "OwnerAddrLine3");
     itemElement4.setTextContent(ownerCity+","+ownerState+","+ownerPos
tal);
           org.w3c.dom.Element             itemElement5             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
           itemElement5.setAttribute("NAME", "accountId");
           itemElement5.setTextContent(acctId);
           org.w3c.dom.Element             itemElement6             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
           itemElement6.setAttribute("NAME", "emailAddress");
           itemElement6.setTextContent(emailId);
```

```
          org.w3c.dom.Element             itemElement7             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
          itemElement7.setAttribute("NAME", "phone");
          itemElement7.setTextContent(phone);
          org.w3c.dom.Element             itemElement8             =
com.splwg.base.support.scripting.XMLUtils.addElement(root1, "FIELD");
          itemElement8.setAttribute("NAME", "services");
          itemElement8.setTextContent(services);
          String datFileName = null;
          String datFolderData = null;
          String inputFileName = null;


          datFileName = this.extractFileName + "/" + "extractxml" + "/"
+ ccId;
          inputFileName = this.extractFileName + "/" + "extractxml" +
"/" + ccId + ".xml";


          datFolderData = "XML_FILE_INPUT" + " " + inputFileName + "\n";
          String sourceXML = "";
          try {

     com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil
commonUtil
=com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil.Fact
ory.newInstance();
               sourceXML = finalDocument.documentElement as String;
               commonUtil.fileWrite(this.extractFileName   +   "/"   +
"extractxml", ccId, sourceXML, "xml");
               commonUtil.fileWrite(this.extractFileName   +   "/"   +
"extractxml", ccId,datFolderData,null);
          } catch (Exception e) {
               logger.error("Exception occurred" + e.getMessage());
          }
     }
```

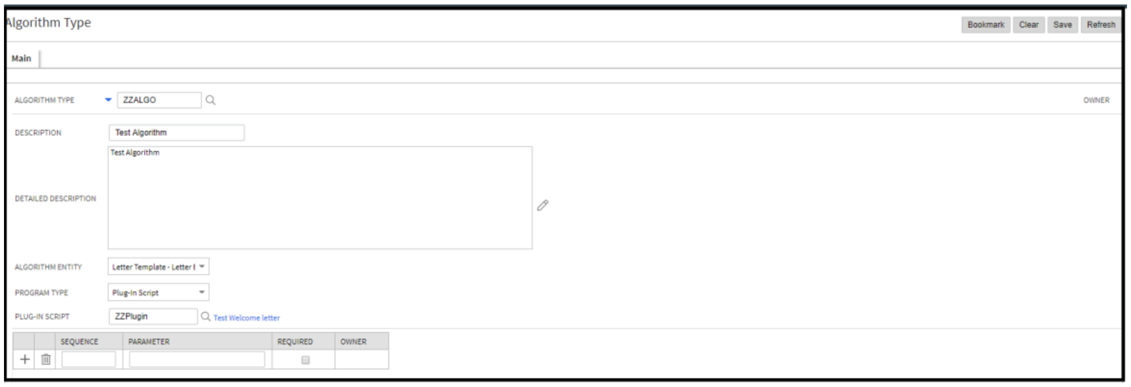
Note: You need to create two files in '`extractxml directory`', xml file and dat file. Dat file will have no extension and its content will be '`XML_FILE_INPUT <xml file path>`'

2. **Creating Algorithm** - Algorithms will be attached to letter template. When you click Display Letter button, the algorithm is executed, which in turn runs executable linked to it. To create algorithm:

a. From the **Admin** menu, select **A**, click **Algorithm Type**. A sub-menu appears.

b. Click **Add** option from the **Algorithm Type** sub-menu.

c. Select **Algorithm Entity** as ‘Letter Template-Letter Extract’, **Program Type** as Plug-In Script. Provide name of the script already created. Save the changes.



ALGORITHM TYPE: ZZALGO
OWNER
DESCRIPTION: Test Algorithm
DETAILED DESCRIPTION: Test Algorithm
ALGORITHM ENTITY: Letter Template - Letter E
PROGRAM TYPE: Plug-In Script
PLUG-IN SCRIPT: ZZPlugin Test Welcome letter

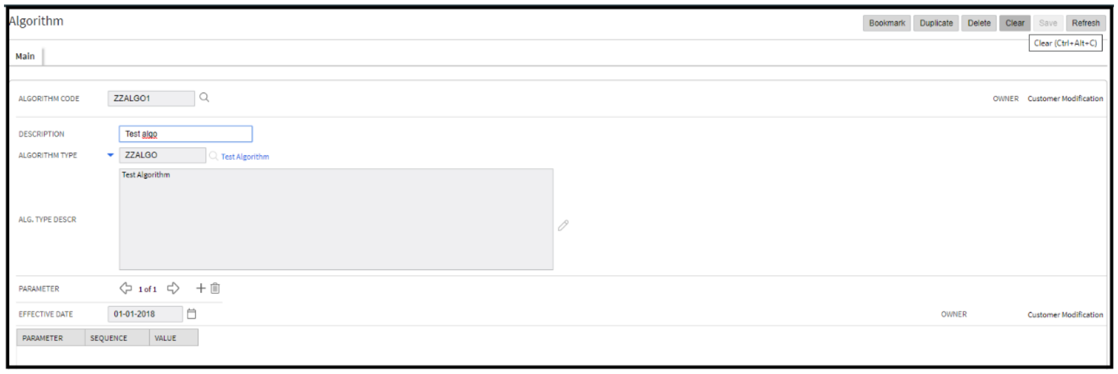
		SEQUENCE	PARAMETER	REQUIRED	OWNER
+	🗑			☐	

Figure 57: Algorithm

An Algorithm Type can have parameters. When you create algorithm, you can pass values to these parameters which will be provided to executing algorithm. To create Algorithm:

d. From the **Admin** menu, select **A** and then click **Algorithm Type**. A sub-menu appears.

e. Click **Add** option from the **Algorithm Type** sub-menu.



ALGORITHM CODE: ZZALGO1
OWNER: Customer Modification
DESCRIPTION: Test algo
ALGORITHM TYPE: ZZALGO Test Algorithm
ALG. TYPE DESCR: Test Algorithm
PARAMETER: 1 of 1
EFFECTIVE DATE: 01-01-2018
OWNER: Customer Modification

PARAMETER	SEQUENCE	VALUE

Figure 58: Algorithm

f. Provide Algorithm Name, and select Algorithm Type.

g. Save the changes.

Note: Groovy Plugin script gets all the required data and creates an XML. This XML is used by FOP algorithm (Online Letter Display Algorithm) to generate PDF.

A sample code for creating Test or Sample XML is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
<DOCSET>
<FIELD NAME="ProcessDt">20/NOV/2017</FIELD>
```

```
<FIELD NAME="OwnerFullNm">Smith, Michael</FIELD>
<FIELD NAME="OwnerAddrLine1"> 2-4-27 Dojima, Kita-ku </FIELD>
<FIELD NAME="OwnerAddrLine2">Osaka 530-0003</FIELD>
<FIELD NAME="OwnerAddrLine3">Japan</FIELD>
<FIELD NAME="accountId">2000000000</FIELD>
<FIELD NAME="emailAddress">michael.smith@green.com</FIELD>
<FIELD NAME="phone">BUS:+1.603.555.0100</FIELD>
<FIELD
NAME="services">PERFDISCOUNT,STATETAX,PERFFLAT,PERFTHRESH,PERFSTEP,</F
IELD>
</DOCSET>
</DOCUMENT>
```

A sample code for APACHE FOP XSLT to generate PDF is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
     <xsl:template match="/">
          <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
               <fo:layout-master-set>
                    <fo:simple-page-master master-name="simpleA4"
                            page-height="29.7cm"
                page-width="21cm"
                margin-top="1cm"
                margin-bottom="0.1cm"
                margin-left="0.8cm"
                margin-right="1.0cm" >
                        <fo:region-body  margin-top="2.5cm"  margin-
bottom="2.5cm"/>
                        <fo:region-before extent="2.0cm"/>
                        <fo:region-after extent="2.0cm"/>
                    </fo:simple-page-master>
               </fo:layout-master-set>
               <fo:page-sequence master-reference="simpleA4"
              initial-page-number="1">
```

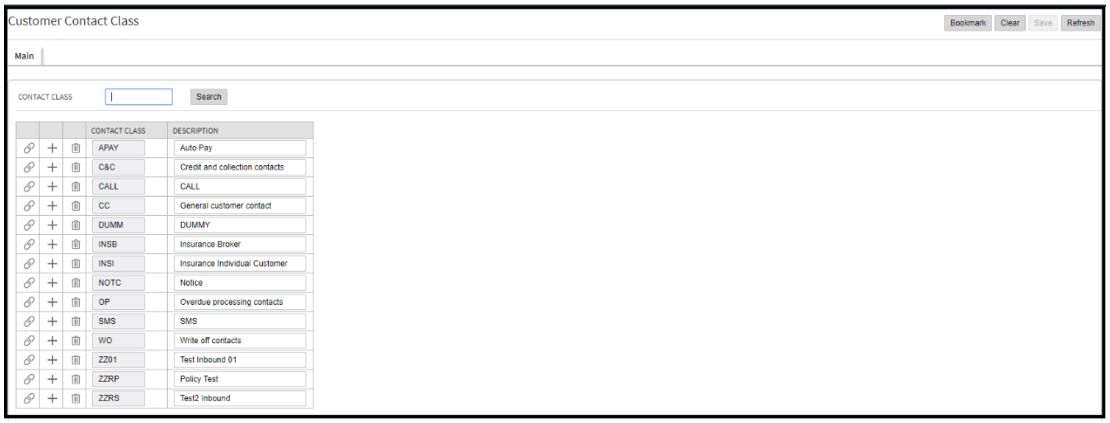
```
                    <fo:static-content flow-name="xsl-region-before">
                         <fo:block font-size="7pt">

                         </fo:block>
                    </fo:static-content>

                    <fo:static-content flow-name="xsl-region-after">


                         <fo:block      font-size="9.0pt"      font-
family="sans-serif"
                      padding-after="2.0pt"      space-before="2.0pt"
text-align="right"
                      border-top-style="ridge"         border-bottom-
width="0.5pt">


                         </fo:block>


                    </fo:static-content>


                    <fo:flow flow-name="xsl-region-body">
                         <xsl:apply-templates
select="DOCUMENT/DOCSET" />
                         <fo:block id="last-page"> </fo:block>
                    </fo:flow>
               </fo:page-sequence>
          </fo:root>
     </xsl:template>



     <xsl:template match="DOCUMENT/DOCSET">
          <fo:block text-align="right" font-size="9px">
               <xsl:value-of select="//FIELD[@NAME='ProcessDt']"/>
          </fo:block>


          <fo:block    text-align="right"    font-size="9px"    space-
before="0.37in" >Vision Corporation</fo:block>
```

```
          <fo:block  text-align="right"  font-size="9px"  >401  Island
Parkway, Redwood Shores, CA 94065</fo:block>


          <fo:block text-align="left" font-size="9px">
               <xsl:value-of select="//FIELD[@NAME='OwnerFullNm']"/>
          </fo:block>
          <fo:block text-align="left" font-size="9px">
               <xsl:value-of
select="//FIELD[@NAME='OwnerAddrLine1']"/>
          </fo:block>
          <fo:block text-align="left" font-size="9px">
               <xsl:value-of
select="//FIELD[@NAME='OwnerAddrLine2']"/>
          </fo:block>
          <fo:block text-align="left" font-size="9px">
               <xsl:value-of
select="//FIELD[@NAME='OwnerAddrLine3']"/>
          </fo:block>
          <fo:block    text-align="left"    font-size="9px"    space-
before="0.37in">Re: Welcome</fo:block>
          <fo:block    text-align="left"    font-size="9px"    space-
before="0.10in">Customer                        Account:<xsl:value-of
select="//FIELD[@NAME='accountId']"/>
          </fo:block>
          <fo:block    text-align="left"    font-size="9px"    space-
before="0.37in">Dear                                    <xsl:value-of
select="//FIELD[@NAME='OwnerFullNm']"/>,</fo:block>


          <fo:block    text-align="left"    font-size="9px"    space-
before="0.10in">Thank you for Choosing Vision Products. We welcome you
to our customer loyalty program.
          </fo:block>
          <fo:block    text-align="left"    font-size="9px"    space-
before="0.10in">You    are    currently    subscribed    to    below
services.</fo:block>
          <fo:block         text-align="left"         font-size="9px"
>Services:<xsl:value-of select="//FIELD[@NAME='services']"/>
          </fo:block>
          <fo:block    text-align="left"    font-size="9px"    space-
before="0.10in">Your contact details with us are as below:</fo:block>
```

```
           <fo:block         text-align="left"         font-size="9px"
>Email:<xsl:value-of select="//FIELD[@NAME='emailAddress']"/>
           </fo:block>
           <fo:block         text-align="left"         font-size="9px"
>Phone:<xsl:value-of select="//FIELD[@NAME='phone']"/>
           </fo:block>


           <fo:block    text-align="left"    font-size="9px"    space-
before="0.10in">If you have any questions about our customer policy or
loyalty program, please call us at +1.650.555.0185. We appreciate your
business   and   are   here   to   help   you   with   all   your   business
needs.</fo:block>
           <fo:block    text-align="left"    font-size="9px"    space-
before="0.37in">Sincerely,</fo:block>
           <fo:block text-align="left" font-size="9px"> Vision Customer
Care.</fo:block>
      </xsl:template>
      <!--Template -->
      <xsl:attribute-set name="foCellAttributeSet">
           <xsl:attribute name="padding-right">2mm</xsl:attribute>
           <xsl:attribute name="padding-left">2mm</xsl:attribute>
           <xsl:attribute name="padding-top">2mm</xsl:attribute>
           <xsl:attribute name="padding-bottom">2mm</xsl:attribute>
      </xsl:attribute-set>
</xsl:stylesheet>
```

Name of the FOP XSLT will be same as letter template code. For example, ‘DD-WELCOME.xsl’.

3. **Creating Customer Contact** - Letters are generated over customer contact. You can add this contact type to existing Contact Class or create new Contact Class. To create new Contact Class:
   a. From the **Admin** menu, select **C** and then click **Customer Contact Class**. A sub-menu appears.
   b. Click **Add** option from the Customer Contact Class sub-menu.
   c. Add a new Class.



CONTACT CLASS	DESCRIPTION
APAY	Auto Pay
C&C	Credit and collection contacts
CALL	CALL
CC	General customer contact
DUMM	DUMMY
INSB	Insurance Broker
INSI	Insurance Individual Customer
NOTC	Notice
OP	Overdue processing contacts
SMS	SMS
WO	Write off contacts
ZZ01	Test Inbound 01
ZZRP	Policy Test
ZZRS	Test2 Inbound

Figure 59: Customer Contact

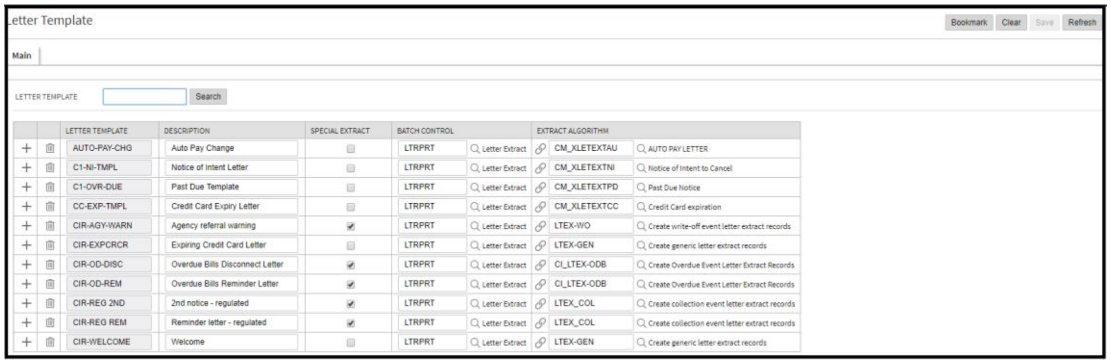
d. Click **Save**.

4. **Creating Letter template -** You need to create letter template to link the algorithm. To create letter template:

a. From the **Admin** menu, select **L** and then click **Letter Template**.

Note: The letter template code should be same as FOP XSL. For example: **'DD-WELCOME'**

b. Add new letter template and link the algorithm to the created letter template.



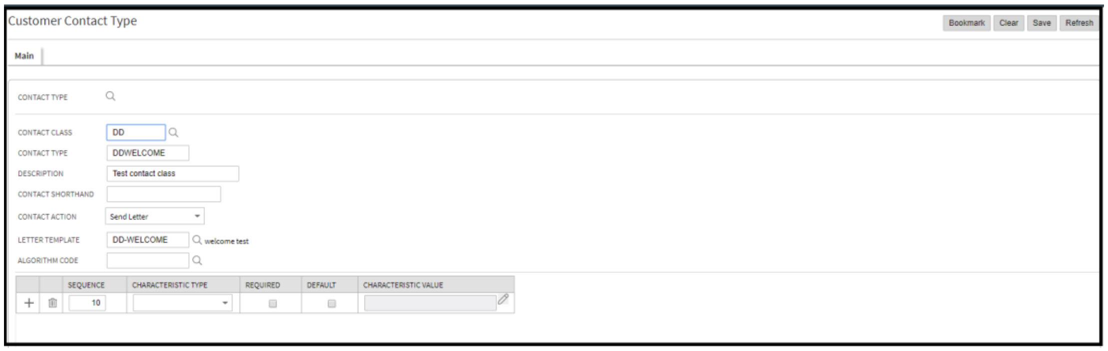
LETTER TEMPLATE	DESCRIPTION	SPECIAL EXTRACT	BATCH CONTROL		EXTRACT ALGORITHM	
AUTO-PAY-CHG	Auto Pay Change	☐	LTRPRT	Letter Extract	CM_XLETEXTAU	AUTO PAY LETTER
C1-NI-TMPL	Notice of Intent Letter	☐	LTRPRT	Letter Extract	CM_XLETEXTNI	Notice of Intent to Cancel
C1-OVR-DUE	Past Due Template	☐	LTRPRT	Letter Extract	CM_XLETEXTPD	Past Due Notice
CC-EXP-TMPL	Credit Card Expiry Letter	☐	LTRPRT	Letter Extract	CM_XLETEXTCC	Credit Card expiration
CIR-AGY-WARN	Agency referral warning	✓	LTRPRT	Letter Extract	LTEX-WO	Create write-off event letter extract records
CIR-EXPCRCR	Expiring Credit Card Letter	☐	LTRPRT	Letter Extract	LTEX-GEN	Create generic letter extract records
CIR-OD-DISC	Overdue Bills Disconnect Letter	✓	LTRPRT	Letter Extract	CI_LTEX-ODB	Create Overdue Event Letter Extract Records
CIR-OD-REM	Overdue Bills Reminder Letter	✓	LTRPRT	Letter Extract	CI_LTEX-ODB	Create Overdue Event Letter Extract Records
CIR-REG 2ND	2nd notice - regulated	✓	LTRPRT	Letter Extract	LTEX_COL	Create collection event letter extract records
CIR-REG REM	Reminder letter - regulated	✓	LTRPRT	Letter Extract	LTEX_COL	Create collection event letter extract records
CIR-WELCOME	Welcome	☐	LTRPRT	Letter Extract	LTEX-GEN	Create generic letter extract records

Figure 60: Create Letter template

c. To create Contact Type,

i. From the **Admin** menu, select **C** and then click **Customer Contact Type**. A sub-menu appears.

ii. Click **Add** option from the Customer Contact Type sub-menu.



CONTACT TYPE

CONTACT CLASS	DD
CONTACT TYPE	DDWELCOME
DESCRIPTION	Test contact class
CONTACT SHORTHAND	
CONTACT ACTION	Send Letter
LETTER TEMPLATE	DD-WELCOME welcome test
ALGORITHM CODE	

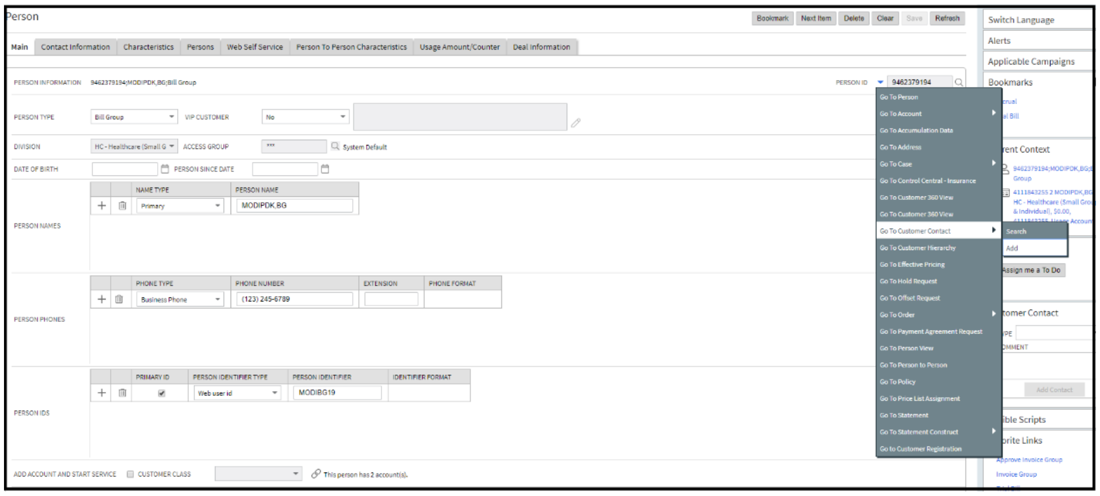
SEQUENCE	CHARACTERISTIC TYPE	REQUIRED	DEFAULT	CHARACTERISTIC VALUE
10				

Figure 61: Contact Type

iii. Provide Contact Class, Contact Type Code, and Letter Template. Click **Save**.

8. **Creating Customer Contact** - To create customer contact, navigate to Open person details.

   a. Click on person context menu, go to customer contact and add.



PERSON INFORMATION 9462379194;MODIPDK,BG;Bill Group

PERSON ID 9462379194

PERSON TYPE Bill Group VIP CUSTOMER No

DIVISION HC - Healthcare (Small G ACCESS GROUP *** System Default

DATE OF BIRTH PERSON SINCE DATE

PERSON NAMES

NAME TYPE	PERSON NAME
Primary	MODIPDK,BG

PERSON PHONES

PHONE TYPE	PHONE NUMBER	EXTENSION	PHONE FORMAT
Business Phone	(123) 245-6789		

PERSON IDS

PRIMARY ID	PERSON IDENTIFIER TYPE	PERSON IDENTIFIER	IDENTIFIER FORMAT
✓	Web user id	MODIBG19	

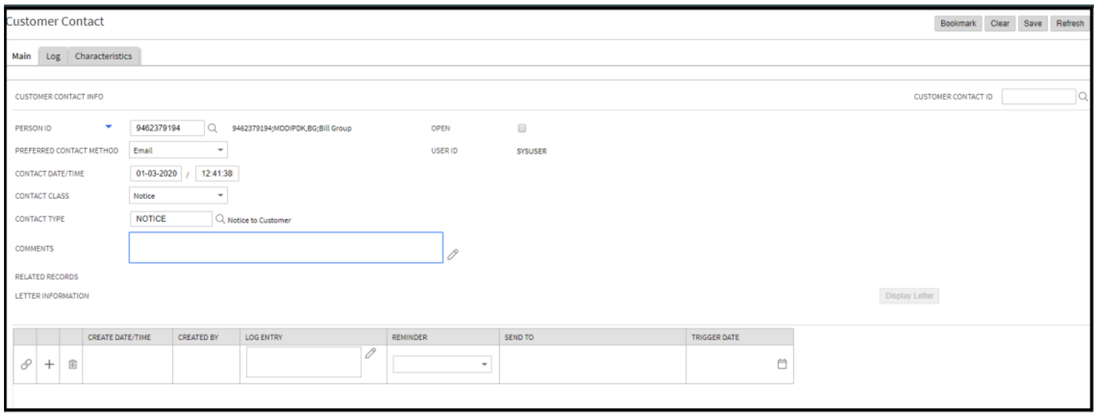
ADD ACCOUNT AND START SERVICE CUSTOMER CLASS This person has 2 account(s).

9462379194;MODIPDK,BG;Bill Group

4111843255 2 MODIPDK,BG HC - Healthcare (Small Group & Individual), $0.00,

Figure 62: Customer Contact

   b. Provide Customer Contact Class, Contact Type and save the changes.



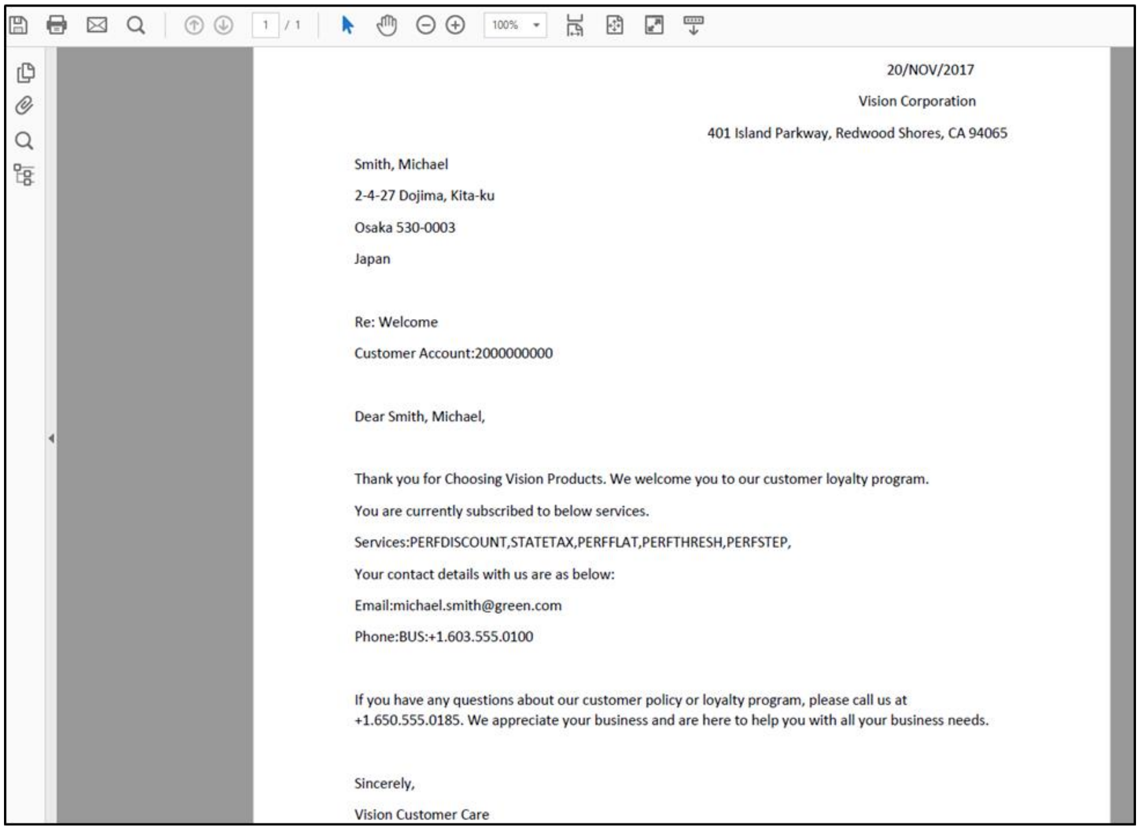
CUSTOMER CONTACT INFO CUSTOMER CONTACT ID

PERSON ID	9462379194 9462379194;MODIPDK,BG;Bill Group	OPEN	
PREFERRED CONTACT METHOD	Email	USER ID	SYSUSER
CONTACT DATE/TIME	01-03-2020 / 12:41:38		
CONTACT CLASS	Notice		
CONTACT TYPE	NOTICE Notice to Customer		
COMMENTS			
RELATED RECORDS			
LETTER INFORMATION			

CREATE DATE/TIME	CREATED BY	LOG ENTRY	REMINDER	SEND TO	TRIGGER DATE

Figure 63: Customer Contact

   c. Click on **Display Letter** button. The output is displayed.



20/NOV/2017

Vision Corporation

401 Island Parkway, Redwood Shores, CA 94065

Smith, Michael

2-4-27 Dojima, Kita-ku

Osaka 530-0003

Japan

Re: Welcome

Customer Account:2000000000

Dear Smith, Michael,

Thank you for Choosing Vision Products. We welcome you to our customer loyalty program.

You are currently subscribed to below services.

Services:PERFDISCOUNT,STATETAX,PERFFLAT,PERFTHRESH,PERFSTEP,

Your contact details with us are as below:

Email:michael.smith@green.com

Phone:BUS:+1.603.555.0100

If you have any questions about our customer policy or loyalty program, please call us at +1.650.555.0185. We appreciate your business and are here to help you with all your business needs.

Sincerely,

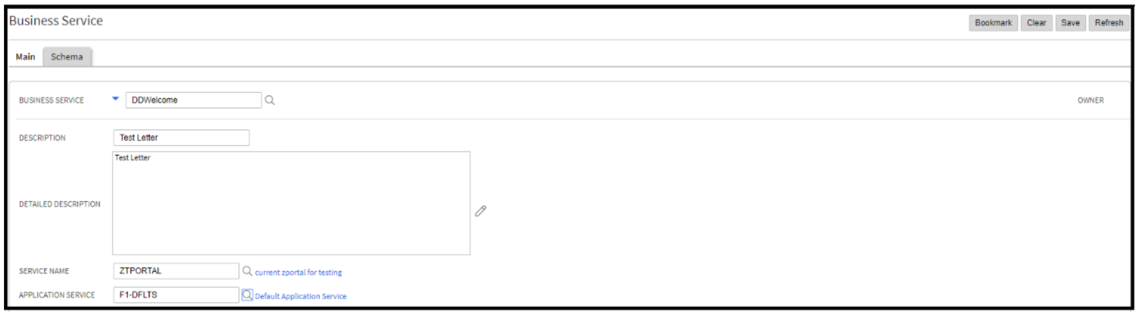
Vision Customer Care

Figure 64: Report

3.2.12.4 New Letter Using Custom Business Service

You can create new letter template using Business Service. Creating new letter templates using custom business services involves below steps:

1. Creating Business Service, which will extract required data and send it back to generic letter extract algorithm with its Schema.



BUSINESS SERVICE	DDWelcome
OWNER	
DESCRIPTION	Test Letter
DETAILED DESCRIPTION	Test Letter
SERVICE NAME	ZTPORTAL current zportal for testing
APPLICATION SERVICE	F1-DFLTS Default Application Service

Figure 65: Business Service

A sample code for Business Service Schema is listed below:

```
<schema pageAction="read">
    <customerContactId mapField="CC_ID"/>
    <letterTemplate mapField="LTR_TMPL_CD"/>
    <accountId mapField="ACCT_ID"/>
    <emailAddress mapField="EMAILID"/>
    <phone mapField="PHONE"/>
    <description mapField="DESCR"/>
</schema>
```

You must define two fields, ‘customerContactId mapField’ and ‘letterTemplate mapField’ in Business Service Schema. These fields are passed as input to Business Service by Generic Letter Extract Algorithm. The rest of the fields in schema are extracted by Business Service and returned to Letter Extract Algorithm to prepare XML to be used by FOP for PDF generation.

Note: To implement new Letter Template using Business Service, you need two XSLs:

Generic Letter Extract Algorithm to convert business service response to XML

FOP XSL, used by Apache FOP to generate PDF

A sample code for XSL for letter extract algorithm is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<DOCUMENT                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"    VERSION="11.5"
TYPE="RPWIP">
<DOCSET NAME="">
<FIELD          NAME="ProcessDt"><xsl:value-of           select='//C1-
BaseLetterExt/processDate'/></FIELD>
<FIELD          NAME="OwnerFullNm"><xsl:value-of          select='//C1-
BaseLetterExt/ownerName'/></FIELD>
<FIELD        NAME="OwnerAddrLine1"><xsl:value-of        select='//C1-
BaseLetterExt/ownerAddress1'/></FIELD>
<xsl:if test="//C1-BaseLetterExt[ownerAddress2 = '' and ownerAddress3 =
'']">
     <FIELD NAME="OwnerAddrLine2"></FIELD>
</xsl:if>
<xsl:if test="//C1-BaseLetterExt[ownerAddress2 != '' or ownerAddress3 !=
'']">
     <FIELD      NAME="OwnerAddrLine2"><xsl:value-of      select='//C1-
BaseLetterExt/ownerAddress2'/>,       <xsl:value-of       select='//C1-
BaseLetterExt/ownerAddress3'/></FIELD>
</xsl:if>
```

```
<xsl:if test="//C1-BaseLetterExt[city = '' and state = '' and postal =
'']">
     <FIELD NAME="OwnerAddrLine3"></FIELD>
</xsl:if>
<xsl:if test="//C1-BaseLetterExt[city != '' or state != '' or postal !=
'']">
     <FIELD      NAME="OwnerAddrLine3"><xsl:value-of      select='//C1-
BaseLetterExt/city'/>,           <xsl:value-of           select='//C1-
BaseLetterExt/state'/>,          <xsl:value-of           select='//C1-
BaseLetterExt/postal'/></FIELD>
</xsl:if>
                        Custom Business Service
<FIELD           NAME="accountId"><xsl:value-of           select='//DD-
WelcomeLetterBS/accountId'/></FIELD>
<FIELD          NAME="emailAddress"><xsl:value-of         select='//DD-
WelcomeLetterBS/emailAddress'/></FIELD>
<FIELD             NAME="phone"><xsl:value-of             select='//DD-
WelcomeLetterBS/phone'/></FIELD>
<FIELD           NAME="services"><xsl:value-of            select='//DD-
WelcomeLetterBS/description'/></FIELD>


</DOCSET>
</DOCUMENT>
</xsl:template>
</xsl:stylesheet>
```

Here, Letter Extract Algorithm internally calls another Business Service **C1-BaseLetterExt**, which provides basic person information. These fields are added in XSL. Once the fields are added in XSL, the fields provided by custom Business Service are added in the algorithm. (Refer Custom Business Service section in above sample code)

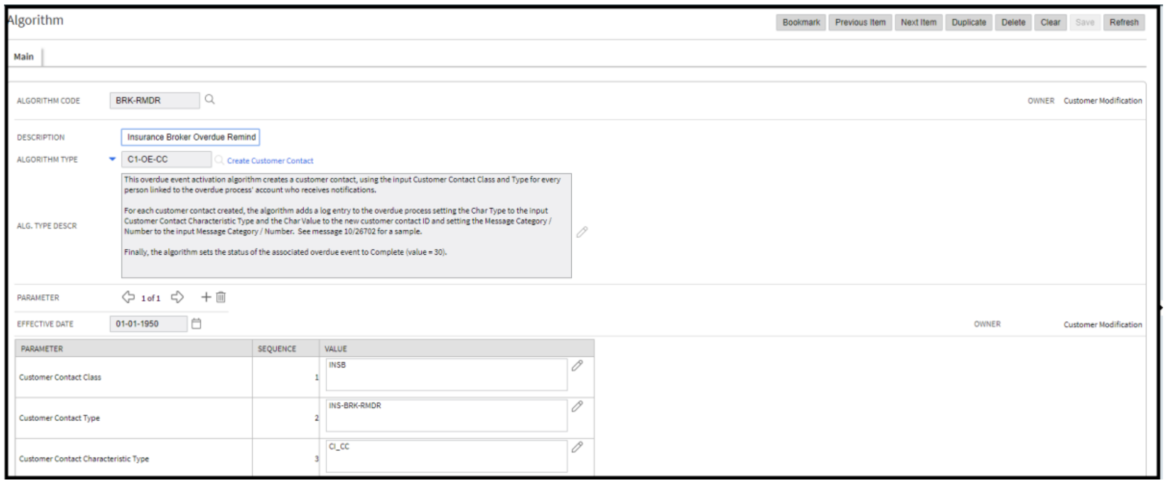
A sample code for XML generated by Letter Extract Algorithm from XSL and Business Service Response is listed below:

```
<?xml            version="1.0"            encoding="UTF-8"?><DOCUMENT
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"             TYPE="RPWIP"
VERSION="11.5">
<DOCSET NAME="">
<FIELD NAME="ProcessDt">November 20, 2017</FIELD>
<FIELD NAME="OwnerFullNm">Michael Smith</FIELD>
<FIELD NAME="OwnerAddrLine1">401 Island Parkway</FIELD>
```

```
<FIELD NAME="OwnerAddrLine2"> Redwood Shores</FIELD>

<FIELD NAME="OwnerAddrLine3">CA, 94065</FIELD>

<FIELD NAME="accountId">2000000000</FIELD>

<FIELD NAME="emailAddress">michael.smith@starwars.com</FIELD>

<FIELD NAME="phone">BUS: +1.603.555.0200</FIELD>

<FIELD
NAME="services">PERFDISCOUNT,STATETAX,PERFFLAT,PERFTHRESH,PERFSTEP</FI
ELD>

</DOCSET>

</DOCUMENT>
```

To setup algorithm,

1. Create new letter. Extract algorithm of Type generic Letter Extract Algorithm.
2. From the **Admin** menu, select **A,** and then click **Algorithm Type**. A sub-menu appears.
3. Click **Add** option from the **Algorithm Type** sub-menu.
4. Provide newly created business service in Letter Specific Business Service Field.
5. Provide complete file path of XSL used by Letter Extract Algorithm in 'Name of XSL file for generating output XML' field.
6. Save the changes.



ALGORITHM CODE: BRK-RMDR
OWNER: Customer Modification
DESCRIPTION: Insurance Broker Overdue Remind
ALGORITHM TYPE: C1-OE-CC Create Customer Contact
ALG. TYPE DESCR: This overdue event activation algorithm creates a customer contact, using the input Customer Contact Class and Type for every person linked to the overdue process' account who receives notifications.

For each customer contact created, the algorithm adds a log entry to the overdue process setting the Char Type to the input Customer Contact Characteristic Type and the Char Value to the new customer contact ID and setting the Message Category / Number to the input Message Category / Number. See message 10/26702 for a sample.

Finally, the algorithm sets the status of the associated overdue event to Complete (value = 30).

PARAMETER: 1 of 1
EFFECTIVE DATE: 01-01-1950
OWNER: Customer Modification

PARAMETER	SEQUENCE	VALUE
Customer Contact Class	1	INSB
Customer Contact Type	2	INS-BRK-RMDR
Customer Contact Characteristic Type	3	CI_CC

Figure 66: Letter Extract Algorithm

A sample code for APACHE FOP XSLT – required by FOP algorithm (online letter display algorithm) to generate PDF is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0"

    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">

     <xsl:template match="/">

           <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

```
                <fo:layout-master-set>
                     <fo:simple-page-master master-name="simpleA4"
                              page-height="29.7cm"
                 page-width="21cm"
                 margin-top="1cm"
                 margin-bottom="0.1cm"
                 margin-left="0.8cm"
                 margin-right="1.0cm" >
                          <fo:region-body  margin-top="2.5cm"  margin-
bottom="2.5cm"/>
                          <fo:region-before extent="2.0cm"/>
                          <fo:region-after extent="2.0cm"/>
                     </fo:simple-page-master>
                </fo:layout-master-set>
                <fo:page-sequence master-reference="simpleA4"
                initial-page-number="1">

                     <fo:static-content flow-name="xsl-region-before">
                          <fo:block font-size="7pt">

                          </fo:block>
                     </fo:static-content>


                     <fo:static-content flow-name="xsl-region-after">
                          <fo:block      font-size="9.0pt"      font-
family="sans-serif"
                       padding-after="2.0pt"     space-before="2.0pt"
text-align="right"
                       border-top-style="ridge"        border-bottom-
width="0.5pt">
                          </fo:block>
                     </fo:static-content>


                     <fo:flow flow-name="xsl-region-body">
                          <xsl:apply-templates
select="DOCUMENT/DOCSET" />
```

```
                          <fo:block id="last-page"> </fo:block>
                     </fo:flow>
                </fo:page-sequence>
           </fo:root>
      </xsl:template>

      <xsl:template match="DOCUMENT/DOCSET">
           <fo:block text-align="right" font-size="9px">
                <xsl:value-of select="//FIELD[@NAME='ProcessDt']"/>
           </fo:block>

           <fo:block    text-align="right"    font-size="9px"    space-
before="0.37in" >Vision Corporation</fo:block>
           <fo:block  text-align="right"  font-size="9px"  >401  Island
Parkway, Redwood Shores, CA 94065</fo:block>

           <fo:block text-align="left" font-size="9px">
                <xsl:value-of select="//FIELD[@NAME='OwnerFullNm']"/>
           </fo:block>
           <fo:block text-align="left" font-size="9px">
                <xsl:value-of
select="//FIELD[@NAME='OwnerAddrLine1']"/>
           </fo:block>
           <fo:block text-align="left" font-size="9px">
                <xsl:value-of
select="//FIELD[@NAME='OwnerAddrLine2']"/>
           </fo:block>
           <fo:block text-align="left" font-size="9px">
                <xsl:value-of
select="//FIELD[@NAME='OwnerAddrLine3']"/>
           </fo:block>
           <fo:block    text-align="left"    font-size="9px"    space-
before="0.37in">Re: Welcome</fo:block>
           <fo:block    text-align="left"    font-size="9px"    space-
before="0.10in">CustomerAccount:<xsl:value-of
select="//FIELD[@NAME='accountId']"/>
           </fo:block>
```

```
          <fo:block    text-align="left"    font-size="9px"    space-before="0.37in">Dear <xsl:value-of select="//FIELD[@NAME='OwnerFullNm']"/>,</fo:block>


          <fo:block    text-align="left"    font-size="9px"    space-before="0.10in">Thank you for Choosing Vision Products. We welcome you to our customer loyalty program.

          </fo:block>

          <fo:block    text-align="left"    font-size="9px"    space-before="0.10in">You are currently subscribed to below services.</fo:block>

          <fo:block         text-align="left"         font-size="9px" >Services:<xsl:value-of select="//FIELD[@NAME='services']"/>

          </fo:block>

          <fo:block    text-align="left"    font-size="9px"    space-before="0.10in">Your contact details with us are as below:</fo:block>

          <fo:block         text-align="left"         font-size="9px" >Email:<xsl:value-of select="//FIELD[@NAME='emailAddress']"/>

          </fo:block>

          <fo:block         text-align="left"         font-size="9px" >Phone:<xsl:value-of select="//FIELD[@NAME='phone']"/>

          </fo:block>


          <fo:block    text-align="left"    font-size="9px"    space-before="0.10in">If you have any questions about our customer policy or loyalty program, please call us at +1.650.555.0185. We appreciate your business and are here to help you with all your business needs.</fo:block>

          <fo:block    text-align="left"    font-size="9px"    space-before="0.37in">Sincerely,</fo:block>

          <fo:block text-align="left" font-size="9px"> Vision Customer Care.</fo:block>

     </xsl:template>

     <!--Template -->

     <xsl:attribute-set name="foCellAttributeSet">

          <xsl:attribute name="padding-right">2mm</xsl:attribute>

          <xsl:attribute name="padding-left">2mm</xsl:attribute>

          <xsl:attribute name="padding-top">2mm</xsl:attribute>

          <xsl:attribute name="padding-bottom">2mm</xsl:attribute>

     </xsl:attribute-set>
```

```
</xsl:stylesheet>
```

1. Name of the FOP XSLT will be same as letter template code. For example, DD-WELCOME.xsl.
2. Create 'Customer Contact Class', 'Customer Contact Type', 'Letter Template' respectively.
3. Create Customer Contact using new contact type On person and generate letter.

3.3 XML Generation

The pre-requisite for the Wrapper API is DOM Object. XML record generation is a two-step process:

- Conversion in XML
- Writing a File

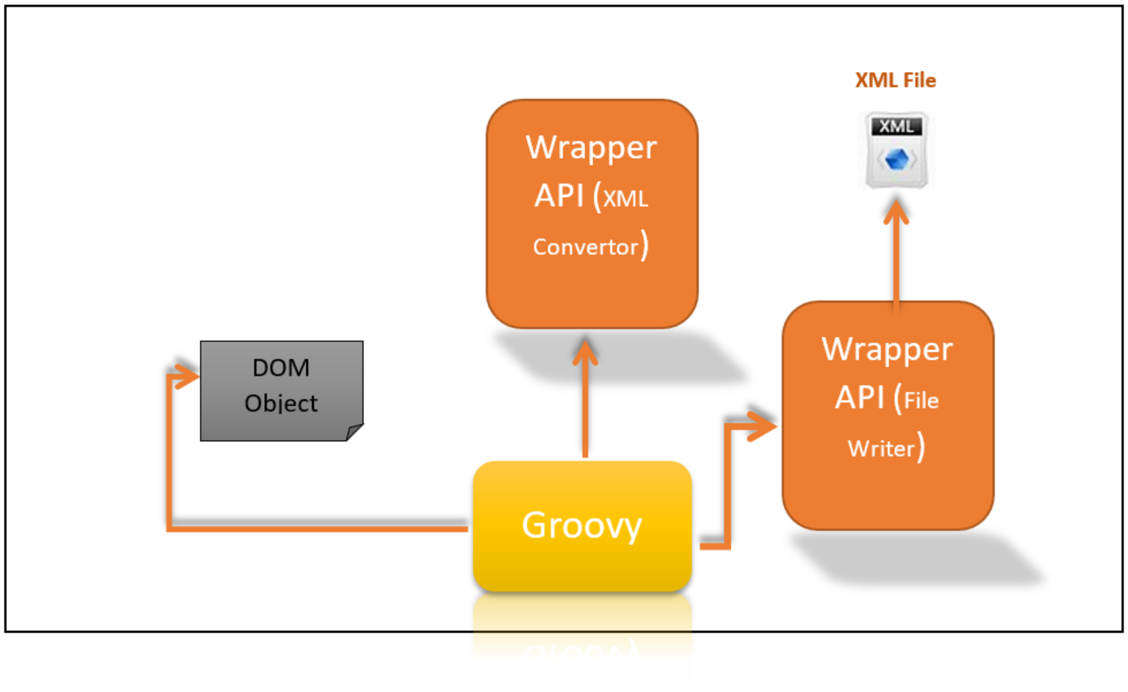


Figure 67: XML Generation Process

3.3.1 Conversion in XML

A wrapper API 'getXMLRecord()' receives 'Document Object' as an input. The Document object follows standard tree structure consisting of element nodes and text nodes. The API function will process the Document object and return generated XML content as string for the same document object.

For every record, API function is called and returned result will be added to one final XML string.

3.3.2 Writing a File

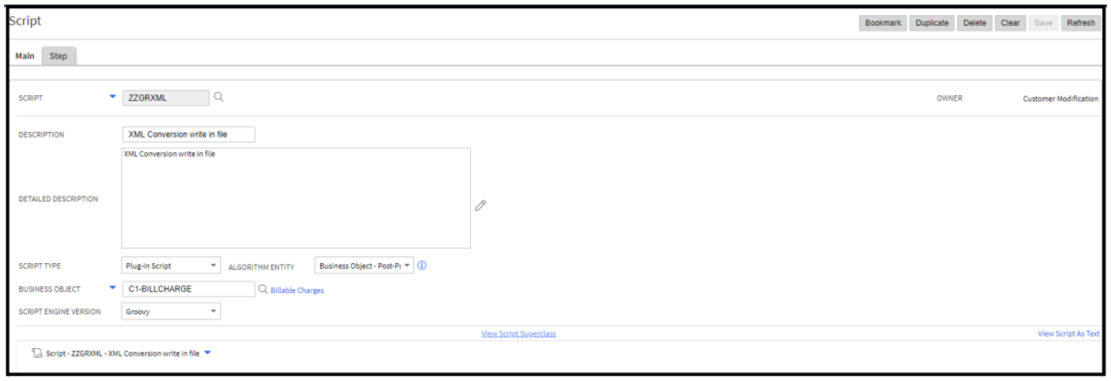
In this step, XML string is passed along with file name, file path, file content and file extension to API function **fileWrite()** which writes the given file content in XML format at a desired path.

The below is the example of groovy script for XML format.

3.3.3 XML Example

An example of a script converted in xml format is listed below:

- **Script**: ZZGRXML
- **Script Type**: Plug-In Script
- **Script Engine Version**: Groovy



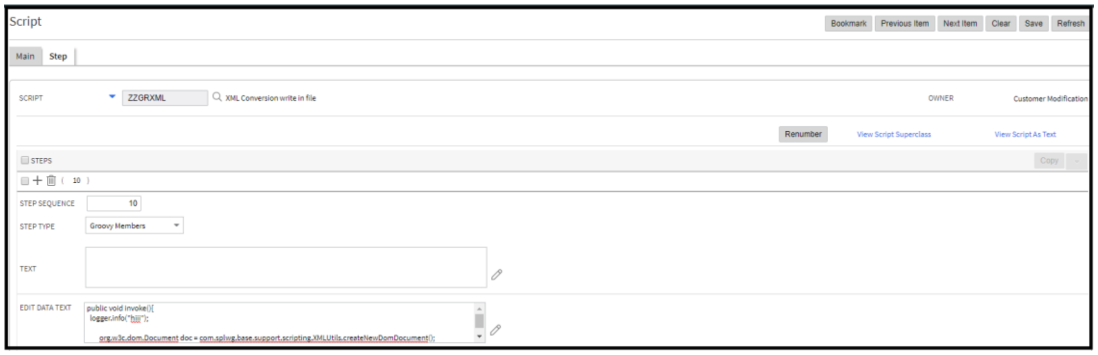
Script

SCRIPT	ZZGRXML
OWNER	Customer Modification
DESCRIPTION	XML Conversion write in file
DETAILED DESCRIPTION	XML Conversion write in file
SCRIPT TYPE	Plug-In Script
ALGORITHM ENTITY	Business Object - Post-Pı
BUSINESS OBJECT	C1-BILLCHARGE (Billable Charges)
SCRIPT ENGINE VERSION	Groovy

Script - ZZGRXML - XML Conversion write in file

Figure 68: Script Example

Steps: It has only one Step Type: ‘Groovy Members’



Script

SCRIPT	ZZGRXML (XML Conversion write in file)
OWNER	Customer Modification
STEP SEQUENCE	10
STEP TYPE	Groovy Members
TEXT	

EDIT DATA TEXT

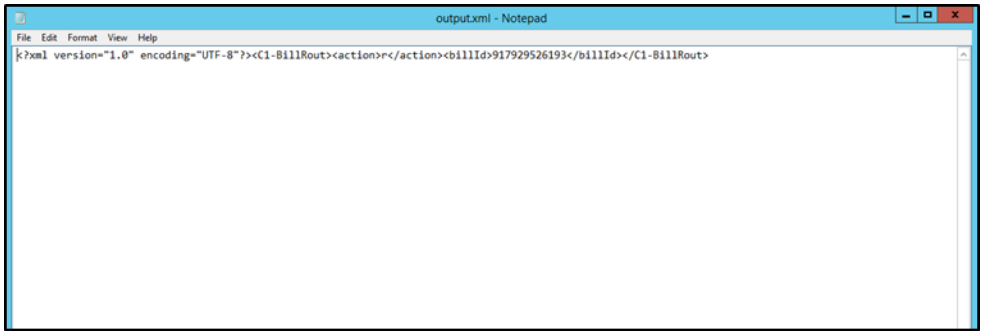
```
public void invoke(){
 logger.info("hiii");

    org.w3c.dom.Document doc = com.splwg.base.support.scripting.XMLUtils.createNewDomDocument();
```

Figure 69: Script - Step

- **Step Sequence**: 10. This value can be any unique integer value.
- **Step Type**: Groovy Members
- **Edit Data Text**: A sample code is listed below:

```
public void invoke(){
  com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil
commonUtil                                                        =
com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil.Facto
ry.newInstance();
              //Set up the DOM document.


        org.w3c.dom.Document                   doc                =
com.splwg.base.support.scripting.XMLUtils.createNewDomDocument();
        org.w3c.dom.Element root = doc.createElement("C1-BillRout");
        doc.appendChild(root);

        org.w3c.dom.Element                itemElement                  =
com.splwg.base.support.scripting.XMLUtils.addElement(root, "action");
        itemElement.setTextContent("r");
                        org.w3c.dom.Element       itemElement2         =
com.splwg.base.support.scripting.XMLUtils.addElement(root, "billId");
        itemElement2.setTextContent("917929526193");
        String xmlString = "";
        try {
            xmlString = commonUtil.getXMLRecord(doc);
        } catch (Exception e) {
        }

        String outputFilePath = "D:/spl/C1";
        String outputFileName = "output";
        String fileContent = xmlString;
        String fileExtension = "xml";

        commonUtil.fileWrite(outputFilePath,            outputFileName,
fileContent, fileExtension);
}
```

When the script is trigger by any means, it will convert the document object to xml string and write it in output.xml file.



```
<?xml version="1.0" encoding="UTF-8"?><C1-BillRout><action>r</action><billId>917929526193</billId></C1-BillRout>
```

Figure 70: Output

3.4 JSON Generation

The pre-requisite for the Wrapper API is Map Object. JSON record generation is a two-step process:

- Conversion in JSON
- Writing a File

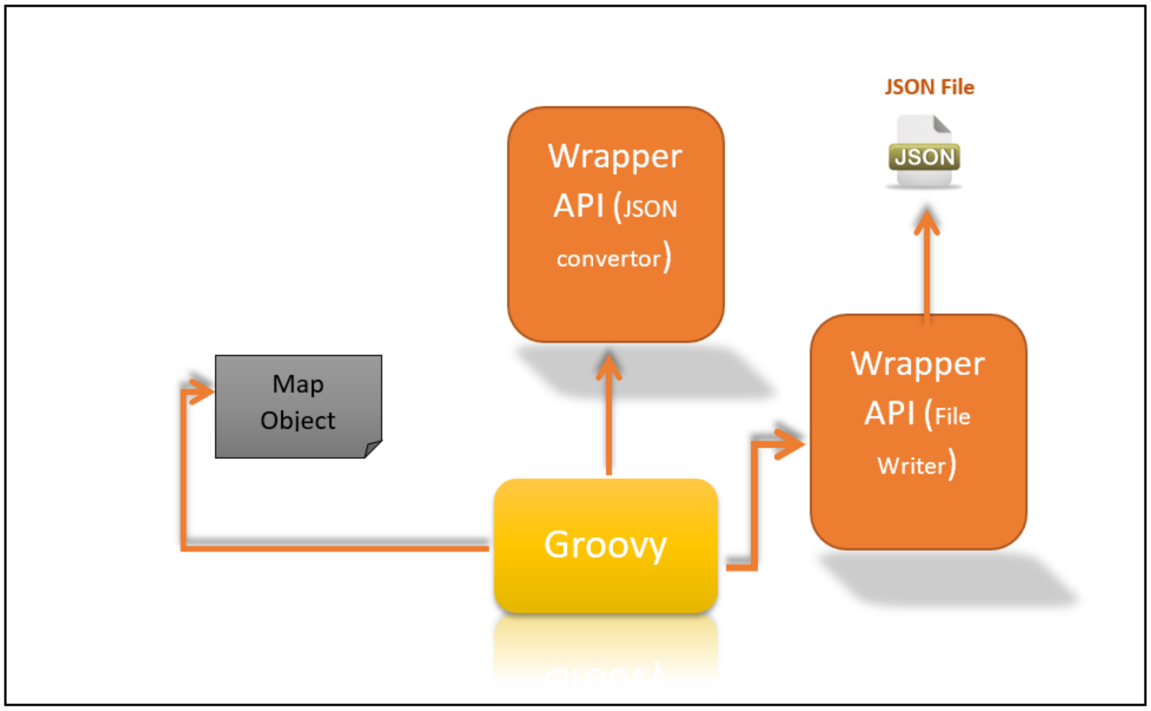


Figure 71: JSON Creation Process

3.4.1 Conversion in JSON

A wrapper API 'getJSONRecord()' receives 'Map' object as an input. The Map object maintains key value pair such as field and its respective value. The API function will process the Map object and return generated CSV content as string for the same Map object.

For every record, API function is called and returned result will be added to one final JSON string.

3.4.2 Writing a File

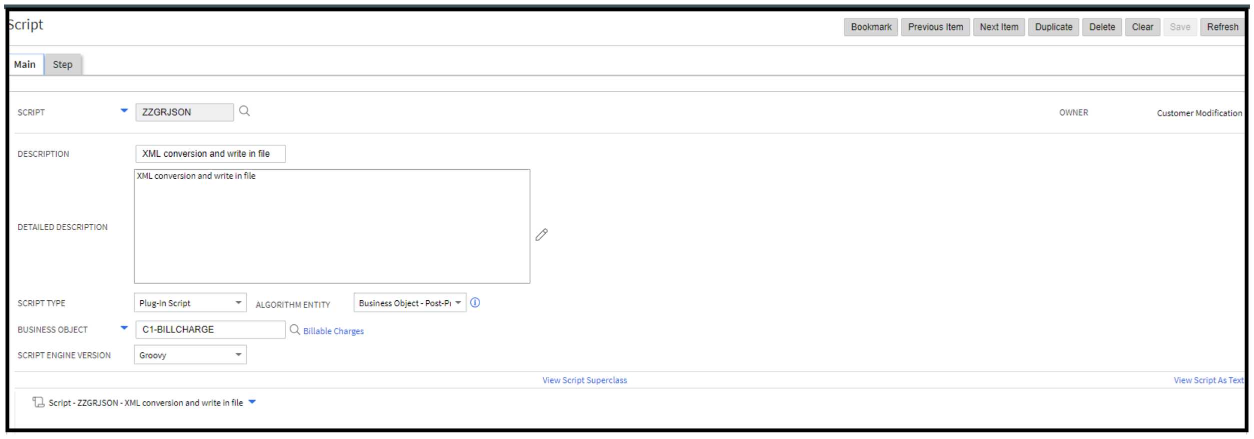
In this step, the final JSON string is passed along with file name, file path, file content and file extension to API function **fileWrite()** which writes the given file content to specific file at a desired path.

The below is the example of groovy script for JSON generation.

3.4.3 JSON Example

An example of a script converted in JSON format is listed below:

- **Script**: ZZGRJSON
- **Script Type**: Plug-In Script
- **Script Engine Version**: Groovy

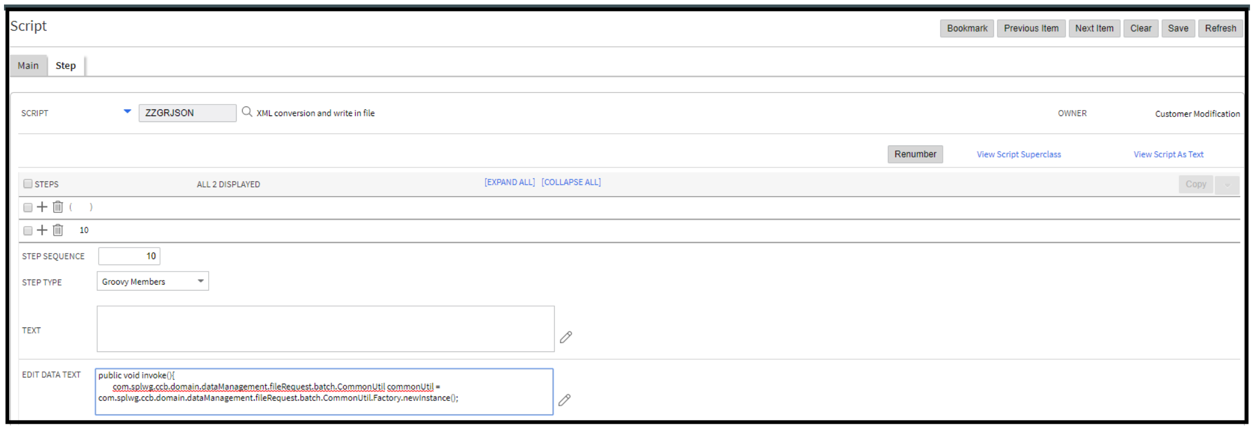


SCRIPT: ZZGRJSON
OWNER: Customer Modification
DESCRIPTION: XML conversion and write in file
DETAILED DESCRIPTION: XML conversion and write in file
SCRIPT TYPE: Plug-In Script
ALGORITHM ENTITY: Business Object - Post-Pı
BUSINESS OBJECT: C1-BILLCHARGE Billable Charges
SCRIPT ENGINE VERSION: Groovy

Script - ZZGRJSON - XML conversion and write in file

Figure 72: JSON Example

Steps: It has only one Step Type: 'Groovy Members'.



SCRIPT: ZZGRJSON XML conversion and write in file
OWNER: Customer Modification

STEPS ALL 2 DISPLAYED

( )

10

STEP SEQUENCE: 10
STEP TYPE: Groovy Members
TEXT:
EDIT DATA TEXT:

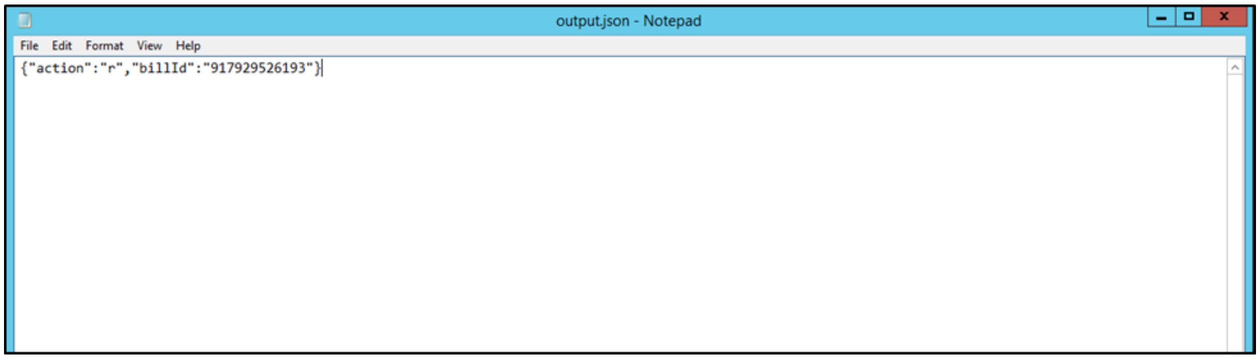
```
public void invoke(){
     com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil commonUtil =
com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil.Factory.newInstance();
```

Figure 73: Script - Step

- **Step Sequence**: 10. This value can be any unique integer value.
- **Step Type**: Groovy Members
- **Edit Data Text:** A sample code is listed below:

```
public void invoke(){

com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil
commonUtil                                                          =
com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil.Facto
ry.newInstance();


        Map fileContentMap = new HashMap<String, String>();
        fileContentMap.put("action", "r");
        fileContentMap.put("billId", "917929526193");


        String jsonString = "";
        try {
            jsonString = commonUtil.getJsonRecord(fileContentMap);
        } catch (Exception e) {
        }
        String outputFilePath = "D:/spl/C1";
        String outputFileName = "output";
        String fileContent = jsonString;
        String fileExtension = "json";
        commonUtil.fileWrite(outputFilePath,            outputFileName,
fileContent, fileExtension);
  }
```

When the script is trigger by any means, it will convert the Map object to JSON string and write it in output.json file.



```
{"action":"r","billId":"917929526193"}
```

Figure 74: Output

3.5 CSV Generation

The pre-requisite for the Wrapper API is List Object. CSV record generation is a two-step process:

- Conversion in CSV
- Writing a File

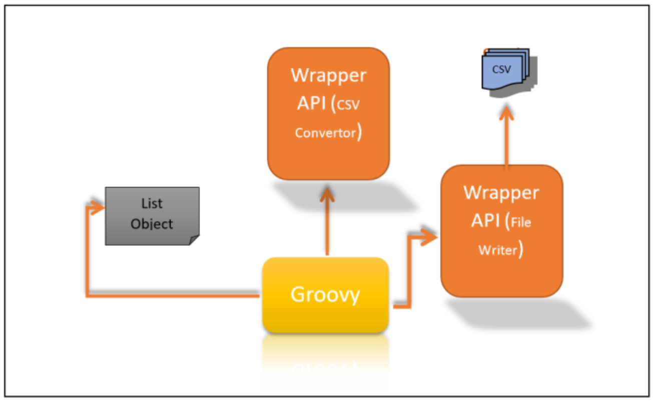


Figure 75: CSV Generation

3.5.1 Conversion in CSV

A wrapper API ‘getCSVRecord()’ receives ‘List Object’ as an input. The List object maintains all the values required in CSV record. The API function will process the List object and return generated CSV content as string for the same List object.

For every record, API function is called and returned result will be added to one final CSV string.

3.5.2 Writing a File

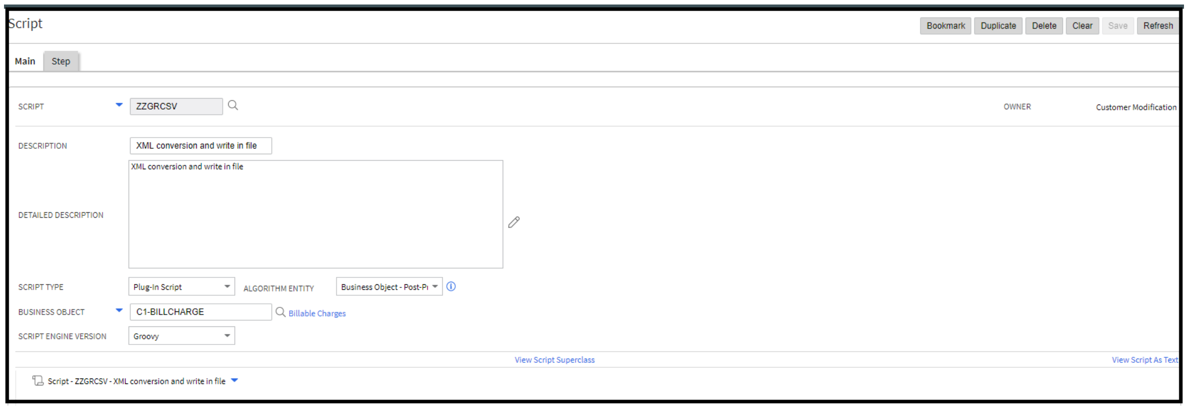
In this step, the final CSV string is passed along with file name, file path, file content and file extension to API function **fileWrite()**, which writes the given file content to specific file at a desired path.

The below is the example of groovy script for CSV generation.

3.5.3 CSV Example

An example of a script converted in CSV format is listed below:

- **Script**: ZZGRCSV
- **Script Type**: Plug-In Script
- **Script Engine Version**: Groovy

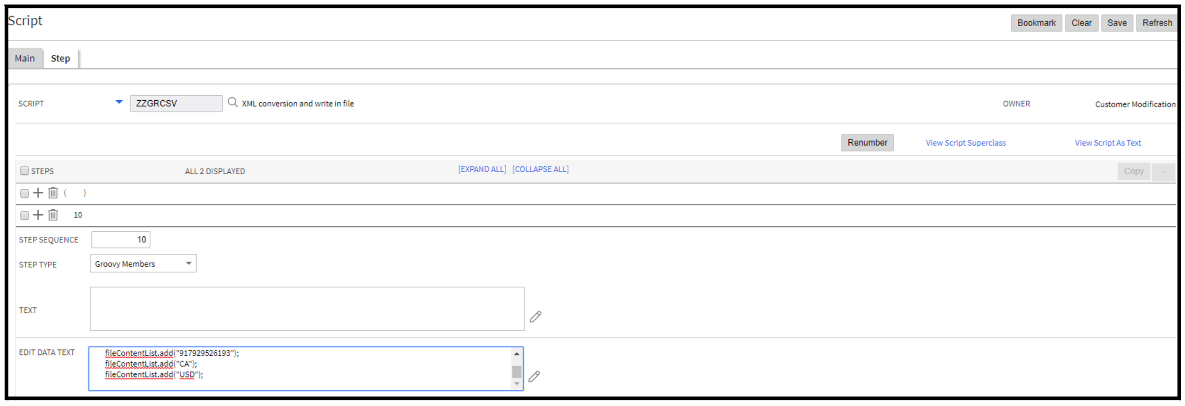


SCRIPT	ZZGRCSV
OWNER	Customer Modification
DESCRIPTION	XML conversion and write in file
DETAILED DESCRIPTION	XML conversion and write in file
SCRIPT TYPE	Plug-In Script
ALGORITHM ENTITY	Business Object - Post-Pı
BUSINESS OBJECT	C1-BILLCHARGE Billable Charges
SCRIPT ENGINE VERSION	Groovy

Script - ZZGRCSV - XML conversion and write in file

Figure 76: CSV Example

Steps: It has only one Step Type: ‘Groovy Members’



SCRIPT	ZZGRCSV XML conversion and write in file
OWNER	Customer Modification
STEP SEQUENCE	10
STEP TYPE	Groovy Members
TEXT	

EDIT DATA TEXT

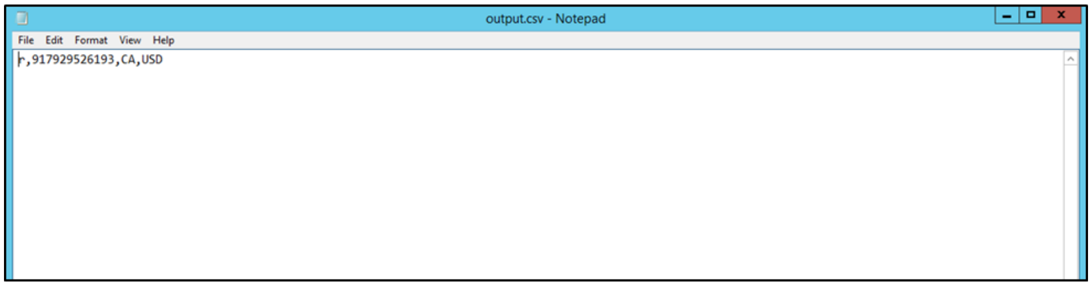
```
fileContentList.add("917929526193");
fileContentList.add("CA");
fileContentList.add("USD");
```

Figure 77: Script - Step

- **Step Sequence**: 10. This value can be any unique integer value.
- **Step Type**: Groovy Members
- **Edit Data Text**: A sample code is listed below:

```
public void invoke(){

com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil
commonUtil                                                      =
com.splwg.ccb.domain.dataManagement.fileRequest.batch.CommonUtil.Facto
ry.newInstance();
        List fileContentList = new ArrayList<String>();
        fileContentList.add("r");
        fileContentList.add("917929526193");
        fileContentList.add("CA");
        fileContentList.add("USD");

        String csvString = "";
        try {
            csvString = commonUtil.getCSVRecord(fileContentList);
        } catch (Exception e) {
        }
                String outputFilePath = "D:/spl/C1";
        String outputFileName = "output";
        String fileContent = csvString;
        String fileExtension = "csv";

        commonUtil.fileWrite(outputFilePath,            outputFileName,
fileContent, fileExtension);
  }
```

When the script is triggered by any means, it will convert the above List object mentioned in script to CSV string and write it in `output.csv` file.



```
r,917929526193,CA,USD
```

Figure 78: Output