

Oracle® Retail Accelerators

*Developing ADF Reports for RPAS
Release 13.2*

June 2011



Note: The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Contents

Contents	ii
1 Introduction	1
Prerequisites	1
About Oracle Retail	1
About Oracle Application Development Framework.....	2
2 RPAS ODBC Data Query	1
Metadata.....	1
Fact and Dimension Tables.....	2
3 Setting Up a Database Connection	1
Prerequisites	1
Setting Up the RPAS Data Source in JDeveloper.....	1
Setting Up an IDE Database Connection	4
Connecting Oracle Database to RPAS.....	6
Installing ODBC client on Windows	7
Connecting Oracle Database to RPAS.....	9
4 Setting Up an ADF-based Application	1
Creating an ADF-based Application.....	1
Creating Model Layer Objects.....	3
Creating the Channels View Object	4
Setting Up the ChannelsVO View Object for LOV.....	4
Setting Up the Application Module	7
Setting Up the User Interface	8
Setting Up an ADF Task flow.....	14
Review the Output.....	16
Creating a Managed Java Class.....	17
A Best Practices	1
Prefer High Level Fact Tables Over Lower Level Fact Tables	1
Prefer Calculations in RPAS	1
Move the Data to the Relational Database.....	2
Guidelines to Create View Objects Used in the Application	3
Measures View Object.....	3
SubClass View Object.....	3
Year View Object.....	3

Introduction

The Oracle Retail Predictive Application Server (RPAS) platform supports multidimensional databases and hierarchical data. Users usually manipulate the data (in the RPAS domains) using the workbooks (graphical user interface) or by calling the RPAS API. Since the RPAS domains are not SQL compliant, the data cannot be accessed directly using ADF. To solve this issue, Oracle Retail has developed the RPAS ODBC package that contains the following components:

- RPAS ODBC Driver – This component resides on the system where the WebLogic Server is installed.
- RPAS ODBC Server – This component runs on the system where the RPAS applications are installed.

Using this ODBC interface, you can then use ADF to access the RPAS domains. This interface enables you to create reports on measures across multiple RPAS domains. Since ADF reports can be embedded in ADF task flows, there are now more ways to present the RPAS data.

This document illustrates how you can use the Oracle Application Development Framework 11g (ADF) to generate reports that provide insights from the Oracle Retail Predictive Application Server (RPAS) based applications. Information included in this document is based on a RPAS and ADF proof of concept project completed within Oracle. In this project, ADF based reports were created on the Oracle Retail Merchandise Financial Planning Retail (MFP Retail) domain to demonstrate how ADF and RPAS can be integrated using ODBC or JDBC. As a connecting tool, this document also introduces RPAS ODBC.

Prerequisites

Before proceeding, you must take the following into consideration:

- This document is intended for application integrators and implementation personnel who are familiar with the Oracle Application Development Framework (ADF).
- In addition to ADF, you must also access to and have knowledge of Oracle Retail Predictive Application Server Release 13.2.1, Oracle Database 11g, and Oracle JDeveloper 11g.

About Oracle Retail

Oracle is the number one provider of innovative and comprehensive industry software solutions for retailers – enabling organizations to serve their customers better by applying insight into daily business decisions for more profitable results. With software that provides supply chain, operations, merchandising, store systems, optimization as well as enterprise applications and infrastructure software, Oracle partners with the world's leading retail companies, including 20 of the 20 top retailers worldwide, to transform the economics of their businesses.

Some of the Oracle Retail applications are based on the Retail Predictive Application Server (RPAS) platform. For example, Oracle Retail Merchandise Financial Planning (MFP), Oracle Retail Item Planning (IP), and Oracle Retail Item Planning configured with Clearance Optimization Engine (IPCOE) are solutions that run on the RPAS-based

platform. To access these solutions, users will need to log on to an RPAS client configured to run the domains of the relevant solutions.

About Oracle Application Development Framework

Oracle Application Development Framework 11g (ADF) is an application development framework that features various development functionalities, including reporting. You can use ADF to create quick KPI reports off live data in order to accomplish the insight-driven retailing experience.

RPAS ODBC Data Query

The RPAS ODBC server calls low-level RPAS API to access the RPAS domains. It presents the RPAS domain data (dimensions, measures) into tables, which are SQL compliant. This chapter describes the database objects that can be used for the RPAS ODBC data query.

It includes the following topics:

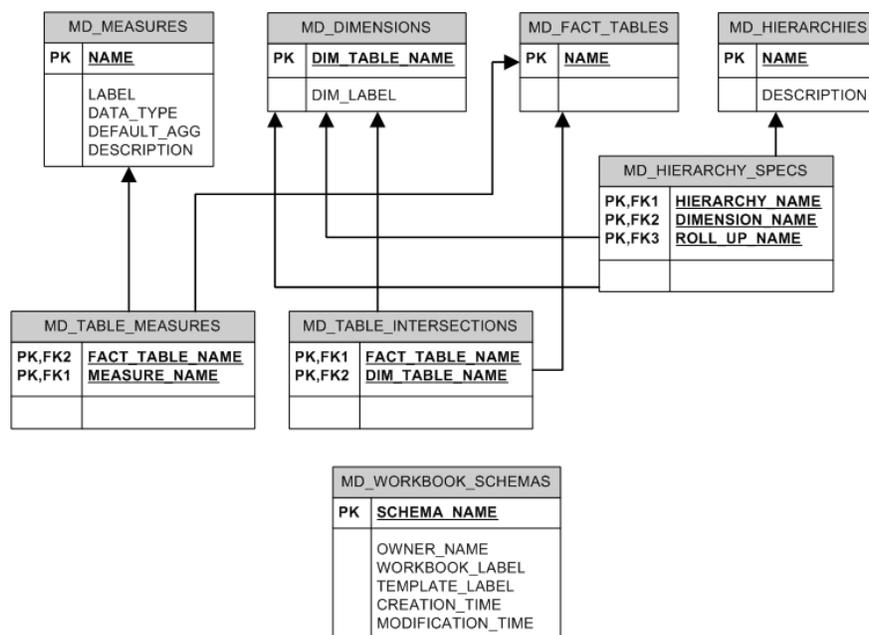
- [Metadata](#)
- [Fact and Dimension Tables](#)

Metadata

The following figure shows the metadata tables available in a domain or workbook. These tables can be used to examine the structure of the domain, such as:

- Measures and dimensions that exist within the database.
- Hierarchies that exist and their rollup structure.
- Fact tables that are available.
- Measures that exist at the intersections they represent.

When connected to a domain, an additional table (MD_WORKBOOK_SCHEMAS) is available to list all accessible workbooks within the domain with their schema names.



Database Diagram for All Metadata Tables in a Domain or in Each Workbook

Note: The MD_WORKBOOK_SCHEMAS table is not included in workbooks.

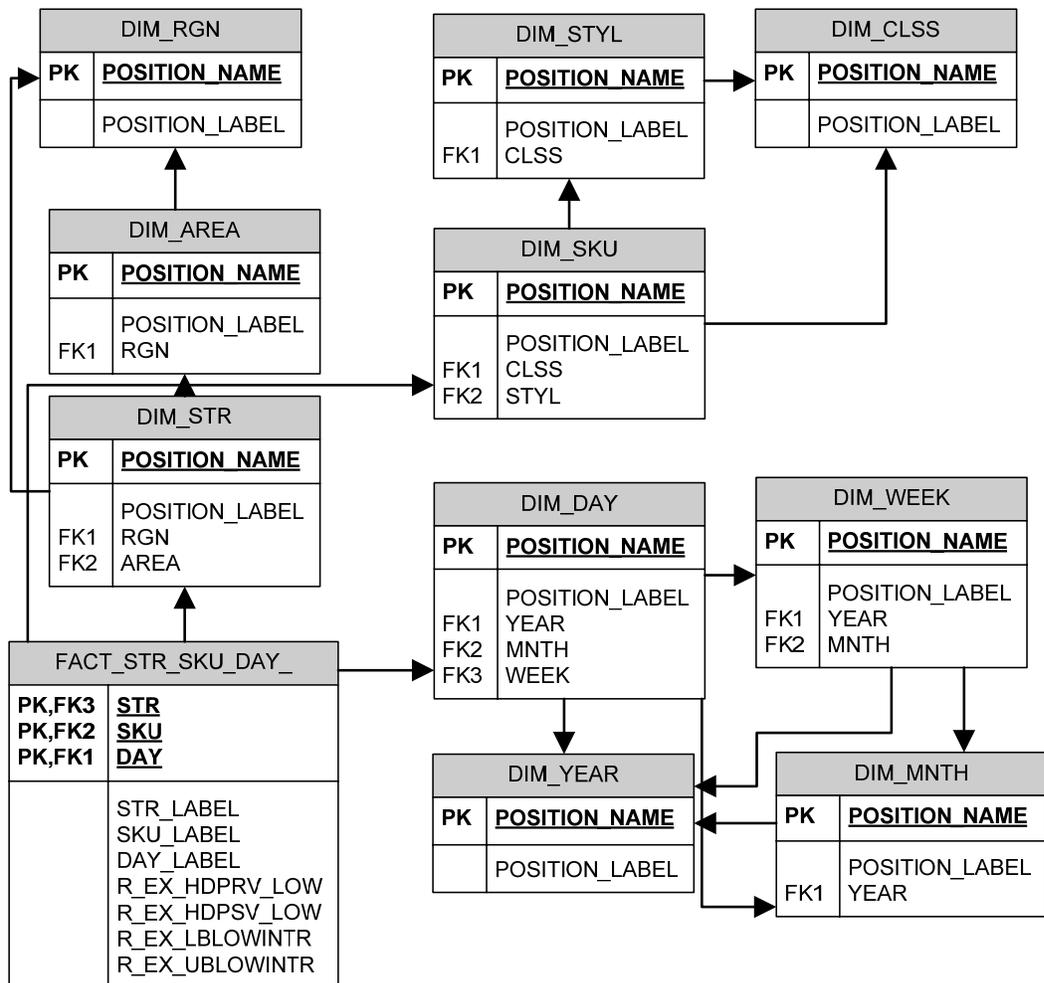
Fact and Dimension Tables

The following figure shows an example of the structure of fact and dimension tables and the relationships between them. A fact table represents an intersection where one or more measures data is stored. Each measure is represented by a column in the table.

Additionally, each dimension on the intersection is represented by a column. A record in the fact table is uniquely identified by a unique combination of position names for the intersecting dimensions.

A dimension table represents a dimension. It includes a column to list all position names, their labels, and their rollup mapping to each dimension at higher levels in the hierarchy. The fact and dimension tables have foreign key relationships between them to represent the intersection and maintain data integrity between the dimensions and the facts.

Dimension tables have foreign key relationships with other dimension tables to represent the hierarchical relationships between them.



Example of Star – De-normalized Schema to Represent Facts and Dimensions in RPAS

At connection time, all intersections at which any measure is stored at its base level are available as fact tables within the database. Additional aggregate level intersections may be made available in the database by specifying them in a custom connection property.

These fact tables are a part of the set of database entities that will be visible to reporting tools at connection time. However, the RPAS ODBC/JDBC driver supports dynamic aggregate level fact tables that can be queried even though they are not available at connection time. These tables include all intersections that are logically above the base intersection fact tables and have at least one measure in them when manifested. If the measure existence condition is not met, the driver returns an error that the fact table could not be found.

These dynamic fact tables are queried in the same fashion as the tables that are available at connection time. The name of the fact table can be constructed by piecing together dimension names (not labels) that make up the intersection in the order in which they would exist within the domain. For example, if someone wants to query facts at the store/class/day level but the fact table is not available at connection time, they can construct the fact table name as: `FACT_STR_CLSSDAY_`. Note that dimension names have been concatenated in the same order as the intersection and have been prefixed with `'FACT_'`. Also, note that a dimension name is assumed to be four characters long and if the dimension name is less than four characters, it is padded with `'_'` characters to make it four characters long.

By issuing ODBC SQLs against these dimension tables and fact tables, you can access the RPAS data in ADF.

Setting Up a Database Connection

This chapter describes how you can set up a database connection using the RPAS ODBC utility. It includes the following topics:

- [Prerequisites](#)
- [Setting Up the RPAS Data Source in JDeveloper](#)
- [Setting Up an IDE Database Connection](#)
- [Connecting Oracle Database to RPAS](#)

Prerequisites

Before you proceed, ensure that you have the following installed:

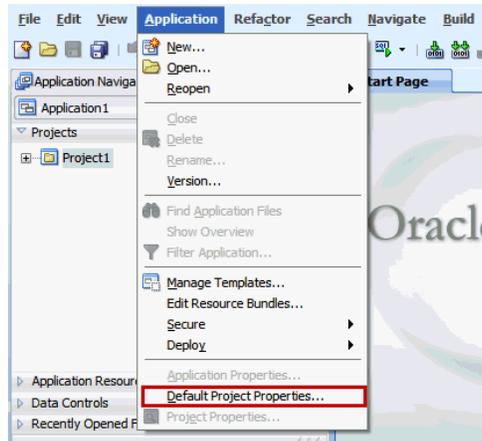
- **Oracle Retail Predictive Application Server (RPAS) Release 13.2 or later and ODBC Components**
This includes the RPAS ODBC Server and RPAS ODBC client. For more information on installing RPAS and setting up the ODBC components, refer to the *Oracle Retail Predictive Application Server Installation Guide Release 13.1.1 and Release 13.2*.
In case you install the RPAS ODBC Server on a Linux-based system, ensure that you set up the Time Zone (TZ) environment variable.
- **MFP Retail domain**
This includes the Merchandise Financial Planning Retail Release 13.2 solution and domain. For more information on installing Merchandise Financial Planning Retail, refer to the *Oracle Retail Merchandising Financial Planning Retail Installation Guide Release 13.2*.
- **Oracle JDeveloper 11g**

Setting Up the RPAS Data Source in JDeveloper

Any Java-based application can be connected to the RPAS domain. This section describes how you can configure the RPAS JDBC client in the JDeveloper environment, in such a manner that any application project in JDeveloper has access to the RPAS data source.

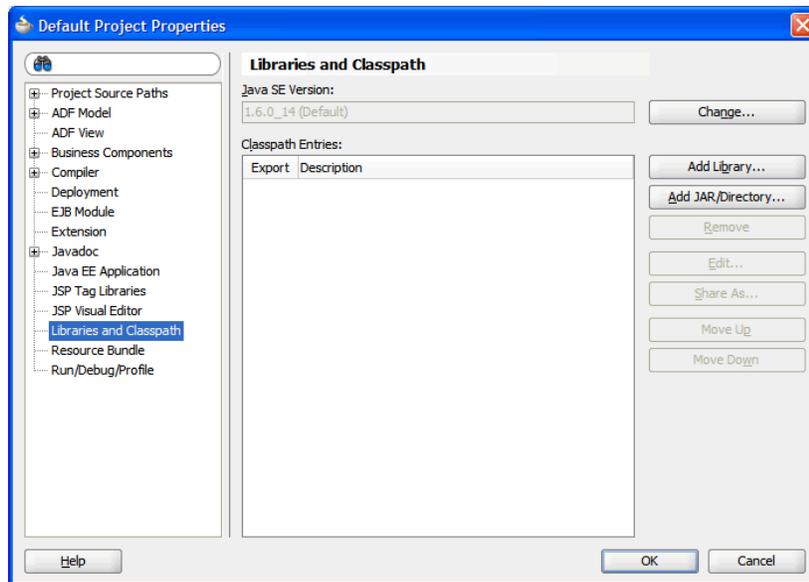
To set up an RPAS data source in JDeveloper:

1. In your RPAS installation, go to the directory where the RPAS server is installed (this is the location set as the RPAS_HOME environment variable).
2. From this location, download the JDBC client JAR files to a folder in your Windows-based system. The JDBC client is present as a ZIP file in this folder.
3. Launch **Oracle JDeveloper**.
4. From the main menu, under **Application**, click **Default Project Properties**. The **Default Project Properties** window appears.



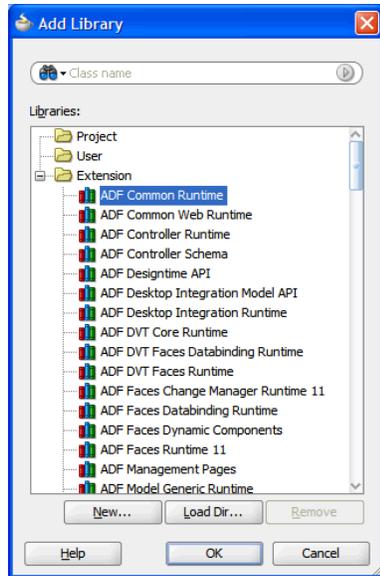
Default Project Properties Option

5. In the **Default Project Properties** dialog box, select **Libraries and Classpath**.



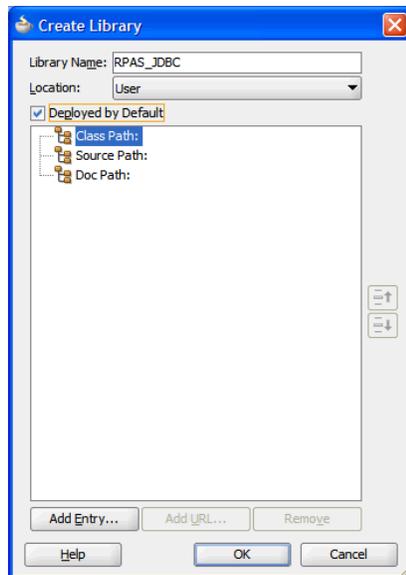
Default Project Properties Dialog Box

6. Click the **Add Library** button. The **Add Library** dialog box appears.



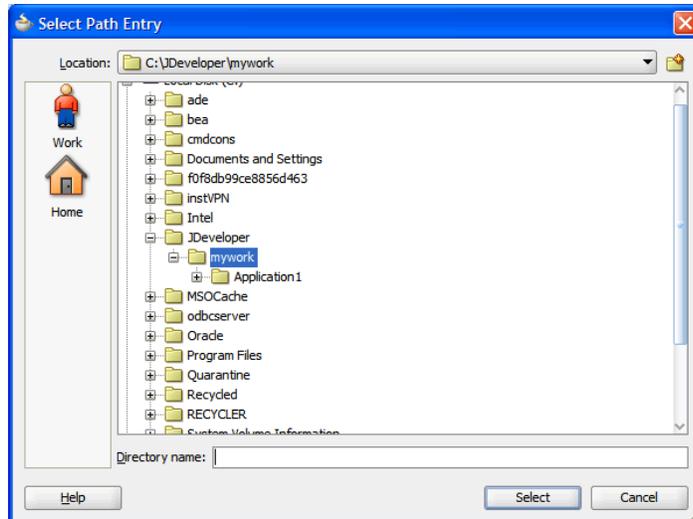
Add Library Dialog Box

7. In the **Add Library** dialog box, click the **New** button. The **Create Library** dialog box window appears.



Create Library Dialog Box

8. In the **Create Library** dialog box, enter following details:
 - In the **Library Name** field, enter **RPAS_JDBC**.
 - In the **Location** field, set **User**.
 - Select the **Deployed By Default** check box.
9. Select the **Class Path** entry, and click the **Add Entry** button. The **Select Path Entry** dialog box appears.



Select Path Entry Dialog Box

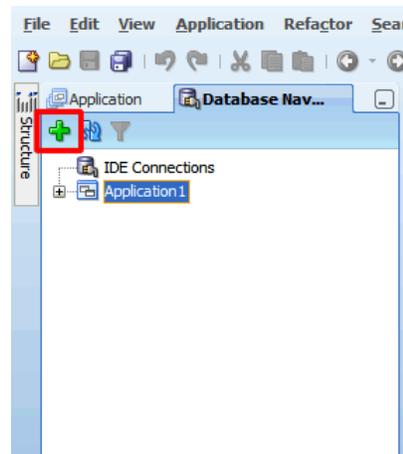
10. In the **Select Path Entry** dialog box, browse to the location where you have downloaded the JDBC client JAR files in Step 2.
11. Select the following JAR files:
 - **iaik_jce_full.jar**
 - **ORjc.jar**
 - **ORssl14.jar**
12. In the **Create Library** dialog box, click **OK**.
13. In the **Add Library** dialog box, click **OK**. Note the new library with the name **RPAS_JDBC** added under the User tree.
14. In the **Default Project Properties** dialog box, click **OK**.

Setting Up an IDE Database Connection

Now that we have added the JDBC client Jars in JDeveloper Library, we would use it to create an IDE database connection.

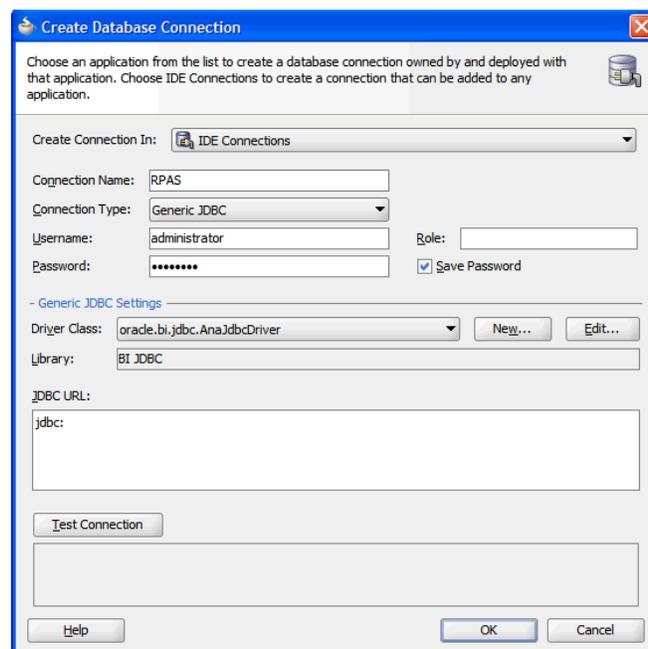
To set up an IDE database connection:

1. In **JDeveloper**, open the **Database Navigator** (from the **View** menu, point your cursor on **Database**, and then click **Database Navigator**).



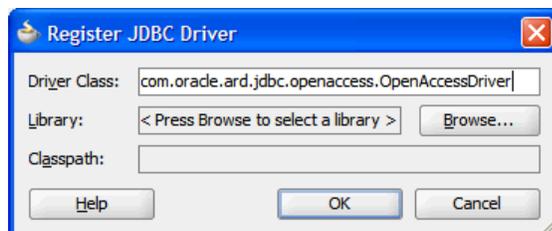
Database Navigator Pane

- In the **Database Navigator** pane, click the **New Connection** button. The **Create Database Connection** dialog box appears.



Create Database Connection Dialog Box

- In the **Create Database Connection** dialog box, add following details:
 - In the **Connection Name** field, enter **RPAS**.
 - In the **Connection Type** list, set the value to **Generic JDBC**.
 - In the **Username** field, enter the user associated with your RPAS domain.
 - In the **Password** field, enter the password associated with the user name entered above.
- In the **Generic JDBC Settings** section, next to the **Driver Class** list, click the **New** button. The **Register JDBC Driver** dialog box appears.



Register JDBC Driver Dialog Box

5. In the **Register JDBC Driver** dialog box, add the following details:
 - In the **Driver Class** field, enter **com.oracle.ard.jdbc.openaccess.OpenAccessDriver**.
 - In the **Library** field, click **Browse**, and select the **RPAS_JDBC** library you created in the section above.
6. Click **OK**.
7. On the **Create Database Connection** dialog box, under **JDBC URL**, enter `jdbc:RPAS://<RPAS host>:<RPAS ODBC server port>;ServerDataSource=<domain name>`.

Note: In the URL above, ensure that you replace `<RPAS host>`, `<RPAS ODBC server port>`, and `<domain name>` with the relevant RPAS information.

8. Click **Test Connection** to ensure that the connection is set up accurately.
9. Click **OK**.

You have now configured a database connection to the RPAS domain in your IDE. You can use this connection in any application you will create in JDeveloper IDE.

Connecting Oracle Database to RPAS

Using the information from the sections above, you can connect JDeveloper to RPAS using the JDBC client. Although the RPAS data source connection can now be used in an application you create in JDeveloper, the connection can only refer to the RPAS data. A typical business requirement may need you to combine the RPAS data with other data sources, such as Oracle Retail Merchandising System (RMS). This section describes how you can achieve this business requirement.

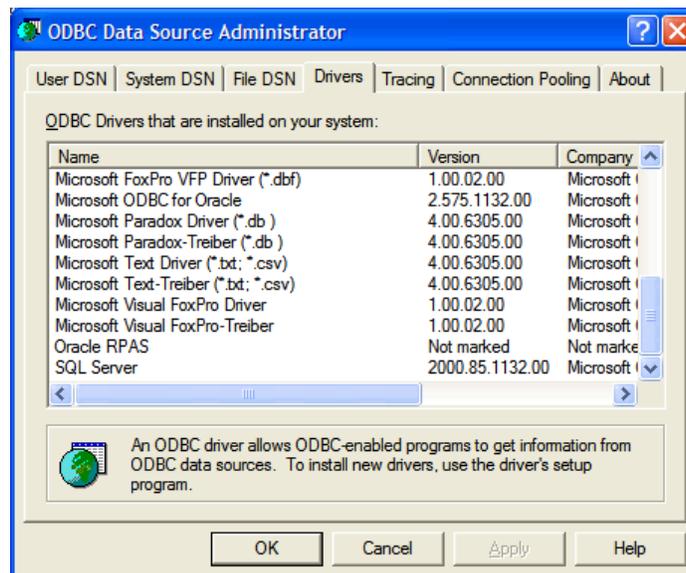
The RPAS installation also includes an ODBC client which applications can use to connect to it. You may use this ODBC driver to connect an Oracle database to RPAS. The Oracle database installation includes built-in components, such as Heterogeneous Services, Database Gateway 4 ODBC that will allow the database to connect to any disparate data source transparently (provided that the disparate data source has an ODBC driver). In this context, Oracle database is where the RMS database schema resides. Once you connect this database to RPAS, you can combine RMS and RPAS data together.

Before you proceed, ensure that you have installed the RPAS ODBC client specific to the operating system where the Oracle Database 11g is installed. This section includes information based on the assumption that Oracle Database 11g is installed on a Microsoft Windows XP system. To proceed further, you will require the RPAS ODBC Client for Windows. It is packaged, by default, in the RPAS installation package for Windows.

Installing ODBC client on Windows

This section describes how you can install the ODBC client on Windows. To install the ODBC client on Windows:

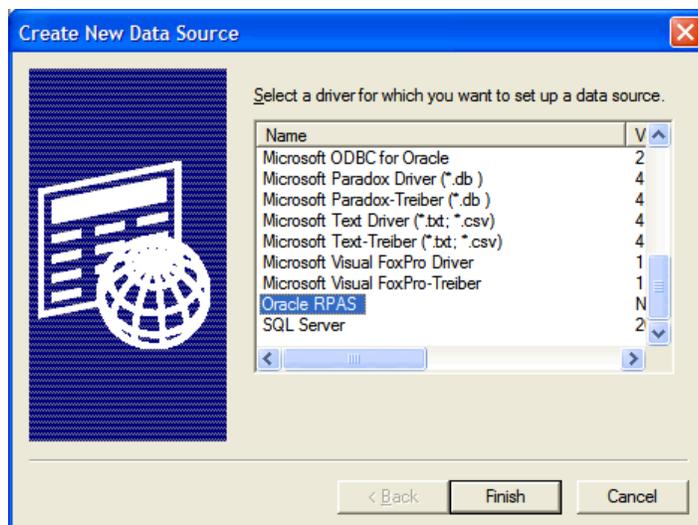
1. Download **RPAS ODBC** client for Windows. By default, this is included in the RPAS for Windows installation package.
2. In the **ODBC client** folder, run the **setup.exe** file.
3. Follow the instructions in the installation wizard and complete the RPAS ODBC driver setup in Windows machine.
4. Once the installation wizard is complete, from the **Windows Start** menu, launch the **Control Panel** from the **Settings** menu.
5. In the **Control Panel** window, double-click **Administrative Tools**.
6. In the **Administrative Tools** window, double-click **Data Sources (ODBC)**. The **ODBC Data Source Administrator** window appears.
7. In the **ODBC Data Source Administrator** window, click the **Drivers** tab.
8. Ensure that Oracle RPAS is in the list of drivers.



ODBC Data Source Administrator Window

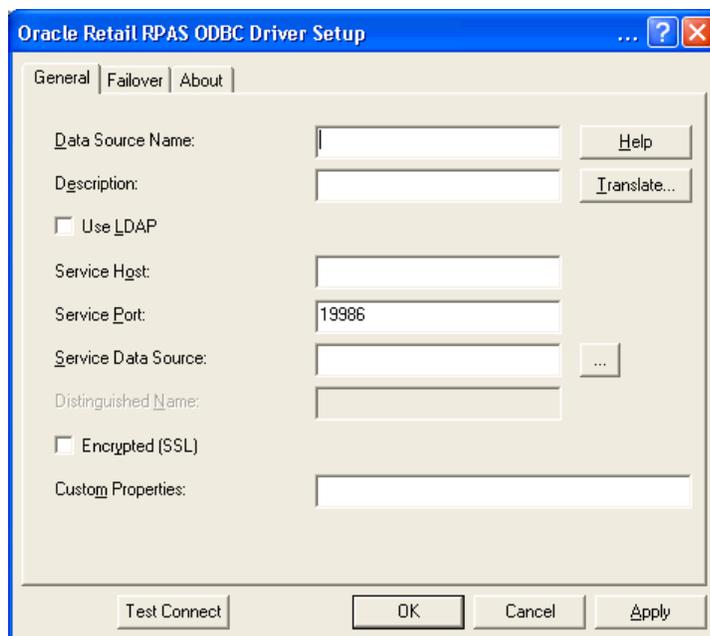
Note: You will now create a connection to the RPAS Domain through the ODBC driver installed above.

9. On the **ODBC Data Source Administrator** window, click the **System DSN** tab.
10. On the **System DSN** tab, click **Add**. The **Create New Data Source** dialog box appears.



Create New Data Source Dialog Box

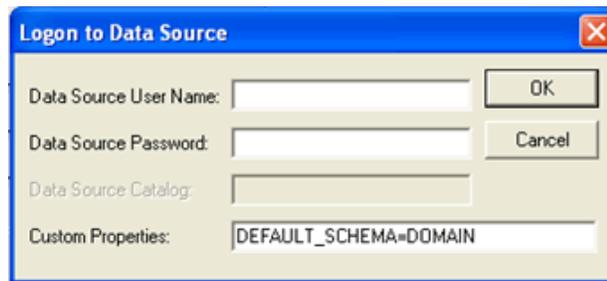
11. In the **Create New Data Source** dialog box, select the **Oracle RPAS** driver from the list, and click **Finish**. The **Oracle Retail RPAS ODBC Driver Setup** dialog box appears.



Oracle Retail RPAS ODBC Driver Setup Dialog Box

12. In the **Oracle Retail RPAS ODBC Driver Setup** dialog box, enter the following details:
 - In the **Data Source Name** field, enter a relevant data source name.
 - In the **Description** field, enter a relevant description for the data source.
 - In the **Service Host** field, enter the relevant RPAS host machine name where the ODBC server is running.
 - In the **Service Data Source** field, click the **Browse** button. The **Service Data Sources** window appears with a list of all domains running in the RPAS application.

13. Select the domain you want to connect to, and click **OK**.
14. Click the **Test Connect** button to test the connection. You will be prompted to enter user name and password for the RPAS domain.



Logon to Data Source Dialog Box

15. Enter the relevant credentials and click **OK**. The test results will appear. You have now established a connection from your Windows machine to an RPAS domain using the RPAS ODBC client.

Connecting Oracle Database to RPAS

Using the connection established in the previous section, you must now connect the Oracle database installed on this Windows system with RPAS using the following steps:

Note: It is assumed that you have installed Oracle Database 11g on your Windows system.

1. Start the database instance and ensure that the following services are running in the Services console:
 - TNS Listener
 - ORCL service
2. Navigate to the directory where Oracle database is installed. This location is set as the ORACLE_HOME environment variable.
3. Navigate to the **hs\admin** directory within the ORACLE_HOME directory.
4. Create a file named **init<someStringForTargetConn>.ora**. For example, **initRPAS.ora**.

Note: You need separate file for each ODBC datasource you want to connect to.

5. Enter the following text in the .ora file:


```
# This is a sample agent init file that contains the HS parameters that are
# needed for the Database Gateway for ODBC
#
# HS init parameters
#
HS_FDS_CONNECT_INFO = <odbc data_source_name>
HS_FDS_TRACE_LEVEL = <trace_level>

#
# Environment variables required for the non-Oracle system
#
#set <envvar>=<value>
```

For example, assuming that the ODBC data source name you created in the Windows machine was named **gdom_domain**, the content of this file will look similar to the following:

```
# This is a sample agent init file that contains the HS parameters that are
# needed for an OLEDB Agent.
#
# HS init parameters
#
HS_FDS_CONNECT_INFO = gdom_domain
HS_FDS_TRACE_LEVEL = ON

#
# Environment variables required for the non-Oracle system
#
#set <envvar>=<value>
```

6. Navigate to the directory **ORACLE_HOME\NETWORK\ADMIN**.

7. In the file **listener.ora**, add the following content:

```
SID_LIST_LISTENER=
  (SID_LIST=
    (SID_DESC=
      (SID_NAME=<Name of the init file without the init prefix or ora
extension>)
      (ORACLE_HOME=<Your Oracle home>)
      (PROGRAM=dg4odbc)
    )
  )
```

For example, in the step above, if you named the init file in **ORACLE_HOME\hs\admin** as **initRPAS.ora**, the content above will look similar to the following:

```
SID_LIST_LISTENER=
  (SID_LIST=
    (SID_DESC=
      (SID_NAME=RPAS)
      (ORACLE_HOME=D:\app\product\11.1.0\db_1)
      (PROGRAM=dg4odbc)
    )
  )
```

8. In the same directory, in the file **tnsnames.ora**, add the following content:

```
<Unique_ID> =
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=localhost)(PORT=1521))
    (CONNECT_DATA=(SID=<SID name in listener.ora>))
    (HS=OK)
  )
```

A UniqueID will be used in the next steps when creating a DBLINK. In the steps above, if you named the **SID_NAME** in **listener.ora** as **RPAS**, the content of this file will look similar to the following:

```
rpas_gdom_domain =
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=localhost)(PORT=1521))
    (CONNECT_DATA=(SID=RPAS))
    (HS=OK)
  )
```

9. Restart the Listener for Oracle database.

10. Open SQLPlus and connect to your schema.

11. Create a DBLINK using the **rpas_gdom_domain** gateway for ODBC that you created in the steps above. Use the following SQL statement:

```
Create public database link RPAS_LINK connect to "<RPAS domain user name>"  
identified by "<RPAS user password>" using 'rpas_gdom_domain';
```

The DBLINK created above can now be used to fetch RPAS data. You can create a view over the RPAS table in your Oracle schema, which you can then join with your local schema tables to combine data.

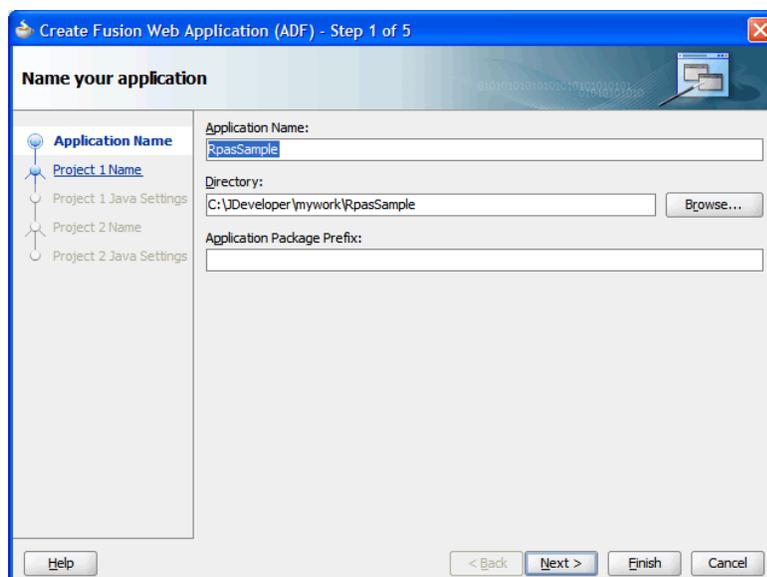
Setting Up an ADF-based Application

Once the database connection is created, you can now use it in an application that provides reports on the data in the RPAS domain. This chapter provides an illustration on how you can create an ADF-based application and set it up to include RPAS-based reports. It includes the following topics:

Creating an ADF-based Application

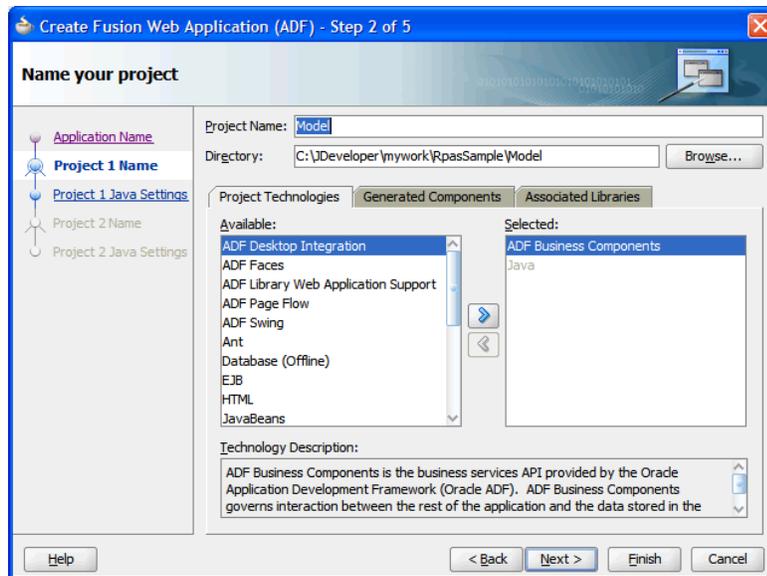
To create an ADF-based application:

1. In **JDeveloper**, from the **File** menu, click **New**. The **New Gallery** wizard appears.
2. On the **New Gallery** wizard, in the **Categories** section, select **Applications** under **General**.
3. From the **Items** section, select **Fusion Web Application (ADF)**, and click **OK**. The **Create Fusion Web Application (ADF)** wizard appears.



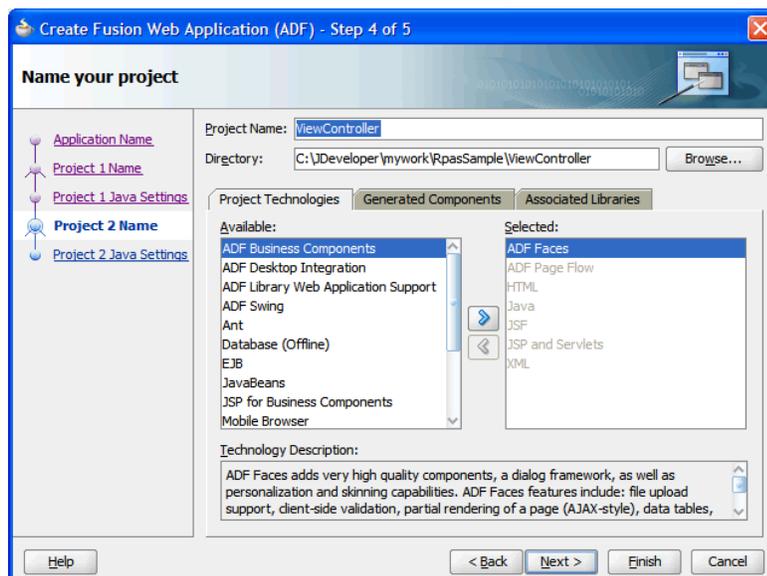
Name Your Application Screen in Create Fusion Web Application (ADF) Wizard

4. On the **Name your application** screen, set the application name to **RpasSample**, and click **Next**. The **Name your project** screen appears.



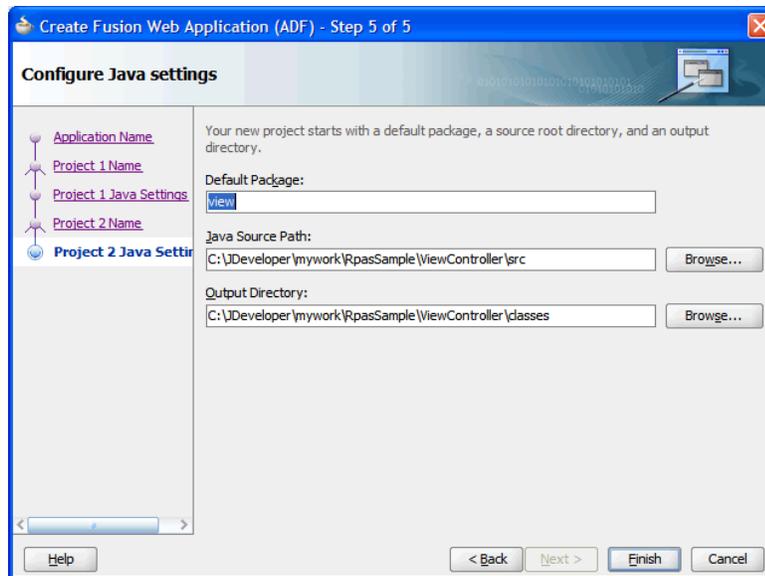
Name your project Screen

5. On the **Name your project** screen, specify the model name as **model**, and click **Next**. The **Configure Java Settings** screen appears.
6. On the **Configure Java Settings** screen, leave the settings As Is, and click **Next**. The **Name your project** screen appears.



Name your project Screen

7. On the **Name your project** screen, set the viewController project name to **viewController**, and then click **Next**. The **Configure Java Settings** screen appears.



Configure Java Settings Screen

8. On the **Configure Java Settings** screen, enter a default package name for the project. For example, *oracle.retail.merch20*.
9. Click **Finish**.

Note: An ADF-based Web application normally has two projects, Model (intended to provide model layer with business components) and ViewController (intended to provide the view layer with ADF faces). This is the recommended practice to separate out Model layer from View layer.

Creating Model Layer Objects

The RPAS Application will contain several View objects (ADF Business component's View Objects). These View Objects will be used to read data from the relevant tables in RPAS database and display it in the user interface. Since these View Objects will be used only to read data, they will be *Read only* view objects. View objects that will be used in this application are as follows:

View Object name	Related table in RPAS schema	Description
ChannelsVO	DIM_CHNL	The list of available channels.
SclsVO	DIM_SCLS	Sub-class dimension table.
SelectYear	DIM_YEAR	Year dimension table.
MeasuresVO	FACT_CHNLSCLSWEEK	Measures the user requires.

The following sections provide you an example of how to declare a read only view object in JDeveloper. For illustration purposes, the Channels view object and Measures view object have been used:

- [Creating the Channels View Object](#)
- [Configuring the Channels View Object for LOV](#)

Creating the Channels View Object

The scenario is that a user must be able to search through all channels in the RPAS schema, and then select one. This can be easily achieved by defining a view object for Channels in the following manner:

1. In **JDeveloper**, with the application open, right-click the **Model** project in the **Application Navigator** panel.
2. From the right-click menu, click **New**. The **New Gallery** wizard dialog box appears.
3. On the **New Gallery** wizard, in the **Categories** section, select **ADF Business Components** under **Business Tier**.
4. In the **Items** section, click **View Object**, and then click **OK**. The **Initialize Business Components Project** dialog box will appear if you do not have a database connection defined for the application. Add a database connection using the **Add (+)** button. Once the database connection is established, you will be directed back to the view object creation wizard. The **Create View Object** wizard dialog box appears.
5. On the **Create View Object** wizard dialog box, in the **Name** screen, select the package as **oracle.retail.merch20.reports.rpas.model**.
6. In the **Name** field, select the view object name as **ChannelsVO**.
7. Under the **Select the data source type you want to use as the basis for this view object**. Section, select the **Read only access through SQL query** option.
8. Click **Next**. The **Query** screen appears.
9. In the **Query Statement** section, enter the following query

```
select CHNL_LABEL,CHNL,concat(concat(CHNL_LABEL, '-'), CHNL) AS Channel_Search  
from DIM_CHNL;
```
10. Click **Next**.
11. Click **Finish**.

You must now fine tune the ChannelsVO view object so that it is used in the user interface as a list of values (LOV).

Setting Up the ChannelsVO View Object for LOV

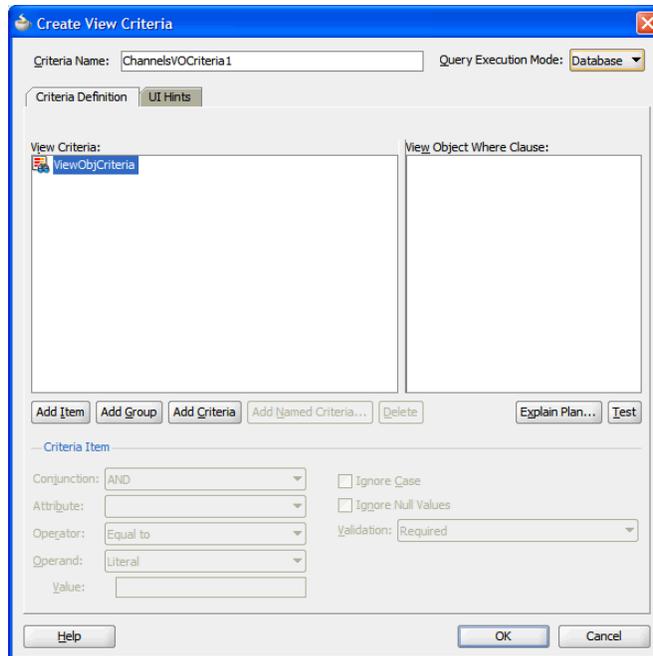
This section describes how you can configure the ChnlLabel attribute of the ChannelsVO view object as a list of values (LOV). After defining the attribute as an LOV, you will then be able to use the attribute on a JSF page as an ADF LOV Input component. For more information, see the [Creating the User Interface](#) section.

Such components are rendered with an input box and a search icon. Users may click the search icon to launch a pop-up window with a list of all possible values for the attribute. They can then select a relevant value from the list.

To achieve this, you must first define the view criteria for the view object. The view criteria is used to filter out rows of interest from the result set that the query of the view object returns.

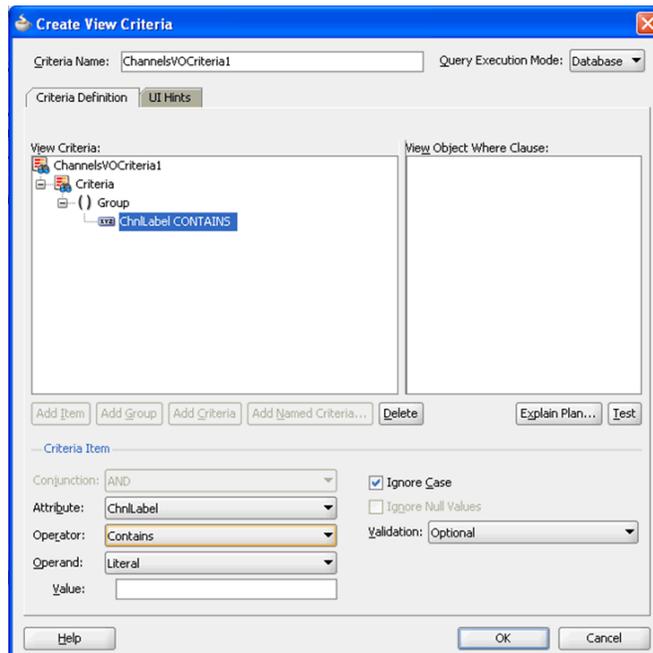
To set up the ChannelsVO view object for LOV:

1. In **JDeveloper**, open the **ChannelsVO** view object from the **model** package.
2. Click the **Query** vertical tab on the left hand side.
3. On the **Query** tab, click the **Add (+)** button in the **View Criteria** section. The **Create View Criteria** dialog box appears.



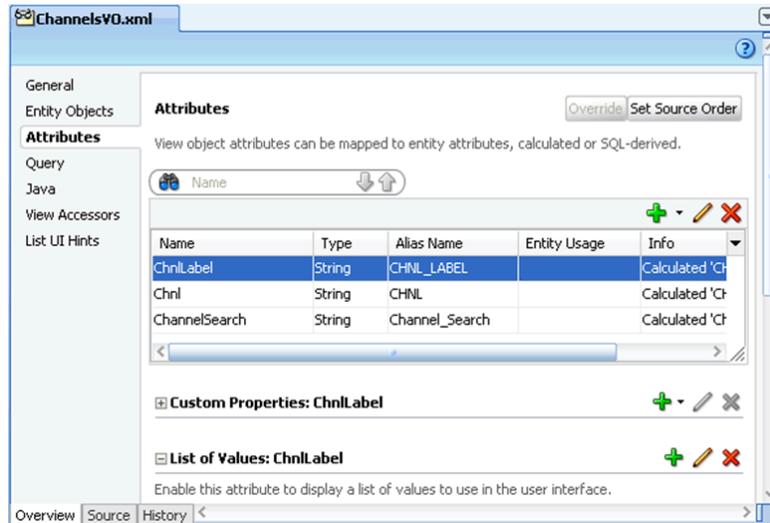
Create View Criteria Dialog Box

4. In the **Create View Criteria** dialog box, complete the following steps:
 - a. Select the default criteria name, and click **Add Item**.
 - b. In the **Criteria Item** section, select **DepartmentSearch** in the **Attribute** list.
 - c. In the **Operator** list, select **Contains**.
 - d. In the **Operand** list, select **Literal**.
 - e. Select the **Ignore Case** check box.



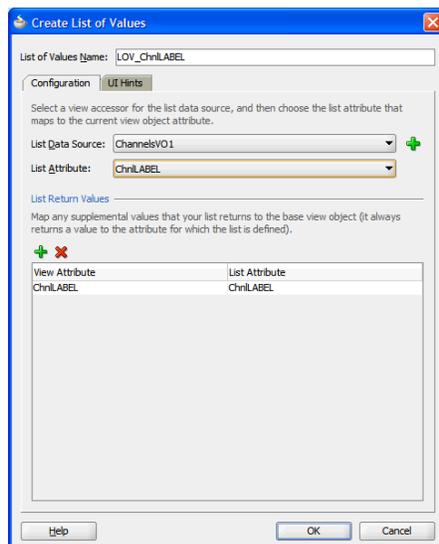
Create View Criteria Dialog Box with Options Selected

5. Use the following steps to define the attribute **ChnlLabel** of the view object as a LOV:
 - a. With the **ChannelsVO** view object open, click the **Attributes** vertical tab on the left hand side.
 - b. On the **Attributes** tab, select **ChnlLabel**, and then click the **Add (+)** button in the **List of Values** section. The **Create List of Values** dialog box appears.



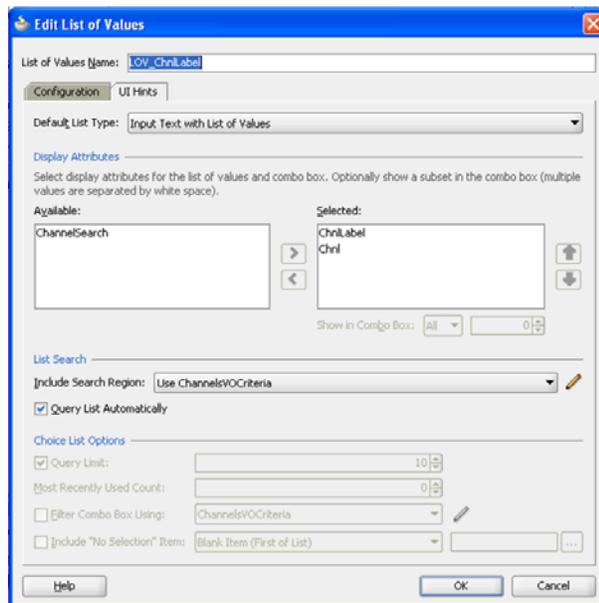
ChnlLabel Attribute Selected on the Attributes Tab

- c. In the **Create List of Values** dialog box, click the **Configuration** tab.
- d. On the **Configuration** tab, click the **Add (+)** button next to **List Data Source**. The **View Accessors** dialog box appears.
- e. In the **View Accessors** dialog box, select the **ChannelsVO1** view object instance from the **Available View Object** list and move it to the **View Accessors** list.
- f. Click **OK**.
- g. In the **Create List of Values** dialog box, select **ChnlLabel** from the **List Attribute** list.



Configuration Tab on the Create List of Values Dialog Box

- h. Click the **UI Hints** tab.
- i. On the **UI Hints** tab, select **Input Text with List of Values** in the **Default List Type** list.



UI Hints Tab on the Create List of Values Dialog Box

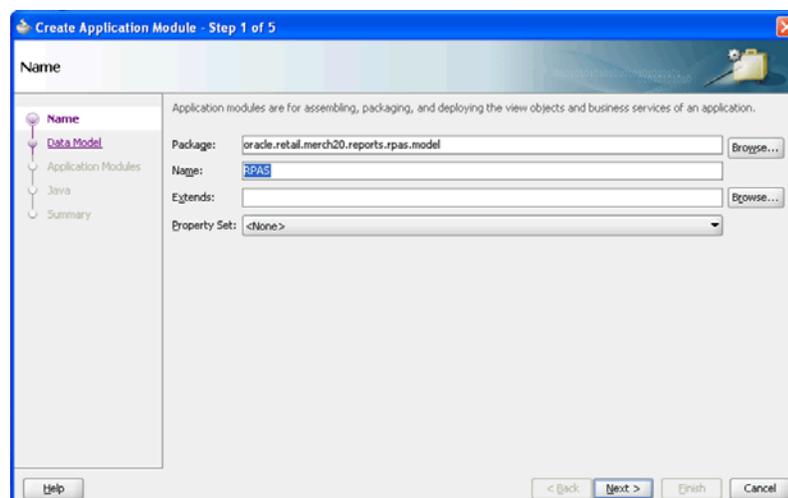
- j. Move the **ChnlLabel** and **Chnl** attributes from the **Available** list to the **Selected** list.
- k. In the **List Search** section, select **ChannelsVO** criteria in the **Include Search Region** list.
- l. Select the **Query List Automatically** check box.
- m. Click **OK**.

Setting Up the Application Module

Application module represents a collection of all view objects and services that are used together to complete a unit of work. This section describes how you can create an application module and include all the view objects created for the project.

To set up an application module:

1. In **JDeveloper**, right-click the **Model** folder in the **Application Navigator** panel.
2. From the right-click menu, click **New**. The **New Gallery** wizard dialog box appears.
3. On the **New Gallery** wizard, in the **Categories** section, select **ADF Business Components** under **Business Tier**.
4. In the **Items** section, select **Application** module, and click **OK**. The **Create Application Module** dialog box appears.
5. On the **Create Application Module** dialog box, in the **Name** screen, enter the package as **oracle.retail.merch20.reports.rpas.model**.
6. In the **Name** field, enter **RPAS_AM**.



Name Screen on the Create Application Module Wizard

7. Click **Next** and navigate to the **Application Modules** screen.
8. In case you have created all the required view objects, they will be appear in **Available View Objects** list.
9. Select all the view objects and move them to the **Data Model** list. This step can also be completed after **Application** module is formed.
10. Click **Finish**.

Setting Up the User Interface

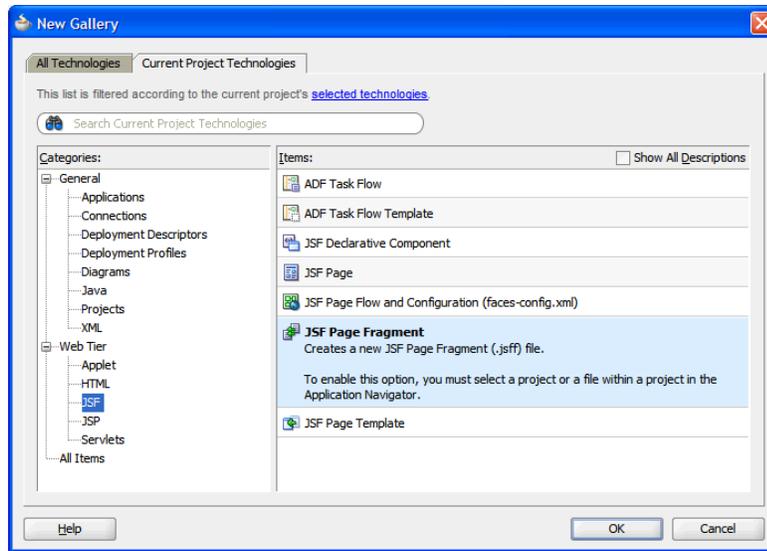
The user interface will include a form, where users will enter all the relevant information to create an item. This section describes how you can create the user interface using the ADF faces. It also includes information on the code you may want to write for capturing the values entered by the users.

The user interface will be created as a JSF page fragment. The page fragment will then be used to create the ADF task flow.

This user interface form may contain several user interface components. For illustration purposes, this section covers only the usage of the Channels LOV input field and the Measures bar graph. Full code listing for this page fragment along with the corresponding backing bean and page definition files are provided in the appendix section.

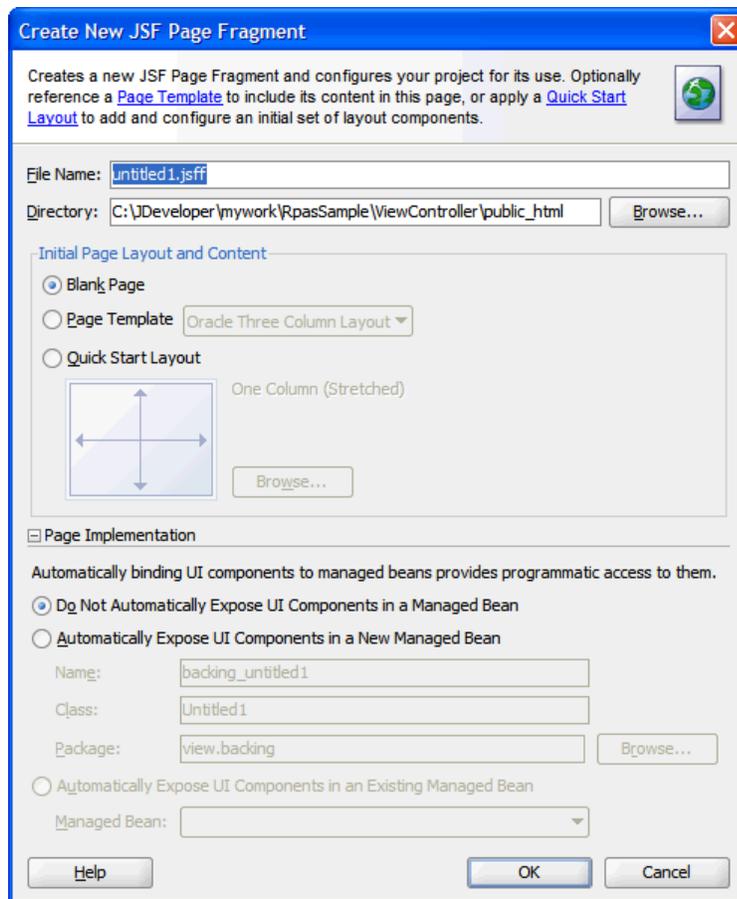
To set up the user interface:

1. In **JDeveloper**, expand the **ViewController** project in the **Application Navigator**.
2. Right-click on the **Web Content** folder, and then click **New**. The **New Gallery** wizard appears.
3. On the **New Gallery** wizard, in the **Categories** section, select **JSF** under **Web Tier**.



New Gallery Wizard

- In the **Items** section, click **JSF Page Fragment**, and then click **OK**. The **Create New JSF Page Fragment** wizard appears.



Create New JSF Page Fragment Window

- On the **Create New JSF Page Fragment** window, enter **dynamicRPAS.jsff** in the **File Name** field.

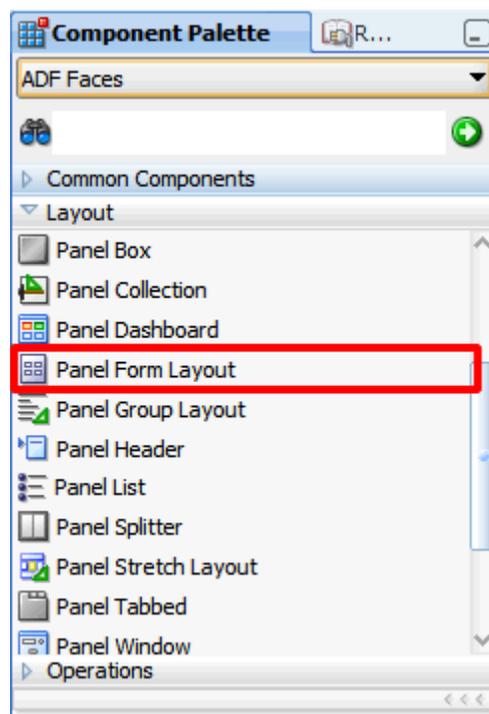
6. In the **Page Implementation** section, select the **Automatically Expose UI Components in a New Managed Bean** check box.
7. Click **OK**.

A new page fragment will be created in the Web Content directory. The corresponding backing bean will be created in the project's application sources under the package `oracle.retail.merch20.reports.rpas.view.backing`.

Note: Backing bean is useful to define any custom user interface functionality that you may require. It is a good practice to have one backing bean per page.

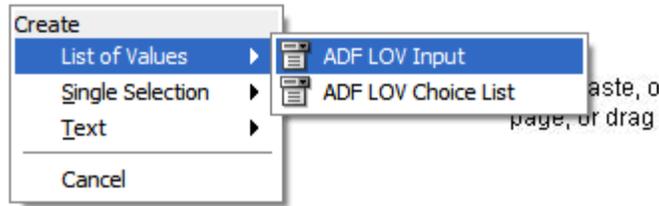
Once the page is created, open the page to construct the user interface as follows:

1. From the **Components Palette**, drag and drop the **Panel Form Layout** under **ADF Faces** category.



Component Palette

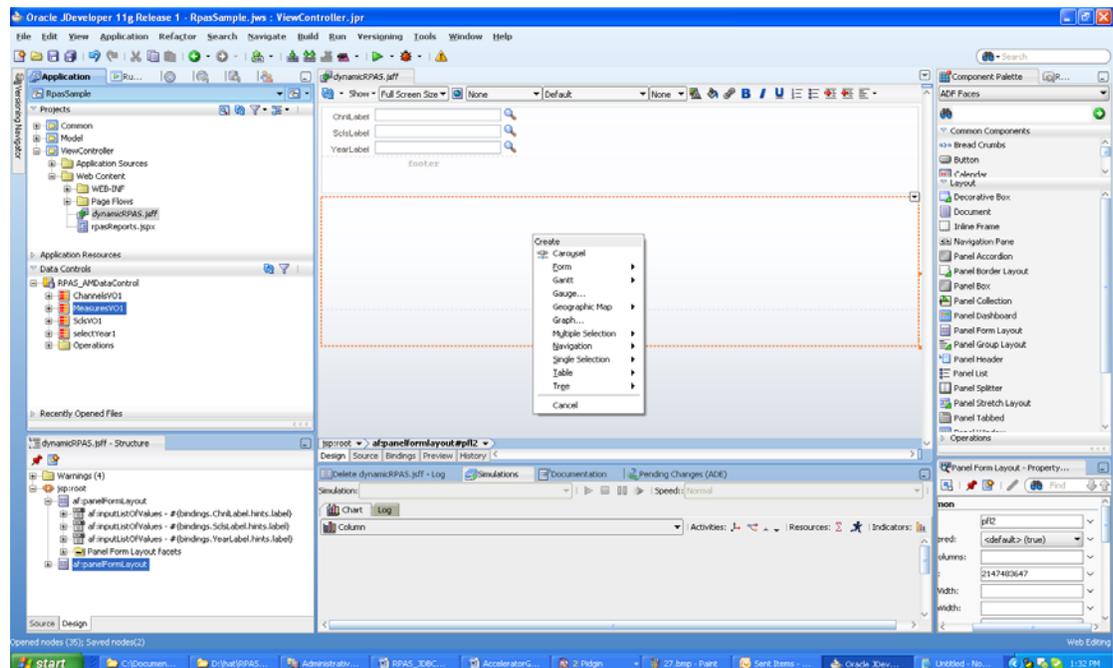
2. In the **Application Navigator** panel, expand the **Data Control** area.
3. Under **Data Control**, expand **RPAS_AMDDataControl**.
4. Expand the **ChannelsVO1** view object instance.
5. Drag and drop **ChnlLabel** attribute to the panel form layout.
6. From the **Drop** option menu, select **List of Values**, and then select **ADF LOV Input**.



Drop Option Menu

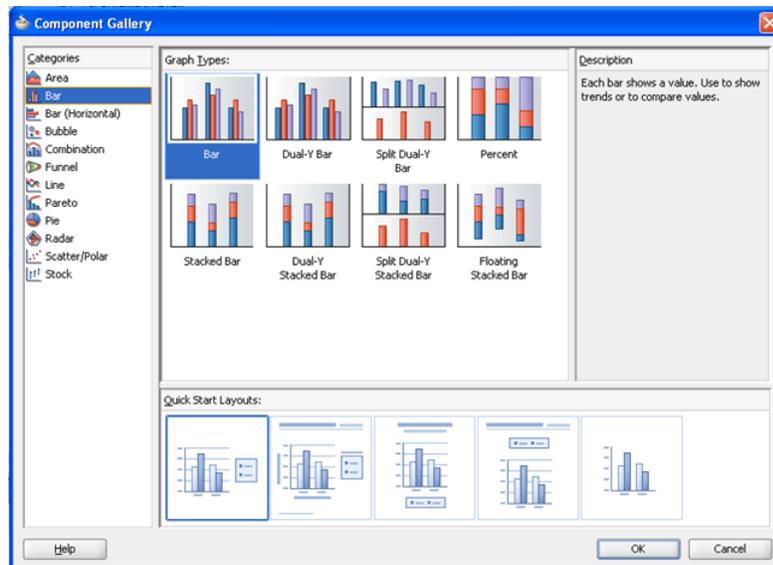
An input box will appear with a search icon on right.

7. Right-click on the page fragment, and click **Go to Page Definition**.
8. In the page definition file, on the **Binding and Executables** tab, observe that a **ChnlLabel** binding is created along with a **ChannelsVOIterator** executable.
9. Drag and drop the **MeasuresVO** as a bar graph on the page.



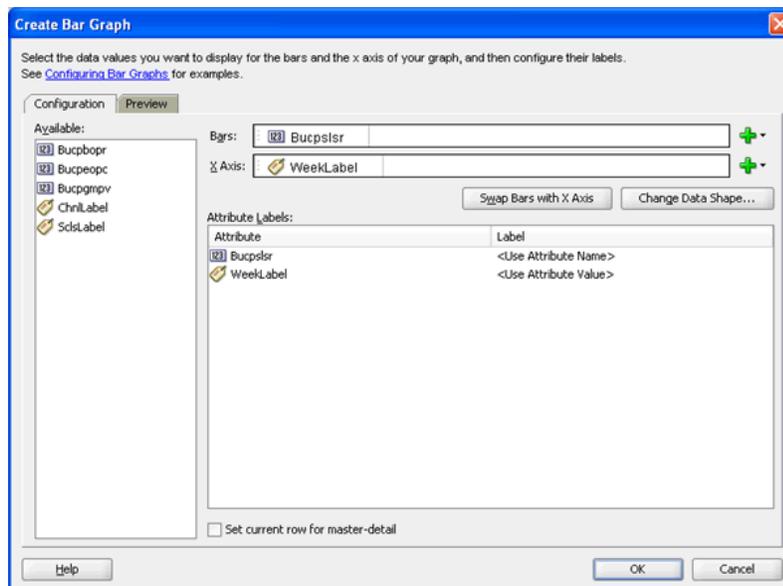
Drag and Drop the MeasuresVO as a Bar Graph

10. From the **Component Gallery**, select **Bar** graph, and click **OK**.



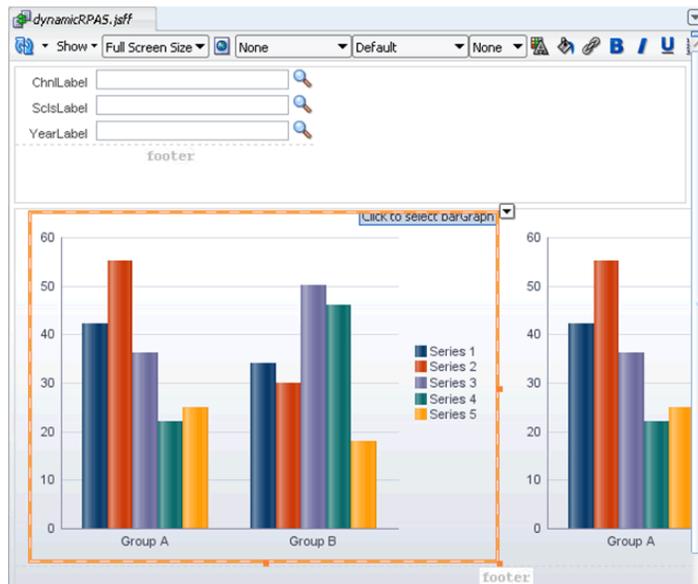
Component Gallery Window

11. On the **Create Bar Graph** window, specify the attribute for the **Bars** and **X Axis**.



Create Bar Graph Window

Repeat this process for all the input components required in the user interface. You may copy the code listing for this page along with the associated backing bean and page definition code from the appendix section. After adding all the user interface components, the page will appear as below.

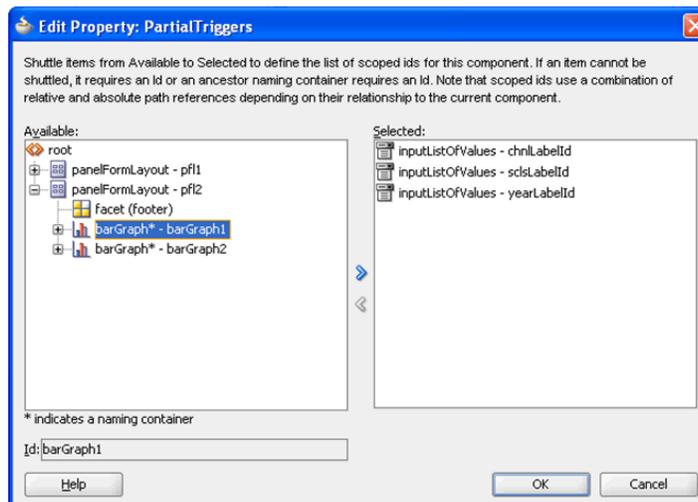


Screen with All User Interface Components

Note: Setting the *auto submit* property to *true* ensures that the value entered for the ChnlLabel will be available to the server in the same request cycle without explicitly submitting the page (Partial page submit).

Ensure that the *auto submit* property is set to *true* for all the LOVs.

Set the partial trigger of the bar graph such that the bar graph reflects the values selected in LOV. As shown below, the partial trigger has been set to the channel, subclass, and year LOV. This way the the bar graph reflects the changes made to any of these LOVs.



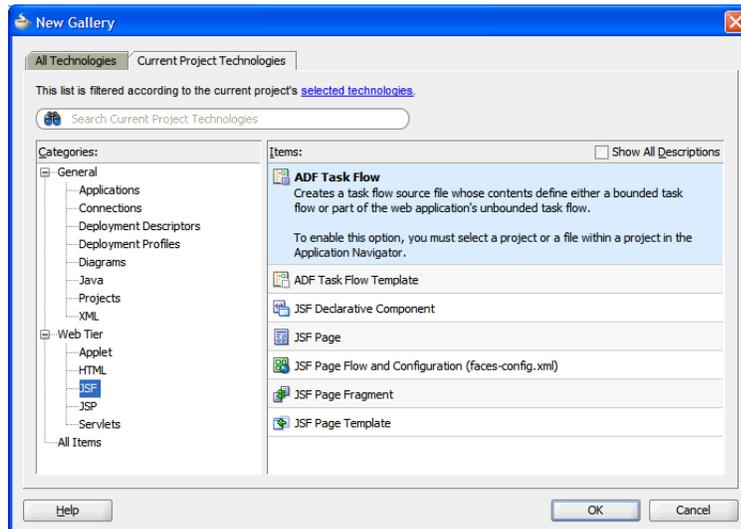
Edit Property: Partial Triggers Window

Setting Up an ADF Task flow

The user interface set up in the previous section is a JSF page fragment. It cannot run independently, unless it is included as a region in a JSF page. This section describes how you can use this JSF page fragment to define a task flow. Once the task flow is defined, it can then be included as a region in any JSF page.

To set up an ADF task flow:

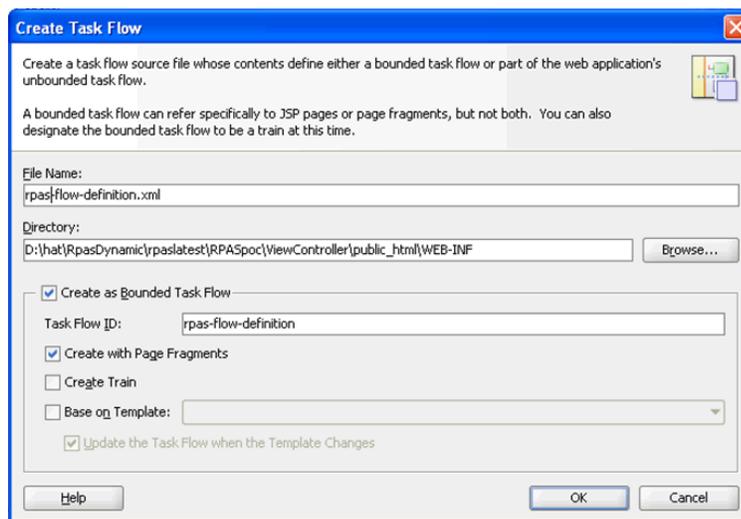
1. In **JDeveloper**, under the **Application Navigator** panel, right-click the **WEB-INF** folder within the **Web Content** folder of your project.
2. From the right-click menu, click **New**. The **New Gallery** wizard appears.
3. On the **New Gallery** wizard, in the **Categories** section, click **JSF** under **Web Tier**.
4. In the **Items** section, click **ADF Task flow**, and then click **OK**.



New Gallery Wizard

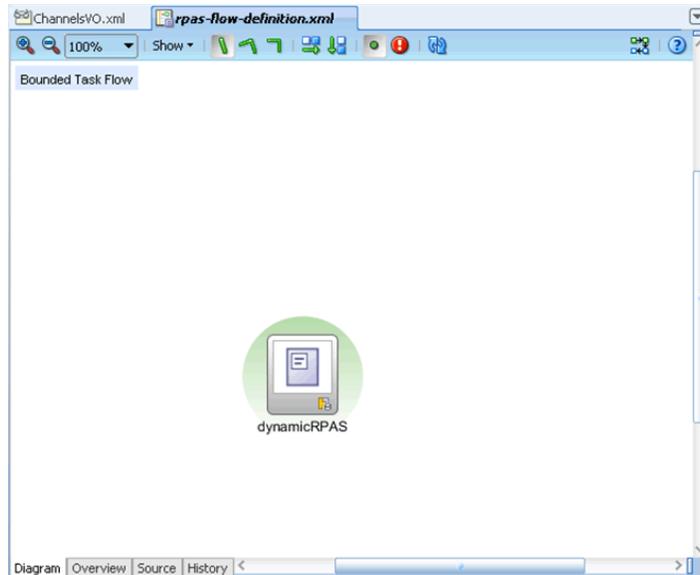
The **Create Task Flow** window appears.

5. On the **Create Task Flow** window, in the **File Name** field, set the file name as **rpas-flow-definition.xml**.



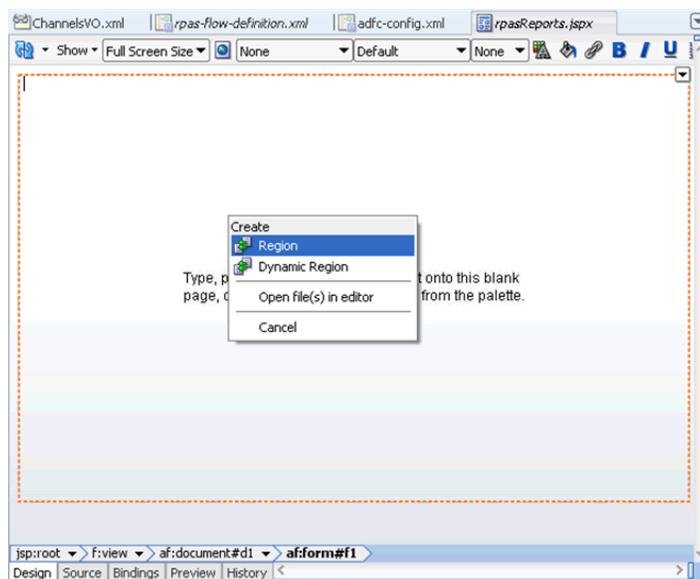
Create Task Flow Window

6. Click **OK**. A task flow definition file is created in the folder.
7. Open the task flow definition file.
8. Drag and drop the page fragment created in the previous section in the task flow diagram.



Page Fragment Added to the Task Flow Diagram

9. Save all files.
10. Now create a new JSF page.
11. Drag and drop the task flow definition file onto the JSF file as a region.



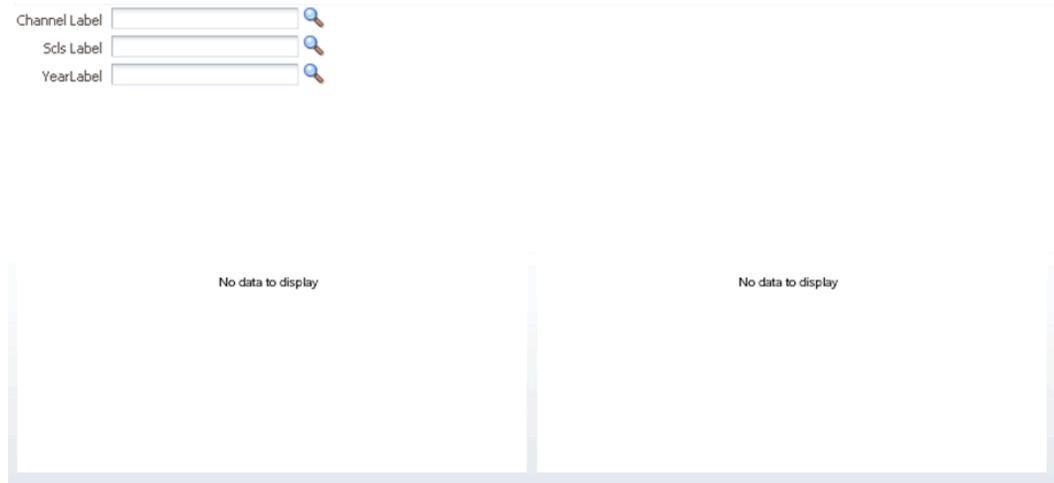
Drop Menu to Drag and Drop the JSF File as a Region

12. Run the JSF page.
- The project will deploy in the built-in WebLogic server and the page will appear in a Web browser window.

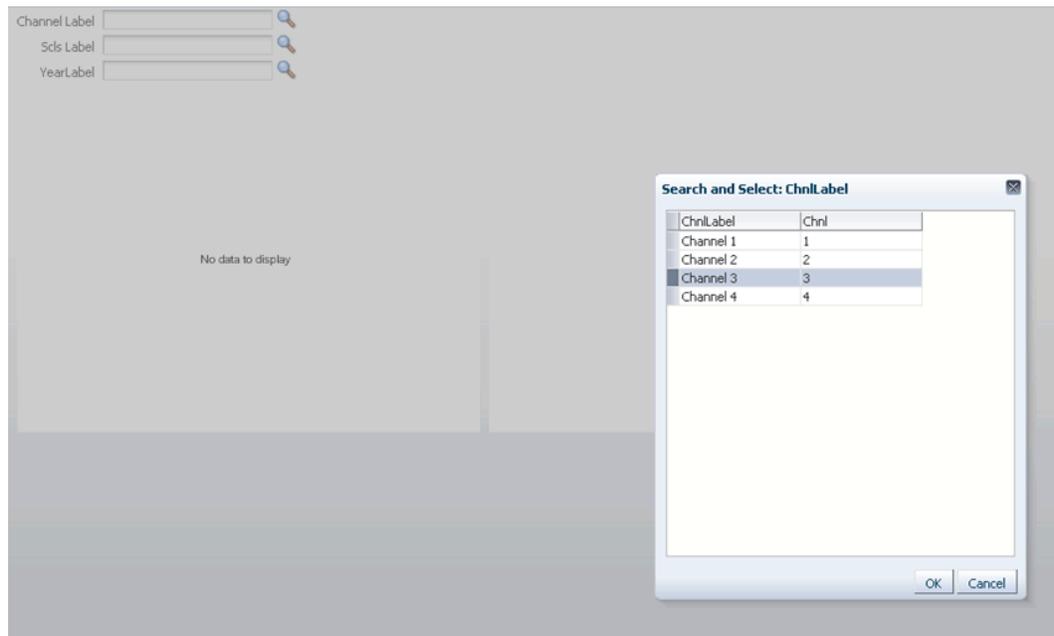
Review the Output

Once the page appears in the Web browser window, review the JSF page output. This section describes the typical work flow to use or review the JSF page output.

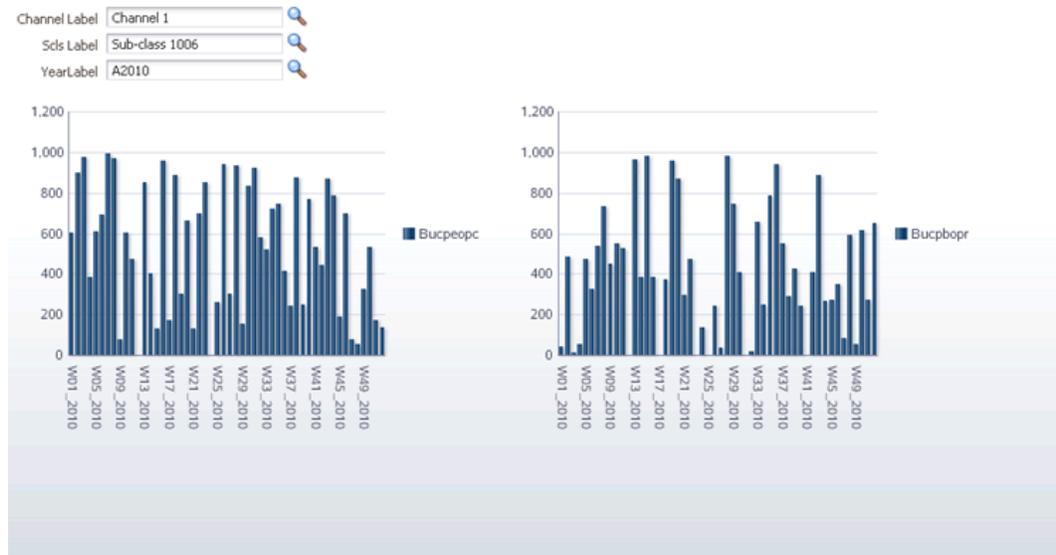
The following figure illustrates how the page appears in the Web browser window:



The following figure highlights the Search and Select: ChnlLabel window that appears when you click the Search icon next to the Channel Label field:



The bar chart will appear once you select valid values in all the three LOV fields. The following figure highlights the bar chart output:

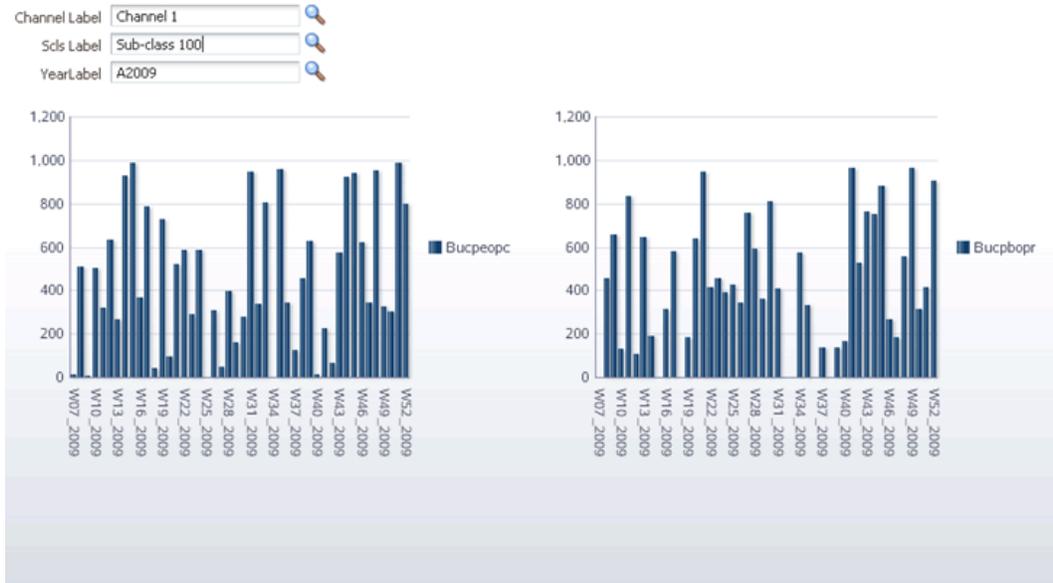


Creating a Managed Java Class

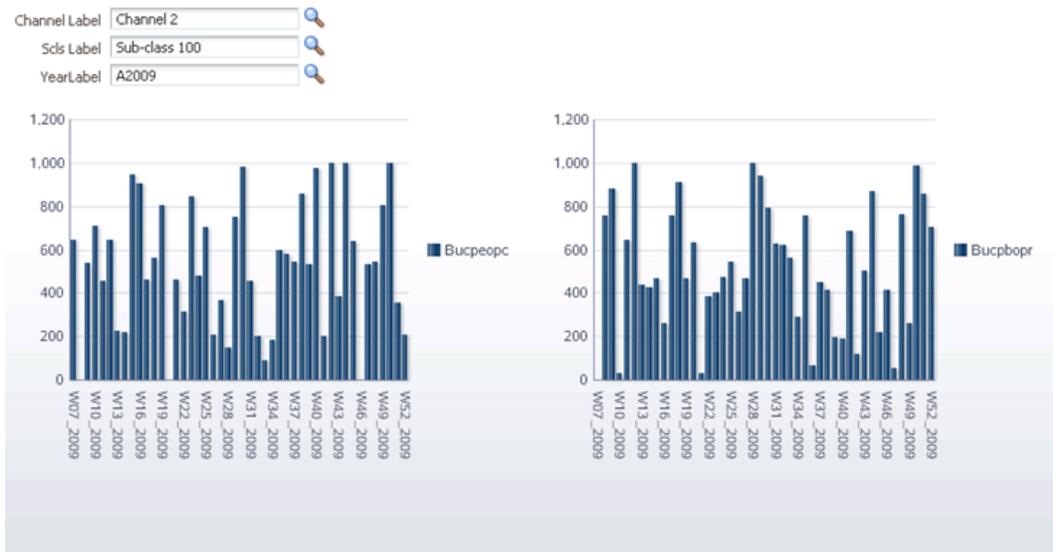
You can create a managed Java class to set the where clause of the MeasuresVO query before rendering the page. This is a requirement specific to this sample application. If this is not one of the business requirements of the application you created, you may skip this section.

To create a managed Java class:

1. In **JDeveloper**, under the **Application Navigator** panel, navigate to **rpas** within **Application Sources**.
2. Right click on **rpas**, and from the right-click menu, click **New**. The **New Gallery** wizard appears.
3. On the **New Gallery** wizard, in the **Categories** section, click **General**.
4. In the **Items** section, click **Java Class**, and then click **OK**. The **Create Java Class** window appears.
5. In the **Create Java Class** window, enter **rpasManaged** in the **Name** field and enter **oracle.retail.merch20.reports.rpas.managed** in the **Package** field.
6. Click **OK**.
7. Define this managed bean in the task flow as shown in the figure below:



When users change the value in the Channel field (to channel 2), the bar chart gets refreshed to reflect the updated information (as shown in the figure below).



Appendix A

Best Practices

This appendix describes some of the best practices that will help you plan your implementation. It also lists the guidelines for creating view objects used in the application discussed in this document.

Prefer High Level Fact Tables Over Lower Level Fact Tables

An RPAS measure has a base intersection. For example, the Regular Sales Retail measure in the MFP Retail domain has a base intersection of CHNL, SCLS, and WEEK.

The RPAS ODBC presents this measure in different tables, such as fact_chnlsclsweek, fact_chnlclsweek, fact_chnldeptweek, and so on. High level fact tables are those having dimensions higher in the hierarchy. For example, fact_chnldeptweek is a higher level table than fact_chnlclsweek because the dept dimension is at a higher level than cls dimension.

When possible, query the high level fact tables other than lower level fact tables, since the high level fact table has fewer records.

Prefer Calculations in RPAS

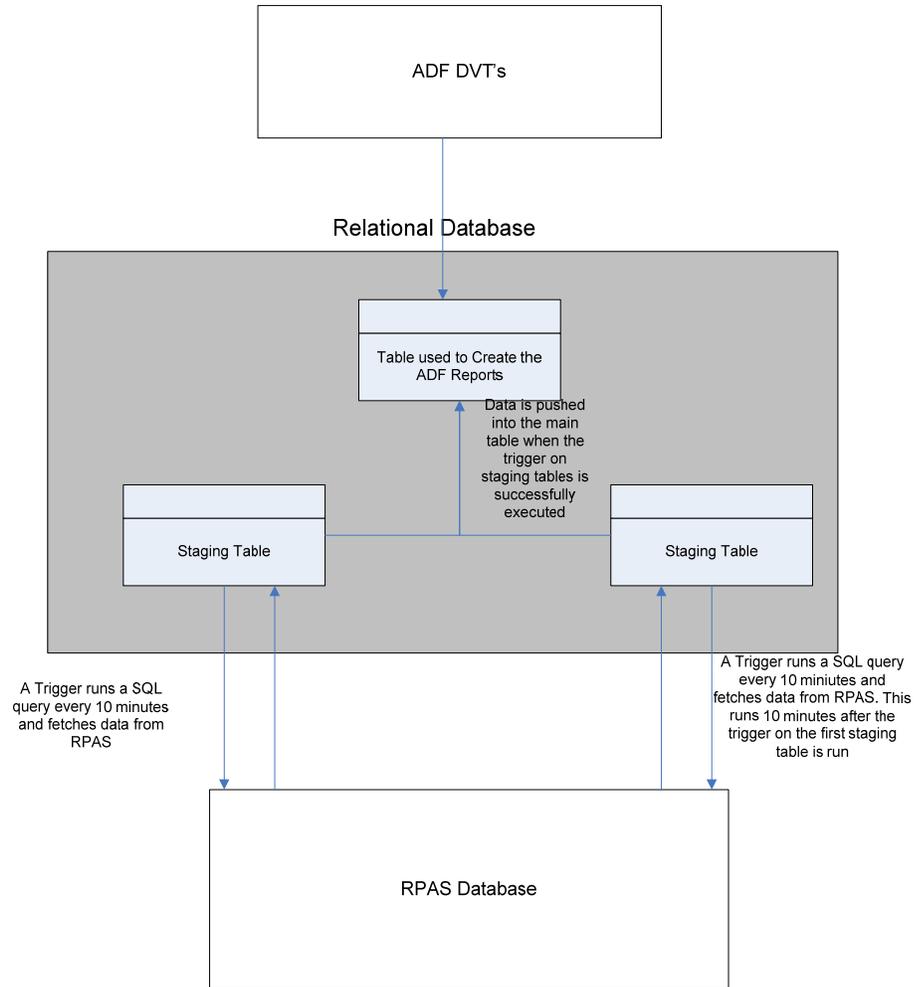
Calculations can be done either in ADF or RPAS. There are 2 benefits when doing calculations in ADF:

- Reporting on real time data. If calculating in RPAS, users have to run a batch before ADF can read the calculated measure. There will be time delay between the batch and the read.
- No need to store the calculated measure in anywhere.

Calculations are faster when they are done in RPAS. Since ADF issues less SQLs, RPAS engine is closer to the data and the batch run already does the calculation before ADF reads it.

Move the Data to the Relational Database

There are various advantages on moving the data to relational database using the architecture outlined below.



This eliminates the performance issues encountered while using RPAS JDBC/ODBC drivers when dealing with large amount of data.

Guidelines to Create View Objects Used in the Application

There are several read only view objects used in this application. All these view objects are based on RPAS schema. For illustration purposes, the Channels view objects are already explained in detail in the sections above. For rest of the view objects, all required information is provided below:

Measures View Object

Name: MeasuresVO

Category: Read only view object

Related table in RMS: FACT_CHNLCLSWEK

Query used:

```
select DC.CHNL_LABEL, DS.SCLS_LABEL, DW.WEEK_LABEL, BUCPSLSR,
BUCPGMPV,BUCPEOPC,BUCPBOPR
From
FACT_CHNLSCLSWEK F,
DIM_CHNL DC,
DIM_SCLS DS,
DIM_WEEK DW
Where
DC.CHNL = F.CHNL and
DS.SCLS= F.SCLS and
DW.WEEK = F.WEEK
```

Note: The “where” clause is appended in the backing bean dynamically as and when the user selects values from LOV

SubClass View Object

Name: SclsVO

Category: Read only view object

Related table in RPAS: DIM_SCLS

Query used:

```
select SCLS,SCLS_LABEL,concat(concat(SCLS_LABEL, '-'), SCLS) AS Scls_Search from
dim_scls group by scls,scls_label
```

Bind variables used:

- None

Attributes to be used in UI:

- SclsLabel: It is intended to be used as ADF LOV input component in UI. This requires the attribute to be configured for LOV as explained in ChannelsVO creation for ChnlLabel attribute.

Year View Object

Name: SelectYearVO

Category: Read only view object

Related table in RMS: DIM_YEAR

Query used:

```
select YEAR, YEAR_LABEL,concat(concat(YEAR_LABEL, '-'), YEAR) AS Year_Search from
DIM_YEAR
```

Attributes to be used in UI:

- **Year_label:** It is intended to be used as ADF LOV input component in UI. This requires the attribute to be configured for LOV as explained in ChannelsVO creation for ChnlLabel attribute.



Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Copyright © 2011, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.