Retail Enabled Oracle Data Integrator Integration

Implementation Guide

Release 13.3

January 2012

ORACLE®

Retail Enabled Oracle Data Integrator Integration Implementation Guide, Release 13.3

Primary Author: Barrett Gaines

Licensing Note: This media pack includes a Restricted Use license for Oracle Retail Predictive Application Server (RPAS) - Enterprise Engine to support Retail Enabled Oracle Data Integrator Integration only.

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(ii) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(iii) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(iv) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

3 ODI Setup

4 Running ODI

5 Fashion Planning Bundle Integration

6 RPO, APC-RPO, RDF Integration

A Building Custom Data Integration Interfaces

List of Figures

List of Tables

Send Us Your Comments

Retail Enabled Oracle Data Integrator Integration Implementation Guide, Release 13.3

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at http://www.oracle.com.

Preface

This Implementation Guide provides critical information about the processing and operating details of Retail Enabled ODI Integration, including the following:

- System configuration settings
- Technical architecture
- Functional integration dataflow across the enterprise
- Batch processing

Audience

This guide is for:

- Systems administration and operations personnel
- Systems analysts
- Integrators and implementers
- Business analysts who need information about Retail Enabled ODI Integration processes and interfaces

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documentation

For more information, see the following documents in the Retail Enabled Oracle Data Integrator Integration Release 13.3 documentation set:

- *Retail Enabled Oracle Data Integrator Integration Release Notes*

For more information about the Fashion Planning Bundle applications that use Retail Enabled ODI Integration, see the following documentation sets:

- Oracle Retail Item Planning documentation
- Oracle Retail Item Planning Configured for COE documentation
- Oracle Retail Clearance Optimization Engine documentation
- Oracle Retail Assortment Planning documentation
- Oracle Retail Merchandise Financial Planning documentation
- Oracle Retail Size Profile Optimization documentation

For more information about the RDF, RPO, and APC-RPO applications that use Retail Enabled ODI Integration, see the following documentation sets:

- Oracle Retail Demand Forecasting documentation
- Oracle Retail Regular Price Optimization documentation
- Oracle Retail Analytic Parameter Calculator for Regular Price Optimization documentation

For more information about installing ODI, see the following document:

- *Oracle Data Integrator Installation Guide*

This guide, along with other ODI documentation, can be found at

http://www.oracle.com/technology/products/oracle-data-integrator/index.html

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 13.3) or a later patch release (for example, 13.3.1). If you are installing the base release, additional patch, and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix releases can contain critical information related to the base release, as well as information about code changes since the base release.

Oracle Retail Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also available on the following Web site:

http://www.oracle.com/technology/documentation/oracle_retail.html

(Data Model documents are not available through Oracle Technology Network. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

Conventions

The following text conventions are used in this document:

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

1

Oracle Data Integrator Overview

Retail Enabled ODI Integration is a set of packages that allows users to easily execute data transfers among select RPAS applications.

Retail Enabled ODI Integration leverages Oracle Data Integrator (ODI) to store information about data interfaces among applications. ODI presents a user-friendly graphical interface for user-initiated data transfers and runtime monitoring. It also provides the ability to host application domains on different machines on a network, an ability not available with prior non-ODI integration strategies.

The RPAS applications that use Retail Enabled ODI Integration are Retail Demand Forecasting (RDF), Regular Price Optimization (RPO), Analytic Parameter Calculator for Regular Price Optimization (APC-RPO), and the Fashion Planning Bundle applications: Merchandise Financial Planning Retail (MFP Retail), Merchandise Financial Planning Cost (MFP Cost), Item Planning (IP), Item Planning Configured for COE (IP COE), Assortment Planning (AP), and Size Profile Optimization (SPO).

Oracle Data Integrator (ODI) is a widely used data integration software product. It provides a declarative design approach for defining data transformation and integration processes, resulting in faster and simpler development and maintenance. Based on its unique ELT (Extract, Load, and Transform) architecture (as opposed to the traditional ETL architecture), ODI guarantees the highest level of performance possible for the execution of data transformation and validation processes.

ODI is developed entirely in Java. Refer to ODI documents for version-specific requirements on Java Virtual Machine. ODI helps with the data integration and sharing among heterogeneous hardware platforms and software systems. Specifically, data integration among Relational Databases (such as Oracle DBMS) and RPAS-based applications, including data transfer between RDBMS and RPAS domains, and data transfer/sharing across multiple RPAS domains.

Note: Currently, ODI-RPAS integration includes batch data integration only. Support for real-time data transfer/replication is not included.

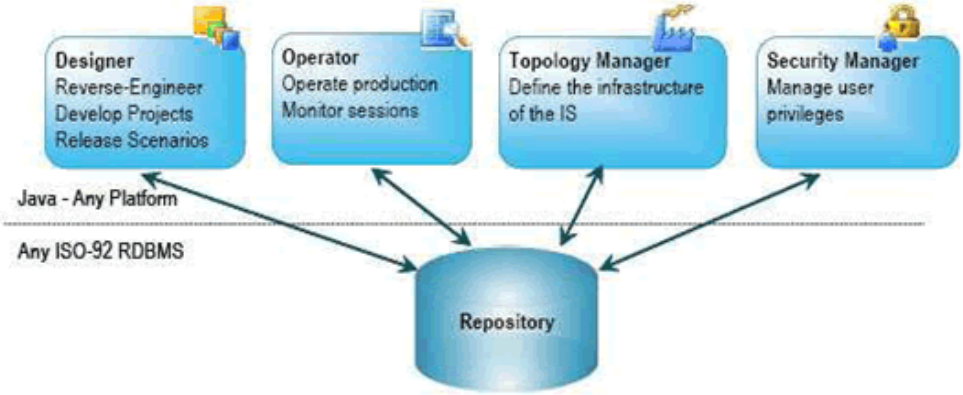
ODI Architectural Overview

ODI is built on several components all working together around a centralized metadata repository. Among the components, there are graphical modules that ODI users directly interact with, and run time components (ODI Agents) that run on source and target systems.

Graphical Modules

The graphical modules are Designer, Operator, Topology Manager, and Security Manager. They can be installed on any graphical platform that supports JVM 1.5.

Figure 1–1 Graphical Modules in ODI



- **Designer** defines declarative rules for data transformation and data integrity. All project development takes place in this module; this is where database and application metadata are imported and defined. The Designer module uses metadata and rules to generate scenarios for production. This is the core module for developers and metadata administrators.
- **Operator** manages and monitors production. It is designed for production operators and shows execution logs with error counts, the number of rows processed, execution statistics, the actual code that is executed, and so on. At design time, developers can use the Operator module for debugging purposes.
- **Topology Manager** defines the physical and logical architecture of the infrastructure. The administrators of the infrastructure or project register servers, schemas, and agents in the master repository through this module.
- **Security Manager** manages user profiles and their access privileges. Security Manager can also assign access privileges to objects and features. Security administrators generally use this module.

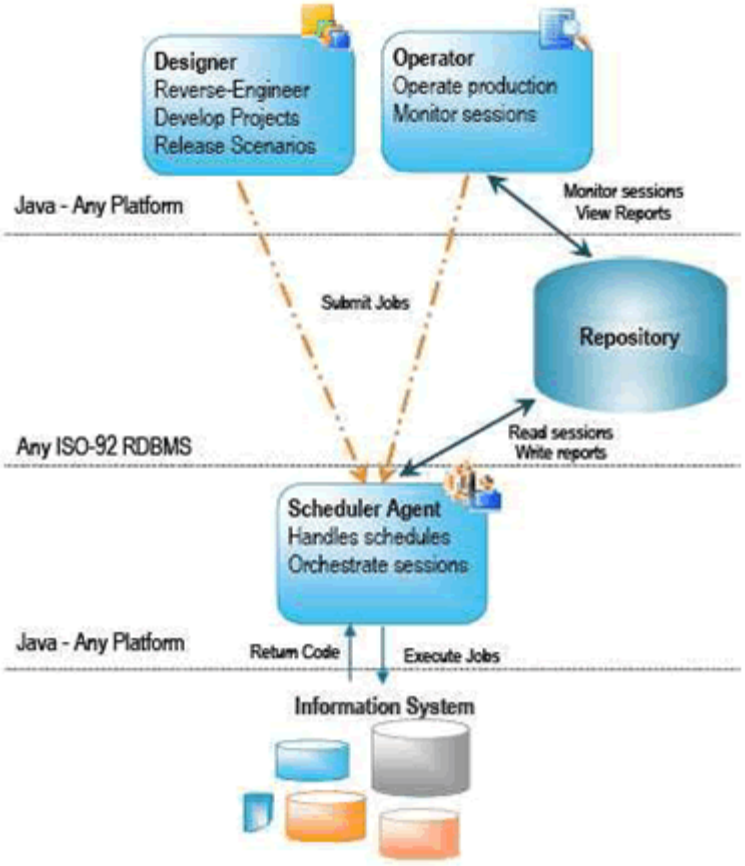
All modules store their information in the centralized repository.

Run-Time Components

At run time, the Scheduler Agent coordinates the execution of the scenarios (compiled data integration interfaces). Refer to ODI documents for platform and software requirements related to running the Scheduler Agent. Execution can be launched from one of the graphical modules or built-in scheduler, or triggered by a third-party scheduler.

The Agents need to be installed on tactical locations in the (distributed) information system to coordinate the integration processes and leverage existing systems.

Figure 1–2 Run-Time Components in ODI

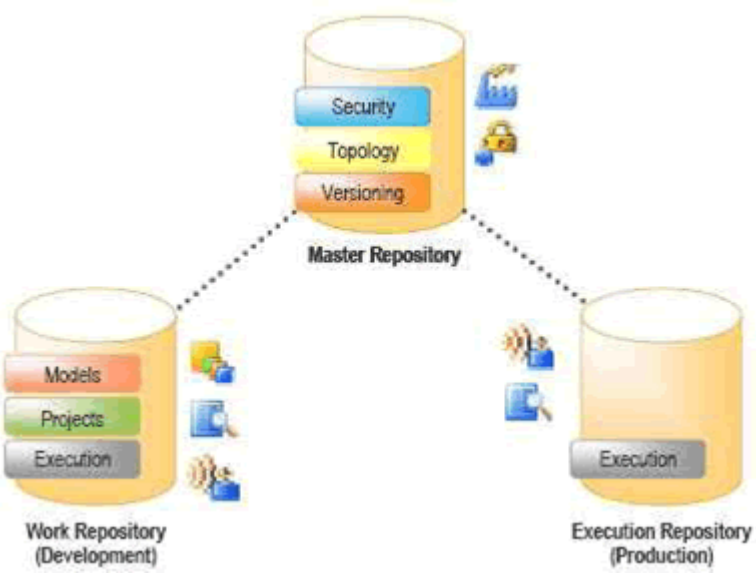


The Repository

The repository consists of a master repository and several work repositories. These repositories are databases stored in relational database management systems. All objects that the modules configure, develop, or use are stored in one of these repositories, and are accessed in client-server mode by the various components of the architecture.

There is usually only one master repository, which contains the security information (user profiles and privileges), the topology information (definitions of technologies and servers), and the versions of the objects. The information contained in the master repository is maintained with Topology Manager and Security Manager. All modules have access to the master repository, because they all store topology and security information there.

Figure 1–3 Repositories



Project objects are stored in the work repository. Several work repositories can coexist in the same installation. This is useful to maintain separate environments or reflect a particular versioning lifecycle-for example, development, qualification, and production environments.

A work repository stores information for the following:

- Models—including datastores, columns, data integrity constraints, cross references, and data lineage.
- Project—including declarative rules, packages, procedures, folders, knowledge modules, and variables.
- Run-time information—including scenarios, scheduling information, and logs.

Users manage the content of a work repository with the Designer and Operator modules. The Agent at run time also accesses work repositories. When a work repository is used to only store execution information (typically for production purposes), it is called an execution repository. An execution repository is accessed at run time with the Operator interface and by Agents.

Note: All work repositories are always attached to one and only one master repository.

2

Getting Started

This chapter describes how to use ODI with RPAS.

Requirements and Assumptions

Before setting up Fashion Planning Bundle integration or RPO, APC-RPO, RDF integration, it is assumed that you have installed the following:

- RPAS, including the RPAS JDBC Client and ODBC Server. For information, see the *Oracle Retail Predictive Application Server Installation Guide*.
- One of the following application sets on the same network-accessible server. Each application has one RPAS domain, each with one or more users already created. For more information, see the application's installation guide.
  - Two or more Fashion Planning Bundle applications: Size Profile Optimization (SPO), Assortment Planning (AP), Merchandise Financial Planning (MFP), Item Planning (IP), or Item Planning Configured for COE (IP COE).
  - Two or more of the following applications: Retail Demand Forecasting (RDF), Regular Price Optimization (RPO), or Analytic Parameter Calculator for Regular Price Optimization (APC-RPO).
- Oracle Database 10g or 11g.
- ODI Server 10.1.3.5 on the same machine as the domains and have set up a master repository and work repository in the Oracle Database.
- ODI Client 10.1.3.5 on one or more PCs. For more information, see the *Oracle Database Integrator Installation Guide*, 10g Release 3 (10.1.3).

ODI Installation

The ODI Client is used to create ODI project components as well as to manage and execute those components. The ODI Client contains graphical modules and therefore must be installed on one or more machines with graphical capability; typically this is a Windows machine. All required JDBC drivers (including RPAS JDBC driver) must be installed in the ODI driver directory in order for the graphical modules to connect to the ODI repositories, source, and target databases.

For detailed instructions on ODI installation, see the *ODI Installation Guide*.

ODI Configuration

A major task of ODI configuration is creating and configuring Master and Work repositories.

For detailed instructions on creating repositories in a relational database, refer to the *ODI Installation Guide*. There are no RPAS-specific requirements for ODI repositories.

Deployment of ODI and RPAS

After ODI and RPAS are both installed, perform the following steps:

1. Unzip the RPAS_HOME overlay using a zip program or from a command line (UNIX or MKS) as follows:

   ```
   cd "$RPAS_HOME"
   unzip -p "$FPB_HOME"/FPBServerRpasHomeOverlay.tar.zip | tar xpf -
   ```

2. Unzip the ODI_HOME overlay using a zip program or from a command line (UNIX or MKS) as follows:

   ```
   cd "$ODI_HOME"
   unzip -p "$FPB_HOME"/FPBServerOdiHomeOverlay.tar.zip | tar xpf -
   ```

3. Transfer the FPBClientOdiHomeOverlay.zip file to a machine that has the ODI client (graphical modules) installed and place it in the ODI import/export directory named impexp.

   ```
   $FPB_HOME/FPBClientOdiHomeOverlay.zip
   ```

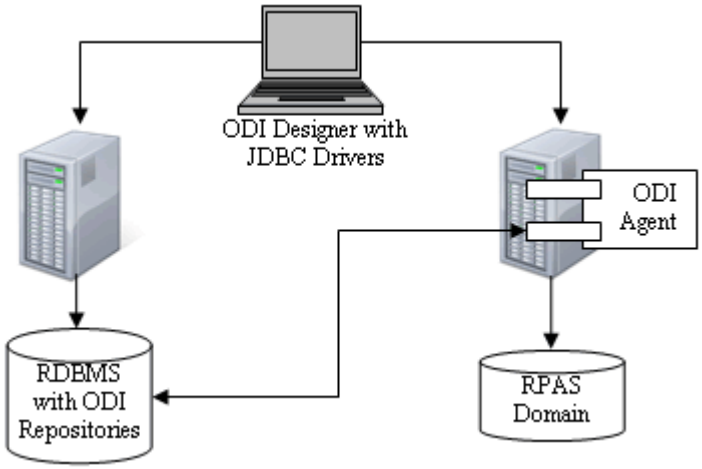
4. Unzip the file using a zip program or from a command line (UNIX or MKS) as follows:

   ```
   cd $ODI_HOME/impexp
   unzip -p FPBClientOdiHomeOverlay.tar.zip | tar xpf -
   ```

Data Integration Between a Relational Database and the RPAS Domain

In this configuration, one of the data sources is an RPAS domain. The other is a relational database management system (RDBMS), which can serve as ODI repositories. An ODI agent is required to be installed on at least the RPAS server.

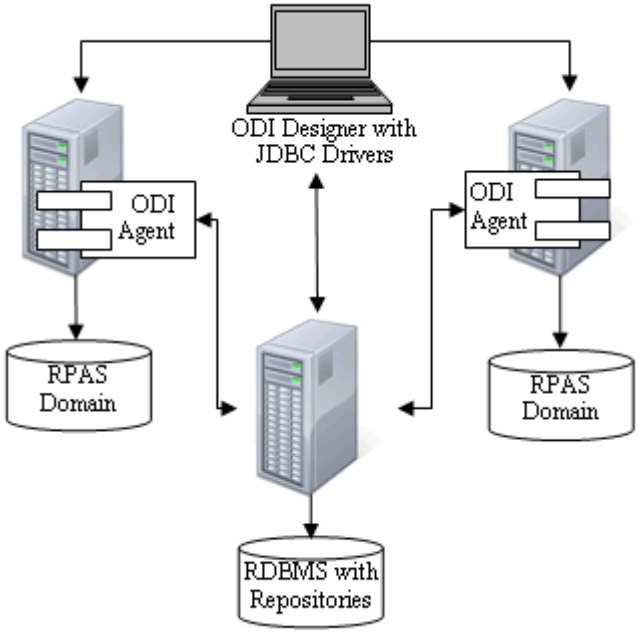
Figure 2–1 Data Integration Between Relational Database and RPAS Domain



Data Integration Between Two RPAS Domains

In this configuration, both data sources are RPAS domains. A third database, which must be an RDBMS, is required to serve as the ODI repository. The RPAS JDBC driver does not support table creation, so the RPAS domain cannot serve as an ODI repository. ODI agents are also required to be installed on the server of the target data source.

Figure 2–2 Data Integration between Two RPAS Domains



Installing ODI Agents with RPAS

At least one ODI agent must be set up to run on the servers hosting the domains.

On the RDBMS server, an ODI agent is not required, but is recommended for the data transfer job to take advantage of database-specific utilities.

Note: The ODI agent should not be confused with the RPASA ODBC agent.

In an ODI-RPAS deployment, RPAS load utilities are used for data loading. ODI agent must be installed and run on the servers where the target RPAS domains are located. The agent must have write access to the input and input/processed directories of the domain. The agent executes the RPAS load utilities for importing data to the domain.

Installation of the ODI agent is as easy as copying directories. For more information, refer to the *ODI Installation Guide*.

Perform the following steps to install ODI agents with RPAS:

1. Modify the file retailparams.sh located in the $ODI_HOME/bin directory. Instructions are in comments at the top of the file. As an example, the following are typical custom variables set up within this file:

   - Fashion Planning Bundle Example:

     ```
     :   ${FPB_APPS=AP MFP ITEMPLAN SIZEOPT}
     :   ${FPB_HOME=/vol/newyork5/retail}
     :   ${AP_HOME=/vol/apps/APFA}
     :   ${MFP_HOME=/vol/apps/mfpcst}
     :   ${ITEMPLAN_HOME=/vol/apps/itemplan}
     :   ${SIZEOPT_HOME=/vol/apps/sizeopt}
     :   ${FPB_HUB=$FPB_HOME/hub}
     :   ${AGENTLOGDIR=$FPB_HOME/log}
     ```

   - RPO/APC-RPO Example:

     ```
     :   ${FPB_APPS=RPO APCRPO}
     :   ${FPB_HOME=/vol/newyork6/retail}
     :   ${RPO_HOME=/vol/apps/RegPrice}
     :   ${APCRPO_HOME=/vol/apps/apcrpo}
     :   ${AGENTLOGDIR=$FPB_HOME/log}
     ```

   If the above syntax is used to set the variables, it is possible to override one or more of these variables in your .profile (or the equivalent shell session setup script, depending on the shell being used) or on the command line. If this has been done prior to running an ODI agent, then retailparams.sh sets only those variables that are not yet defined.

   Note: When running the ODI integration packages, you are asked to specify a context. The Topology Manager maps each context to exactly one ODI agent. At runtime, package-related scripts and programs are spawned by the specified ODI agent and therefore inherit the same environment that the agent had at the time it was started. For example, it is assumed that rpaslogin.ksh has been run prior to starting ODI agents, so that RPAS is available to programs launched from within ODI.

2. If you need to customize scripts called from ODI and those scripts use environment variables, ensure that the needed variables are set and exported before starting each ODI agent. If you are running multiple ODI agents, the values for these variables can be the same or different for each agent, as needed.

   - Set the environment variable AGENT_CONTEXT and export it before running each agent. For example:

     ```
     AGENT_CONTEXT="RETAIL_QA1" ; export AGENT_CONTEXT
     ```

   - Start the ODI agent from ODI's bin directory:

     ```
     ./agent.sh -port=<port number> -name=<agent name>
     ```

     where `<agent name>` is the physical agent name that is defined in the agent configuration in ODI Topology Manager.

For example, to run an ODI agent under the name bundle1_production using port 20100, start the agent as follows:

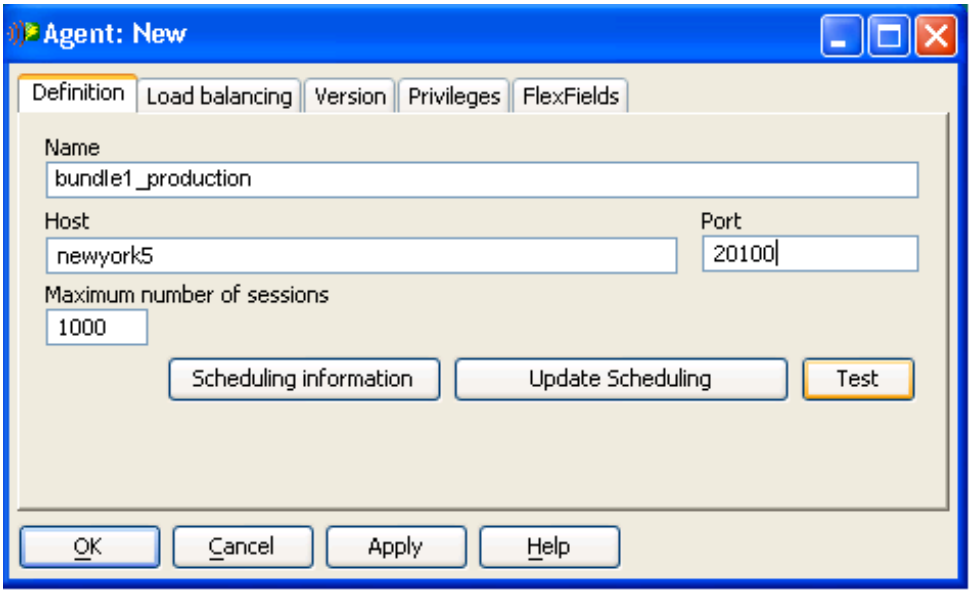
```
cd "$ODI_HOME/bin"
./agent.sh -port=20100 -name=bundle1_production > agent-20100.log 2>&1 &
```

- To stop this agent, use the agentstop.sh script (also in the ODI bin directory):

```
./agentstop.sh -port=20100
```

3. Add the ODI agent name (for example, bundle1_production) to the list of agents in Topology Manager:
   a. Open the Physical Architecture pane in Topology Manager.
   b. Right-click **Agents** and select **Insert Agent**.
   c. The Agent: New window appears. Enter the agent information.

Figure 2–3 Agent: New



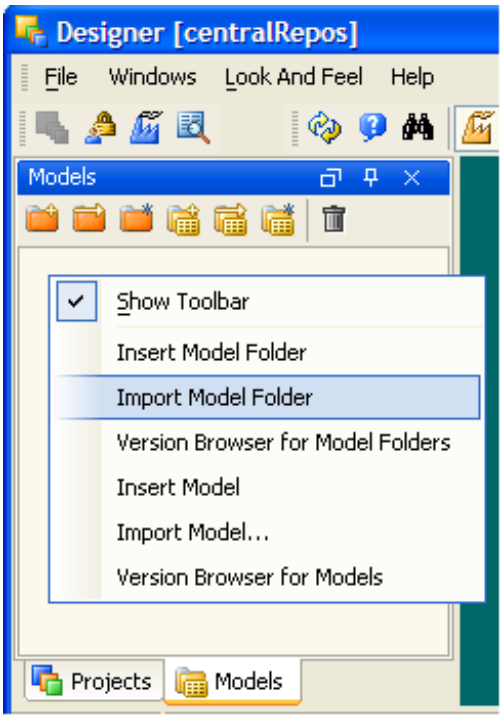
   d. Click **Test** to verify a connection can be made.
   e. Click **OK**.

Importing a Project

To import an ODI project in the master repository, perform the following steps on a machine with the ODI client installed.

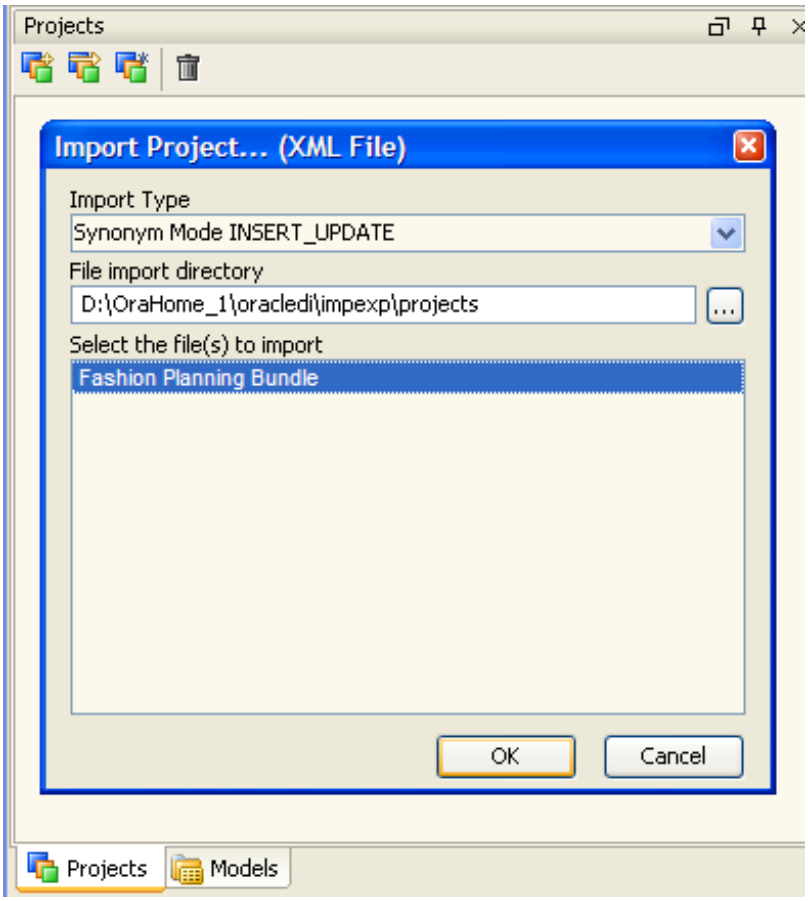
1. Unzip the FPBClientOdiHomeOverlay.tar.zip as described in the Deployment of ODI and RPAS section.
2. Open the ODI Topology Manager.
3. From the File menu, select **Import** and then select **Topology**.
4. Input the ODI impexp directory path when prompted.
5. Open the ODI Designer program.
6. Right-click a blank part of the Models pane background.
7. Select **Import** and then **Import Model Folder**, as shown in Figure 2–4.

Figure 2–4 Importing the Project



8. Navigate to the ODI impexp/projects directory and select the project, as shown in Figure 2–5.

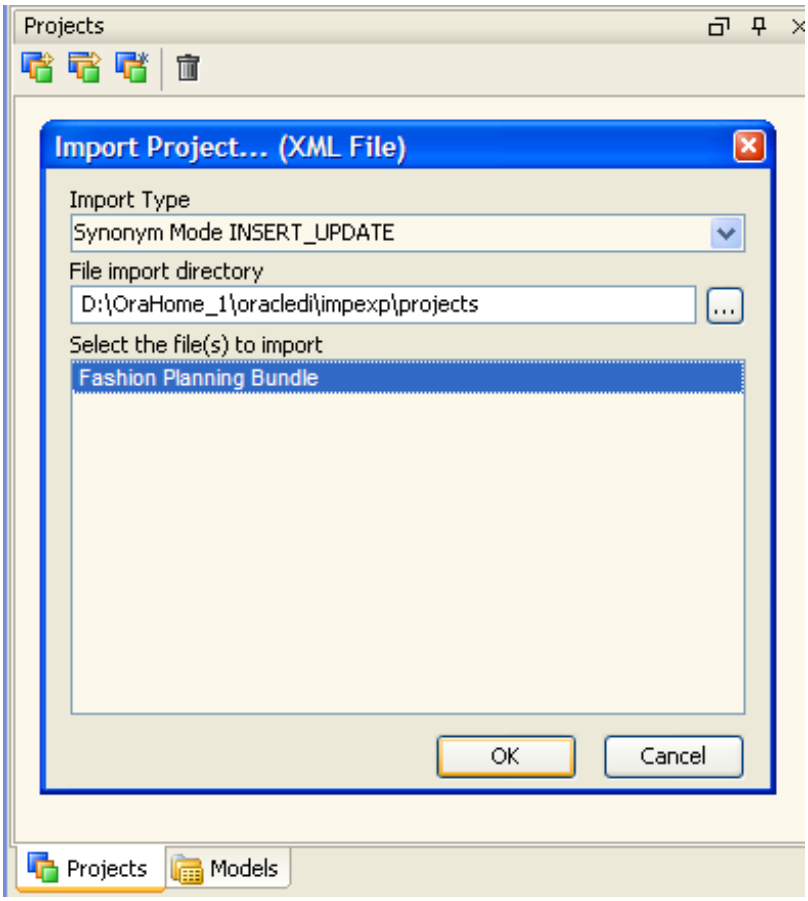
Figure 2–5 Import Project Folder (XML File)



9. Navigate to the Designer's Projects pane.
10. Right-click the pane background and select **Import Project**.
11. Navigate to the impexp/projects folder.

12. Select a project and click **OK**, as shown in Figure 2–6.

Figure 2–6 Import Project (XML File)



The project, along with its associated data models and knowledge modules, is installed to the master repository.

RPAS offers two knowledge modules and one technology for batch data integration. The two knowledge modules are used for:

- Loading measure data using loadMeasure utility
- Loading hierarchy data

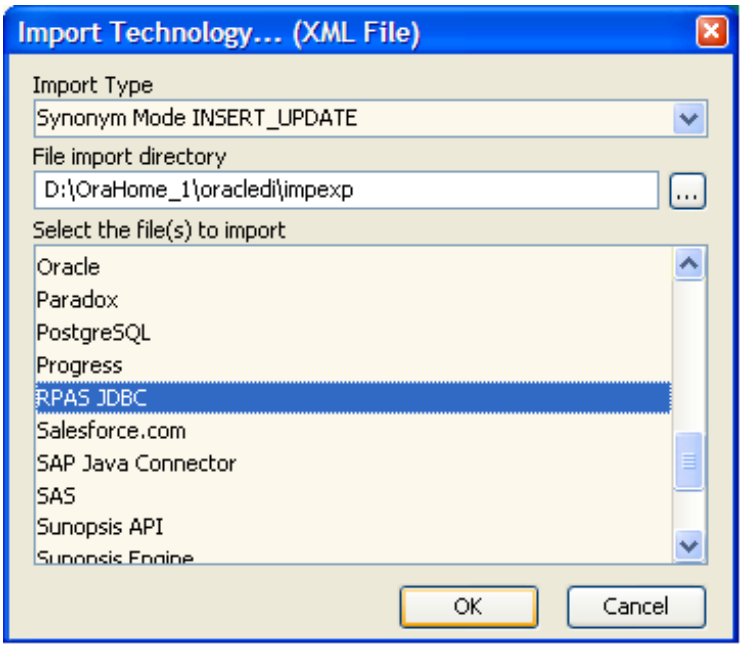
> **Note:** There is no RPAS-specific knowledge module for exporting data from RPAS because all data exports are performed using SQL through the RPAS JDBC driver.

Installing RPAS JDBC Technology

RPAS applications require installation of the RPAS JDBC technology. To install RPAS JDBC, perform the following steps:

1. Launch the ODI graphical module Topology Manager.
2. Open the Physical Architecture pane.
3. In the Technologies folder, right-click any technology (such as **File**).
4. Select **Import**, and then select **Import Technology**.
5. Navigate to the $ODI_HOME\impexp folder.
6. Select **RPAS JDBC** and click **OK**.

Figure 2–7 Import Technology (XML File)



Starting RPAS ODBC Agent and RPAS Data Service

Perform the following steps to start the RPAS ODBC agent and RPAS Data Service:

1. Before setting up ODI topology, ensure the RPAS Data Service is running on the same machine as your domains.
2. Run the customInstall.sh script, which is located in $RPAS_HOME/odbc, on the server hosting your domains.

   This script must be run at least once on the server. It creates the directory $RPAS_HOME/odbcserver and generates helper scripts under $RPAS_HOME/odbcserver/admin.

   It is acceptable to use all the default values suggested by the script prompts. It is not necessary to specify a domain path. When prompted for the domain path, press **Enter**.
3. Start the RPAS ODBC agent and the RPAS Data Service on the same machine as your domains. Enter the following:

   ```
   cd "$RPAS_HOME"/odbcserver/admin
   nohup ./startRPASDataService  > dataService.log 2>&1 &
   nohup ./startRPASODBCAgent > odbcAgent.log 2>&1 &
   ```

   These programs may be stopped independently by running the scripts stopRPASDataService and/or stopRPASODBCAgent (located in the same directory).

Creating Agents and RPAS Data Servers/Schemas

There are two technologies used in the integration: FILE and RPAS JDBC. Each data server uses one of these technologies.

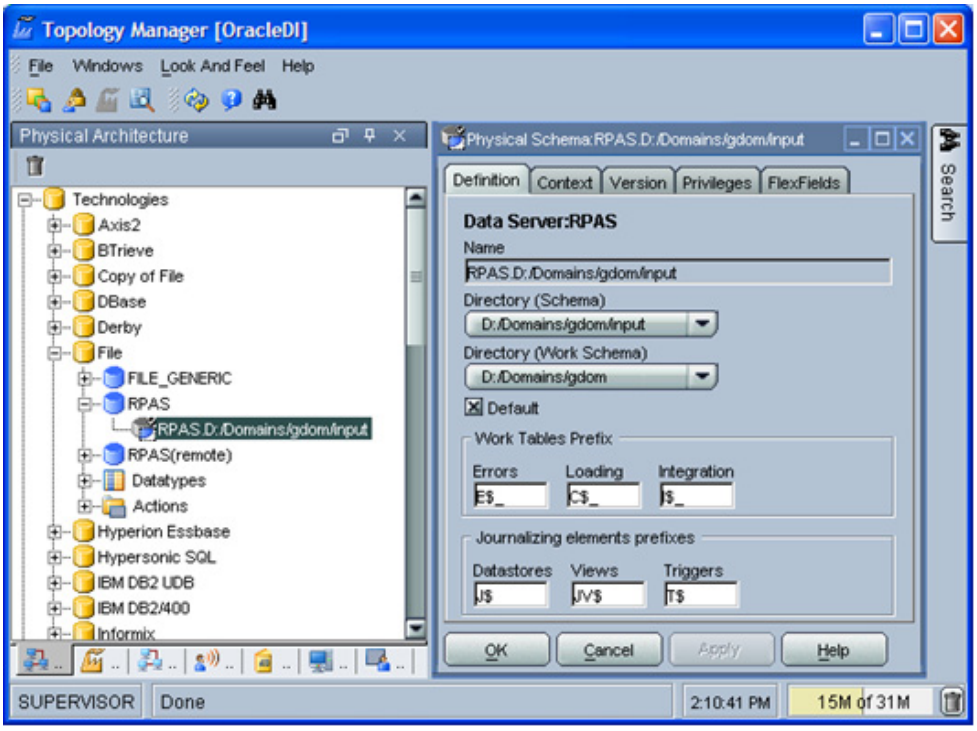
- The FILE technology is used by RPAS domains receiving data. This is the data server needed for the two RPAS integration knowledge modules (IKM) that use RPAS load utilities.
- The RPAS JDBC technology is used for the domain that is the data source.

> **Note:** In the Physical Architecture pane of the Topology Manager, there is exactly one RPAS JDBC data server for each source domain and one FILE data server for each target domain. A domain (either source or target) should have one data server of each type.

Perform the following steps to create agents and RPAS data servers and schemas:

1. In ODI Topology Manager, create logical and physical agents to connect to the listener agents that run on the machines where RPAS domains are located.
2. Create two types of physical data servers for RPAS: one under the preexisting FILE technology, and another under the RPAS JDBC Technology.
3. In the Directory(Schema) field, enter the path of the RPAS domain input directory, as shown in Figure 2–8.

Figure 2–8 Topology Manager Screen Showing RPAS Data Server Definition



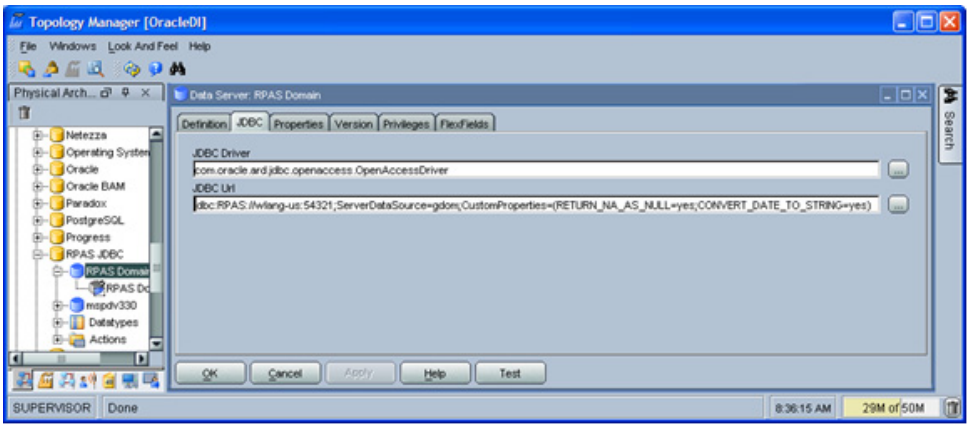
4. In the user and password fields, enter the domain's user name and password. Note that the user must have administration privileges. For instructions on setting administration privileges, see the "User Maintenance" chapter of the *RPAS Administration Guide for the Classic Client* or the *RPAS Administration Guide for the Fusion Client*.
5. Enter the following two CustomProperties attributes in the JDBC URL field:
   - RETURN_NA_AS_NULL

     When RETURN_NA_AS_NULL is set to yes, the NA value in RPAS domain is returned as null. This is useful if you do not want to send the NA value to the target database. It can also help filter the data records that contain only NA values. The filter can be written as "measure1 is not null OR measure2 is not null OR measure3 is not null OR …."
   - CONVERT_DATE_TO_STRING

     When CONVERT_DATE_TO_STRING is set to yes, the date type measures is returned as a string in the yyyymmdd format. This is required if the target data source is RPAS (which implies loadMeasure utility is used to load data).

   Figure 2–9 shows the settings for RPAS JDBC driver.

Figure 2–9 Topology Manager Screen Showing JDBC Settings for RPAS Data Server



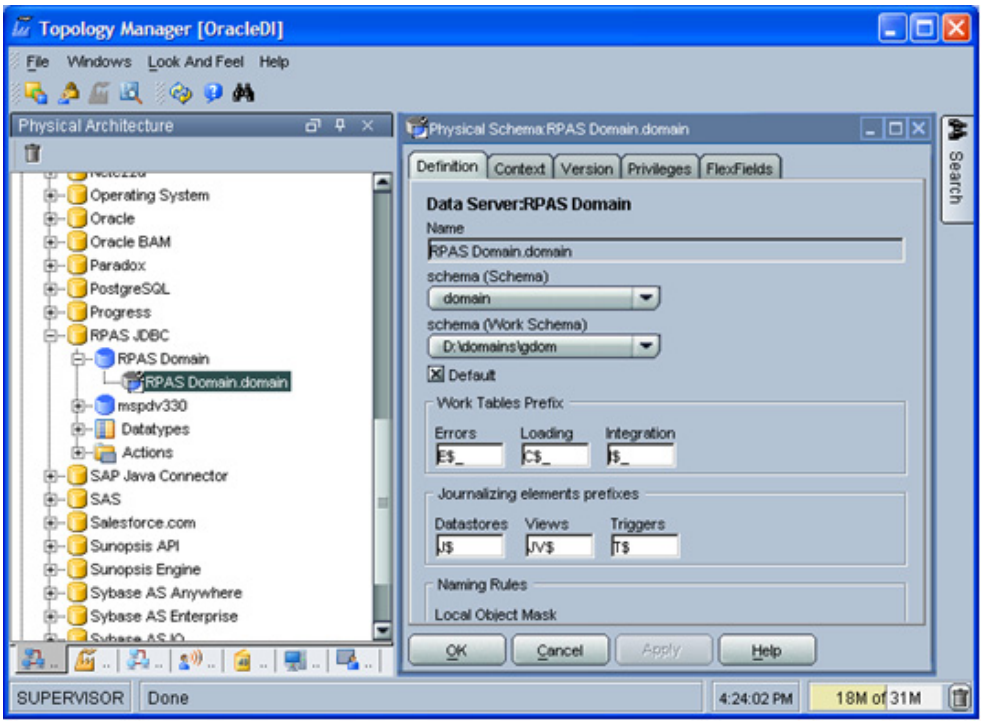
For more information about the usage of the RPAS JDBC driver and for a complete list of custom properties, see the *Oracle Retail Predictive Application Server Installation Guide*.

Configuring Contexts and Topology

The schema (Schema) property of this data server should be set to domain when the associated database is an RPAS domain (as opposed to workbooks). The **schema (Schema)** property should refer to the workbook name (that is, domain_t0) when the datastore is a workbook (a very rare case).

Note: RPAS datastores can only be global domains. ODI datastores cannot refer to RPAS local domains.

Figure 2–10 Topology Manager Screen Showing RPAS Domain Data Server Definition



It is recommended that multiple contexts be created in Topology Manager: one for development environment, one for testing, and one for production.

It is common practice that multiple physical schemas are created for each technology type: one physical schema for development environment, one for testing, and one for production. One logical schema should be created for all the physical schemas above. This single logical schema is mapped to one (and only one) physical schema in each context.

It is recommended that you create a topology map showing the relationship between the physical architecture, the logical architecture, and the ODI Contexts. Table 2–1 is an example that represents an environment with MFP Retail and IP installed:

Table 2–1 Topology Map

	ODI Context		
	DEV	**QA**	**PROD**
MFP Source	RPAS_JDBC_MFP_DEV	RPAS_JDBC_MFP_QA	RPAS_JDBC_MFP_PROD
MFP Target	FILE_MFP_DEV	FILE_MFP_QA	FILE_MFP_PROD
IP Source	RPAS_JDBC_IP_DEV	RPAS_JDBC_IP_QA	RPAS_JDBC_IP_PROD
IP Target	FILE_IP_DEV	FILE_IP_QA	FILE_IP_PROD

This strategy allows you to plan at a more abstract level, such as sending data from MFP to IP, rather than sending data from MFP on host ABC at directory /devel/x/y.

At runtime, the user selects in which context or environment to execute. ODI's Topology Manager stores the locations of all domains (directory and IP address/hostname) for each application/context and selects the appropriate host machine and directory path at runtime based on the context selected.

For example, to set up the DEV context shown in Table 2–1, open the Contexts pane in Topology Manager and perform the following:

1. Right-click the Context pane and select **Insert Context**.
2. Name the context DEV and give it the same code.
3. Select a password for the context. You are asked for this password whenever you change contexts in Designer.
4. On the Agents tab, associate each logical agent with one physical agent.

   Note: One physical agent is required per host, at minimum. You can more than one logical agent associated with the same physical agent.

5. On the Schemas tab, associate each logical schema to use with a physical schema. (The physical schemas were defined in the Physical Architecture tab.)
6. Click **OK** to save your changes.

Development Considerations

There can be two different ODI development scenarios, depending on the developer's accessibility to the production environment.

- The development environment is separate from the production environment (that is, the development team does not have access to production databases). In this scenario, the development work is done in a development Context, tested in a QA Context, then exported and packaged into XML files and shipped to the production environment. The packages are installed and imported into ODI Designer or Operator at the production site, ready for execution.

  In this scenario, it is required that the development, QA, and production domains must have conforming RPAS measures and hierarchies (same structure and dimensions). Other non-RPAS databases must also have conforming schema structures across development, QA, and production environments. For best portability, Oracle Retail recommends that the development environment and the production environment use the same names for ODI logical schemas.

- The development team has access to production databases, meaning a production Context can be set up in the ODI instance in which the developers work.

  In this scenario, it is recommended that an ODI Context is created for development, QA, and production. Each context should have its own physical schema. The ODI administrator can assign different levels of privilege to developers and users to ensure proper security.

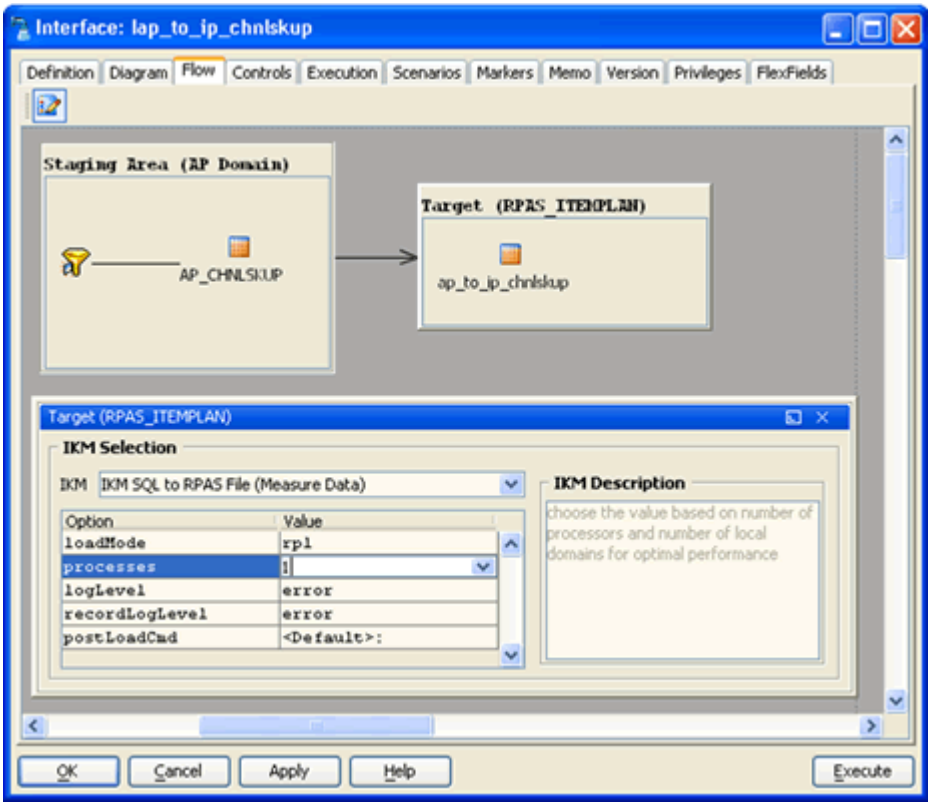
- To increase performance of the integration phase, it is possible to set the number of processes that loadmeasure uses from one to two or more, allowing local domains of the target and destination to be loaded in parallel. It is recommended that the number of processes equal the maximum number of local domains targeted for that particular ODI Interface.

  This may be done as follows:

  1. In Designer's Projects pane, double-click an interface in the Interfaces folder to open it.
  2. Open the **Flow** tab.
  3. Click once on the title bar of the Target window.

4. If the Properties Panel is not visible, click the **Display Hide Properties Panel** toggling icon. A window opens with options and values for the target, as shown in Figure 2–11.

Figure 2–11 Increasing Performance of the Integration Phase



5. Change the value of the **processes** option to the desired value.

3

ODI Setup

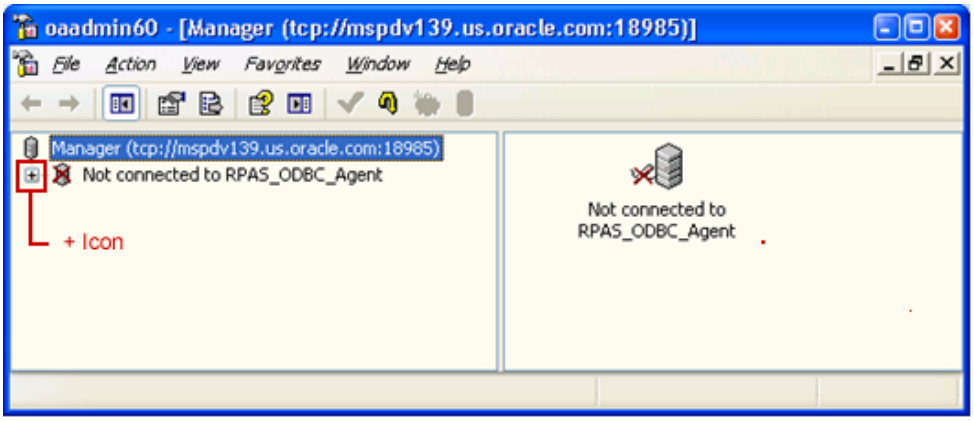
This chapter describes steps required to setup Retail Enabled ODI Integration.

Setting Up Data Servers

To set configure the ODBC data servers, perform the following steps:

1. From Windows, open the Management Console (oaadmin60) from the Oracle RPAS ODBC Server folder on the start menu. The following window appears:

Figure 3–1 oaadmin60 - Management Console

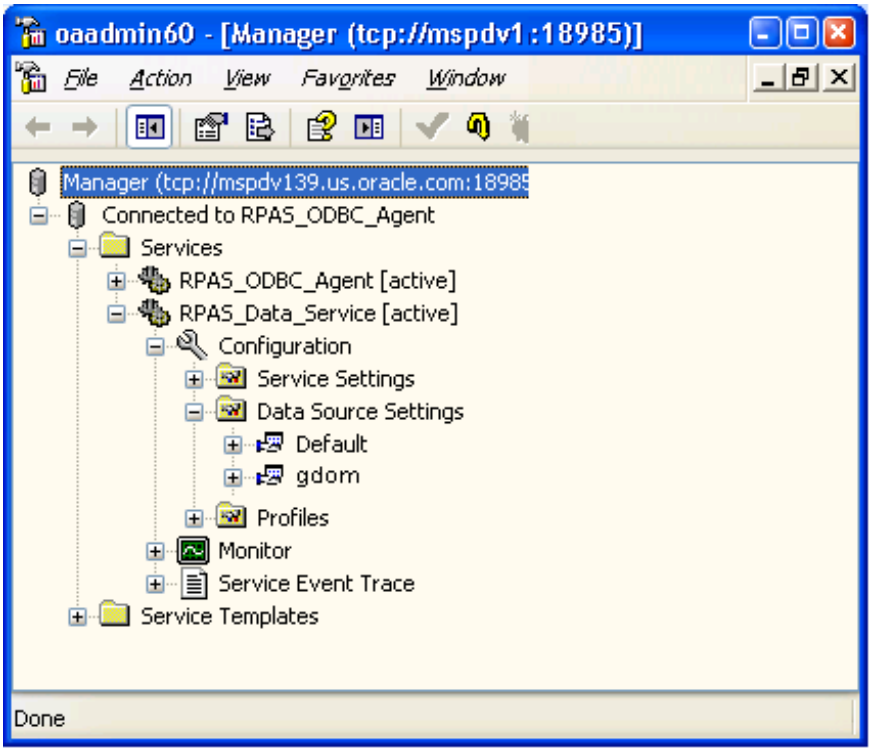


2. Click the **+** icon next to the **Not connected to RPAS_ODBC_Agent** icon, as shown in Figure 3–1. It automatically changes to Connected to RPAS_ODBC_Agent.

   If it does not connect, then check whether the process <rpas_home>/odbcserver/bin/oaagent -n RPAS_ODBC_Agent is running. If it is not running, rerun startRPASODBCAgent.

3. Use the + icons to open the **Services** folder, then **RPAS_Data_Service**, then **Configuration**, and finally the **Data Source Settings** folder, as shown in Figure 3–2.

Figure 3–2 oaadmin60 - Data Source Settings folder

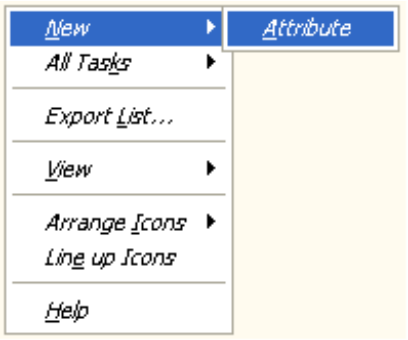


Within the Data Source Settings folder is the gdom data service. It was created by customInstall.sh with all properties and attributes set up except for the DOMAIN_PATH property.

4. To make the gdom settings the default for all newly added data services, copy all of gdom's settings to the Default data service.

5. If the Default data service is missing IP parameter attributes that are present in gdom, add them by right-clicking the background of the right pane and selecting **New**, and then **Attribute**, as shown in Figure 3–3.

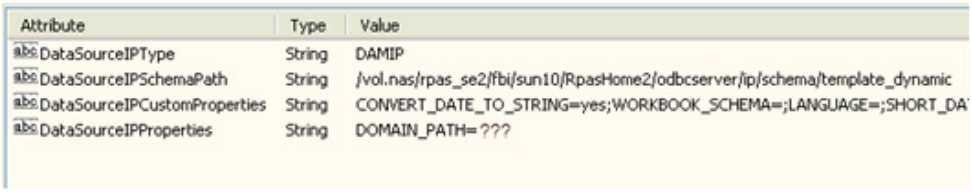
Figure 3–3 Selecting New / Attribute



A list of available attributes is presented.

When the Default data service is set up correctly, it looks like Figure 3–4:

Figure 3–4 Default Data Service



In Figure 3–4, DOMAIN_PATH=??? is a placeholder. The actual path should be entered for each data source. See the Creating New Data Sources section for more information.

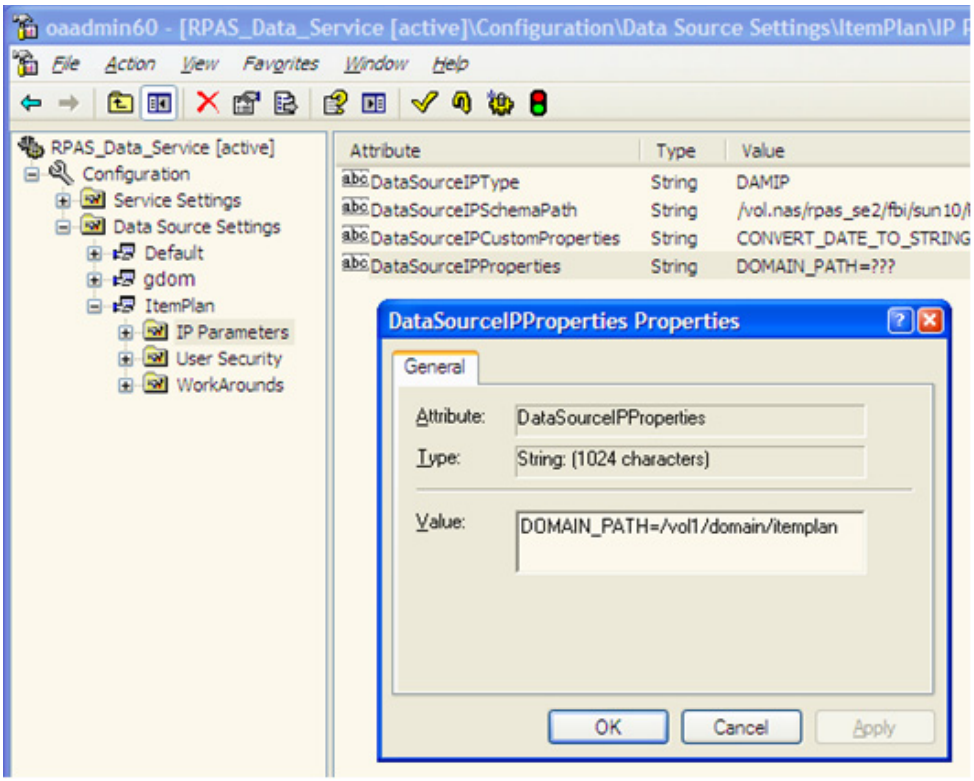
Creating New Data Sources

For each RPAS domain, you must create a new data source. For example, to create a new data source for Item Planning, perform the following steps:

1. Right-click **Data Source Settings**.
2. Click **New** and then click **Data Source**.
3. You are prompted to enter a name for the data source. This name is used within ODI's Topology Manager.

   Type **ItemPlan** and press **Enter**.
4. Click the **+** icon next to ItemPlan to expand its folder.
5. Select **IP Parameters**. Its parameters appear in the right pane.
6. To set the domain path for the domain, double-click the **DataSourceIPProperties**, as shown in Figure 3–5.

Figure 3–5 DataSourceIPProperties



7. Click **OK**.

Repeat these steps for each RPAS domain.

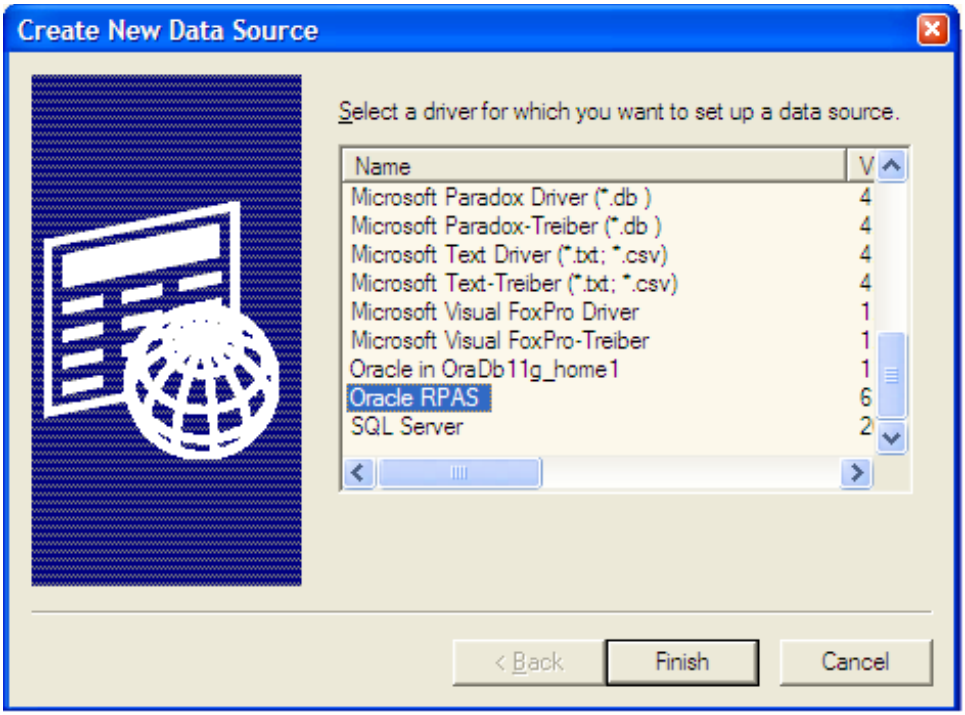
Once you have created a new data source for each RPAS domain, you need to associate that data source with a hostname. See Associating Data Sources and Host Names for more information.

Associating Data Sources and Host Names

To associate each data source name with a particular hostname, perform the following steps:

1. Start the ODBC Administrator application from the Oracle RPAS ODBC Driver folder on the Windows Start menu.
2. Open the DNS tab and click **Add**. The Create New Data Source dialog window opens, as shown in Figure 3–6:

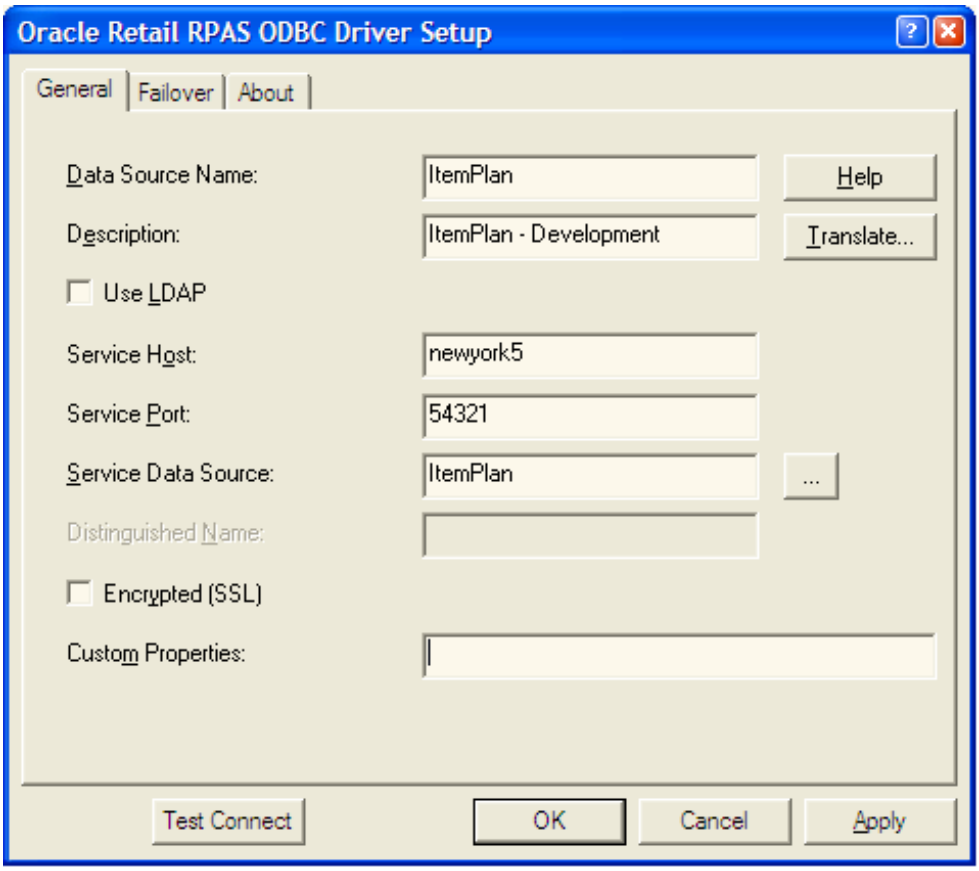
Figure 3–6 Create New Data Source



3. From the list, select Oracle RPAS and click **Finish**.

4. The Oracle Retail RPAS ODBC Driver Setup window opens.

Figure 3–7 Oracle Retail RPAS ODBC Driver Setup



Enter the following information:

- **Data Source Name**: The data source name must agree with the data source name you entered in the ODBC Management Console, as described in the Creating New Data Sources section.
- **Description**: A helpful description of the association.
- **Service Host**: The host name of the server where the domain is located, or a server that can access the domain over an NFS mount.
- **Service Port**: Enter the port number of the ODBC Data Service. This is the same port number selected when customInstall.sh was run. If you used the default port, it is 54321.
- **Service Data Source**: The service source name, like the data source name, must agree with the data source name you entered in the ODBC Management Console, as described in the Creating New Data Sources section.

5. Click **Test Connect** to test whether you can connect to the domain.
6. Click **OK**.

Repeat these steps for each RPAS domain.

4

Running ODI

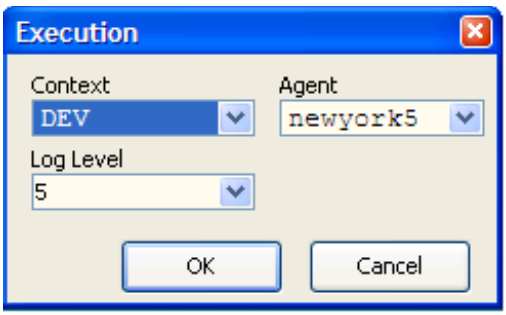
This chapter describes how to setup the domain paths and the IP addresses of the domains, which is done on each PC.

Executing Data Transfers with Packages

You can initiate a data transfer between RPAS applications from the ODI Designer module. To do this, perform the following steps:

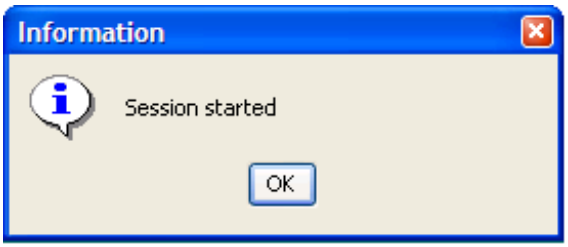
1. Open Designer's Projects pane.
2. Open the Main Folder/Packages folder.
3. Right-click the appropriate package. For example, right-click **AP to IP** and select **Execute**.
4. The Execution window appears. Select the desired context and an agent that is running on the same host as the target domain and click **OK**.

Figure 4–1 Execution Window



When the session is ready, the Information window appears, informing you that the session has started.

Figure 4–2 Information Window



You can monitor the progress of the data transfer processes in ODI's graphical module Operator. To open the Operator module, click the **Operator** icon:

Next, you should set up the data mappings between the RPAS applications. For Fashion Planning Bundle data mapping, see Chapter 5, "Fashion Planning Bundle Integration". For RDF, RPO, and APC-RPO data mapping, see Chapter 6, "RPO, APC-RPO, RDF Integration".

5

Fashion Planning Bundle Integration

This chapter describes the overall flow of data among the Fashion Planning Bundle applications, the data mappings between the applications, and the integration of hierarchies.

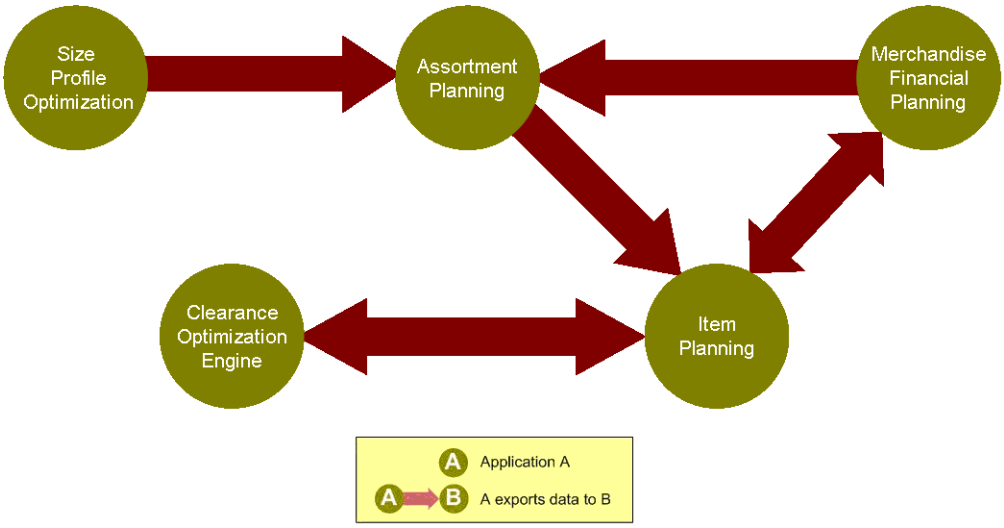
Overview of the Fashion Planning Bundle

This section describes the integration between solutions within the Fashion Planning Bundle. It does not describe external integrations.

The Fashion Planning Bundle is a full-suite planning solution for fashion retailers that integrates the following applications: Item Planning (IP), Item Planning Configured for Clearance Optimization Engine (IP COE), Assortment Planning (AP), Merchandise Financial Planning (MFP), and Size Profile Optimization (SPO).

Figure 5–1 shows the conceptual overview of the integration of these products.

Figure 5–1 Conceptual Overview



This solution supports data sharing among these applications. Note that the data sharing functionality is not dependent on the presence of all these applications. The defined data sharing between any of the applications works for the entire suite as well as for a subset of the applications.

Integration Interface Data Flow Description

These descriptions explain the each of the data flows in Figure 5–1.

From Size Profile Optimization to Assortment Planning

The following data is imported from SPO to AP:

- Store-level size profiles (current and archived versions) with corresponding escalation levels
- Store-clustered size profiles
- Prepack definition configuration at style-color/size/prepack-ID and prepack validity periods at prepack-ID week

The size profiles are by AP to obtain more detailed buy plans at style-color/size or prepack per store per receipt week. The prepack optimization module of SPO addresses the optimization of not only the number of units within a pack, but also the size ratios for each style-color within the pack. This enables the product to be tailored to the consumer selling patterns at each specific location.

SPO can group stores based on historical data into clusters, and these clusters can be used as a basis for creating prepacks.

For more information on SPO, see the Oracle Retail Size Profile Optimization documentation.

From Assortment Planning to Item Planning and an Allocation Application

The following data is exported from AP to IP and an allocation application:

- Buy plan

AP sends the buy plan to IP and the allocation application. The allocation application can use the sales plan or receipt plan created by AP to determine what inventory is allocated to stores. IP uses the pre-buying period assortment plan to track the performance of items during the buying period and to provide a framework to respond accordingly.

From Item Planning to Merchandise Financial Planning

The following data is exported from IP to MFP:

- Item plan

  Approved data is exported from IP and loaded into MFP at the subclass level. The aggregation takes place within IP when exporting. Only the plans that have been approved since the last export in IP are imported into MFP.

  MFP users can review and reconcile their merchandise financial plans (which are planned at the subclass level) to the approved item plans from IP.

From Merchandise Financial Planning to Item Planning

The following data is exported from MFP to IP:

- MFP current and original plans

  The current and original plans are used by IP to help with the creation of item plans.

From Merchandise Financial Planning to Assortment Planning

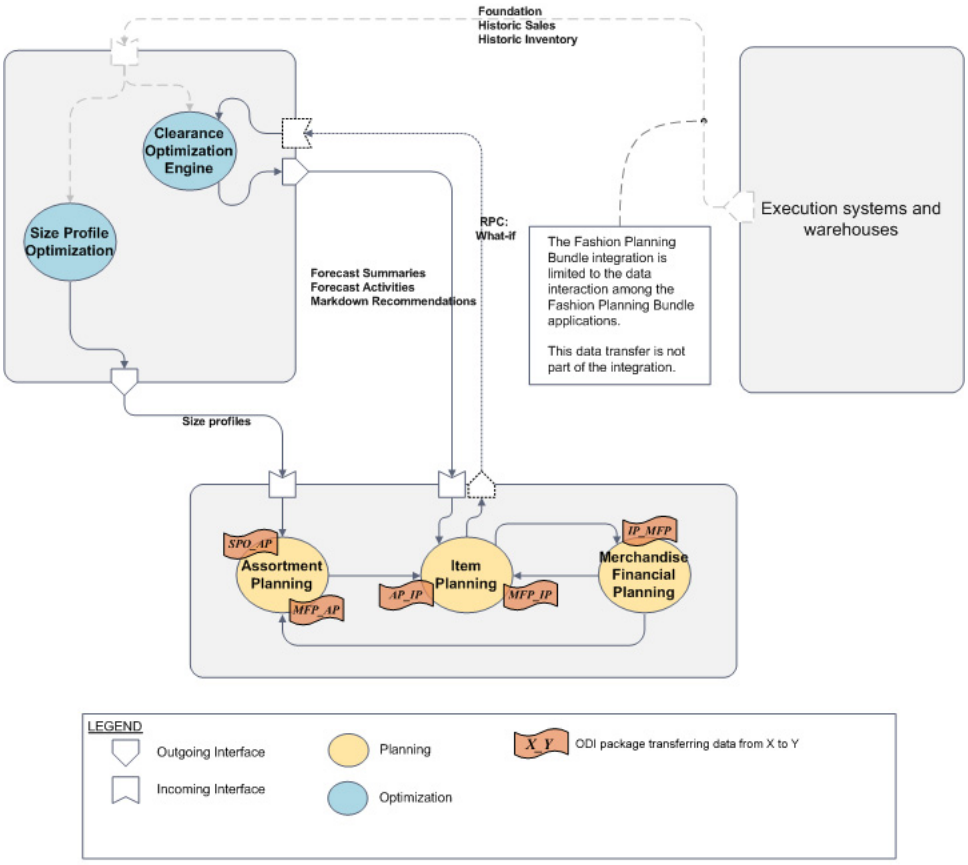
The following data is exported from MFP to AP:

- MFP current plans

  MFP current plans are critical inputs into the AP process. They define the financial goals for that buying period. AP uses these plans as targets to determine the number of options it should carry, the number of weeks it should carry them, and the amount of markdown it can afford in order to meet the targets for sales and gross margin.

Figure 5–2 shows the applications and data flow that are part of the Fashion Planning Bundle.

Figure 5–2 Overview of Fashion Planning Bundle Integration



The applications shown in the diagram are in the following categories:

- Execution Applications
- Optimization Applications
- Planning Applications

Execution Applications

> **Note:** Full data transfers from the execution applications are not part of the Fashion Planning Bundle integration. They are included in the diagram to illustrate that historical data needs to be obtained from applications outside the applications included in the Fashion Planning Bundle.

The execution applications provide the foundation and historical data that is input to the optimization applications.

- A data warehouse application provides the initial load of sales and inventory data.
- A merchandising application provides the daily and weekly data updates, or deltas, of the sales and inventory data. It provides data updates of master information such as styles, prices, stores, and so on.

Optimization Applications

The optimization applications take the foundation and historical data and produce size and markdown recommendations that are used by the planning applications.

- SPO creates profiles of the optimal size distribution by both merchandise category and store. This gives insight into consumer demand patterns by size. These size profiles feed into AP.

  For more information on SPO, see the Oracle Retail Size Profile Optimization documentation. For more information on AP, see the Oracle Retail Assortment Planning documentation.

- COE provides markdown recommendations and forecasts that enable retailers to make informed markdown decisions. What-if data feeds back into COE from IP. COE is only available with IP configured for COE.

  For more information on COE, see the Oracle Retail Clearance Optimization Engine documentation. For more information on IP, see the Oracle Retail Item Planning documentation.

Planning Applications

The planning applications take data from the optimization applications and create plans for the retailer.

- AP creates buy plans for the retailer based on size profiles from SPO and the subclass plans from MFP. The buy plans feeds into IP.
- IP takes the buy plan from AP and the subclass plan from MFP to create an item plan. When configured for COE, markdown data from COE is also used. IP output feeds back into MFP. What-if data feeds back into COE to create new markdown plans.
- MFP takes data from IP which is consulted when creating the subclass plan in MFP. Based on financial goals, the subclass plans determine how much merchandise should be available in a store in order to meet the goals. Subclass plans are fed into AP and IP.

Measure Data Integration

The following data integration points for each application-to-application package are described in this section:

- SizeOpt to AP Package
- SizeOpt to AP with PrepackOpt Package
- IP to MFP Retail Package
- IP to MFP Cost Package
- AP to IP Package
- MFP Cost to AP Package
- MFP Retail to AP Package
- MFP Cost to IP Package
- MFP Retail to IP Package
- MFP Finalize Exports Package

The scripts listed in each section are listed in the following directory:

$RPAS_HOME/scripts/integration/ODI

SizeOpt to AP Package

The following information is about the SizeOpt (also known as SPO) to AP integration.

> **Note:** For customers who use Assortment Planning with the Prepack component, see the SizeOpt to AP with PrepackOpt Package section.

Scripts Used By the SizeOpt to AP Package

- sizeopt_to_ap_init.ksh
- sizeopt_to_ap_finalize.ksh

Data Mapping for SizeOpt to AP Package

Data is sent when the expression is populated with a value other than the default.

Table 5–1 SizeOpt to AP Data

SizeOpt Expression	Expression Type	Expression Default Value	AP Target Measure
eptpackdef	Integer	0	eptpackdef
esprofxxlaxg	Real	-1	sztyarchsp1up
esprofxxlbxg	Real	-1	sztyarchsp2up
esprofxxlcxg	Real	-1	sztyarchsp3up
esprofxxldxg	Real	-1	sztyarachsp4up
sku2sizexxlxg	Boolean	False	sztyclssszmapb
exportprofxxlxg	Real	-1	sztysizeprflup
exportescxxlxg	String	""	szwpescleveltx
sku2atcdxxlxg	Boolean	False	szwpssncodeb

SizeOpt to AP with PrepackOpt Package

The following information is about integrating SPO to AP if AP has the Prepack component.

> **Note:** This package is for customers who use Assortment Planning with the Prepack component.
>
> For customers who use Assortment Planning without the Prepack component, see the SizeOpt to AP Package section.

This package is the same as the SizeOpt to AP Package package, except that the eptpackdef data is not sent from SizeOpt to AP. This allows SizeOpt and AP to have independent versions of the prepack definitions data.

Scripts Used By the SizeOpt to AP with Prepack Package

- sizeopt_to_ap_init.ksh
- sizeopt_to_ap_finalize.ksh

Data Mapping for SizeOpt to AP with Prepack Package

Data is sent when the expression is populated with a value other than the default value.

Table 5–2 SizeOpt to AP with Prepack Data

SizeOpt Expression	Expression Type	Expression Default Value	AP Target Measure
esprofxxlaxg	Real	-1	sztyarchsp1up
esprofxxlbxg	Real	-1	sztyarchsp2up
esprofxxlcxg	Real	-1	sztyarchsp3up
esprofxxldxg	Real	-1	sztyarachsp4up
sku2sizexxlxg	Boolean	False	sztyclssszmapb
sku2sizexxlxg	Boolean	False	skup2msrn
sku2sizexxlxg	Boolean	False	skup2msiz
exportprofxxlxg	Real	-1	sztysizeprflup
exportescxxlxg	String	""	szwpescleveltx
sku2atcdxxlxg	Boolean	False	szwpssncodeb

IP to MFP Retail Package

The following information is about the IP to MFP Retail integration.

Scripts Used By the IP to MFP Retail Package

- ip_to_mfp_init.ksh
- ip_to_mfp_finalize.ksh

Data Mapping for IP to MFP Retail Package

The CP Approved data is sent when ipcpappnewb is set to TRUE.

Table 5–3 IP to MFP Retail: CP Approved Data

IP Expression	Expression Type	Expression Default Value	MFP Retail Target Measure
ipcpbopc	Real	0	ipcpbopc
ipcpbopr	Real	0	ipcpbopr
ipcpbopu	Real	0	ipcpbopu
ipcpeopc	Real	0	ipcpeopc
ipcpeopr	Real	0	ipcpeopr
ipcpeopu	Real	0	ipcpeopu
ipcpgmpv	Real	0	ipcpgmpv
ipcprecc	Real	0	ipcprecc
ipcprecr	Real	0	ipcprecr
ipcprecu	Real	0	ipcprecu
ipcpslsr	Real	0	ipcpslsr
ipcpslsu	Real	0	ipcpslsu

The OP Approved data is sent when ipopappnewb is set to TRUE.

Table 5–4 IP to MFP Retail: OP Approved Data

IP Expression	Expression Type	Expression Default Value	MFP Retail Target Measure
ipopbopc	Real	0	ipopbopc
ipopbopu	Real	0	ipopbopu
ipopbopr	Real	0	ipopbopr
ipopeopc	Real	0	ipopeopc
ipopeopr	Real	0	ipopeopr
ipopeopu	Real	0	ipopeopu
ipopgmpv	Real	0	ipopgmpv
ipoprecc	Real	0	ipoprecc
ipoprecr	Real	0	ipoprecr
ipoprecu	Real	0	ipoprecu
ipopslsr	Real	0	ipopslsr
ipopslsu	Real	0	ipopslsu

IP to MFP Cost Package

The following information is about the IP to MFP Cost integration.

Scripts Used By the IP to MFP Cost Package

- ip_to_mfp_init.ksh
- ip_to_mfp_finalize.ksh

Data Mapping for IP to MFP Cost Package

The CP Approved data is sent when ipcpappnewb is set to TRUE.

Table 5–5 IP to MFP Cost: CP Approved Data

IP Expression	Expression Type	Expression Default Value	MFP Cost Target Measure
ipcpbopc	Real	0	ipcpbopc
ipcpbopu	Real	0	ipcpbopu
ipcpeopc	Real	0	ipcpeopc
ipcpeopu	Real	0	ipcpeopu
ipcpgmpv	Real	0	ipcpgmpv
ipcprecc	Real	0	ipcprecc
ipcprecu	Real	0	ipcprecu
ipcpslsc	Real	0	ipcpslsc
ipcpslsr	Real	0	ipcpslsr
ipcpslsu	Real	0	ipcpslsu

The OP Approved data is sent when ipopappnewb is set to TRUE.

Table 5–6 IP to MFP Cost: OP Approved Data

IP Expression	Expression Type	Expression Default Value	Target MFP Cost Measure
ipopbopc	Real	0	ipopbopc
ipopbopu	Real	0	ipopbopu
ipopeopc	Real	0	ipopeopc
ipopeopu	Real	0	ipopeopu
ipopgmpv	Real	0	ipopgmpv
ipoprecc	Real	0	ipoprecc
ipoprecu	Real	0	ipoprecu
ipopslsc	Real	0	ipopslsc
ipopslsr	Real	0	ipopslsr
ipopslsu	Real	0	ipopslsu

AP to IP Package

The following information is about the AP to IP integration.

Scripts Used By the AP to IP Package

- ap_to_ip_init.ksh
- ap_to_ip_finalize.ksh

Data Mapping for AP to IP Package

Data is sent when the expression is populated with a value other than the default value.

Table 5–7 AP to IP Data

AP Expression	Expression Type	Expression Default Value	IP Target Measure
bpcpasgn2clsb	Boolean	False	apcpasgn2strb
bpcsbopc	Real	0	apcpbopc
bpcsbopr	Real	0	apcpbopr
bpcsbopu	Real	0	apcpbopu
bpcsfpcstu	Real	0	apcpcogsc
bpwcdelfrequ	Integer	1	apcpdelfrequ
bpcseopc	Real	0	apcpeopc
bpcseopr	Real	0	apcpeopr
bpcseopu	Real	0	apcpeopu
bpcpexitweekdt	Date	[Jan 1, 1900]	apcpexitd
acwpfabrictx	String	""	apcpfabrictx
sptyitmat1maptx	String	""	apcpitmat1maptx
sptyitmat2maptx	String	""	apcpitmat2maptx
bpcsmkdclrr	Real	0	apcpmkdclrr
bpcsmkdpermr	Real	0	apcpmkdpermr
bpcsmkdpromor	Real	0	apcpmkdpror
bpcpprepacku	Integer	1	apcppckszu
sccpperfgrpll	String	""	apcpperfgrptx
bpcsclrcstu	Real	0	apcpprcclrc
bpcsclrprcu	Real	0	apcpprcclrr
bpcsfprtlu	Real	0	apcpprcinir
bpcspromoprcr	Real	0	apcpprcpror
bpcspermprcr	Real	0	apcpprcr
bpwcpresminu	Integer	0	apcppresminu
bpcsrecc	Real	0	apcprecc
bpcsrecr	Real	0	apcprecr
bpcsrecu	Real	0	apcprecu
sccpsizgrpll	String	""	apcpsftystcku
acwpsizrangetx	String	""	apcpsizgrptx
actyvendorll	String	""	apcpsizrangetx
bpcsslsclrc	Real	0	apcpslsclrc
bpcsslsclrr	Real	0	apcpslsclrr
bpcsslsclru	Real	0	apcpslsclru
bpcsfpslsc	Real	0	apcpslsregc
bpcsfpslsr	Real	0	apcpslsregr
bpcsfpslsu	Real	0	apcpslsregu

Table 5–7 (Cont.) AP to IP Data

AP Expression	Expression Type	Expression Default Value	IP Target Measure
bpcpstartweekdt	Date	[Jan 1, 1990]	apcpslsstartd
bpcsstrcntu	Real	0	apcpstru
actyvendorll	String	""	apcpvendortx
bpcpwksclru	Integer	0	apcpweeksclru
bpcpwksregu	Integer	0	apcpweeksregu

MFP Cost to AP Package

The following information is about the MFP Cost to AP integration.

Scripts Used By the MFP Cost to AP Package

- mfp_to_ap_init.ksh
- mfp_to_ap_finalize.ksh

Data Mapping for MFP Cost to AP Package

Data is sent when mowpappcpnewb is set to TRUE.

Table 5–8 MFP Cost to AP Data

MFP Cost Expression	Expression Type	Expression Default Value	AP Target Measure
bucpgmpv	Real	0	mfcpposgmr
bucprecc	Real	0	mfcprecc
bucprecu	Real	0	mfcprecu
bucpslsr	Real	0	mfcpslsr
bucpslsu	Real	0	mfcpslsu

MFP Retail to AP Package

The following information is about the MFP Retail to AP integration.

Scripts Used By the MFP Retail to AP Package

- mfp_to_ap_init.ksh
- mfp_to_ap_finalize.ksh

Data Mapping for MFP Retail to AP Package

Data is sent when mowpappcpnewb is set to TRUE.

Table 5–9 MFP Retail to AP Data

MFP Retail Expression	Expression Type	Expression Default Value	AP Target Measure
bucpagmpv	Real	0	mfcpposgmr
bucprecc	Real	0	mfcprecc
bucprecr	Real	0	mfcprecr

Table 5–9 (Cont.) MFP Retail to AP Data

MFP Retail Expression	Expression Type	Expression Default Value	AP Target Measure
bucprecu	Real	0	mfcprecu
bucpslsregr + bucpslsclrr + bucpslspror	Real	0	mfcpslsr
bucpslsregu + bucpslsclru + bucpslsprou	Real	0	mfcpslsu

MFP Cost to IP Package

The following information is about the MFP Cost to IP integration.

Scripts Used By the MFP Cost to IP Package

- mfp_to_ip_init.ksh
- mfp_to_ip_finalize.ksh

Data Mapping for MFP Cost to IP Package

The CP Approved data is sent when mowpappcpnewb is set to TRUE.

Table 5–10 MFP Cost to IP: CP Approved Data

MFP Cost Expression	Expression Type	Expression Default Value	IP Target Measure
bucpbopc	Real	0	bucpbopc
bucpbopu	Real	0	bucpbopu
bucpeopc	Real	0	bucpeopc
bucpeopu	Real	0	bucpeopu
bucpgmpv	Real	0	bucpgmpv
bucprecc	Real	0	bucprecc
bucprecu	Real	0	bucprecu
bucpslsc	Real	0	bucpslsc
bucpslsr	Real	0	bucpslsr
bucpslsu	Real	0	bucpslsu
bucpslsnetr	Real	0	bucpslsnetr

The OP Approved data is sent when mowpappopnewb is set to TRUE.

Table 5–11 MFP Cost to IP: OP Approved Data

MFP Cost Expression	Expression Type	Expression Default Value	IP Target Measure
buopbopc	Real	0	buopbopc
buopbopu	Real	0	buopbopu
buopeopc	Real	0	buopeopc

Table 5–11 (Cont.) MFP Cost to IP: OP Approved Data

MFP Cost Expression	Expression Type	Expression Default Value	IP Target Measure
buopeopu	Real	0	buopeopu
buopgmpv	Real	0	buopgmpv
buoprecc	Real	0	buoprecc
buoprecu	Real	0	buoprecu
buopslsc	Real	0	buopslsc
buopslsr	Real	0	buopslsr
buopslsu	Real	0	buopslsu
buopslsnetr	Real	0	buopslsnetr

MFP Retail to IP Package

The following information is about the MFP Retail to IP integration.

Scripts Used By the MFP Retail to IP Package

- mfp_to_ip_init.ksh
- mfp_to_ip_finalize.ksh

Data Mapping for MFP Retail to IP Package

The CP Approved data is sent when mowpappcpnewb is set to TRUE.

Table 5–12 MFP Retail to IP: CP Approved Data

MFP Retail Expression	Expression Type	Expression Default Value	IP Target Measure
bucpbopc	Real	0	bucpbopc
bucpbopr	Real	0	bucpbopr
bucpbopu	Real	0	bucpbopu
bucpeopc	Real	0	bucpeopc
bucpeopr	Real	0	bucpeopr
bucpeopu	Real	0	bucpeopu
bucpigmpv	Real	0	bucpgmpv
bucpmkdclrr	Real	0	bucpmkdclrr
bucpmkdpermr	Real	0	bucpmkdpermr
bucpmkdpromor	Real	0	bucpmkdpror
bucprecc	Real	0	bucprecc
bucprecr	Real	0	bucprecr
bucprecu	Real	0	bucprecu
bucpslsclrr	Real	0	bucpslsclrr
bucpslsclru	Real	0	bucpslsclru
bucpslspror	Real	0	bucpslspror

Table 5–12 (Cont.) MFP Retail to IP: CP Approved Data

MFP Retail Expression	Expression Type	Expression Default Value	IP Target Measure
bucpslsprou	Real	0	bucpslsprou
bucpslsregr	Real	0	bucpslsregr
bucpslsregu	Real	0	bucpslsregu
bucpslsnetr	Real	0	bucpslsnetr

The OP Approved data is sent when mowpappopnewb is set to TRUE.

Table 5–13 MFP Retail to IP: OP Approved Data

MFP Retail Expression	Expression Type	Expression Default Value	IP Target Measure
buopbopc	Real	0	buopbopc
buopbopr	Real	0	buopbopr
buopbopu	Real	0	buopbopu
buopeopc	Real	0	buopeopc
buocpeopr	Real	0	buocpeopr
buopeopu	Real	0	buopeopu
buopicogsc	Real	0	buopcogsc
buopigmpv	Real	0	buopgmpv
buopmkdclrr	Real	0	buopmkdclrr
buopmkdpermr	Real	0	buopmkdpermr
buopmkdpromor	Real	0	buopmkdpromor
buoprecc	Real	0	buoprecc
buoprecr	Real	0	buoprecr
buoprecu	Real	0	buoprecu
buopslsclrr	Real	0	buopslsclrr
buopslsclru	Real	0	buopslsclru
buopslspror	Real	0	buopslspror
buopslsprou	Real	0	buopslsprou
buopslsregr	Real	0	buopslsregr
buopslsregu	Real	0	buopslsregu
buopslsnetr	Real	0	buopslsnetr

MFP Finalize Exports Package

There are four packages that export data from MFP Cost and MFP Retail to Assortment Planning (AP) and Item Planning (IP). These packages have two flag measures called Newly Approved. These measures indicate which data has been approved since the last export. The purpose of the MFP Finalize Exports package is to give you a way to reset these flags after MFP exports are completed.

These two flags are boolean measures at the channel/class/week level. A value of TRUE indicates that the corresponding channel/class/week level data has been approved and is ready for export. One measure controls the Current Plan data and the other controls the Original Plan data.

Table 5–14 MFP Finalize Exports Measures

Measure Name	Label
mowpappcpnewb	Wp Newly Approved CP (Current Plan flag)
mowpappopnewb	Wp Newly Approved OP (Original Plan flag)

For instance, if you use MFP Retail, AP, and IP, and you approve an MFP plan for fiscal year 2011 (FY2011), this sets the Newly Approved flags to TRUE in the January time frame (for the channels/classes).

If you wanted to export this plan to both AP and IP, perform the following steps:

1. Run the MFP Retail to AP Package and MFP Retail to IP Package. It does not matter which package you run first.
2. Run the MFP Finalize Exports package to reset Newly Approved flags to FALSE.

If you do not run the MFP Finalize Exports package and later you approve an MFP plan for FY2012, the FY2011 data is still flagged as Newly Approved. During the next data export, this causes the FY2011 plan to be sent unnecessarily with the FY2012 plan. This slows the export process and may cause discrepancies since MFP exports any saved changes to the FY2011 data since the previous export.

Script Used By the MFP Finalize Exports Package

- mfp_finalize_exports.ksh

Hierarchy Integration

If all of the Fashion Planning Bundle applications are on one server or their domains are accessible from one server (for example, through NFS mounts), you only need to run each hierarchy integration package once. However, if you are integrating applications across different servers, the listed integration steps must be repeated with different contexts and agents after each server is integrated.

Adding New Products: PROD Hierarchy Integration

Sometimes it is required to add new products/SKUs to the PROD hierarchies. But each application's PROD hierarchy typically contains different dimensions. Therefore, the Fashion Planning Bundle provides an integration package named Broadcast PROD Hierarchy that adds new products while maintaining the hierarchy synchronization between all applications.

To add new products, create a prod.csv file (in CSV format) that defines the following information for each new product:

Table 5–15 Prod.csv File Dimension Positions/Labels

Dimension Position / Label
SKU
SKU_LABEL
SKUP
SKUP_LABEL
SKUG
SKUG_LABEL
CLR
CLR_LABEL
SKP1
SKP1_LABEL
SKP2
SKP2_LABEL
VDRC
VDRC_LABEL
VNDR
VNDR_LABEL
CLGP
CLGP_LABEL
SCLS
SCLS_LABEL
CLASS
CLASS_LABEL
ITGP
ITGP_LABEL

Table 5–15 (Cont.) Prod.csv File Dimension Positions/Labels

Dimension Position / Label
PL1
PL1_LABEL
PL2
PL2_LABEL
DEPT
DEPT LABEL
PGRP
PGRP_LABEL
DVSN
DVSN_LABEL
CMPP
CMPP_LABEL

The dimensions in Table 5–15 are the union or superset of the PROD hierarchies for all Fashion Planning Bundle applications. An example file named prod.csv is provided under the $RPAS_HOME/doc directory. You can edit a copy of the file using a spreadsheet program or text editor.

Once the new products and their attributes have been added to prod.csv, perform the following steps to add the products to all applications:

1. Place the updated prod.csv file in the $RPAS_HOME directory.
2. In the Projects pane in Designer, open the **Interfaces** folder, then the **Packages** folder.
3. Double-click **Broadcast PROD Hierarchy**.
4. Select the context/agents appropriate to the application domain locations.

   Note that the value of the FPB_APPS environment variable for the agent.sh process determines which domains to update. See Installing ODI Agents with RPAS for an example of the FPB_APPS setup.

For more information about each application's PROD hierarchy, see that application's user guide.

Adding New Stores: LOC Hierarchy Integration

The procedure for adding new stores is similar to that of adding new products. The package Broadcast LOC Hierarchy adds new stores, while maintaining the hierarchy synchronization between applications.

To add new stores, create a loc.csv file (in CSV format) that defines the following information for each new store:

Table 5–16 Loc.csv File Dimension Positions/Labels

Dimension Position/Label
STR
STR_LABEL
DSTR
DSTR_LABEL
RGN
RGN_LABEL
AREA
AREA_LABEL
CHNL
CHNL_LABEL
CHN
CHN_BAEL
COMP
COMP_LABEL

The dimensions in Table 5–16 are the union or superset of the LOC hierarchies for all Fashion Planning Bundle applications. An example file named loc.csv is provided under the $RPAS_HOME/doc directory. You can edit a copy of the file using a spreadsheet program or text editor.

Once the new stores and their attributes have been added to loc.csv, perform the following steps to add the stores to all applications:

1. Place the updated loc.csv file in the $RPAS_HOME directory.
2. In the Projects pane in Designer, open the **Interfaces** folder, then the **Packages** folder.
3. Double-click **Broadcast LOC Hierarchy**.
4. Select the context/agents appropriate to the application domain locations.

   Note that the value of the FPB_APPS environment variable for the agent.sh process determines which domains to update. See Installing ODI Agents with RPAS for an example of the FPB_APPS setup.

For more information about each application's LOC hierarchy, see that application's user guide.

6

RPO, APC-RPO, RDF Integration

This chapter describes the integration among RPO, APC-RPO, and RDF using Oracle Data Integrator (ODI).

Measure Data Integration

The following data integration points for each application-to-application package are described in this section:

- APC-RPO to RPO Package
- APC-RPO to RDF Package
- RDF to RPO Package
- RPO to RDF Package

The scripts listed in each section are listed in the following directory:

$RPAS_HOME/scripts/integration/ODI

APC-RPO to RPO Package

The following information is about the APC-RPO to RPO integration package. This package sends anchor prices, self elasticities, and historical minimum and maximum prices from APC-RPO to RPO.

Data Mapping for APC-RPO to RPO Package

Data is sent when the expression is populated with a value other than the default value, which is zero.

Table 6–1 APC-RPO to RPO Data

APC-RPO Expression	RPO Target Measure
exprtelst	ol1gammasp
anchorprice	ol1anchprc
minhistprice	ol1hisloprc
maxhistprice	ol1hishiprc

APC-RPO to RDF Package

The following information is about the APC-RPO to RDF integration package. This package sends anchor prices, self elasticities, and historical minimum and maximum prices from APC-RPO to RDF.

Data Mapping for APC-RPO to RDF Package

Data is sent when the expression is populated with a value other than the default value, which is zero.

Table 6–2 APC-RPO to RDF Data

APC-RPO Expression	RDF Target Measure
achprstr	rdfanchprc
exptelsstr	rdfgamma
price	rdfprice

RDF to RPO Package

The following information is about the RDF to RPO integration package. This package sends item based demand from RDF to RPO.

Data Mapping for RDF to RPO Package

Data is sent when the expression is populated with a value other than the default value, which is zero.

Table 6–3 RDF to RPO Data

RDF Expression	RPO Target Measure
appf01xb	dl1itbdsp

RPO to RDF Package

The following information is about the RPO to RDF integration package. This package sends item prices from RPO to RDF.

Data Mapping for RPO to RDF Package

Data is sent when the expression is populated with a value other than the default value, which is zero.

Table 6–4 RPO to RDF Data

RPO Expression	RDF Target Measure
fappitpc	rdfprice

A

Building Custom Data Integration Interfaces

The purpose of this appendix is to familiarize an ODI developer for building custom ODI integration projects on RPAS.

A relational database must be available to serve as the ODI repository. The RPAS JDBC driver does not support table creation, so the RPAS domain cannot serve as ODI repository.

The RPAS JDBC Driver

The RPAS JDBC driver supports reverse engineering, and provides a standard SQL interface for an RPAS domain to serve as a source database. This means the RPAS domain can be viewed as a relational database, and all the existing ODI Load knowledge modules (IKM) that work for a generic relational database should work for an RPAS domain. This is one of the reasons that an RPAS-specific knowledge module is not needed to read from an RPAS domain.

The RPAS Specific Knowledge Modules

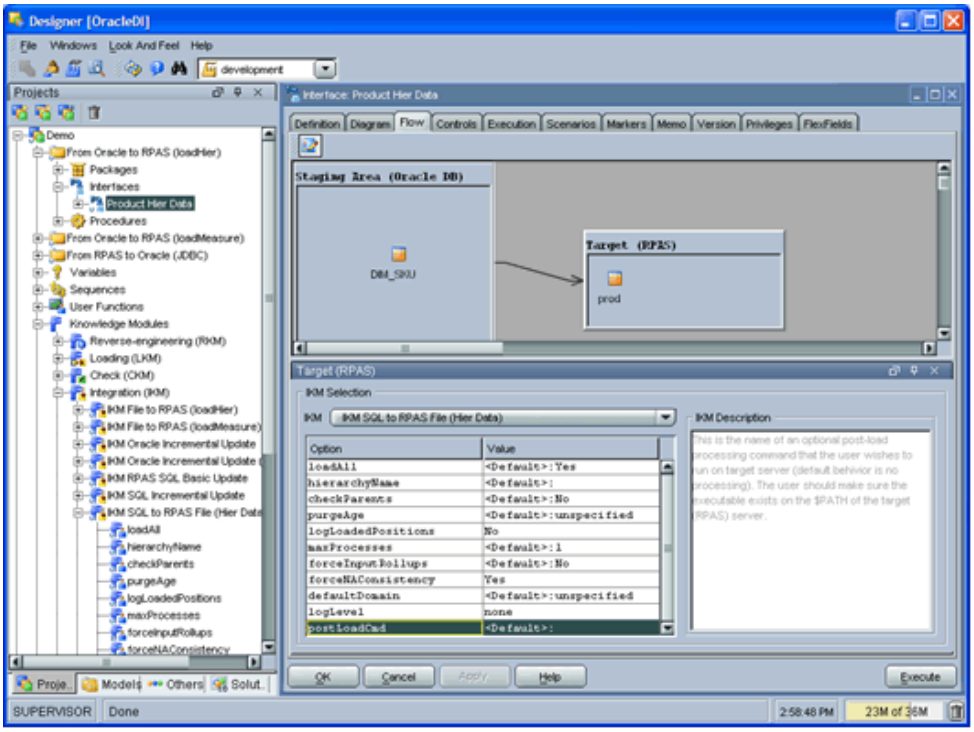
RPAS provides two IKMs for data load: one for loading measure data, and another one for loading hierarchy data into an RPAS domain. Oracle Retail does not recommend any other IKM for loading data into an RPAS domain.

Loading Hierarchy Data

Use the **IKM SQL to RPAS File (Hier Data)** knowledge module to load hierarchy data into an RPAS domain. This is the only IKM that can load hierarchy data into an RPAS domain. This IKM is designed for data integration interfaces with the source being RPAS or a generic relational database and the target being an RPAS domain. It uses SQL query to extract data from the source database, and ODI's File JDBC driver to write the data to a comma-separated value (CSV) file in the target RPAS domain. After that, it calls the RPAS loadHier utility using the CSV file as input to the target domain. Lastly, it performs an optional step for any post-load processing.

This knowledge module implements options to support all the parameters of the RPAS loadHier utility, and a postLoadCmd option to support the last optional step, as shown in Figure A–1. For description of loadHier parameters, refer to the "Loading Hierarchies - loadHier" section in the *Oracle Retail Predictive Application Server Administration Guide for the Classic Client* or the *Oracle Retail Predictive Application Server Administration Guide for the Fusion Client*.

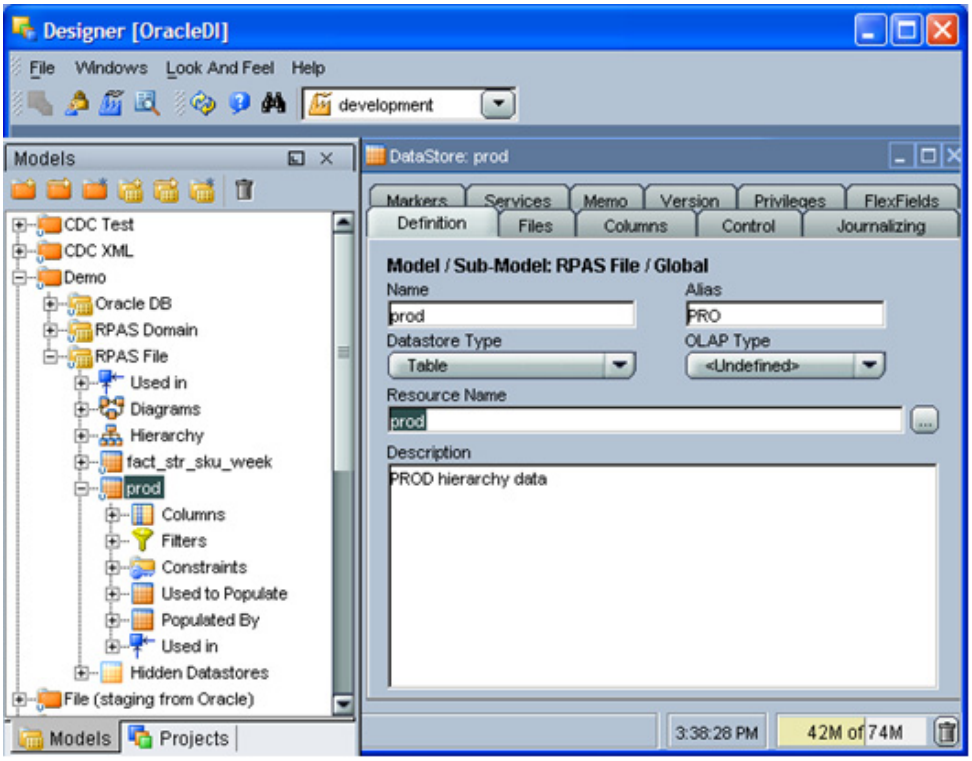
Figure A–1 Designer: loadHier Options



In order to use this IKM, the target technology should be File.

When creating the data model for the target datastore, the table must be named after the hierarchy name defined in RPAS. The ODI Resource Name of the table must be the same as the table name, as shown in Figure A–2.

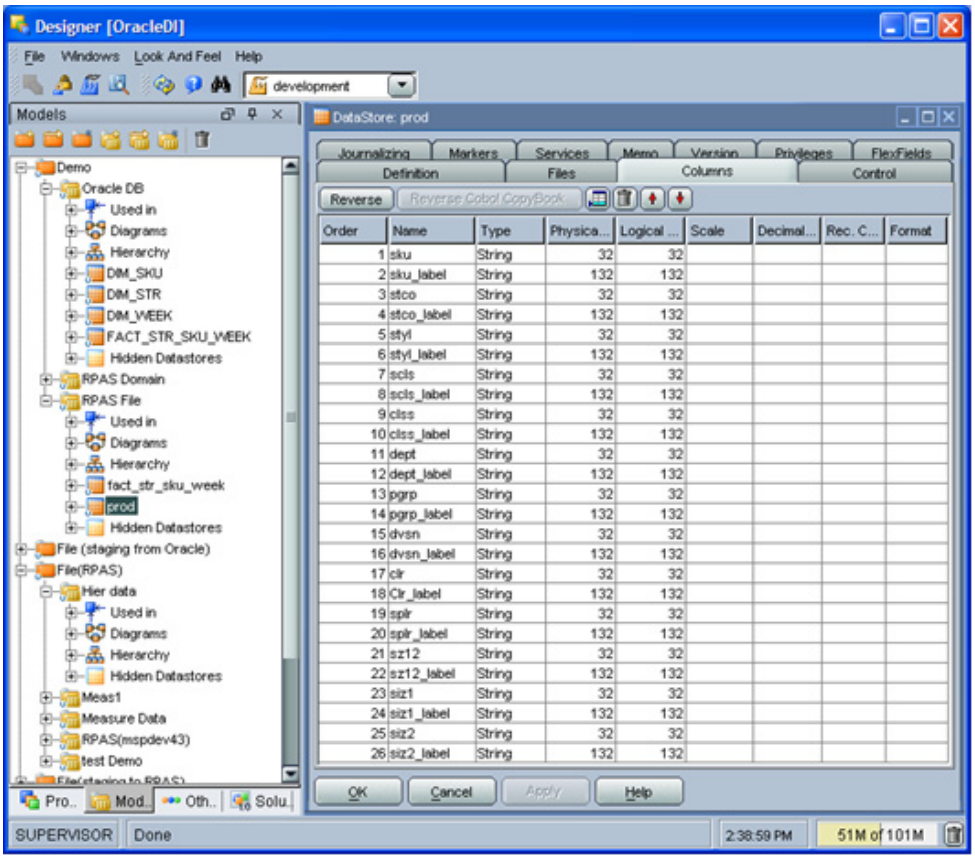
Figure A–2 ODI Resource Name



The table must have one header line, and must contain columns for all dimensions of the hierarchy being loaded with two columns for each dimension: one for position name and one for position label. The position name columns must be named after the dimension names of the hierarchy, the name of label columns should be the name of dimension appended with _label.

Figure A–3 shows a column definition of prod table (prod is the name of product hierarchy in a test RPAS domain).

Figure A–3 Designer: Column Definition of prod Table



Note: The data type for all columns must be set to String as shown in Figure A–3. This is because the table (which uses ODI FILE technology) contains a header line for column names which are all strings.

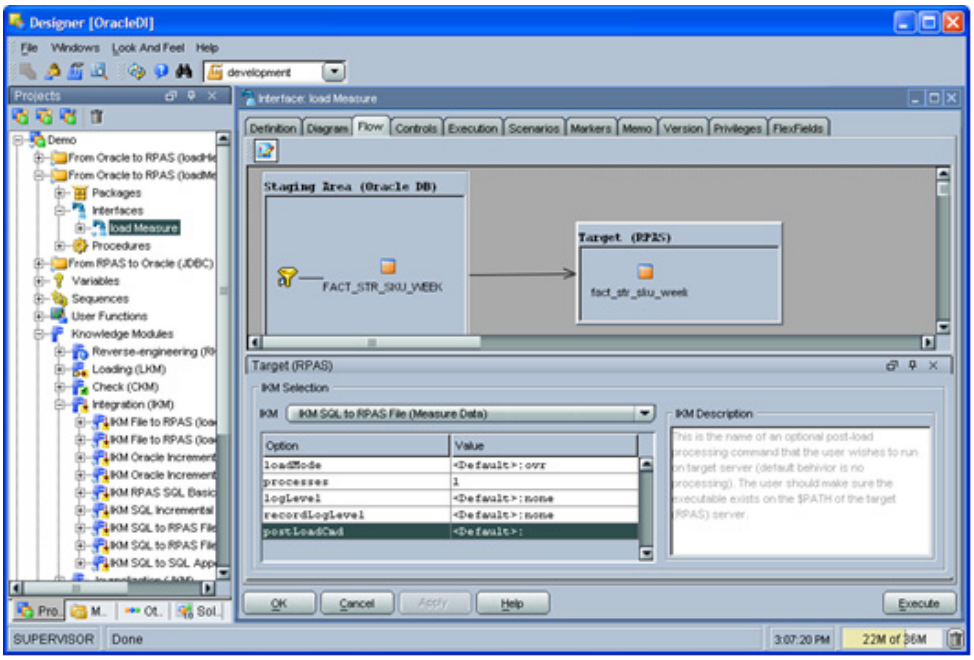
The developers who use this IKM are expected to know the hierarchy names defined in RPAS and the dimension names within each hierarchy.

Loading Measure Data

An integration knowledge module, IKM SQL To File (Measure data), is used for batch measure data transfer. The SQL To File knowledge module was designed for bulk data transfer, with source being an RPAS domain or a relational database and target being an RPAS domain. It uses SQL query to extract data on source, and then calls RPAS loadmeasure on the target domain. It also has an optional step for any post-load processing that the user wants to run. Examples of postload processing can include batch calculations or specific custom scripts that may be needed for processing newly loaded data in the target domain.

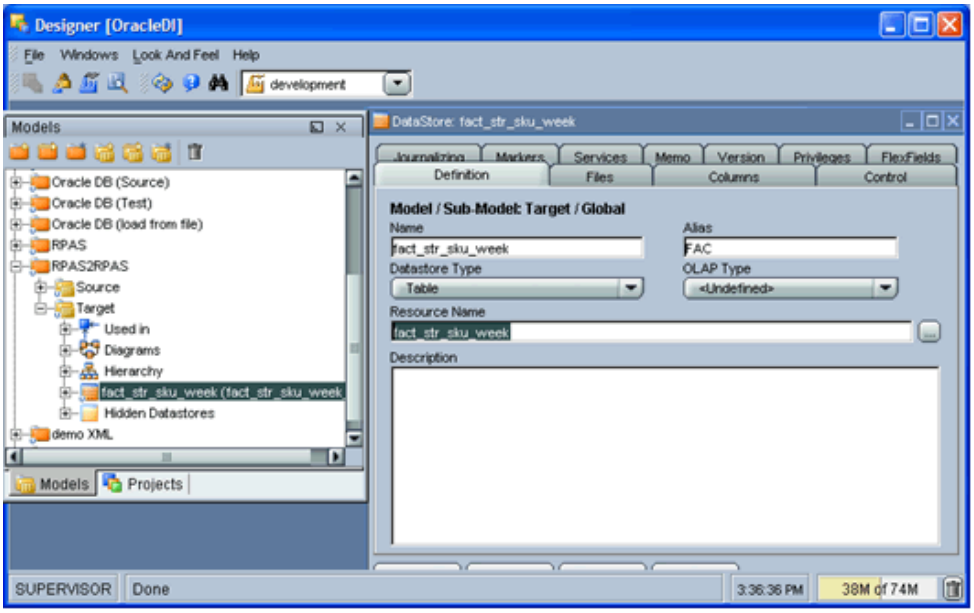
This knowledge module implements options for selecting load type, number of parallel processes (of loadmeasure utility), log level, and post-load processing. Values for the options should be entered prior to the execution of the data transfer job. A description of each option is displayed in the IKM Description window when the option is highlighted (see Figure A–4).

Figure A–4 Designer: loadMeasure Options



In order to use this IKM, the data model of the target database should be a File type table. The comma-separated value (CSV) file format must be used. There is no restriction on the name of the file. However, the ODI Resource Name of the table must be the same as the table name, as shown in Figure A–5.

Figure A–5 ODI Resource Name

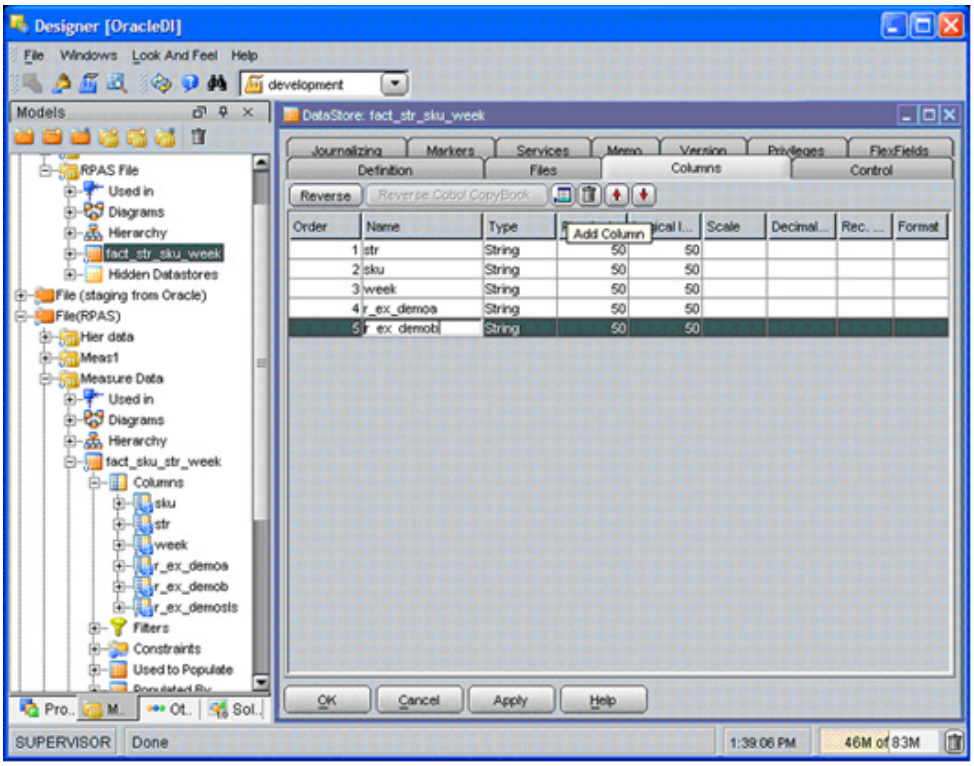


The file must have a header line (for column names). The file (or table) should contain dimension columns and measure columns. The name of the dimension columns should be consistent with the RPAS dimension names, and the name of the measure columns should be the same as RPAS measure names. The combination of the dimension columns should correspond to the load intersection or the base intersection of the measures contained in the table.

Figure A–6 shows column definitions of a target measure data table: the first three columns (str, sku, week) define the base intersection of the measures (r_ex_demoa and r_ex_demob in this example). The data type of the columns can be simply defined to be string, regardless of the real types in RPAS because the type only affects the ODI File JDBC driver when writing to the CSV file.

Note: The data type for all columns must be set to String as shown in Figure A–6, regardless of the real types in RPAS. This is because the table (which uses ODI's FILE technology) contains a header line for column names which are all strings.

Figure A–6 Designer: Column Definitions of Target Measure Data Table



Other Recommendations

To achieve optimal performance, use multiple interfaces when reading from a partitioned RPAS domain. The interfaces all connect to the master domain, but have different filters so that each interface reads data from different partitions (local domains). At execution time, all of the interfaces can run in parallel. The number of interfaces should be determined based on the number of partitions, number of processors of the server, and other loading on the server. If the number of measures to be transferred is large, vertical partitioning can also be considered: multiple interfaces can be designed where each interface transfers a subset of measures, and the interfaces can run in parallel at execution time.

Limitations and Potential Issues

- ODI repositories cannot be created in an RPAS domain. For data integration across multiple RPAS domains, an additional relational database is required for storing the repository.
- RPAS JDBC Driver does not support SQL INSERT. RPAS Data/hierarchy load utilities must be used when the target database is an RPAS domain.
- When loading measure data into an RPAS domain using the IKM SQL to File knowledge module, data can be loaded at the load intersection or base intersection only.
- If a data transfer (to RPAS) job fails, manual cleanup might be needed in the RPAS domain input directory.