Oracle® Retail Predictive Application Server

Administration Guide
Release 13.0

April 2008

ORACLE

Oracle® Retail Predictive Application Server Administration Guide, Release 13.0

Copyright © 2008, Oracle. All rights reserved.

Primary Author: Gary O'Hara

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software—Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

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.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Value-Added Reseller (VAR) Language

(i) the software component known as **ACUMATE** developed and licensed by Lucent Technologies Inc. of Murray Hill, New Jersey, to Oracle and imbedded in the Oracle Retail Predictive Application Server – Enterprise Engine, Oracle Retail Category Management, Oracle Retail Item Planning, Oracle Retail Merchandise Financial Planning, Oracle Retail Advanced Inventory Planning and Oracle Retail Demand Forecasting applications.

(ii) 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.

(iii) the **SeeBeyond** component developed and licensed by Sun MicroSystems, Inc. (Sun) of Santa Clara, California, to Oracle and imbedded in the Oracle Retail Integration Bus application.

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

(v) the software component known as **Crystal Enterprise Professional and/or Crystal Reports Professional** licensed by Business Objects Software Limited ("Business Objects") and imbedded in Oracle Retail Store Inventory Management.

(vi) 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.

(vii) 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.

(viii) the software component known as **Style Report™** developed and licensed by InetSoft Technology Corp. of Piscataway, New Jersey, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

(ix) the software component known as **WebLogic™** developed and licensed by BEA Systems, Inc. of San Jose, California, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

(x) the software component known as **DataBeacon™** developed and licensed by Cognos Incorporated of Ottawa, Ontario, Canada, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

Contents

Preface ..... **xi**
Audience xi
Related Documents xi
Customer Support xi
Review Patch Documentation xi
Oracle Retail Documentation on the Oracle Technology Network xii
Conventions xii
1 Introduction **1**
Overview 1
System Administration Workbooks 1
Workbook and Wizard Descriptions 1
General Workbook Procedures 2
Change a Workbook's Calculation Method 2
Refresh and Export Data 2
Global Domain 4
Overview 4
Measure Data 4
2 Domain Administration **7**
DomainDaemon 7
Usage 7
Starting the DomainDaemon 8
Monitoring the DomainDaemon 9
Stopping the DomainDaemon 9
Losing a Client-Server Connection 9
Environment Variables 9
Lock Timeout Variable 10
Log File Backups 10
Profiling Logging 11
Centralized Administration 11
Administrative Workbook Templates and Wizards 11
RPAS Utilities 12
Domain Upgrade 12
Upgrade a Simple or Global Domain Environment 12
convertDomain 13
upgradeDomain 14
Upgrade Configurations 15
rpasinstall 17

3 Security and User Administration 19
Functional Overview 19
User Logon Security 19
Measure Level Security 20
Position Level Security 20
Workbook Security 21
User Administration 22
Overview 22
Access the User Administration Tab 23
Add a User 23
Add a User Group 24
Delete a User 24
Delete a User Group 24
Edit a User 25
Security Administration Workbook 25
Overview 25
Using the Security Administration Workbook 28
4 Hierarchy Maintenance 31
Overview 31
Hierarchy Maintenance Workbook 31
Hierarchy Maintenance Wizard 32
Hierarchy Maintenance Worksheet 32
Access Hierarchy Maintenance 34
Maintain a User-Defined Dimension within a Hierarchy 34
5 Measure Analysis 37
Overview 37
Measure Analysis Workbook 37
Measure Analysis Wizard 37
Measure Analysis Worksheet 37
Access Measure Analysis 38
Review and Edit Sales or Other Registered Measure Data 38
6 Workbook Auto Build Maintenance 39
Overview 39
Workbook Auto Build Maintenance Wizard 39
Accessing the Auto Build Maintenance Workbook 39
Add a Workbook to the Auto Build Queue 39
Delete a Workbook from the Auto Build Queue 40
7 Internationalization 41
Translation 41
Translation Administration 41
Translation Administration Workbook 42

8 Commit as Soon as Possible 45
Overview 45
Using Commit ASAP 45
Managing the Workbook Queue – showWorkbookQueues 46
Usage 46
Commit ASAP Settings – configCommitAsap 47
Usage 47
Logging and Technical Information 48
9 Batch Processes and RPAS Utilities 49
Overview 49
RPAS Utilities Logging Options 49
Log Levels 49
Utilities with Standard Logging 50
Using Shell Scripts to Run Batch Processes 53
A Sample Shell Script 53
Common Information and Parameters for RPAS Utilities 54
Configuration Tools Log Files 55
Error Files 55
10 Hierarchy Management 57
Overview 57
Placeholder Positions in the Domain 57
Position Repartitioning 58
Loading RDF and Curve Parameters after Repartitioning 58
Enabling Dummy Positions 59
Configuring and Scheduling the Rebuffering Process 59
Loading Hierarchies – loadHier 60
Notes 62
Reconfiguring the Partitions of a Global Domain – reconfigGlobalDomainPartitions 63
Updating or Reporting the Dummy Position Buffer – positionBufferMgr 66
Renaming Positions – renamePositions 68
Setting Properties for Dimensions – dimensionMgr 69
Setting Informal Positions to Formal – updateDpmPositionStatus 71
Exporting Hierarchy Data – exportHier 72
11 Data Management 73
Loading Measure Data – loadmeasure 73
Usage 73
Exporting Measure Data – exportMeasure 75
Exporting Measure Data – exportData 77
Usage 78
Mapping Data Between Domains – mapData 82
Usage 82

Moving Data between Arrays – updateArray 82
Usage 83
12 Operational Utilities 85
Find Alerts – alertmgr 85
Usage 85
Copying Domains – copyDomain 86
Usage 86
Move a Domain – moveDomain 89
Usage 89
Setting Miscellaneous Domain Properties – domainprop 91
Usage 91
Calculation Engine – mace 92
Usage 92
Managing Users – usermgr 95
Usage 95
Managing the Workbook Batch Queue – wbbatch 96
Usage 96
Workbook Manager – wbmgr 97
Usage 97
13 Informational Utilities 99
Retrieving Domain Information – domaininfo 99
Usage 99
Checking the Validity of a Domain – checkDomain 100
Usage 100
Determining RPAS Server Version – rpasversion 101
Usage 101
List Contents of a Database – listDb 101
Usage 101
Printing Data from Arrays – printArray 101
Usage 101
Printing Data from Measures – printMeasure 103
Usage 103
14 RPAS Reporter Tool User Guide 105
Technical Information about the ODBC Environment 105
Required Files 105
ODBC Configuration for Windows 106
Overview 106
Defining the ODBC Server Configuration Settings 106
Defining the ODBC Client Configuration 109
Creating the ODBC Windows Data Source 110
Starting the RPAS ODBC Server Process 112
Testing the Connection 112

ODBC Configuration for UNIX....113
Overview....113
Server Configuration....113
Client Configuration....113
Starting the RPAS ODBC Server Process....113
Testing the Connection....114
Tuning the ODBC Server....114
Installing and Using the RPAS JDBC Driver....115
Installing and Setting Up the Oracle RPAS JDBC Driver on Windows....115
Updating Environment Variables for JDBC Driver....120
Installing and Setting Up the JDBC Client on UNIX....122
Using the RPAS JDBC Driver....122
Using the jdbcisql Utility Provided with RPAS JDBC Driver....123
Using Oracle SQL Developer....124
Using Oracle JDeveloper....124
Using a Java Program....125
Data Query....126
Re-using Aggregate RPAS Reporter Tool Queries....126
Metadata....126
Fact and Dimension Tables....128
Measure Security in the ODBC Driver....130
Use Cases....130
Using Metadata Tables to Explore the Structure of a Domain or a Workbook....130
Querying Fact Data....132
Connecting to a Workbook....132
Requesting Additional Aggregate Tables....133
Clients....133
Oracle Business Intelligence Enterprise Edition (OBIEE)....134
Microsoft Access....135
JDeveloper....135
XML Publisher....136
Interactive SQL (ISQL) Utility....137
A Appendix: Integration Guide....139
RPAS, RDF and Planning Integration with RMS and Price....139
Summary of Integration Approach with RMS....139
Environment Variable Setup....140
RDF and Merchandise Financial Planning Transformation Programs....140
RDF and Merchandise Financial Planning Transformation Matrix....144
Common Programs for Extracts....145
Extract of Forecast Data for RMS....146
Load of Extracted Forecast Data and Standard Deviations to RMS....147
Extract of Diff Profile Data for RMS....147

Extract of Store Grade Data for RMS 148
Extract of Receipt Plan for RMS....149
Extract of Markdown Budget Data for Price....150
Extract of Current Plan Data for RDW....151
Extract of Original Plan Data for RDW....153
RDF and Merchandise Financial Planning Extract Matrix....155
Integration with Oracle Retail Workspace 156
Oracle Single Sign-on Overview....157
What is Single Sign-On?....157
What Do I Need for Oracle Single Sign-On?....157
Can Oracle Single Sign-On Work with Other SSO Implementations? 157
Oracle Single Sign-on Terms and Definitions....158
What Single Sign-On is not....159
How Oracle Single Sign-On Works....159
Installation Overview....161
User Management....162
B Appendix: Curve Administration Guide 163
curvevalidate 163
Usage 163
curvebatch....165
Usage 165

Preface

Oracle Retail Administration Guides are designed so that you can view and understand the application's "behind-the-scenes" processing, including such information as the following:

- Key system administration configuration settings
- Technical architecture
- Functional integration dataflow across the enterprise
- Batch processing

Audience

This document is intended for the users and administrators of Oracle Retail Predictive Application Server. This may include merchandisers, buyers, and business analysts.

Related Documents

For more information, see the following documents in the Oracle Retail Predictive Application Server Release 13.0 documentation set:

- *Oracle Retail Predictive Application Server Release Notes*
- *Oracle Retail Predictive Application Server Licensing Information*
- *Oracle Retail Predictive Application Server Installation Guide*
- *Oracle Retail Predictive Application Server User Guide*
- *Oracle Retail Predictive Application Server Online Help*
- *Oracle Retail Predictive Application Server Configuration Tools User Guide*
- *Oracle Retail Predictive Application Server Configuration Tools Online Help*

Customer Support

- https://metalink.oracle.com

When contacting Customer Support, please provide:

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

Review Patch Documentation

For a base release (".0" release, such as 13.0), Oracle Retail strongly recommends that you read all patch documentation before you begin installation procedures. Patch documentation can contain critical information related to the base release, based on new information and code changes that have been made since the base release.

Oracle Retail Documentation on the Oracle Technology Network

In addition to being packaged with each product release (on the base or patch level), all Oracle Retail documentation is available on the following Web site:

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

Documentation should be available on this Web site within a month after a product release. Note that documentation is always available with the packaged code on the release date.

Conventions

Navigate: This is a navigate statement. It tells you how to get to the start of the procedure and ends with a screen shot of the starting point and the statement "the Window Name window opens."

> **Note:** This is a note. It is used to call out information that is important, but not necessarily part of the procedure.

```
This is a code sample
    It is used to display examples of code
```

A hyperlink appears like this.

1

Introduction

Overview

All RPAS-based products require setup and the following administration activities to be performed:

- Domain administration
- User account management
- User and workbook template administration
- Hierarchy maintenance
- Measure analysis
- Workbook auto build maintenance
- Translation administration

System Administration Workbooks

Using the administration workbooks, designated employees manage other employees' use of the Oracle Retail Predictive Solutions. System administrators use the administration workbooks to perform the following:

- Set up and maintain users and user groups.
- Manage user access to specific workbook templates and individual measures.
- Edit the contents of translation tables to support multiple-language use of the application.
- Specify the type, frequency, and format of workbooks in the automatic build queue.

Workbook and Wizard Descriptions

RPAS contains the following workbooks and wizards to administer the system:

- User Account Management wizards – A group of wizards for setting up and maintaining users and user groups.
- Security Administration workbook – A workbook for setting up and maintaining user/template, user/measure, and template/measure access rights.
- Translation Administration workbook – A workbook for managing the foreign language translation of strings and label text throughout the application.
- Workbook auto build maintenance – A workbook for managing the workbook auto build queue.

General Workbook Procedures

Change a Workbook's Calculation Method

There are two types of calculation modes that can be set in the RPAS Client:

1. "Deferred" calculation mode (the most common) allows the user to make multiple edits in a workbook before recalculating the data. In this mode, the edits are effectively "queued" and executed once **Calculate** is selected.
2. "Automatic" calculation mode forces the workbook to be recalculated every time a cell is changed. This forces immediate communication from the worksheet back to the database. In this mode, there may be a pause between one data change and the user's ability to effect the next change.

For efficiency and usability purposes, Oracle recommends that operation be run in "deferred" calculation mode.

Set the Workbook to Deferred Calculation Mode

Click the **Edit** menu, and select **Manual Calculation**.

Return the Workbook to Automatic Calculation Mode

Click the **Edit** menu, and select **Automatic Calculation**.

Send the Queue of Data Changes to the Server

Click the **Edit** menu, and select **Calculate Now**.

Refresh and Export Data

Refresh the Data in a Worksheet

The Refresh feature allows the user to update a workbook with the data that is currently stored in the domain. This lets the user work with the most current data without having to rebuild the workbook. However, refresh rules in the configuration governs the refresh process. Please see the *RPAS Configuration Tools User Guide* for more information on setting up refresh rules.

Export the Current Worksheet View to an Output File

Navigate: From the File menu, select **Export Sheet**. The Save As dialog box displays.

1. In the **Save In** field, select a directory to save the export file.
2. In the **File** Name field, type a name for the export file.
3. Click the **Save As Type** list, and select a file type for the export file.
4. Make a selection for each of the following:
   a. Delimiter – Specify the character that is used to separate information in the output file. Standard choices are Tab, Comma, or Space; but different delimiters can be specified by selecting the **Other** radio button.

b. Labels – Specify the format of the label headers across the top of the cells in the output file. The options are:
   - **Do Not Include** – No labels provided
   - **Include Once** – One label placed across the top of each section of related cells
   - **Repeat** – A separate label, repeated as necessary, appears atop each cell
   - **Descriptions** – Specify whether to identify dimensional positions in the output file with concise system names (for instance, SKU00012) or the descriptive labels (for instance, Cashmere Sweater – L – beige) assigned to each position

5. Select **Save** to export the file.
6. Click **OK**.

Insert Measures into an Open Worksheet

Given the necessary access rights, the user can insert a measure or group of measures into a workbook that is already open. This functionality reduces the need to rebuild a workbook whenever a view of currently unrepresented measures is required. The ability to insert new measures into open workbooks is particularly useful when establishing access to alerts.

Use the following procedure to select a measure or group of measures to be inserted in a workbook that is currently open.

> **Note:** A worksheet must be open and active for the Insert Measure menu option to be enabled. Any measure(s) that are to be inserted in the workbook will be placed on the currently active worksheet.

Navigate: From the Edit menu, select **Insert Measure.**

1. Select the desired measure(s) in the **Insert Measures** list box.
2. Click **OK**.

> **Note:** Measures with writeable access that are added to the workbook using **Insert Measure** can be edited in the workbook. However, these measures have no commit rule, thus user edits cannot be committed to the domain.

Global Domain

Overview

"Global Domain" is a type of domain structure that provides users with the ability to view data from multiple domains and to administer common activities of an RPAS domain and solution.

Domains can be built in one of two methods:

- **Simple domain** – This is the traditional, stand-alone domain that has no visibility to other domains.
- **Global domain** – This is a domain environment that contains two or more "local" domains (or "sub-domains") and a "master" domain that has visibility to all local domains that are part of that environment.

Using a Global Domain environment has two primary functional benefits. The first feature allows users to have a global view of data in workbooks. Users can build workbooks with data from local domains, refresh global workbook data from local domains, save global workbooks, and commit the data from global workbooks to the individual local domains.

"Local" domains are typically organized ("partitioned") along organizational structures that reflect user roles and responsibilities. Most users will only work within the local domain(s) that contain their area of responsibilities, and they may not need to be aware of the Global Domain environment. For performance and user contention reasons, Global Domain usage should be limited to relatively infrequent processes that require data from multiple local domains.

The other primary feature of Global Domain is centralized configuration and administration. Most of the mechanisms that are required to build and administer a domain have been centralized and they need only be run in the "master" domain, which either propagates data to the local domains or stores the data centrally so that the local domains reference it in the master domain.

> **Note:** For a Global Domain environment to function properly, all local domains must be structurally identical.

Measure Data

In a global domain environment, measure data can be physically stored in two different ways:

1. across the local domains
2. in the master domain

Measure data that is stored in local domains is split across domains based on a pre-determined level of a given hierarchy. This level is defined during the configuration process, and it is referred to as the "partition" level.

The base intersection of a measure (for instance, what dimensions a measure contains) determines whether data is stored in the local domains or in the master domain. The data will be stored in the master domain if the base intersection of a measure is above the "partition" level or if it does not contain the hierarchy on which the Global Domain environment is partitioned. This type of measure is referred to as a "Global Domain measure," or a "Higher Base Intersection measure."

Consider a global domain environment where the partition-level is based on the Department dimension in the Product hierarchy. Data for measures that have a base intersection in the Product hierarchy at or below Department are stored in the local

domain based on the Department to which the underlying position in the Product hierarchy belongs. Other hierarchies are irrelevant for this discussion.

However, measures that have a higher base intersection in the Product hierarchy than Department (for instance, Division) or measures that do not contain the Product hierarchy (such as a measure based at Store-Week) cannot be split across the local domains. These measures will reside in the master domain and will be accessed from there when these measures are required in workbooks.

All measures will be registered in the master domain, and they are automatically registered in all local domains. RPAS automatically determines where the measure needs to be stored by comparing the base intersection of the measure against the designated partition-level of the Global Domain environment.

The physical location of the measure data will be invisible to the user after the measure has been registered. However, administrators must know where data for each measure is to be stored (master vs. local) as the data must be loaded in the proper location.

2

Domain Administration

DomainDaemon

The RPAS DomainDaemon is a process that is used to enable the communication channel between RPAS Clients and RPAS domains.

The DomainDaemon runs on the server side and waits for requests from RPAS Clients on a given port. Once DomainDaemon receives a request from a client, it starts a server process that the client connects to. From this point, the client and server communicate directly. The system administrators may choose to have one single DomainDaemon process for all of the users, or they may choose to have separate processes per domain, per enterprise, and so on.

The DomainDaemon is installed in the [RPASDIR]/bin directory. [RPASDIR] stands for the full path to the directory where the RPAS Server is installed. The system administrators can start, stop, and monitor the DomainDaemon processes by using scripts that are provided in this directory.

Usage

The following table provides descriptions of the arguments used by the `DomaimDaemon` utility.

Argument	Description
`–version`	Prints the RPAS version, revision, and build information of the utility.
`-start`	Starts a DomainDaemon on the specified port.
`-port` *`portNum`*	The *`portNum`* must be between 1025 and 65535 (inclusive). If *`portNum`* is set to `auto`, it will find any free port.
`-debug`	The `–debug` option enables additional logging.
`-loglevel`	This option enables additional logging.
`-timeout` *`milliseconds`*	Specifies the number of milliseconds to wait for the server to start. A value of -1 means no timeout.
`-server` *`serverProgramName`*	Specifies the name of the RPAS database server program. Defaults to `RpasDbServer`.
`-no_auto_add`	Disables the registering of domains in response to client requests to start a RPAS database server.
`-stop`	Stops the DomainDaemon on the specified port.
`-ping`	Reports the status of a DomainDaemon process.
`-showDomains`	Shows all domains managed by this daemon.
`-add pathToDomain`	Adds the specified domain to the list of domains managed by a DomainDaemon.

Argument	Description
`-activate` *`pathToDomain`*	Reactivates a previously deactivated domain. Specify the port number and the complete path to the domain.
`-deactivate` *`pathToDomain`*	Marks a domain as temporarily unavailable. Deactivating a domain also terminates all user sessions in that domain. A message will be displayed in the client to notify users when this occurs. Domains are most commonly deactivated before beginning a routine nightly/weekly batch process. This ensures that no users make updates to the system during these processes. Specify the port number and the complete path to the domain.
`-remove` *`pathToDomain`*	Removes the specified domain from the list of domains managed by a DomainDaemon.
`-showActiveServers`	Shows all active server processes. Specifying a port number is required. For each active server, the DomainDaemon shows the process ID, domain, and user ID.
`-stopActiveServers`	Stops all active servers. Specify a port number and a process ID.
`-stopServer` *`processId`*	Stops the server using the specified processed.
`-stopUser` *`userId`*	Stops the server using the specified userId.

Starting the DomainDaemon

In order to start the DomainDaemon, execute the script called DomainDaemon in the installation directory. The port number where the DomainDaemon will be running must be passed in as an argument. The port number must be between 1025 and 65535. If **auto** is specified instead of a number, the DomainDaemon is started on any available port.

> **Note:** In the following examples, [RPASDIR] stands for the full path to the directory where the RPAS Server is installed.

Example:

Issuing the following command from a UNIX shell will start a DomainDaemon on port 55278:

```
([RPASDIR]/bin)$ DomainDaemon –port 55278 –start &
```

Monitoring the DomainDaemon

The –ping argument can be used to see whether a DomainDaemon is active. The port number must also be passed as an argument. If the DomainDaemon is active on the port, a message will be printed, and the script will return true. Otherwise, the script will return false.

Example:

```
([RPASDIR]/bin)$ DomainDaemon –port 55277 –ping
DomainDaemon on port 55277 is alive.
```

Stopping the DomainDaemon

Use the –stop argument to stop the DomainDaemon running on a given port.

Example:

```
([RPASDIR]/bin)$ DomainDaemon –port 55277 –stop
```

Losing a Client-Server Connection

There are certain procedures to follow if the connection between the RPAS Client and the RPAS Server is lost. This connection can be lost for any number of reasons, but most commonly when the user's computer crashes or if the network connection is lost.

If this situation occurs, notify the system administrator if you do not have access to the server processes.

The system administrator needs to perform two steps:

1. Find the lost RPAS Server process that is associated with that user. This is done by using the `-showActiveServers` argument as specified below. Make note of the user's process ID.
2. Stop the user's RPAS Server process, which will remove any locks and allow the user to log into the RPAS Client and begin a new RPAS Server process. This is done using the `-stop server command` and the user's process ID as specified below.

The user can then log back into the RPAS Client.

Environment Variables

RPAS includes a number of environment variables that are set at the system level in UNIX. At the system level, the variables are applicable to all RPAS Servers (DomainDaemons) that are run on the system.

The common syntax for setting these variables is as follows:

```
Export ENVIRONMENT_VARIABLE=XXXXXX
```

`ENVIRONMENT_VARIABLE` is a defined variable that is recognized by RPAS. `XXXXXX` is an appropriate value for the variable, which could be a string, Boolean, or numeric data type. If the value represents time, this number normally represents time in milliseconds.

> **Note:** The DomainDaemon must be restarted after setting any environment variables. An example of how this process is completed is as follows:
> `DomainDaemon -port 55123 -start -debug &`

Lock Timeout Variable

When performing certain operations, it is possible for two or more users to be "contending" for access to the same database (.gem file), which happens most commonly when two users attempt to simultaneously commit/save the same data back to the domain. By default, RPAS is set up to wait one minute before returning a lock contention error when this situation occurs.

If desired, an administrator can override this default value by setting the "RPAS_LOCK_TIMEOUT" environment variable. This variable is set to the number of milliseconds to wait for a file lock before returning a lock contention error. As with any environmental variable, the variable must be set prior to starting the process that uses that variable. The variable was introduced for use with the RPAS database server, which means that the variable is set for the DomainDaemon.

For example, the two lines below indicate how an administrator would tell RPAS to wait two minutes before returning a lock contention error with the `RpasDbServer` after launching the client and logging in. Any client that connects to that domain daemon would see lock contention after a two minute delay:

```
Export RPAS_LOCK_TIMEOUT=120000
```

The `RPAS_LOCK_TIMEOUT` environment variable is also used to handle issues with firewalls and the `RpasDbServer`. The `RpasDbServer` now checks `RPAS_LOCK_TIMEOUT` to determine what should happen when it has been idle for a period of time. Like any environmental variable that `RpasDbServer` uses this environmental variable must be set prior to starting the DomainDaemon.

The environmental variable `RPAS_REQUEST_TIMEOUT` should be set to a value that is the number of seconds of idle time should pass before the `RpasDbServer` sends a "Server has timeout waiting for a request" exception to the client and exits. In this case idle time is the time waiting for a request. If this variable is not present or is set to zero then the `RpasDbServer` will never timeout.

On the client side the exception sent from the server will appear as a warning dialog box only after some new action is initiated. With the client there will be additional warning dialog boxes displayed to the user.

Log File Backups

The `RPAS_LOG_BACKUPS` environment variables allow an administrator to define the number of log file backups to retain for a given user. A log file is created each time for each session that a user has with the RPAS Client.

The environment is set by executing the following command:

```
Export RPAS_LOG_BACKUPS=X
```

`X` is an integer value that represents the number of backup log files to keep for each user.

Profiling Logging

The following two environment variables may be setup to control profiling logging:

- RPAS_PROFILING_ENABLE

  The RPAS_PROFILING_ENABLE environment variable, when set to true, allows profiling data to be written to the profiling log file. This flag does NOT affect writing to general RPAS log file, which is controlled solely by loglevel.

- RPAS_PROFILING_PATH

  The RPAS_PROFILING_PATH environment variable defaults to "rpasProfile.log" if not present. This variable specifies the profiling log file name. It can be overridden programmatically in the constructor of the profiling timer.

Centralized Administration

Note: If a solution is built in a Global Domain environment, most administrative activities can only be performed in the "master" domain. This applies to RPAS administrative workbook templates and wizards as well as RPAS utilities that are run on the backend against the domain.

Administrative Workbook Templates and Wizards

The following list includes the standard RPAS workbook templates and/or wizards that have been centralized. It can only be run in the master domain of a Global Domain environment. See the individual sections for additional information.

- Alert Manager window – Results of the alert finder run on the global domain are collated and displayed in this window.
  1. This applies to all alerts registered in the global domain.
  2. Results are based on data from all the individual local domains.
  3. Results are consolidated (added together) to display a single result per measure.
- Alert Manager workbook template – This template is used to build alert workbooks from the Alert Manager dialog window. Data will be retrieved from the local domains.
- Measure Analysis – This is used to analyze measure data from local domains.
- Security Administration – This is used to set security by template, measure, and positions. This workbook template can only be used in the master domain, and it is disabled for use in local domains.
- User Administration – User information will be set up and maintained in the global domain, but it will be replicated to the local domains. Updates are effective immediately after the changes are committed. This workbook template can only be used in the master domain, so it is disabled for use in local domains.
- Translation Administration – This is a template that is used to modify the labels of translatable data in RPAS. This workbook template can only be used in the master domain, so it is disabled for use in local domains.
- Hierarchy Maintenance – This is used to set up and maintain positions of user-defined dimensions. User-defined dimensions must be registered in the Global Domain by using the reguserdim utility.

RPAS Utilities

Please refer to the usage or syntax section on each utility for detailed information.

Domain Upgrade

Upgrade a Simple or Global Domain Environment

RPAS supports the upgrade of RPAS 11.0.x, 11.1.x, and 12.x environments to RPAS 13.

> **Note:** See the solution-specific Installation and Administration Guides for potential solution upgrade limitations.

This similar process should also be followed when upgrading between minor (patch) releases of RPAS (for example, RPAS 12.1.1 to RPAS 12.1.2). The following overview of this process calls out upgrade steps that are unique to upgrading to a major or minor release.

Upgrade Process Overview

The following is a high level description of the process that must be followed to upgrade an RPAS domain and configuration.

1. Acquire the new version of the platform, which includes the following key components:
   - RPAS Server
   - RPAS Client
   - RPAS Configuration Tools
2. Create a backup copy of any domains that will be upgraded.
3. Commit any existing workbooks that have not yet been committed (if commit is required). RPAS does not guarantee the upgrade of existing saved workbooks or styles as part of the upgrade process from one major release to the next.
4. Remove from the auto-workbook queue (`wbbatch` utility) any `-commit` and `-refresh` scheduled batch jobs. If upgrading an 11.0.x environment to 12.1.x (or later), all `-build` scheduled batch jobs should also be removed from the queue.

   > **Note:** Skip this step if upgrading a 12.1 or newer environment.

5. Run the `convertDomain` utility on existing domains. In global domain environments this utility is only executed from the master domain.

   > **Note:** Skip this step if upgrading a 12.1 or newer environment.

6. Run the `upgradeDomain` utility on existing domains. In global domain environments this utility is only executed from the master domain.
7. Upgrade existing configurations (see the following section, "Upgrade Configurations" for more information on this step).

   > **Note:** Skip this step if upgrading from 11.0.x, 11.1.x, and 12.0.x to RPAS 12.1.x or newer.

8. Optional: Run the `rpasinstall` utility with the `-patchinstall` argument. This step is only required if changes have been made to the configuration and these changes are ready to be propagated to the domain(s).

convertDomain

Migrating data from 11.0, 11.1 and 12.0 versions of RPAS to RPAS 12.1 (or newer versions) requires the `convertDomain` utility. It contains the necessary functions to read the pre-12.1 database and array formats and writes to the new 12.1 formats. It processes both simple and global domains, regardless of subdomain locations, can optionally remove the previous .gem files, and will recover from a partial conversion. Upon successful completion it will tag the domain as converted, which is visible in the domain history and visible with the `domaininfo` utility.

Usage

```
convertDomain -d domain [-loglevel level] [-purge] [-forcetag]
```

The following table provides descriptions of the arguments used by the `convertdomain` utility.

Argument	Description
`–d` *path*	Path to the domain being converted.
`-purge`	Inclusion of this argument indicates that existing (Acumate) .gem database files should be removed after conversion to reduce the space required for the process. The .gem file will be removed immediately after it has been converted.
`-forcetag`	This argument indicates that the domain should be tagged as converted even if some databases/arrays could not be converted.

The conversion process is done directly to the simple or global domain path specified with the `–d` argument. A `copyDomain` is **not** automatically performed beforehand by the utility.

The process works by first opening the "R_SUBDOMAINPATH%1" array in the "data/configmeasdata" database. If this fails then it is assumed that the domain is a simple domain and not a global one. If it succeeds then the subdomain paths are read from the array. Next the process searches for all .gem files in the subdomains and the master, recursively and converts each one. The order of the conversion should not be considered "predictable"; there is no assurance that subdomains will be converted in order, or that the master will be converted first or last. In order to ensure recoverability the "data/configmeasdata" database is not removed until the entire domain has been converted. Finally, if no errors occurred during conversion (or the `–forcetag` option is supplied) then the new domain is tagged with the "CONVERT" tag and time stamped.

> **Note:** `upgradeDomain` is **not** called by this process. It must be called separately. This allows the user to specify additional parameters to `upgradeDomain`. Once the domain has been converted and upgraded it is ready for use.

In the case of a partial domain conversion it is possible to restart convertDomain. Currently every database will be reconverted unless the –purge option is specified. In that case the conversion will restart with the last database that was being converted. There is no current way to skip converted databases if –purge was not used, nor can individual databases be converted.

upgradeDomain

Usage

```
upgradedomain -d domainPath [OPTIONS]
```

The following table provides descriptions of the arguments used by the upgradedomain utility.

Argument	Description
–d *path*	Path to the domain being upgraded.
–verbose	Optional parameter to show the detail about each change that is applied to the domain.
-n	Optional parameter to report which changes would be applied without applying the changes.
-purgeWorkbooks	Required parameter that purges all existing workbooks and clears the workbook batch queue.
-ignoreSharedNames	Allow upgrade even if dimensions and hierarchies share names
-apptag	Indicate the application and version associated with this upgrade. Parameter must be APP:VERSION.

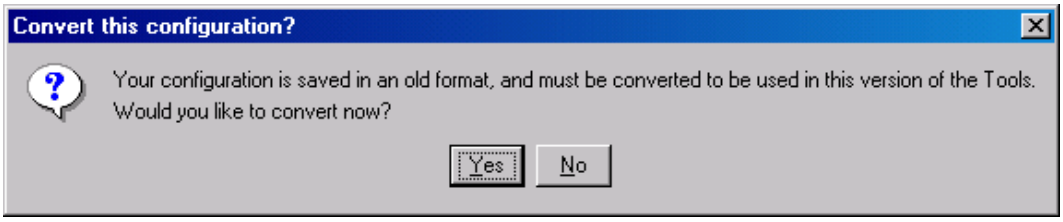
Note: An administrator can also run the Domain Information utility to verify the upgrade process as shown below.

```
domaininfo –d pathtodomain –domainversion
```

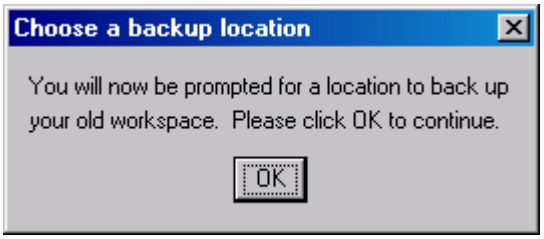
Upgrade Configurations

Once the RPAS Configuration Tools have been installed and the domains have been upgraded, an administrator should upgrade existing configurations to version 13 or the latest version 13 patch.

1. Launch the new version of the Configuration Tools; open the existing configuration using **File–Open**. Select the configuration and click **OK**. A message box appears prompting you to convert the configuration.

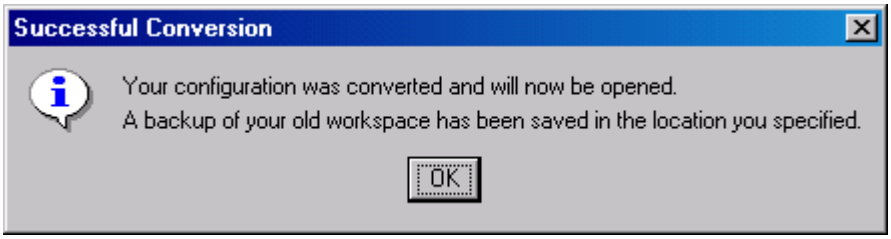


2. Click **Yes**. The Backup Location message box appears. Click **OK** to continue.



3. The administrator must now specify a location to store the backup of the current (non-upgraded) configuration. Use the browser to identify a location and enter the back-up configuration name. The backup may be stored in the same location as the configuration being upgraded only if the name of the backup is changed from its original name.

   The Successful Conversion dialog box appears if the configuration was converted without any issues.



4. The upgraded configuration will now be open in the Configuration Tools. Changes can now be made to the configuration. Note that if changes are made the domain must be updated through the normal patch install process.

> **Note:** If upgrading RDF, Grade or Curve see "Upgrade Application Configurations" for additional steps to complete the application upgrade.

Command-line Support for Configuration Upgrading

The RPAS Configuration Tools also provides command-line support to upgrade configurations. The Configuration Converter is a standalone utility that converts a configuration that was originally created and saved in a prior release of the Configuration Tools. Only configurations created in a prior major release need to be converted. Configurations saved in previous versions of the same major release, but in different minor releases, do not need to be converted. See the *RPAS Configuration Tools Guide* for more information on using the Configuration Converter (RpasConverter.exe) via the command-line.

> **Note:** If upgrading RDF, Grade or Curve see "Upgrade Application Configurations" for additional steps to complete the application upgrade.

Upgrade Application Configurations

Applications such as RDF, Curve and Grade are configured using a plug-in architecture in the RPAS Configuration Tools. This architecture allows for the automation of most configuration activities for the solution. The plug-in requests specific information from the configuration administrator and the solution auto-generation tool automatically generates the solution configuration. Prior to each patch or upgrade to a major release, the auto-generation tool should be executed to ensure the solution configuration is updated with the base configuration changes for the application. See the *RDF*, *Curve*, or *Grade Configuration Guides* for more information on the auto-generation process for each application.

rpasinstall

Usage

`rpasInstall <arguments>`

The following table provides descriptions of the arguments used by the `rpasInstall` utility.

Argument	Description
`-fullinstall`	Required argument.
`-patchinstall`	Required argument.
`-testinstall`	Required argument.
`-ch` *`config_home`*	Required argument. Path to the directory containing the configuration file.
`-cn` *`config_name`*	Required argument. The name of the configuration.
`-in` *`input_home`*	Required argument. Path that includes the directory containing the input files for the domain to be created.
`-log` *`log_name`*	Required argument. Path that includes the name of the log file to be created or updated.
`-configdir` *`config_directory`*	The path to the directory containing the xml files used by RPAS. This is a required argument if the user wants to supply globaldomainconfig.xml or calendar.xml.
`-dh` *`domain_home`*	The path to directory in which the domain will be created. Use if and only if a globaldomainconfig.xml is not used.
`-p` *`dim_name`*	The partitioning dimension. Use if and only if global domain is being implemented without the use of globaldomainconfig.xml.
`-rf` *`function_name`*	The filename of the function to be registered. This pairing may be repeated for multiple functions. This argument is not required if there are no functions to be registered.
`-updatestyles`	This argument will enable the update of styles based on the styles set in the Configuration Tools. If using `-patchinstall`, this results in loss of manual formatting (if any) to measures.

3

Security and User Administration

Functional Overview

This chapter describes the security model in RPAS, which includes workbook templates, workbooks, measures, and positions. The levels of security are defined as measure level, position level, and workbook level.

This chapter also describes user administration and security administration.

> **Note:** If a solution is built in a Global Domain environment, it is only required and possible to perform the administrative activities included in this section in the "master" domain.

User Logon Security

User accounts may be marked as **locked out** by the domain administrator.

This will prevent the user from logging in to the RPAS Client. The account remains locked out until the administrator re-enables the account.

Account lockouts may be set or cleared by the domain administrator by using the User Management utility.

The account may be marked as **must change password**.

This is useful for brand-new accounts. The user will be allowed to logon with the current password and then forced to select a new password.

Must change password may be set or cleared by the domain administrator using the User Management utility.

Account Lockout may be enabled for a domain.

The domain administrator selects a number of failed logon attempts after which the User account will be marked as locked out. The account will remain locked out until the administrator re-enables it.

Account Lockout can be enabled through the `domainprop` utility by using the `-lockAccount` flag.

Password expiration may be enabled for a domain.

The domain administrator selects a number of days after which passwords expire. When the user logs in, the system requires a new password to be entered if the configured number of days has passed since this user entered a new password.

Password expiration can be enabled through the `domainprop` utility using the `-expirePassword` flag.

Password history may be enabled for a domain.

The domain administrator selects a number of passwords to save. When the user attempt to change passwords, the system will not permit any password already stored in the password history to be used again.

Password history may be enabled through the `domainprop` utility using the `-passwordHistory` flag.

Measure Level Security

Measures have access rights; which are read-write, read-only, or denied. Measures that are read-write or read-only may be selected in the extra measures and insert measure dialogs. RPAS ensures that read-only measures are not editable by the user and the presence of read-only measures does not affect the ability to commit a workbook.

Measure security can be specified and changed through the Security Administration workbook. The Measure Rights worksheet allows Read Only, Deny, or Read/Write access to a measure to be specified for each user. These measure rights only apply to the Measure Analysis Workbook.

A workbook template overrides the default security of a measure, but it can only narrow the security of the measure. For example, a measure could have default read-write access for a user and a template could specify that all users have read-only access to the measure when a workbook is built. However, if the default measure security was read-only, the template could not expand the security of that measure to read-write. Measures that are explicitly made read-only by a workbook template will not be expanded to read-write access by RPAS.

Note: Refer to the *RPAS User Guide* from more information on the Measure Analysis workbook.

Position Level Security

Position Level Security allows access control for dimensions on a position-by-position basis. This capability is completely optional. If position level security is not explicitly defined and configured, all users in a domain have access to all positions in all hierarchies. Once position level security is defined, access to a position can be granted or denied for individual users, users in a group, or for all users.

Position level security can be defined at levels (dimensions) at or above base (such as class in the product hierarchy) in any hierarchy other than calendar. As positions are added at a level/dimension lower in the hierarchy than where the position level security is maintained, access to those positions is automatically granted if a user has access to the "parent" position. In other words, if security is maintained at the subclass level, users are automatically granted access to all the SKUs in a given subclass if they have access to that subclass. This includes those that were added after security was established.

Exactly one dimension in each hierarchy can be defined as the security dimension for the hierarchy. If a security dimension is defined for the hierarchy, all dimensions in the hierarchy have position level security enabled, but position security is set at or above the designated dimension. For instance, if the "class" dimension is designated as the security dimension, an administrator can maintain access to positions in the class dimension or at any level above class.

To specify the security dimension for a hierarchy, use the RPAS Configuration Tools or the `hierarchyMgr` utility.

After a security dimension is defined for a hierarchy, all users in the domain default to having access to all positions in any dimension in the hierarchy. Additionally, users automatically have access to newly added positions to a domain. Worksheets in the Security Administration workbook are used to control position access for individual users, user groups, or all users (referred to as "world" or default access). There are three worksheets in this workbook for each hierarchy with a defined security dimension. The default worksheet controls access to positions for all users (for instance, Prod Security Default); one worksheet controls access to positions by user group (for instance, Prod

Security Group); and the last worksheet controls access to positions by individual users (for instance, Prod Security User).

Access must be granted at all levels for a user to have access to a position. This means that a position must have a value of true at the levels default/world, group, and user. The following table demonstrates how access is granted or denied based on all combinations of settings:

Security set by Position **Denied = False** **Granted = True**			**Based on settings on left, user is Granted or Denied access**
User	**User Group**	**World**	**Resulting Access**
Denied	Denied	Denied	Denied
Denied	Denied	Granted	Denied
Denied	Granted	Denied	Denied
Granted	Denied	Denied	Denied
Denied	Granted	Granted	Denied
Granted	Denied	Granted	Denied
Granted	Granted	Denied	Denied
Granted	Granted	Granted	Granted

Position level security is used when a user selects positions in the wizard process before building a workbook. Only positions to which a user has access are available for selection in the 2-tree, which are then included in the build of the workbook.

Workbook Security

Currently, workbook access is either granted or denied. If the user has been granted access to a workbook; s/he can open, modify, and commit the workbook. No distinction is made between read-write-commit, read-write, and read-only access. Workbook access is automatically granted to the user that built it, and it may be shared with multiple groups or the world.

> **Note:** A user must have access to the workbook template in order to access the workbook, even if the workbook has world or group access rights.

Users with administrator status automatically have access to all workbook templates. By default, administrators have access to all workbooks that are saved with world access. If a workbook is saved with group access, administrators can only access the workbook if they are members of the default user group of the user who saved the workbook.

Another aspect of workbook security is the ability to set limits for the number of workbooks that a user can have saved at any given time. Limits can be set for a user per template, for a user group per template, or for a template for all users. The limits are evaluated in the above order, which means that a limit defined at user-template overrides any values defined at group-template or template. If the above limits are not defined, the default value is one billion.

The limits are checked when the workbook build process is initiated. When the limit is reached, an error message displays informing the user that the workbook build process cannot complete because the limit has been reached. The message also lets the user know what that limit is. The wizard process then terminates.

Administrative users have full access to all workbook templates regardless of the access rights that other admin users may assign to them in the Security workbook. The administrative user can build the Security workbook to change the access right back, so the nominal assignment does not matter for administrative users.

Non-administrative users so not have access to Security template and User Administration template groups even if the administrator inadvertently assigns them access rights.

User Administration

Overview

User administration is the process by which administrators add and/or delete authorized system users, create and/or delete user groups, and edit user profiles. These tasks are performed through completion wizards on the User Administration tab. The following procedures are discussed in this area:

- Access the User Administration tab
- Add a user
- Add a user group
- Delete a user
- Delete a user group
- Edit a user's profile

Once users and user groups are set up, access permissions to workbook templates and measures within workbooks can be assigned through Security Administration. Security Administration also supports modification of the label, default workbook template, and/or Admin status associated with individual users.

Access the User Administration Tab

1. Select **New** from the File menu. The New dialog box appears.
2. Select the **User Administration** tab.

Add a User

1. From the File menu, select **New**.
2. Click the **User Administration** tab.
3. Select **Add User**
4. Click **OK**.
5. In the **ID** field, type the ID string that the user will use for logging on.

   Note: Each user ID must begin with a letter and contain no spaces (the underscore character is acceptable).

6. In the **User Label** field, type a label that describes the user (for example, the user's full name). This identifying label appears in various locations throughout the application. For example, labels appear on the File – Open dialog box to identify the owner of a given workbook, and on the Forecast Approval worksheet to specify which user approved a given forecast.
7. In the **Default Group** field, select the user group to which the user will belong.
8. If a user will belong to more than one group, select the additional groups from the list in the **Other Groups** field.
9. In the **Password** field, type a password for the user.
10. In the **Password Verification** field, type the same password.
11. If the user should have Admin status, check the **Administrator** box.

    Note: Admin status enables users to perform the Format menu option Save Format/Admin, which creates new system-wide default styles for workbook templates. If unsure whether a user should be granted this ability, note that a user's Admin status can later be modified on the Users worksheet of the User and Template Administration workbook.

    Note: Granting users Admin status gives them access to all workbook templates, but it does not automatically give them access to all workbooks.

12. If the user must change his or her password when logging on for the first time, check the **Force Password Change** box.
13. Check the **Lock User Account** box to temporarily disable the user's account.
14. Click **Finish** to add the new user to the database.

Workbook template and measure access rights can now be assigned to the user. To do so, access the User and Template Administration workbook.

Add a User Group

User groups provide an intermediate level of security to workbooks that were created and saved by specific users. When new users are assigned to the system, they must be assigned to existing user groups. User groups should consist of individuals with similar job functions or responsibilities. In the Oracle Retail Predictive Planning Suite, the user group corresponds to the user's planning role.

1. Select **New** from the File menu.
2. Click the **User Administration** tab.
3. Select **Add User Group**.
4. Click **OK**.
5. In the **Group Name** field, type a name for the group.
6. In the **Group Label** field, type a descriptive label for the group. This label is displayed when referring to the group throughout RPAS.
7. Click Finish to add the user group to the database.

Delete a User

If a user profile is no longer needed, it should be deleted from the system in order to maintain system security.

1. From the File menu, select **New**.
2. Click the **User Administration** tab.
3. Select **Delete User**.
4. Click **OK**.
5. Select the name of the user to delete.
6. Click **Finish** to delete the user from the system.

Delete a User Group

If a user group no longer exists, the group should be deleted from the system as soon as possible to maintain system security.

> **Caution:** Deleting a user group will delete every user in that group.

1. From the File menu, select **New**.
2. Click the **User Administration** tab.
3. Select **Delete User Group**.
4. Click **OK**.
5. Select the user group to delete.
6. Click **Finish** to delete the user group from the system.

Edit a User

1. From the File menu, select **New**.
2. Click the **User Administration** tab.
3. Select **Edit User**.
4. Click **OK**.
5. Select the user to edit.
6. Click **Next**.
7. Make the necessary changes to the user's profile. The administrator can change anything except the User Name. See "Add a User" for details.
8. Click **Finish** to save the changes.

Security Administration Workbook

Overview

The Security Administration workbook is only available to system administrators. After users and user groups are created, the administrator may set up and maintain access permissions to workbook templates and measures within those workbook templates. This workbook allows the administrator to determine which templates individual users can access, as well as the measures that users can access while manipulating workbooks in the system. S/he can also specify and restrict the measures that are available to be added to a given workbook template. Setting access permissions in this way provides a high degree of measure security, because users can be restricted to viewing and editing only certain relevant measures.

All administrative users have full access to all workbook templates regardless of the access rights that they were assigned in the Security workbook by other administrative users. The administrative user can build the Security workbook to change the access right back, so the nominal assignment does not matter for admin users.

The Security Administration workbook has one or more worksheets for each of the following:

- Workbook Template Rights worksheet
- Workbook Template Measure Rights worksheet
- Measure Rights worksheet
- Dimension Modification Rights
- Position Level Security
- Workbook Template Limits

Security Template Administration also allows the administrator to modify the label, Admin status, and/or default workbook template associated with each user. You also access this workbook template to modify the labels associated with user groups, workbook templates, and workbook template groups. Using this workbook, the administrator can:

- Assign and modify access rights of each user to all workbook templates. User/template permissions are set in the Workbook Template Rights worksheet.
- Determine which optional measures are to be accessible through individual workbook templates. Template/measure permissions are set in the Workbook Template Measure Rights worksheet.
- Assign/restrict user access to individual measures. User/measure permissions are established in the Measure Rights worksheet.

Workbook Template Rights Worksheet

The Workbook Template Rights worksheet is for setting and maintaining access permissions of each user to specific workbook templates.

The worksheet contains a checkbox for each available workbook template and user combination. A checkmark in the cell indicates that the user has access rights to that specific template.

To grant a user access rights to a workbook template, put a checkmark in the checkbox in for that workbook template.

To deny a user access rights to that specific workbook template, leave the checkbox blank or clear the checkmark.

After changing a user's profile, the changes must be committed to the database in order for them to take effect.

The "Read Only" permission on a template applies only to actual workbooks created by the template. For templates that do not generate a workbook, but only run through a wizard process for other purposes, the "Read Only" permission for a user on that template will not prevent them from running through the wizard. This applies to standard RPAS templates, such as Add User and Delete User, but it may also apply to various application-specific templates.

Workbook Template Measure Rights Worksheet

The Workbook Template Measure Rights worksheet allows administrators to determine which registered measures will be available for optional inclusion in newly built workbooks.

When a measure is initially registered as a public measure, all templates default to having access to that measure. This means that it is possible for this measure to be added to a workbook template, even if it is not one of the standard measures displayed when a workbook of that type is built. Some new workbook wizards include a dialog that prompts users to select any additional measures to be included in the workbook build. By default, all newly registered measures are included on this list of available additional measures. The other method of inserting new measures into a workbook is via the Insert Measure command.

The Workbook Template Measure Rights worksheet is used to modify template/measure permissions, which allows only certain templates to optionally include specified measures in new workbook builds.

This worksheet contains a drop-down list for each available workbook template and registered measure combination. Three security options are available: Denied, Read-only, Full Access. Setting a value to Read-only or Full Access will allow users access to the workbook template. Full Access indicates that the user may edit data in the workbook. However, Denied or Read-only measure rights could prevent the user from committing data.

Measure Rights Worksheet

The Measure Rights worksheet allows the administrator to restrict user access to individual measures on a user-by-measure basis. User/measure permissions are initially determined by the system by integrating the current user/template and template/measure settings and applying the following rule: "A user cannot have access to any measure that is not available in at least one template to which the user has access."

Permissions can be made even more restrictive on a user by measure basis by using the Measure Rights worksheet to deny users access to measures that they would normally be permitted to edit.

The worksheet contains a drop-down list for each available user and registered measure combination. Three security options are available: Denied, Read-only, Read/Write. Denied prevents the user from viewing data. Read-only allows the user to view the data. Read/Write allows the user to edit data values. However, a commit rule must be configured for a measure for data to be committed to the RPAS data store.

> **Note:** The Measure Rights worksheet contains only public measures; that is, measures that can be optionally included in a worksheet, depending on choices made in a new workbook wizard. Measures that are registered as private measures will not appear in this worksheet. If there are no public measures available to be displayed in this worksheet, the worksheet will not be built.

Dimension Modification Rights Worksheet

The Dimension Modification Rights worksheet allows the administrator to determine which dimensions, if any, a user can modify. The worksheet contains a checkbox for each available user and dimension combination. A checkmark in the cell indicates that the user is permitted to modify the specified dimension.

After changes are made to a user's dimension modification rights, they must be committed before they take effect.

Position Level Security Worksheets

The position-level security worksheets are used to grant or deny access to positions for individual users, user groups, or all users. Position-level security is set for a specific dimension of a hierarchy (other than calendar). See the *RPAS Configuration Tools User Guide* for more information on setting position-level security dimensions.

For each hierarchy/dimension that has position-level security enabled (normally just a single hierarchy/dimension), there are three worksheets: one each for user, user group, and world/all users.

After changes are made to position-level security, they must be committed before they take effect.

Workbook Template Limits Worksheets

The Workbook Template Limit worksheets are used to limit the number of workbooks that the user can have saved. Limits can be set for a user per template, for a user group per template, or for a template for all users. The limits are evaluated in the above order, which means that a limit defined at user-template will override any values defined at group-template or template. If the above limits are not defined, the default value is 1 billion, but it is not displayed in the workbook.

The limits are checked when the user begins the workbook build process. If the limit has been reached, an error message appears that informs the user that the workbook build process cannot complete because the limit has been reached. The wizard process then terminates.

Using the Security Administration Workbook

Note: These tasks are performed through the Security Administration workbook. This workbook is only available to system administrators.

Access Security Administration

1. From the main menu, select **File – New**. The New dialog box appears.
2. Select the **Administration** tab to display a list of workbook templates for Administration.
3. Highlight **Security Administration**.
4. Click **OK**.

Set or Modify Users' Access to Workbook Templates

1. From the File menu, select **New**.
2. Click the **Administration** tab.
3. Select **Security Administration**.
4. Click **OK**.
5. On the Workbook Template Rights worksheet, select each template for which a user's access rights require modification. Set to Denied, Read-only or Full Access.
6. Changes must be committed to the master database before they take effect. To commit the changes, select **Commit Now** from the File menu.
7. Save the workbook by selecting **Save** from the File menu, if desired.
8. To close the workbook, select **Close** from the File menu.

Set Measure Availability for Workbook Templates

1. From the File menu, select **New**.
2. Click the **Administration** tab.
3. Select **Security Administration.**
4. Click **OK**.
5. On the Workbook Template Measure Rights worksheet, select each registered measure that should be available for inclusion in the associated workbook template. For measures that should not be included in the associated template, make sure there is no check mark.
6. Changes must be committed to the master database before they take effect. To commit the changes, select **Commit Now** from the File menu.

7. Save the workbook by selecting **Save** from the File menu, if desired.
8. To close the workbook, select **Close** from the File menu.

Assign or Restrict User Access to Measures

1. From the File menu, select **New**.
2. Click the **Administration** tab.
3. Select **Security Administration**.
4. Click **OK**.
5. On the Measure Rights worksheet, for each measure that a user should have access to, select either **Read Only** or **Read/Write** from the drop-down list. For measures to which the user should not have access, make sure **Denied** is selected.
6. Any changes made must be committed to the master database before they take effect. To commit the changes, select **Commit Now** from the File menu.
7. Save the workbook by selecting **Save** from the File menu, if desired.
8. To close the workbook, select **Close** from the File menu.

Change a User's Ability to Modify Dimensions

1. From the File menu, select **New**.
2. Click the **Administration** tab.
3. Select **Security Administration**.
4. Click **OK**.
5. On the Dimension Modification Rights worksheet, select each dimension for which the user needs modification rights. For dimensions that the user should not be able to modify, make sure there is no check mark.
6. Any changes made must be committed to the master database before they take effect. To commit the changes, select **Commit Now** from the File menu.
7. Save the workbook by selecting **Save** from the File menu, if desired.
8. To close the workbook, select **Close** from the File menu.

Set or Modify Access to Positions (if position level security has been enabled)

1. From the File menu, select **New**.
2. Select the **Administration** tab.
3. Select **Security Administration**.
4. Click OK.
5. Select the worksheet for which security needs to be set or modified: **User**, **User Group**, or **World**.
6. By default, the dimension (level) at which position level security is enabled will be displayed. To manage security at a level above the designated level (only levels above are possible), right-click and **Select Rollup** to view the available dimensions.
7. To grant access to a position, click the checkbox of the cell.

> **Note:** A user must have access at the User, User Group, and World levels to have access to a position.

8. Changes must be committed to the domain before exiting in order for them to take effect.

Limit the Number of Workbooks that a User Can Save

1. From the File menu, select **New**.
2. Select the **Administration** tab.
3. Select **Security Administration**.
4. Click **OK**.
5. Select the worksheet for which the limit will be set: User / Template, Group / Template, or Template.
6. Set the values as necessary.
7. Commit the data to the domain before exiting.

4

Hierarchy Maintenance

Overview

Hierarchy Maintenance Workbook

Oracle Retail Predictive Solutions provide the ability to set up and maintain user-named and user-defined dimensions within hierarchies. Hierarchy Maintenance is the means by which custom-created dimensions within a hierarchy can be established and maintained through the application interface in order to meet individual business needs.

When Oracle Retail Predictive Solutions are installed, implementation scripts define the dimensions and hierarchical structures specific to the customer's organization. For example, the system can be built to recognize that SKUs roll up into styles, styles roll up into product classes, and so on within the product hierarchy. Occasionally, you might want to group products according to some ad hoc personal design to suit a particular business need. You can group arbitrary items in a hierarchy to use in functions such as forecasting, replenishment, and measure analysis. These user-defined groupings act as normal dimensional levels. In other words, they allow the user to roll data up from lower levels of aggregation along the hierarchical paths that you define.

For example, suppose experience has shown that the accuracy of forecasts for your top 50 products (A products) reflects the relative accuracy of all forecasts. Therefore, you would like to group elements within a user-defined dimension as the top 50 products by designating them 'A Products.' Then, when you select products in a wizard or look at data in a worksheet, you can change the rollup to your user-defined dimension to see your top 50 products grouped together.

> **Note:** Your collection of 50 products may comprise elements from a wide range of product classes or departments, and your grouping scheme may have little to do with the normal dimensional relationships of these items in the product hierarchy.
>
> The group of items you designate as 'A Products' may change over time as consumer preferences change. From this example, you see that user-defined dimensions can be used to create any ad hoc groupings to provide additional support in analyzing, selecting, or summarizing data in Demand Forecasting. The Hierarchy Maintenance interface allows you to change the nature of the groupings as required.
>
> The number and names of user-definable dimensions are set by your company when an RPAS-based solution is initially installed. The positions within each dimension and their associated labels can be altered and maintained through the hierarchy maintenance process.

Keep in mind that any hierarchy in RPAS can have user-defined dimensions within it as long as they are set up by your company at the time of installation. The examples in this section refer to the Product hierarchy, but other hierarchies could be maintained in the same way.

Hierarchy Maintenance Example

Suppose you want to designate SKUs in your product hierarchy as either A, B, or C products so that you can group these items together when you view information, such as forecasting, replenishment, or measure analysis reports.

To do this, you need to maintain a user-defined dimension that will allow you to map the SKUs to the various positions of your classification scheme (A, B, or C). The user-defined dimension used in the following example is named Product Status. To maintain this user-defined dimension, use the Hierarchy Maintenance Wizard.

Hierarchy Maintenance Wizard

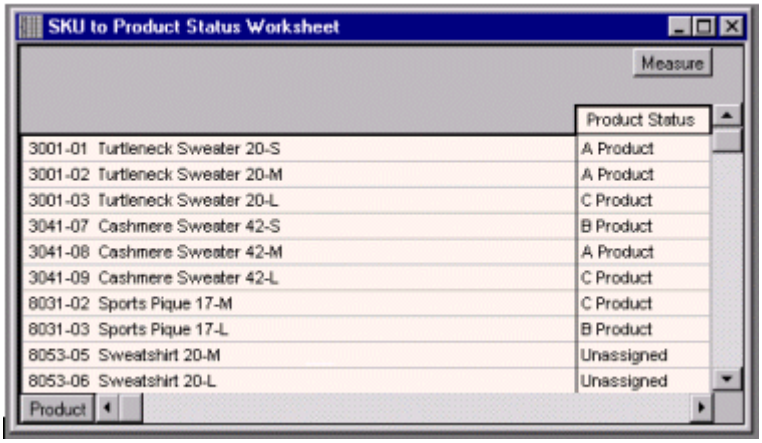
The first step in maintaining hierarchies is to access the Hierarchy Maintenance Wizard. In this wizard, select the SKUs that will be mapped to the various positions of the user-defined dimension. Responses to prompts in the wizard are used to format a new Hierarchy Maintenance workbook.

Hierarchy Maintenance Worksheet

The Hierarchy Maintenance worksheet displays the position assignment fields for the selected custom dimension. Edit the cells associated with the custom dimension as required.

Returning to the example dimension Product Status, you want to classify each selected SKU in your workbook as an A Product, a B Product, or a C Product. This example provides only three positions, or values, in the Product Status dimension; however, you can enter any character string in an individual SKU's Product Status cell. This new string will be treated as a separate user-defined grouping. If this is the first time a particular SKU has been mapped to the Product Status dimension, the label assigned to that SKU will not yet be defined. The Product Status field is automatically filled with 'Unassigned.'

Assign labels to each product with regard to the Product Status dimension. In the following example, products that were previously 'Unassigned' are now designated as A, B, or C Products.

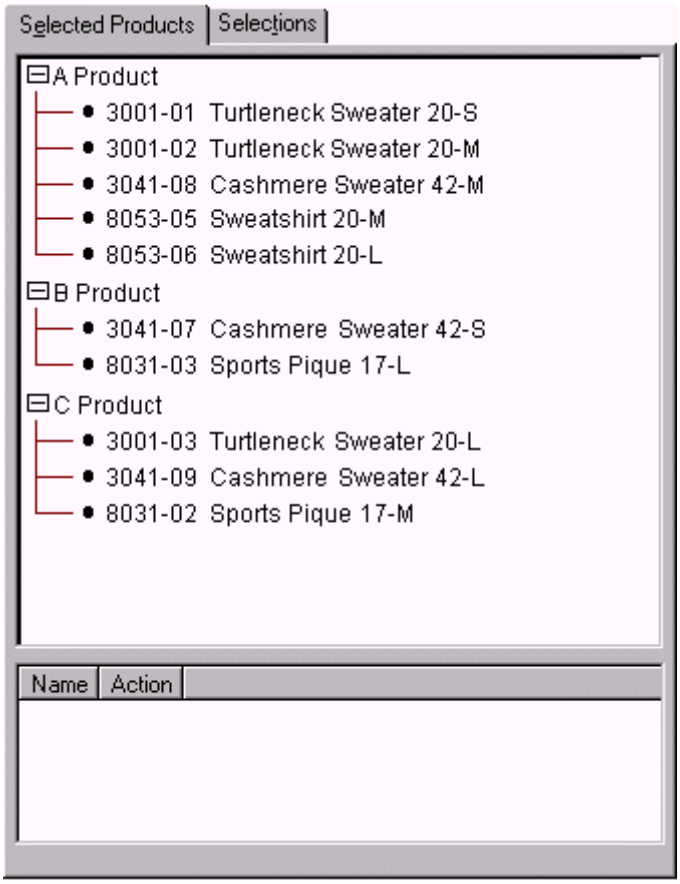


Note: The Oracle Retail Predictive Solutions system is case-sensitive when a new position name (label) is entered in the Hierarchy Maintenance workbook. After the workbook is committed, the typing of the group name is not case-sensitive. For example, "B Product" can later be entered as "b product" after the "B Product" group label has been committed.

After making the A, B, or C Product designations for the selected SKUs, you must commit the workbook for any changes to take effect.

For this example, labels have now been assigned to the various positions within the Product Status dimension, and selected products in the product hierarchy have been classified with regard to the custom dimension. Demand Forecasting treats Product Status, a user-defined dimension, as a normal dimensional level within the product hierarchy.

The following figure displays the results when, in a wizard, access a quick menu and change the rollup to the Product Status dimension. The products shown here are classified according to the position values (A Product, B Product, or C Product) that were assigned while maintaining the Product Status dimension.



Access Hierarchy Maintenance

> **Note:** If a solution is built in a Global Domain environment, it is only required and possible to perform all the administrative activities in this section in the "master" domain.

1. Select **Open** from the File menu to bypass the Hierarchy Maintenance wizard, and open an existing Hierarchy Maintenance workbook, or select **New** from the File menu.
2. Select the **Administration** tab to display the list of Administration templates.
3. Select **Hierarchy Maintenance**.
4. Click **OK**.
5. Select the hierarchy to specify a user-defined dimension (for example, Product or Location). Only the hierarchies that have been set up to contain user-defined dimensions are represented here.
6. Click **Next**.
7. Select the user-defined dimension to be updated. The number and names of available custom dimensions are set at installation.
8. Click **Next**.
9. On the **Available** side of the selection wizard, choose the items to be mapped to positions within the custom dimension.
10. Click the right arrow button to move them to the **Selected** side.
11. Once all items to appear in your workbook have been selected, click **Finish**.

Maintain a User-Defined Dimension within a Hierarchy

Use this procedure to assign product or location items to custom-defined positions within a specialized dimension. Custom-created dimensions are distinct from those in the standard hierarchical roll-ups configured in the system implementation. Users can use these dimensions like normal Demand Forecasting levels, aggregating data along these new hierarchical paths.

1. Select **New** from the File menu.
2. Select the **Administration** tab to display the list of Administration templates.
3. Select **Hierarchy Maintenance**.
4. Click **OK**.
5. Select the hierarchy to specify a user-defined dimension (for example, Product or Location). Only the hierarchies that have been set up to contain user-defined dimensions are represented here.
6. Click **Next**.
7. Select the user-defined dimension to be updated. The number and names of available custom dimensions are set at installation.
8. Click **Next**.
9. On the **Available** side of the selection wizard, choose the items to be mapped to positions within the custom dimension.
10. Click the right arrow button to move them to the **Selected** side.
11. Once all items to appear in your workbook have been selected, click **Finish**.

12. The Hierarchy Maintenance workbook is displayed. In the position assignment field for the custom dimension, assign a value to each product or location position in the workbook. Enter any text string in a cell. Each unique string will be treated as a separate user-defined position within the custom dimension.
13. Select **Commit Now** from the File menu to commit the changes to the master database. If desired, the user may also save the workbook by selecting **Save** from the File menu.
14. To close the workbook, select **Close** from the File menu.

5

Measure Analysis

Overview

Measure Analysis Workbook

The Measure Analysis workbook template allows the user to view data associated with any registered measure in the Oracle Retail Predictive Solutions applications, such as actual sales data for specified product/location/calendar combinations. The user may also use the Measure Analysis workbook to edit values for writeable measures, however commit capability is only allowed to administrative users.

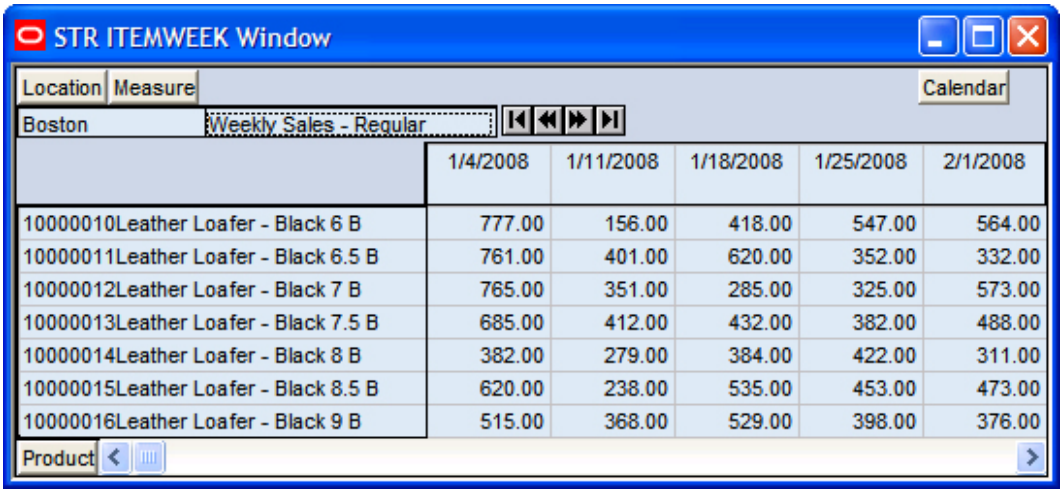
Although a common use of the Measure Analysis workbook is to view actual sales data, the workbook is not restricted to presenting sales data alone. The user can view any data loaded into the Oracle Retail Predictive Solutions master database, such as selling prices, shipments, and orders. The Measure Analysis Wizard provides a list of all stored measures that have an "Insertable" measure property set to true (see the *RPAS Configuration Tools User Guide* for more information on measure properties). The user simply chooses the measures to be displayed in the new workbook.

Measure Analysis Wizard

The Measure Analysis Wizard guides the user through the process of creating a new Measure Analysis workbook in which the user can view selected measure data.

Measure Analysis Worksheet

The Measure Analysis worksheet allows the user to view the chosen measure data for the positions selected from the measure's associated hierarchies. Each Measure Analysis worksheet is displayed at a different dimensional intersection, depending on the measure selections made in the wizard. This dimensional intersection is shown in the worksheet title bar.



Example of Measure Analysis Worksheet

The example above shows a Measure Analysis worksheet that displays Weekly Sales data for several items in a particular store. The location/product/calendar dimensional intersection of this worksheet, as shown in the title bar, is STR (Store), ITEM, WEEK. The Weekly Sales measure, because it is registered as a read/write measure, can be edited in this worksheet. However only an administrative user can commit overwrites to writeable measures in this workbook.

Access Measure Analysis

1. Select **Open** from the File menu to bypass the Measure Analysis wizard and open an existing Measure Analysis workbook, or select **New** from the File menu.
2. On the Analysis tab, select **Measure Analysis**.
3. Click **OK**.

Review and Edit Sales or Other Registered Measure Data

1. To open an existing Measure Analysis workbook: select **Open** from the File menu, double-click on the workbook to be opened, and go to step 8.

   OR

   To open a new workbook, select **New** from the File menu.
2. On the Analysis tab, select **Measure Analysis**.
3. Click **OK**.
4. The Measure Analysis Wizard opens and prompts the user to select the measures to be displayed in the new workbook. Use Ctrl-Click and/or Shift-Click to select multiple measures.
5. Click **Next**.
6. For each hierarchy specified in the base intersection of the measure(s) selected, the user will see a hierarchy wizard from which to select positions to view. Repeat this step for each hierarchy wizard.
7. Click **Next**.
8. Click **Finish** to open the Measure Analysis workbook.
9. On the Measure Analysis Worksheet(s), view the stored data associated with the measures and hierarchy positions selected in the wizard. Make any changes as required. Administrative users may commit changes.

6

Workbook Auto Build Maintenance

Overview

The Workbook Auto Build feature allows users to set up workbook builds to take place on a regular basis during nightly batch runs. Workbooks to be built in this way are added to the auto build queue. Because the workbook build process is automated, users are spared the processing time required to regularly enter the same wizard selections each time a new workbook is built. And because the build process occurs overnight, users are spared the wait time associated with constructing new workbooks.

The Workbook Auto Build feature works through the Workbook Auto Build Maintenance Wizard.

Workbook Auto Build Maintenance Wizard

The Workbook Auto Build Maintenance wizard steps the user through the processes of adding and/or deleting workbooks from the auto build queue.

Accessing the Auto Build Maintenance Workbook

1. Select **New** from the File menu.
2. Select the **Administration** tab.
3. Highlight **Auto Workbook Maintenance**.
4. Click **OK**.

Add a Workbook to the Auto Build Queue

Workbooks in this queue are designated to be built automatically on a specified regular basis as part of the nightly batch run.

1. Select **New** from the File menu.
2. Select the **Administration** tab.
3. Highlight **Auto Workbook Maintenance**.
4. Click **OK**.
5. From the task list, select **Add Workbook**.
6. Click **Next**.
7. Select a workbook template type.
8. Click **Next**.
9. Select an owner for the workbook.
10. Click **Next**.
11. Fill in the workbook **Build Label**, the **Build Frequency** (in days) with which the workbook should be built, and the **Next Build Date**.
12. Specify the Saved Access for the workbook: select **User**, **Group**, or **World**.
13. Select the group that owns the workbook. Choose from the list of groups to which the user belongs.
14. Click **Next** to initialize the wizard for the workbook template selected in step 5 above. The choices made are saved under the name specified for the Build Label.

Delete a Workbook from the Auto Build Queue

1. Select **New** from the File menu.
2. Select the **Administration** tab.
3. Highlight **Auto Workbook Maintenance**.
4. Click **OK**.
5. From the task list, select **Delete Workbooks**.
6. Click **Next**.
7. Select the workbook or workbooks to delete from the auto build queue.
8. Click **Finish** to delete the workbooks from the Auto Workbook Build queue.

7

Internationalization

Internationalization is the process of creating software that is able to be translated more easily. Changes to the code are not specific to any particular market. Oracle Retail Predictive Solutions has been internationalized to support multiple languages.

This section describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated may include the following, among others:

- Graphical user interface (GUI)
- Error messages

The user interface for Oracle Retail Predictive Solutions has been translated into:

- French
- German
- Italian
- Japanese
- Korean
- Portuguese
- Russian
- Simplified Chinese
- Spanish
- Traditional Chinese

Translation Administration

Every product, location, and calendar position can be presented in multiple languages, as can messages presented through the client. However, before translated strings can be viewed in the client, the following processed must be followed to setup the environment to support multiple languages.

1. Build the domain(s) with the Multi-Language setting enable in the Configuration Tool properties.
2. To install the translated client .dll, copy the file **<*language*>.dll** to C:\Program Files\Oracle\RPAS Client\ (or user's client Path). A file named **foundation.exe** must be in the same directory. The **<*language*>.dll** contains translated user interface information—the menus, dialogs, buttons, etc.
3. By default, the language of the RPAS Client is determined by the language of the client-side operating system. You can override this behavior by setting the Language entry in the Options section of the **foundation.ini** file, like this:

   ```
   [Options]
   Language=10
   ```

If the RPAS Client cannot find the library for the language you specify, it will default to English. The following languages are currently released:

Language	Code
English	9
French	12
German	7
Italian	16
Japanese	17
Korean	18
Portuguese	22
Russian	25
Simplified Chinese	2052
Spanish	10
Traditional Chinese	1028

4. Next, place the solution and RPAS .ovr files for the language selected into the 'input' directory of your domain. The input directory should also contain the lngs.dat file that will be loaded prior to loading the .ovr files. If not packaged with the language .ovr files, the current lngs.dat file will be in the "input/processed" folder of the domain. Move it back to the 'input' folder and remove the timestamp extension. Then reload the hierarchy using `loadHier -d /path -load lngs`. Make sure that the language you are testing exists in the lngs.dat file before loading it. Load each .ovr file within the input directory by using the `loadmeasure` utility (for example, `loadmeasure –d /path –measure r_msglabel`).

   Note: The files will be removed from your 'input' folder when they are loaded. So, any that are left in there still need to be loaded.

   Note: For Japanese, Korean, Chinese Simplified/Traditional, Russian, you will need to reboot your machine in each of these languages to properly see text.

5. In **Regional Options**, set the default language to match. On Windows 2000, you set both **Your Locale**" and **Set Default** under **Language** settings for your system. On Windows XP, set the **Standards and formats** list to the desired language. Set **Location** to the appropriate country. Click on the **Advanced** tab and set **Language for non-Unicode programs** to the appropriate language.

Translation Administration Workbook

The Translation Administration workbook contains worksheets for translating text used in measure labels, workbook template names, template group names, user group labels, and general areas (for instance, wizard instructions, and error messages).

Note: RPAS and solution-specific messages to the user should not be modified. If changes are made to these messages they may be overwritten when patching occurs.

Hierarchy Labels Worksheet

The Hierarchy Labels worksheet allows the user to view and edit the translations of hierarchy labels. Translations are supported for each of the system's allowable alternative languages.

Dimension Labels Worksheet

The Dimension Labels worksheet allows the user to view and edit the translations of dimension labels. Translations are supported for each of the system's allowable alternative languages.

Workbook Template Group Labels Worksheet

The Template Group Translations worksheet allows the user to view and edit the translations of template group names. Translations are supported for each of the system's allowable alternative languages. Translations in this worksheet affect the labels on the tabs that appear in the File – New dialog (for example (in English), Administration, Analysis, and Predict).

Workbook Template Labels Worksheet

The Template Translation worksheet allows the user to view and edit the translations of workbook template names. Translations are supported for each of the system's allowable alternative languages.

Measure Labels Worksheet

The Measure Translations worksheet allows the user to view and edit the translations of measure labels. Translations are supported for each of the system's allowable alternative languages.

Measure Descriptions Worksheet

The Measure Descriptions worksheet allows the user to view and edit the translations of measure descriptions. Translations are supported for each of the system's allowable alternative languages.

User Group Labels Worksheet

The User Group Translations worksheet allows the user to view and edit the translations of user group labels. Translations are supported for each of the system's allowable alternative languages. The list of user groups includes the Administration, Default, and Internal user groups, plus any other user group names set up by the system administrator. For products in the Oracle Retail Predictive Planning Suite, the list of user groups also includes the various planning roles.

Message Labels Worksheet

The Message Labels worksheet allows the user to view and edit the translations of messages displayed to users in the RPAS Client. Translations are supported for each of the system's allowable alternative languages.

RGRP Labels Worksheet

The RGRP Labels worksheet allows the user to view and edit the translations of rule group labels displayed to users in the RPAS Client. Translations are supported for each of the system's allowable alternative languages.

8

Commit as Soon as Possible

Overview

Commit As Soon As Possible (Commit ASAP) allows users to schedule the commit process of workbook data so that it executes as soon as all the system resources are available. Commit ASAP is an option in the File menu of the RPAS Client. Procedures for using Commit ASAP are provided in the *RPAS User Guide*.

Commit ASAP takes a copy of the data to be committed. Unlike Commit Later, which adds a workbook commit process to a queue that is run in batch, the data that is eventually committed is the data that was present at the time the commit instruction was issued. With Commit Later, if the user makes further changes to the workbook and saves that workbook before the batch commit process is run, those changes will also get committed.

Using Commit ASAP

After attempting to commit a workbook using Commit ASAP (File\Commit ASAP), the user will see a message in the client that the workbook has been scheduled for a commit. The user can continue with his/her work. The system will then try to commit the workbook as soon as it can, taking into account any other scheduled commits. If the commit cannot be done prior to the domain's Commit ASAP deadline, it will be canceled and listed as failed.

There are four states for commit processes to be added to the Commit ASAP queue.

- Pending – The commit process is queued up to take place at some point in the future.
- Committing – The workbook is currently being committed.
- Success – The commit succeeded.
- Failed – The commit failed.

The status of each commit ASAP process can be viewed by using a dialog window called "**Commit Status**" from the File menu. This dialog window displays all of the Commit ASAP processes with their respective status for all processes that have not been purged (see below). This dialog can be used to sort the tasks based on any of the columns.

Entries can be filtered in a variety of ways. If the checkbox **All Users** is not checked, the user will only be able to view his/her entries. If it is checked, you will see the entries for all users. The checkboxes in the **Status To Display** group allow you to filter the output so that the user sees only the processes with the specified statuses. The window can be updated by using the **Refresh** button. The dialog remembers the settings based on the last use.

Note: If a user attempts to commit a workbook ASAP that already has a process in the queue, the original processes will be removed from the queue. That means that there can only ever be one pending commit ASAP in the queue for a given workbook/user/template name combination.

Note: Workbooks must have been saved at least once before attempting a Commit ASAP. A workbook has not been saved if the label says "untitled."

Managing the Workbook Queue – showWorkbookQueues

The RPAS utility showWorkbookQueues is used for viewing the status of Commit ASAP processes and for purging entries in the Commit ASAP status window. The usage of this utility follows below.

The purge option requires a date before which entries will be removed, as well as specification for which entries to remove: succeeded, failed, or both.

Usage

```
showWorkbookQueues -version
showWorkbookQueues -d domainPath –show
[all|pending|waiting|working|success|failed]*
showWorkbookQueues -d domainPath -purge date [success | failed]*
```

The following table provides descriptions of the arguments used by the showWorkbookQueues utility.

Argument	Description
–version	Prints the RPAS version, revision, and build information of the utility.
-d domainPath	Specifies the path to the domain.
-show	Lists the contents of the queue in the order in which the parameter is specified. Possible values: all, pending, waiting, working, success, and failed.
all	Used with the –show parameter. This lists all of the workbooks in all statuses.
pending	Used with the –show parameter. This lists all workbooks that are waiting to be committed.
waiting	For Oracle Retail development use only.
success	Used with the –show parameter. This lists all workbooks that have been successfully committed.
failed	Used with the –show parameter. This lists all workbooks that did not successfully commit.
-purge *date*	Purges entries in the Commit ASAP status window. Entries before the date provided will be removed. The date should be a string of the following DateTime format: YYYYMMDDHHmm For example "200406071529" equals June 7, 2004 3:29 PM. Administrator must select to purge commit processes that either succeeded or failed.

Commit ASAP Settings – configCommitAsap

There are two settings for Commit ASAP that are managed by an administrator. Both are set using the utility `configCommitAsap`.

- Maximum number of simultaneous commit processes (property `MaxProcesses`, default value is 4).
- Deadline for which all pending processes must be completed, after which they will be cancelled and marked as failed.

  This deadline will likely be used by administrators before beginning nightly batch processes (property deadline, default value is 00:01 [meaning 12:01 AM], in 24-hour time).

  A commit process that starts before the deadline is reached will be processed. Commit requests that were in the queue before the deadline that did not get processed will be cancelled and marked as failed. Commit requests added to the queue after the deadline will use the deadline of the following day.

Usage

```
configCommitAsap -d pathToDomain [-maxProcs numProcs]
[-deadline time] [-display]
```

The following table provides descriptions of the arguments used by the `configCommitAsap` utility.

Argument	Description
`–version`	Prints the RPAS version, revision, and build information of the utility.
`-maxProcs` *`numProcs`*	Sets the maximum number of concurrent commit processes where `numProcs` is an integer greater than 0. Workbooks can be committed in parallel if they do not require access to the same measure databases. If they do share databases, they will be committed sequentially.
`-deadline` *`time`*	The time of the day when all outstanding commit ASAP operations will timeout. If a commit ASAP operation is submitted after this time, it will not timeout until the deadline time on the next day. This string must have the following format: HH:MM For example “13:30” refers to 1:30 PM.
`-display`	Displays the current commit ASAP settings.
`-loglevel` *`level`*	Use this argument to set the logger verbosity level. Possible values: `all`, `profile`, `information`, `warning`, `error`, or `none`.
`-noheader`	To disable timestamp header use.

Logging and Technical Information

A log file is available in the Commit ASAP directory that should be checked if a user reports an error with a Commit ASAP submission. The file is named **rpasServer.log** and is in the following directory: <Path to domain>/commitAsapQueue.

Another log file is generated for each Commit ASAP process and stored in a user's directory (users/<userid>/asapLogs). The format of the log file name is "orig_<original workbook name>asap_<temporary workbook name>.log." RPAS creates a temporary workbook in this process to capture the snapshot of the data that needs to be committed. Temporary workbooks are never viewed by a user. An administrator can use this log if something does not properly commit.

> **Note:** These "snapshot" workbooks cannot be viewed or used in the RPAS Client.

An example of this log file is orig_t1_asap_t5 where "t1" is the name of the original workbook and "t5" is the name of the snapshot workbook.

The following directories are used to store the copies of the workbook as they are processed through the system:

- **pending** directory – Contains one file per submitted commit ASAP that has not yet been processed. These files are, in general, binary and cannot be easily read.
- **working** directory – Contains one file per submitted commit ASAP that is currently in the commit process.
- **success** directory – Contains one file per submitted commit ASAP that has successfully completed its commit process.
- **failed** directory – Contains one file per submitted commit ASAP that either had a failure during its commit process or could not be committed prior to the deadline.
- **unknown** directory – If the Commit ASAP process detects a corrupted queue file, a message gets logged and the file gets moved into the unknown directory.

9

Batch Processes and RPAS Utilities

Overview

Included with an RPAS installation is a collection of stand-alone executables and scripts that are used for a variety of operations. RPAS utilities are run directly against a domain. If in a Global Domain environment, most utilities can only be run on the master domain. RPAS utilities can be categorized into the following groupings:

- Hierarchy management – the loading and refreshing of hierarchies, and the process of updating the data structures in the domain to reflect hierarchy changes
- Measure data – utilities for loading, exporting, and moving data within and between domains
- Miscellaneous – a variety of utilities for performing certain procedures in batch and for setting a number of parameters on an environment/domain
- Information RPAS utilities – a variety of utilities that retrieve information about a domain, data, the RPAS Server code, or an object used by the server

RPAS Utilities Logging Options

RPAS has a number of applications used to control or process data. Currently there are no unified methods for logging output, controlling the level of logging, or directing logging to a particular file. Instead each utility has its own methods, although many are similar. The current behavior for each utility follows.

Log Levels

This is a list of the standard log levels, controlled by the `-loglevel` option. Not all programs use these levels, but most do. Default logging level is "Warning," which means that any log messages that are specified as a warning or higher will be output:

- All – Forces all log levels to be output
- Profile – Performance profiling information
- Audit – User-specific domain and workbook activities. These activities include the following:
  - Workbook build, calculation, save, commit and custom menu operations
  - User login and logout to domain
- Information – General status messages that are not problematic. Outputs status and progress of the operation, in addition to the error and warning messages.
- Warning – Messages indicating a potential problem, but not one that is fatal. Outputs warning messages, in addition to error messages.
- Error – Messages relating to a fatal problem. Outputs only error messages.
- None – No messages. There should be no output if the utility successfully executes.

Each of the lines that contain the above types of feedback is normally preceded with a code that indicates what type of information is being output. Each code should have an angle bracket ("<") in front of it. E indicates the message is an error. W indicates the message is a warning. I indicates the message is informational.

Note: Audit information related to workbook activities gets recorded in rpas.log under each user's working directory. Information related to domain activities, such as user sign-on and sign-off, gets recorded in DomainDaemon.log.

Utilities with Standard Logging

A number of utilities allow for the `-loglevel` option to control which messages are output to the screen. There is no way to log to a file directly. The table below displays the utilities that can use the `-loglevel` option.

- **alertmgr**
- **checkDomain**
- **checkParents**
- **configCommitAsap**
- **createdb**
- **createGlobalDomain**
- **dattrmgr**
- **dbdiff**
- **dimensionMgr**
- **domaininfo**
- **ldrule**
- **listDb**
- **mapData**
- **moveDomain**
- **printArray**
- **printMeasure**
- **regfunction**
- **regmeasattr**
- **regmeasure**
- **regtemplate**
- **regTokenMeasure**
- **reguserdim**
- **rpasverison**
- **rtkappcnfgmeas**
- **showWorkbookQueues**
- **syncNAValue**
- **updateArray**
- **updatestyles**
- **upgradeDomain**
- **usermgr**
- **wbmgr**

Scripts

Shell scripts cannot use standard logging, but may execute the following programs that use it:

convertDomain

All output to the screen.

createRpasDomain

The –v option controls the type of messages sent to the screen.

Utilities with Special Logging

These utilities may use standard logging with additional features, or may use entirely different logging methods.

DomainDaemon

The `DomainDaemon` uses standard logging. Logs output to a file (see below). The file is created either in the current working directory or in the directory specified by the "RPAS_LOG_PATH" environment variable.

The file name depends on the "RPAS_LOG_BACKUPS" environment variable. If it is set to 1 or greater, then:

The log file name is "Daemon_Dyyyymmddhhmmbxx.log" where "yyyy" is the current year, "mm" is the current month, "dd" is the current day, "hh" is the current hour, "mm" is the current minute and "xx" is some number used to make the file name unique.

The number of these log files will be limited to the number provided in the environmental variable "RPAS_LOG_BACKUPS".

Otherwise, the log file name will be "Daemon.log". Any existing log file is renamed to "Daemon.old".

At midnight the current log file is closed and a new one opened, with naming as above.

RpasDbServer

This should only be started from the DomainDaemon as a part of a client request to start an RPAS session. The logging level is controlled by the client's "RPAS_LOG_LEVEL" environment variable. If not set then it defaults to logging messages at the "warning" level.

This utility creates log files in the "domain/users/client" directory, where domain is the current domain path and client is the current client. The actual file name used will be either "rpas Dyyyymmddhhmmbxx.log" or "rpas.log" base on the environmental variable "RPAS_LOG_BACKUPS" (c.f. DomainDaemon, above).

loadHier

The `loadHier` utility uses standard logging. This utility performs part of its processing in child processes, see the entry for `reshapeArrays` as well. Any log messages generated by `reshapeArrays` will go to the log file "reshapeArrays.log" in the current working directory. `loadHier` provides a list of all hierarchy positions that have been changed since the previous hierarchy load. The resulting directory name is:
"<utility><YYMMDDHHMISS><pXXXXX><bYY>"

where utility is the name of the program (for example, `-loadmeasure`), followed by a time/date stamp, then the process id (pXXXX), and then a 2 digit number to avoid conflicts (bYY).

locked

Messages are sent only to the screen.

mace

The `mace` utility uses standard logging. The `-debugRuleEngine` option logs some messages to the file "mace.log" in the current working directory.

positionBufferMgr

Uses standard logging. The `reshapeArrays` process is spawned as a child. See its entry for details.

reconfigGlobalDomainPartitions

Uses standard logging. The `reshapeArrays` and `loadHier` processes may be spawned as children. See their entries for additional details. When the `loadHier` utility is started as a child process it remaps the screen output of to the log file "loadHier.log" contained in the current working directory.

renamePositions

Uses standard logging. The `-log` option overwrites the default log file name of hierName and "Rename.log" in the current working directory. The `-loglevel` parameter does not control the types of messages written to this log file.

regmeasureServer

This application should only be started from the RPAS libraries to process measure registration/deregistration. Each process creates a log file in a newly created directory in the domain "output" directory. The newly created directory name will be formatted as "regServerYYYYMMDDHHMIbXX", where YYYY is the year, MM is the month, DD is the day, HH is the hour, MI is the minute, the character 'b', and XX is two digits used to make the directory name unique. The RPAS libraries will attempt to create at most eight directories for any single application.

Utilities with Multi-Process Logging

Some utilities are based on the multi-processes domain utility framework. These utilities send messages to the screen and a log file "master.log". Any child processes output messages to a log file in the domain/output directory named "subdomain0000.log" where the number indicated the sub-domain being processed. This directory will contain all log files created during the run of that utility. This change has been updated so that the controlling process logs to the screen as well as to a file in that directory. The newly created directory name is formatted as "APPNAMEYYYYMMDDHHMIbXX", where APPNAME is the utility name, YYYY is the year, MM is the month, DD is the day, HH is the hour, MI is the minute, the character 'b', and XX is two digits used to make the directory name unique. The framework will attempt to limit the number of directories created for any single utility to eight. The parameter `-loglevel` can be used to control the type of messages send to the screen and log file.

These utilities are as follows:

- defrag
- exportData
- loadmeasure
- reshapeArrays
- reconfigGlobalDomainPartitions
- updatedpmpositionstatus
- copyDomain
- wbbatch

domainprop

The `domainprop` utility only provides logging to the screen.

hierarchyMgr

The `hierarchyMgr` utility only provides logging to the screen.

configCommitAsap

This utility should be started from the `RpasDbServer` application when the client requests a workbook to be committed.

Using Shell Scripts to Run Batch Processes

Batch processes should be written using scripts that call the RPAS 11 binaries found in the $RPAS_HOME/bin/ directory. Any log files generated by scripts will be in the [DOM]/scripts/err/ directory. Examples of tools include Korn shell, Python, and Perl.

A Sample Shell Script

The following is a sample shell script that loads the product and location hierarchies into a domain. It is assumed that this script is invoked from the [DOM]/scripts/ directory.

```
1  #!/bin/ksh
2  loadHier –d .. –load prod > ./err/loadhier.prod.log
3  loadHier –d .. –load loc >> ./err/loadhier.loc.log
```

Line 1 defines the shell that will execute the script. In this example, it is defined to be the Korn shell. Therefore, this script will always be executed from the Korn shell even if the user's shell is different.

Lines 2 and 3 call the `loadHier` utility to load the latest product and location hierarchy information. Depending on the batch process to be performed by the shell script, lines 2 and 3 can be replaced by one or more lines to call one or more RPAS utilities.

Common Information and Parameters for RPAS Utilities

A number of standard arguments are available for most RPAS utilities. Check the usage of a specific utility to verify whether or not it is available.

Argument	Description
`-version`	Use this argument to get the version information of the utility (for instance, RPAS 11.2.0). It does not require `-d domainPath`.
`-d` *`pathtodomain`*	Common to most utility this specifies the path to the domain against which the utility will run or from which data will be used.
`-loglevel`	See "RPAS Utilities Logging Options" for more information.
`-n`	Certain utilities contain this parameter to perform a dry run. Using this option will show the administrator what would change, but makes no actual changes to the system or data. See the usage of a specific utility to see whether this option is applicable.
`-noheader`	To disable the use of a timestamp in the header of the log file.

`-help`
`-?`
`-usage`
	Any of these arguments will output the utility information and syntax to the terminal window. This can also be accomplished by running the utility with no arguments.

Logger verbosity levels determine how much information is generated on the terminal when running a given utility. An administrator can set these levels for each RPAS utility. The available logger verbosity levels are as follows:

- none – There should be no output if the utility successfully executes
- error – Outputs only error messages
- warning – Outputs warnings in addition to error messages
- information – Outputs status and progress of the operation in addition to the error and warning messages
- all – Outputs all available information generated by the utility, including error, warning, and informational messages

Each of the lines that contain the above types of feedback are normally preceded with a code that indicates what type of information is being output. Each code should have an angle bracket ("<") in front of it. E indicates the message is an error. W indicates the message is a warning. I indicates the message is informational.

Configuration Tools Log Files

For the RPAS Configuration Tools, information is logged in the files **stderr.txt** and **stdout.txt**, which are located in the **bin** sub-directory of the Tools directory. If a problem with the configuration tools is encountered, send these two files to Oracle Retail Customer Care along with a description of the problem.

Error Files

Error files are usually generated during data loading. These files include a list of bad records followed by the total number of records read and related information. Part of the output from a typical data loading batch job is as follows:

```
Loading array TEMP in nonoverlay mode. Zero values are loaded.
Skipping cell with invalid position:
    INFO   1
    DAY 1996D364
    SKU SKU_00726828
    STR STR_0107
Skipping cell with invalid position:
    INFO   1
    DAY 1996D364
    SKU SKU_00726828
    STR STR_0201
Skipping cell with invalid position:
    INFO   1
    DAY 1996D364
    SKU SKU_00726828
    STR STR_4008
Skipping cell with invalid position:
    INFO   1
    DAY 1996D364
    SKU SKU_00726828
    STR STR_4009
Skipping cell with invalid position:
    INFO   1
    DAY 1997D6
    SKU SKU_00726828
    STR STR_0107
Skipping cell with invalid position:
    INFO   1
    DAY 1997D6
    SKU SKU_00726828
    STR STR_0201
Array -- TEMP. Load time:   0:02
LoadComplete -- Records read: 4240, Total cell updates: 4240
  New cells created: 4028
  Cells with invalid positions: 212
  Cell updates: 0
```

10

Hierarchy Management

Overview

There are a number of key concepts and processes that are critical to the hierarchy management process:

- Hierarchy structures are loaded into a domain using the `loadHier` utility.
- The process of updating the data structures in the domain is commonly referred to as "reshaping." This process is handled automatically by the system.
- The length of position names is 24 characters or less by default. RPAS provides the ability to increase this length using the `dimensionMgr` utility.
- RPAS provides the ability to have placeholder positions in the domain that can be used when loading new hierarchy positions. Use of "dummy" positions defers the time consuming process of "reshaping" until a scheduled time, thus saving valuable time in the batch window for time-constrained batch processes.
- RPAS can automatically handle the movement of positions and their corresponding data between local domains when their parent-child relationships change and cause such a scenario. This requires the use of dummy positions and is only applicable in a global domain environment.
- Positions at the partition level in a global domain environment can be moved between local domains using the `reconfigGlobalDomainPartitions` utility.
- New local domains can be added to an existing global domain environment using the `reconfigGlobalDomainPartitions` utility.

Placeholder Positions in the Domain

RPAS provides the ability for a domain to contain "dummy" hierarchy positions, which are placeholder positions that allow new positions to be added to a hierarchy without having to update the data structures in the domain (process referred to as "reshaping"). These dummy positions are not visible to applications or users, so they cannot be seen in workbooks. They should not be confused with any form of 'dummy' or 'placeholder' positions that are part of a business process and visible to users, such as in the process of new store or item introductions.

RPAS provides the ability for any dimension, other than those in the calendar hierarchy, to contain dummy positions. The number of dummy positions is a percentage of total positions for a given dimension in the domain. For example, if the SKU dimension of the PROD hierarchy contains 1 million positions; a dummy position buffer of 1% will allow for 10,000 dummy SKU positions. When dummy positions are enabled for a dimension, positions have an 'internal' name (known only to RPAS and not visible to users or applications) and an 'external' name. Dummy positions are positions with an internal name, but no external name. Reshaping is required whenever the collection of internal names changes.

New positions are added to a domain through the hierarchy load process (`loadHier` utility). If a position is new, RPAS will map an existing unused dummy position to the newly added position so that it can be used in the domain without having to "reshape" the domain.

Similarly, as old positions are deleted, the external name for the internal position is removed from the mapping table (and data for the positions is removed from the arrays), and the position effectively become a dummy position. Therefore, deletes can happen without the need to "reshape" the domain.

As dummy positions are consumed, the number of available dummy positions will be reduced. Dummy positions are held in the "buffer," and the process of updating this is "re-buffering." The buffer can be updated automatically, but unpredictably, when required (based on the lower and upper bounds that are defined for a PNI dimension). It is recommended that the rebuffering process is scheduled to ensure that batch process windows are predictable. The manual rebuffering process is executed with the `positionBufferMgr` utility.

Position Repartitioning

Position repartitioning is the automated process of moving positions and all corresponding measure data between local domains. This functionality is only available (and relevant) in Global Domain environments. Positions need to be moved between local domains when they are assigned a new parent that exists in a different local domain. Note that moving positions at the partitioning level is a manual process and requires the use of the `reconfigGlobalDomainPartitions` utility.

For example, imagine Style1 belongs to Sub-Class1 in LocalDomain1. If Style1 is reassigned to be a child of Sub-Class2, which is located in LocalDomain2, RPAS will move the Style1 position, Style1's children (if any), and all corresponding data to LocalDomain2. This process is often referred to as "reclassification" by RPAS customers. RPAS refers to this functionality as "Position Repartitioning" because it technically does not handle the many complex functional requirements of true reclassification as most retailers define the term to mean.

Loading RDF and Curve Parameters after Repartitioning

After repartitioning, default parameters for Curve and RDF are not automatically loaded in new subdomain(s). To load these parameters, the following scripts, which are located in the RPAS_HOME/bin directory, need to be executed:

- loadCurveParameters.ksh – Used to load Curve parameters after a repartition.
- loadRdfParameters.ksh – Used to load RDF parameters after a repartition.

These scripts are used to load RDF and Curve parameters to a subdomain that was created as a result of repartitioning. These parameters are usually loaded during a full installation by the plug-ins, but when performing a patchinstall, the parameters are not loaded by default. These parameters include default required method, default source, spreading profile, and others.

You need to run these scripts after repartitioning a domain on the new partition.

Syntax

scriptname -d <full path to domain> -s <full path to subdomain>

Example:

```
loadRdfParameters.ksh -d /vol.nas/forecast/domains/RDF_12 -s /vol.nas/forecast
/domains/RDF_12/ldom0/
```

Enabling Dummy Positions

The use of dummy positions is enabled in a domain per dimension. Dimensions are enabled for dummy positions using the Configuration Tools both before and after a domain has been built. There are two properties in the Hierarchy Tool, and these properties can be set for each dimension except the calendar hierarchy. These properties are the "PNI Buffer Percent" for both the lower and upper bounds of dummy positions.

New positions are added to a domain by including new positions in the hierarchy data input files, and then running `loadHier`. Newly added positions will be immediately available for use unless all dummy positions have been consumed, which launches an automatic rebuffering process. Positions that have been assigned a new parent that require movement between domains will be automatically processed as part of the position repartitioning process.

Configuring and Scheduling the Rebuffering Process

Administrators need to determine the process by which they wish to rebuffer the dummy positions. Rebuffering can be completed automatically as part of the hierarchy load process when a domain runs out of dummy positions, or it can be scheduled using an RPAS utility.

If it is desired to have a predictable batch window, it is recommended that administrators schedule the rebuffering process rather than use the automated process. Scheduling rebuffering will minimize the possibility that the automated rebuffering process occurs during a critical batch process. The automated rebuffering would then only be used as a backup and run as an exception.

If customers do wish to reschedule the rebuffering, they will need to determine an approach that fits their business and batch processes. One approach might be to schedule all dimensions (with dummy positions enabled) in all domains to be rebuffered together on a weekly or monthly basis when there is a large amount of system down-time. Another approach could be to rebuffer a few domains on a cyclical basis, such as a few a day or week.

There are high and low settings for the dummy position buffer. Within the Configuration Tools, these settings are the **Buffer % Low** and **Buffer % High**. When executing a scheduled rebuffering process using the `positionBufferMgr` utility, the buffer for a dimension in a given local domain is updated when the number of dummy positions falls outside the high or low target buffer percentages. The number of dummy positions is calculated by taking the average of the high and low percentages multiplied by the total number of positions of the dimension in the local domain.

Deciding the buffer percentages will depend on a number of criteria. The goal is to have sufficient dummy positions to allow for growth in the local domains without having to execute an automated rebuffering process. Determining the targeted number of dummy positions will be a product of the anticipated growth in a given time period (for instance, 100 SKU's per week) and the frequency of the scheduled rebuffering process (for instance, rebuffering once a week or month).

These buffer settings and rebuffering processes are managed by the `positionBufferMgr` utility.

Loading Hierarchies – loadHier

The `loadHier` utility is used to load and refresh a hierarchy. All hierarchy data files are saved in fixed width (or space delimited) files with a .DAT file extension. The width of positions (number of characters) is specified in a configuration file before a domain is built. The width of positions can be increased after a domain has been built using the `dimensionMgr` utility or by changing a property in the Configuration Tools and patching the domain.

`loadHier` supports comma separated value (CSV) or fixed width flat files. The utility has also been enhanced to allow a simple compression method that can skip duplicated values line by line. To use a CSV file, instead of a fixed-width file, you must include .csv before the file extension.

Example of `loadHier` using a CSV File:

```
loadHier –load calendar.csv.dat
```

Due to the compression algorithms used in the CSV format, the `–nosort` argument is automatically used when CSV is detected.

> **Note:** The `exportHier` and `loadHier` utilities will skip any user defined dimensions. This is true for both fixed width and CSV formats. At this time, there is no way to export or import user-defined dimensions.

The hierarchy load utility also handles the process of "reshaping" data in the domain after adding or removing positions. This reshaping process is required to update the underlying data structures to reflect the hierarchical changes; however, note that the use of dummy positions can reduce the frequency of reshaping (see Enabling Dummy Positions for more information).

`loadHier` supports both the loading of hierarchy positions and purging data in parallel. When RPAS deletes a partition position through purging, RPAS adjusts the cache data in parallel to maintain the correct position/domain mapping.

RPAS allows for multiple input files to be loaded for the same hierarchy. The extra input files should be named with a secondary extension (for example, ' msgs.dat.1'). The extra input files can be loaded only with the main input file. For example, you cannot load 'msgs.dat.1' in a separate `loadHier` call. The primary use for this is loading hierarchy data for multiple languages. Hierarchy files are in the format *hier*.dat or *hier*.dat.*language* where hier is the name of a registered hierarchy, and multiple hierarchy files can have different language extensions. For example, a run of the `loadHier` utility would load **prod.dat**, **prod.dat.japanese**, and **prod.dat.spanish** if those files were available in the input folder. RPAS currently supports translations of the following languages: English, French, German, Spanish, Brazilian Portuguese, Japanese, Korean, Simplified Chinese, Traditional Chinese, and Russian.

RPAS automatically generates a backup copy of hierarchy files prior to performing a load for a hierarchy. If any type of error occurs during the load process, the hierarchy is restored from the backup copy.

Usage

loadHier –d *domainPath* –load *hierName* –loadAll {–purgeAge *purgeage*} {–checkParents} {–noClean}{–loglevel *level*} {-defaultDomain ldom#, ldom#}

The following table provides descriptions of the arguments used by the loadHier utility.

Argument	Description
–d domainPath	The domain in which to load the hierarchy data.
–load hierName	The name of the hierarchy to load and refresh.
–loadAll	Loads all hierarchy input files (with a .dat file extension) that are located in the input directory of the domain. Including this argument disables the reshaping process until all files have been loaded.
–checkParents	Verifies the one-to-one mapping of a child position being loaded to its parent position.
–purgeAge *purgeage*	Specifies the purgeage during loadHier.If not specified, loadHier gets purgeage from domain. In Global Domains, -purgeAge supports the purge of partition positions when the *purgeage* is reached.
–noClean	This is a switch that prevents the removal of input files and temporary data files that are generated during the hierarchy load process. Input files will remain in the "input" directory of the domain after the process is completed. This option is often used for debugging or troubleshooting purposes.
-noSort	Use this argument for instances when the RPAS process of sorting the hierarchy file prior to loading the values is not needed due to preprocessing (external to RPAS). This flag will prevent the input <hier.dat> file from being sorted while loading the hierarchy. This argument is often needed when loading calendar hierarchy data. This option can be eliminated if an alphabetic sort arranges calendar data in chronological order. Due to the compression algorithms used in the CSV file format, the –noSort option will automatically be used when CSV is detected. It is recommended that you presort the data before exporting it into a CSV format.
-purgeBackups	This argument prevents the reshapeArrays utility from backing up original databases to dbName.bak.
-logLoadedPositions	This argument will enable the logging of successfully loaded input file lines into a "loaded[HIERNAME].dat" file under the processed directory.
-maxProcesses *count*	If specified, some parts of loadHier will run in parallel, meaning that it will use a maximum of the defined processes, which are specified by *count*.

Argument	Description
`-forceInputRollups`	This argument enforces new hierarchy roll-up changes. New roll-up changes will override or dominate existing hierarchy roll-ups in the event they conflict with the rollups specified in the input file. This allows you to load a hierarchy file that reclassifies one or more upper level positions while removing one or more discontinued base-level positions that roll-up to the reclassified position.
`-forceNAConsistency`	Use this argument to force NA consistency when the current NA value is different from the originally defined NA value for the measure.
`-defaultDomain ldom#, ldom#,`	Use to specify comma separated default domain paths that will be used for accommodating new partitions. The domain paths can point to existing local domains or to new (non-existing) local domain. The local domain names are specified by a fully qualified path. To specify more than one local domain, separate local domain paths with a comma. **Example:** `loadHier –defaultDomain ldom1, ldom2, ldom3`

Notes

When using `–defaultDomain, loadHier` adds the new partition positions to the specified default domains one by one. The list of default domains is performed in the given order until each new partition positions is added.

Example:

Let's say we have a global domain that consists of two local domains, ldom0 and ldom1.

Let's assume we use the following `loadHier` command:

```
loadHier ... -defaultDomain ldom1,ldom2,ldom3 ...
```

Let's say that in this call, there are three new partition positions (part1,part2, and part3) in the input file. When the `loadHier` finishes, there will be two new local domains, ldom2 and ldom3 with the following new partition positions included in them:

```
ldom1 --> part1
ldom2 --> part2
ldom3 --> part3
```

In the previous example, if we only add two new partition positions (part1 and part2), when the `loadHier` finishes there will only be one new local domain - ldom2. New partition positions will be located as follows:

```
ldom1 --> part1
ldom2 --> part2
```

Using the same example, if we add 5 partition positions (part1,part2,part3,part4 and part5), when the `loadHier` finishes there will be two new local domains, ldom2 and ldom3. New partition positions will be located as follows:

```
ldom1 --> part1,part4
ldom2 --> part2,part5
ldom3 --> part3
```

Reconfiguring the Partitions of a Global Domain – reconfigGlobalDomainPartitions

It is common for many customers to regularly add, remove, or change the parent-child relationships for positions in hierarchies, most commonly for positions in the product hierarchy. While this movement/reassignment of positions is normally handled automatically within the `loadHier` utility, a special process must be followed for positions at the partition level of a Global Domain environment.

The RPAS utility `reconfigGlobalDomainPartitions` is used for the following activities in a global domain environment:

- Add a new position along the partition dimension and allocate it to an existing or new local domain.
- Remove an existing position from the partition dimension.
- Remove local domains (this is automatic if all partition-level positions in a local domain are removed or moved).
- Move an existing partition position from one local domain to an existing or new local domain.

Runs `loadHier` to apply the position addition/removal to hierarchy.

> **Note:** This utility can only be used on a master domain of a global domain set.

The following processes must be followed to add, remove, or move positions at the partition level in a Global Domain environment:

- The administrator must be notified in advance that positions at the partition level are being added, removed, or moved.
- The administrator should run the utility `reconfigGlobalDomainPartitions` to by specifying the sub-domain to which the positions do or will belong.
- This utility calls the `loadHier` utility at the end of the reconfiguration process to apply the hierarchy changes to the domain. When adding positions (using the `–add` argument) an updated hierarchy file must be available in the input directory when the `reconfigGlobalDomainPartitions` utility is called. Otherwise the utility will fail. Updated hierarchy files are not required to remove (using the `–remove` argument) or move positions (using the `–move` argument).

> **Note:** The use of this utility is only required for positions at the partition level. Positions below the partition level can be added, removed, or moved between local domains by loading a modified hierarchy input file with these changes.

Usage

```
reconfigGlobalDomainPartitions –d pathToMasterDomain –add posName1,posName2, ... –
sub pathToSubDomain
reconfigGlobalDomainPartitions –d pathToMasterDomain –remove posName1, posName2,
...
reconfigGlobalDomainPartitions –d pathtoMasterDomain –move posName1,posName2, ...
–sub pathToSubDomain
reconfigGlobalDomainPartitions –d pathToMasterDomain –input pathToInputDir
```

The following table provides descriptions of the arguments used by the `reconfigGlobalDomainPartitions` utility.

Argument	Description
–d *pathToMasterDomain*	Specifies the path to the master domain in a Global Domain environment.
–add *posName1, posName2, …*	Adds one or more positions at the partition level to a specified local domain. The path to the local domain must follow the list of positions to add, using the –sub argument. If the specified path is to a local domain that does not yet exist, the system will create a new local domain with the specified positions at the partition level. This argument cannot be used with –remove or –input.
–remove *posName1, posName2, ...*	Removes the designated positions from the local domain to which the positions belong. The path to the local domain does not need to be specified with this argument. The local domain will be deleted if all the positions at the partition level in a local domain are removed. This argument cannot be used with –add or –input.
–move *posName1, posName2, ...*	Moves the specified positions at the partition level from the current domain in which the positions are located to the specified local domain. This argument requires specification of the –sub argument. To move positions, all dimensions below the partition level must be enabled for dummy positions.
–sub *pathToSubDomain*	Specifies the path to the local domain to which positions are being added or the destination local domain for positions being moved. When a new domain path is specified using -sub option, a new local domain will be created. This argument is required for the –add argument and –move argument.
–input *pathToInputDir*	Specifies the path to the input directory that contains an xml configuration file (reconfigpartdim.xml) to specify positions to either add or move. The file must have all the information to run the process including the command name, position names to add or move, and paths to the local domains. This option is useful for adding or moving positions to multiple local domains. This argument does not handle both adding and moving in the same call. This argument cannot be used with –add or –remove.

Argument	Description
`-maxProcesses` *count*	If specified, some parts of `reconfig` utility will run in parallel, utilizing up to the given number of processes.
`-forceInputRollups`	This argument will prevent this utility from failing in instances where there is a roll-up conflict in the input file provided to the utility. This argument enforces new hierarchy roll-up changes such that they dominate existing hierarchy roll-ups in case they conflict with the roll-ups specified in the input file.

Using an Input File

When using the `-input` argument, the file must be in a particular format and must contain the "add" or "move" commands, the path to each local domain to which positions are being added or the destination for positions being moved, and the list of positions for each local domain. The file must be XML and named "**reconfigpartdim.xml**".

> **Note:** The `-input` argument only supports the addition or movement of positions.

Below is the required format of the input file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
    <rpas>
         <command name="command_name">
             <subdomain>
                <subpath>path_to_local_domain_1</subpath>
                <subpositions>sample_pos_1</subpositions>
            </subdomain>
            <subdomain>
                <subpath>path_to_local_domain_2</subpath>
                <subpositions>sample_pos_2,sample_pos_3</subpositions>
            </subdomain>
        </command>
    </rpas>
```

> **Note:** The entries in bold are the parameters that must be specified in the input file.

The following table provides descriptions of the arguments used by the input file.

Argument	Description
`command_name`	Valid values are "add" or "move."
`path_to_local_domain`	Path to the local domain to which positions are being added or moved.
`sample_pos`	One or more positions that are being added or moved to the designated local domains.

Notes, Assumptions, and Limitations

- Position names are separated by commas and must be valid **external** position names without the prefix of a dimension (e.g., 0102,0144,0152,0160).
- The utility backs up the required data and will automatically restore the domains to the original state in case of failure.
- In a single call to the utility without using the –input argument, positions can only be added or removed or moved. That is, the `–add`, `–remove`, and `–move` arguments cannot be mixed in the same call.
- Multiple positions can be added or moved to a single local domain in a single call to the utility using the `–add` or `–move` option, respectively.
- Multiple positions can be added or moved to multiple local domains in a single call to the utility using the `–input` option.
- When adding positions at the partition level, an updated hierarchy file must be available in the input directory when running this utility as the `loadhier` utility is called after adding positions. If the updated hierarchy file is not present in the input directory when attempting to add positions, the utility will fail.
- No updated hierarchy file is required when moving or removing positions. If a hierarchy file is in the input directory, the utility will back up this file.
- A log file (loadHier.log) will be created in the root directory if `loadHier` fails.

Updating or Reporting the Dummy Position Buffer – positionBufferMgr

The position buffer manager is used to manually rebuffer dummy positions in the domain or to report on the status of dummy positions. This utility can only be used if dummy positions have been enabled in the domain environment

Usage

```
positionBufferMgr -d pathtodomain [-rebuffer|-report] {-hier hierName}* {-
partitionPositions "pos1,pos2..."}
```

The following table provides descriptions of the arguments used by the `positionBufferMgr` utility.

Argument	Description
`–d` *`pathToDomain`*	Specifies the path to the domain.
`–hier` *`hiername`*	The hierarchy that will be rebuffered. If no hierarchy is specified, the utility will rebuffer or report on all hierarchies configured to have dummy positions other than calendar.
`–partitionPositions` *`“pos1, pos2, …”`*	By specifying positions at the partition level, the utility will only rebuffer local domains to which the specified positions belong. Positions specified that are not at the partition level will be ignored.
`–rebuffer`	Adjust all dimensions of the specified hierarchy to have the number of dummy positions based on the configuration of the dimension for the environment. If a hierarchy is not specified with this argument, the utility will run on all hierarchies other than calendar configured to have dummy positions.

Argument	Description
`–report`	Report the number of buffer positions for all dimensions of the provided hierarchy. The minimum buffer size is also displayed if applicable. If a hierarchy is not specified with this argument the utility will run on all hierarchies configured to have dummy positions other than calendar.
`-maxProcesses` *max*	If specified, some parts of `positionBufferMgr` will run in parallel, utilizing up to the given number of processes.

Notes

The minimum buffer size appears as a new column when `positionBufferMgr` is called with the `-report` argument. Also, the minimum buffer size is factored into the output indicating whether rebuffering is needed by `positionBufferMgr`.

The following example of output from `positionBufferMgr -report` where the STYL dimension needs to be rebuffered since it has 3 available positions but a minimum size of 4.

Example of `positionBufferMgr –report`:

```
Percent     Percent     Min         Available   Allocated   Dimension
Low         High        Size
0           0           0           0           5           CLR
0           0           0           0           10          CLSS
0           0           0           0           7           DEPT
0           0           0           0           1           DVSN
0           0           0           0           2           PGRP
0           0           0           0           12          SCLS
0           0           0           0           1           SDCD
0           0           0           0           4           SIZ1
0           0           0           0           5           SIZ2
25          35          0           6           24          SKU
0           0           0           0           12          SPLR
10          20          0           2           24          STCO
0           0           4           3           18          STYL should add 1
                                                            positions
0           0           0           0           5           SZ12
```

Renaming Positions – renamePositions

RPAS provides the ability to change the name of a position using an RPAS utility named `renamePositions`. Positions that are to be renamed should be included in a hierarchy data file that is located in a designated input directory (specified when running utility) and that follows the naming convention **hierName.rn.dat** (for instance, "prod.rn.dat").

After the hierarchy data file(s) has been updated and placed in the designated location, an administrator must run the `renamePositions` utility.

Usage

`renamePositions –d domainPath –i inputDirectory –hier hierName {–log logFileName} {–n}`

The following table provides descriptions of the arguments used by the `renamePositions` utility.

Argument	Description
`–d domainPath`	Specifies the full path to the domain.
`–i inputDirectory`	Input directory where input file with positions to rename is located. Utility looks for hierarchy data files with "rn" between hierarchy name and .dat extension (for instance, prod.rn.dat).
`–hier hierName`	Hierarchy for which positions are being renamed.
`–log logFileName`	Optional parameter to generate log file to file name other than default. The default file name is hierRename.log.
`–n`	Does not apply changes, but generates a report that identifies changes that would be applied.

Notes:

The `–n` is a dry run, which means that it does everything as a true run (for instance, writing to a log file) except that it does not actually commit the changes to the domain.

`–log` is an optional argument that can be used to name the log file other than the default, which will be created as hierRename.log in the current directory.

Input File

There will be three columns of data in each input line. These columns correspond to the dimension name, the old position name, and the new position name. The three fields will be tab-delimited. Any line not formatted this way will be ignored, and empty lines are also ignored.

Old position names must be an existing position name.

New position names cannot be an already used external name or an existing internal name. Lines having invalid position names will be ignored and added to the log file.

Old Position Name and New Position Name should not be prefixed with the name of the dimension.

The 'width' attribute in the domain must be greater than or equal to the max length of the new external names in the input file. If the width of the new name is greater than the width attribute of the corresponding dimension, RPAS will print an error in the log file and ignore the record.

Dimensions specified in the input file should belong to the hierarchy that is specified in arguments. Otherwise, the record will be ignored, and RPAS will print an error in the log file.

Setting Properties for Dimensions – dimensionMgr

The dimension manager utility is used for setting a number of parameters for dimensions and positions. These parameters are set when using the functionality of Position Name Indirection (PNI). This feature provides the ability to have position names that are longer than the default 24 characters and for a dimension to have dummy positions.

Usage

`dimensionMgr –d pathToDomain –dim dimensionName [COMMAND]`

The table below provides descriptions of the arguments used by the `dimensionMgr` utility.

> **Note:** This utility includes arguments that are not documented in this guide as it is recommended that those operations be configured using the Configuration Tools to ensure consistency between the configuration and the domain.

Argument	Description
`–d` *`pathToDomain`*	Specifies the path to the domain.
`–dim` *`dimensionName`*	Specifies the name of the dimension to which the settings will apply.
`–specs`	This argument displays the properties of the specified dimension. The dimension properties will indicate whether the DPM is enabled for the dimension.
`–width` *`widthVal`*	This argument sets the width of position names for the specified dimension. The default width for positions of a given dimension is 24 characters. Widths can only be extended; they cannot be decreased.
`-minBufferSize` *`minSize`*	This argument sets the minimum number of unused positions specified by *`minSize`*. Using this argument, you can assign an absolute minimum size to a dimension buffer. This feature is not available in the Configuration Tools.
`–bufPctMin` *`minVal`*	This argument sets the minimum percent of unused positions.
`–bufPctMax` *`maxVal`*	This argument sets the maximum percent of unused positions.

Argument	Description
`-enableDPM`	This argument enables Dynamic Position Maintenance (DPM) for the specified dimension (`-dim dimensionName`). In order to enable DPM for a dimension, the specified dimension (and all dimensions that roll up to it) must be PNI enabled, meaning that buffer percent minimum and buffer percent maximum are assigned non-zero values. Using this argument not only enables DPM for the specified dimension, but for all dimensions that roll up to it. DPM cannot be enabled for any dimension in an RPAS-internal hierarchy (DATA, META, RGPS, ADMU, ADMW, MSGS, LNGS).
`-enableTranslation` *`width`*	This argument enables the specified dimension to use translated labels. When enabling translated labels, the numeric parameter passed in the argument (`width`) defines the field width for the translated values in the data file, which is loaded using the `loadMeasure` utility.
`-maxProcesses` *`max`*	If specified, some parts of `dimensionMgr` will run in parallel, using up to the given number of processes defined by `max`.

Notes

To force the `dimensionMgr` utility to add positions to a dimension, both the `–bufPctMin` and `–bufPctMax` must be set. As long as minimum remains at zero, setting a higher maximum has no effect.

Multiple command arguments are allowed.

Buffer Percent Minimum Size and Buffer Percent Maximum Size are specified as a percentage of the total size of the dimension. For example, a dimension with 200 real positions and a Buffer Minimum of 5 and Buffer Maximum of 20 could have between 10 and 40 extra buffer positions at any given time.

If buffers are defined, then the Buffer Maximum must be greater than the Buffer Minimum and less than 10000. To turn off buffering for a dimension, set both Minimum Buffer and Maximum Buffer to zero.

If both the minimum buffer size and buffer minimum percentage are defined, then dimension is rebuffered to generate the appropriate buffer positions based on the higher value. For example, on a dimension with 100 currently in-use positions, if the minimum and maximum percentages were 10% and 20%, then about 15 available buffer positions would be created. However, if minimum buffer size of 25 is defined, then the dimension would get rebuffered to have 25 positions since that is the larger value. However, if the dimension then grew to 200 current in-use positions, the percentage target would grow to 30 buffer positions, and the 25 position minimum would no longer be relevant.

The primary advantage to the new minimum buffer size is that it works well for dimensions which have very few current positions (or even none).

The new minimum buffer size can be set in one of following ways:

- A new optional field is supported in the hierarchy.xml file at domain creation time as shown below:

  ```
  <buffer_size_min>20</buffer_size_min>
  ```

- Using the `-minBufferSize` *`minSize`* parameter, shown in the argument table above.

Deleting a DPM Position

When a DPM position is deleted, the underlying position is not automatically returned to the buffer positions. The "prepForBatch" batch job has to be run to redeem those buffer positions.

Setting Informal Positions to Formal – updateDpmPositionStatus

RPAS supports the ad hoc addition of non-calendar positions to the hierarchy by end users in the RPAS Client. This functionality is referred to as "Dynamic Position Maintenance" (DPM). Positions added outside of the `loadHier` process are assigned an "informal" status. `loadHier` ignores "informal" positions when they are present in input file. `updateDpmpositionStatus` changes the status of a position from "informal" to "formal" and enable loading/purging that position through `loadHier` utility and will disable further DPM activities on the position.

This utility cannot be run in subdomains in a global domain environment.

This utility can be run in parallel by specifying processes option.

This utility expects an input file named as dpmposupdate.xml in the input directory of the domain. Input file must be the XML file format as the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<rpas>
    <dimension name="dim1">
            <positions>pos1,pos2,pos3…</positions>
      </dimension>
      <dimension name="dim2">
        <positions>pos1,pos2,pos3…</positions>
      </dimension>
      <dimension name="dim3">
            <positions>pos1,pos2,pos3…</positions>
      </dimension>
</rpas>
```

Multiple positions belong to multiple dimensions can be specified using this input file in order to change their statuses from "informal" to "formal".

In a global domain the user will place an input file only to the input directory of the master domain. An input file for each subdomain is not required. The utility will split the input file and place it to each of the subdomains' input directories. Only the positions that reside in that subdomain will exist in that input file. The input file will be moved to processed directory under the input directory of the domain. The input file name will be appended with a timestamp.

Usage

```
updatedpmpositionstatus -d domainPath [{PARAMETERS}] {OPTIONS}
```

The following table provides descriptions of the arguments used by the updateDpmPositionStatus utility.

Argument	Description
–d *pathToDomain*	Specifies the path to the domain.
-processes *max*	When run on a Global Domain set defines the maximum number of processes to run in parallel.

Exporting Hierarchy Data – exportHier

exportHier is a command-line utility used to export all the positions in a hierarchy, including their rollup relations, from RPAS. By default, the utility assumes that the file will have a CSV flat file format with fixed width format as an optional argument. The utility will export all hierarchy positions, but the file may be specified to include only formal or informal positions. The resulting file could then be used as a .DAT file with loadHier.

Usage

```
exportHier -d domainPath –hier hier_name –datFile dat_file [-fixedWidth]
[-onlyFormal | onlyInformal]
```

The following table provides descriptions of the arguments used by the exportHier utility.

Argument	Description
–d *pathToDomain*	Specifies the path to the domain.
–hier *hier_name*	The name of a hierarchy in the domain from which the .DAT file will be generated.
-datFile *dat_File*	The path/location where the .DAT file will be created. This .DAT file can then be used with loadHier to load the hierarchy into a domain.
-fixedWidth	This argument indicates that the output .DAT file is a fixed-width file instead of a comma-separated value (CSV) file format.
-onlyFormal	This argument exports only formal positions to the .DAT file. If this is option is specified, informal positions will be skipped.
-onlyInformal	This argument exports only informal positions to the .DAT file. If this is option is specified, formal positions will be skipped.

Note

The exportHier and loadHier utilities will skip any user defined dimensions. This is true for both fixed width and CSV formats. At this time, there is no way to export or import user-defined dimensions.

11

Data Management

Loading Measure Data – loadmeasure

The `loadmeasure` utility is used to load data into the domain. The administrator must specify the measure name(s) and the path to the domain that contains the measure(s).

`loadmeasure` has been enhanced to support the load of measure data into RPAS using a CSV or fixed width flat file format. The utility has also been enhanced to allow a simple compression method that can skip duplicated values line by line. To use a CVS (Comma Separated Value) formatted file to load measure data, place ".csv" before the file extension.

Example: rsal.csv.ovr

In a Global Domain environment, `loadmeasure` is centralized and can only be called in the master domain. The utility will load one or more input files that can contain data from one or all of the local domains within the given Global Domain environment. The utility will split the input files and load them into the required domain, which is the local domain to which the position belongs or the master domain if the measure has a base intersection above the partition level. The split will only occur once in the case of multiple measures. Local domains will be checked for files even if there is no file in the global domain. The utility can be run in parallel in a global domain environment.

Usage

```
loadmeasure –d pathToDomain –measure measureName{,measureName,…} {–logdirectory
directoryName} {–applyloads} {-processes max} {-noClean} {–forcePurge}
{-splitOnly | -noSplit} {-defrag}{–loglevel level}
```

The following table provides descriptions of the arguments used by the `loadmeasure` utility.

Argument	Description
`–d pathToDomain`	Specifies the domain in which to load the measure.
`–measure measureNames`	Specifies the name of the measure(s) to load. Measure names must be lowercase (for example, measurename1,measurename2,measurename3,…). If more than one measure is specified, all the measures must be in the same input file.
`-applyloads`	Use this argument to apply any staged loads for the named measure.

Argument	Description
–processes *max*	Specifies the maximum number of child loadmeasure processes to run concurrently across the local domains in a global domain environment. If this argument is omitted or if less than two processes are specified, the application will do all processing in a single process and no child processes will be created. This only specifies the number of child processes and the controlling process is not included (*max* + 1 is the actual number of processes).
-noClean	Use this option to prevent the input files from being moved to the processed directory.
–forcePurge	Force the purge routine to run even if no new data is loaded. This purges old measure data.
-splitonly	This argument causes the input files in the global domain to be split across the local domains, but it does not do any further processing of the input files. The –noSplit argument can then be used to load these pre-split input files into the local domains. The –splitOnly option is mutually exclusive with the –processes argument. This option should only be used in global domain environments.
-noSplit	Use this argument to load the pre-split input files (created by –splitOnly) into the local domains. This option should only be used in global domain environments.
-defrag	This argument can be used to defragment the domain at the end of the measure loading process to reduce the physical size of the domain. This is achieved by copying the existing data files into files with fully populated BTree pages.
–loglevel *level*	Use this argument to set the logger verbosity level. Possible values: all, profile, information, warning, error, or none.
–logdirectory *directoryName*	Specifies the location of the output error log. The default location is pathToDomain/scripts/err.

Running Scripts

The loadmeasure utility also provides the ability to automatically run scripts before and after the utility is executed. These are referred to as "pre-processing" and "post-processing" scripts.

When loadmeasure is called, the utility checks for the existence of scripts named "pre<*measurename*>.sh" in the "../scripts" directory of the domain.

If scripts exist, they will be run prior to the execution of the utility. Likewise, after the utility has completed running, it checks for the existence of scripts named "post<*measurename*>.sh" and executes them immediately. When loading multiple measures in a single call only the preprocessing script for the first listed measure will have any affect on the data.

Notes

To use a CVS (Comma Separated Value) formatted file, place .csv before the file extension as shown in the example below.

Example: measName.csv.ovr

Loading Image Paths for Positions

A configuration and backend process may also be used to support the load of image paths for one or more positions of a dimension at a time. The paths of the images should be stored in a measure called "r_images_<dimension name>" where "<dimension name>" should be replaced with the RPAS Name of the image-enabled dimension (for example, "r_images_sku" if loading image paths for the "sku" dimension). This measure is single-dimensional, defined on the image enabled dimension. An .ovr file is required with position names and the image paths for those positions formatted according to the RPAS measure load formats. The `loadMeasure` utility is then used to load this data into the domain.

> **Note:** See the *RPAS Configuration Tools User Guide* and the "Position Images" section in the *RPAS User Guide* for more information on Image Display.

Example

```
loadmeasure –d <domain path> -measure r_images_sku
```

`<domain path>` is the path to the domain.

Exporting Measure Data – exportMeasure

`exportMeasure` is a command-line utility that may be used to export domain or workbook measure data from RPAS in either CSV or fixed width file format. A single measure, or multiple measures, may be exported based a specified intersection. If the measure's base intersection is not the same as the export intersection, the measure's default aggregation method will be used to aggregate data to an intersection higher than base, or replication will be used for spreading measure data if the data is required at an intersection lower than base. `exportMeasure` also allows a simple compression method that can skip duplicated values line by line.

This utility:

- Supports export data in a user specified range, which can be a single mask measure, or a range specified on Calendar dimension, or a combination of the two.
- Supports multiple processes for better performance in a global domain environment.

Usage

`exportMeasure –d` *`pathToDomain`* `–out` *`outFile`* `[COMMAND] [OPTIONS]`

The following table provides descriptions of the arguments used by the `exportMeasure` utility.

Argument	Description
`–d` *`pathToDomain`*	Specifies the path to the domain.
`-out` *`outFile`*	Specifies the output file name. It is required and must be a valid file name including the path.

Argument	Description
-meas *"measSpec, measSpec …"*	Must specify one. measSpec is measName.modifier. The –meas argument may be repeated to export multiple measure arrays to the same output file. modifier include the following: ▪ .precision<double>, specify the precision for numeric measure ▪ .format<formatString>, specify the user defined export format The examples below provide valid measure specifications given MeasNameA is a valid real type measure. **Examples:** -meas MeasNameA -meas MeasNameA.precision(0.0001) -meas MeasNameA.format("%13.2f").precision(0.01) -meas MeasNameA.precision(0.01).format("%13.2f") The examples below provide valid measure specifications given MeasNameB is a valid date type measure. **Examples:** -meas MeasNameB -meas MeasNameB.format("YYYYMMDD")
-wb *wbname*	If specified, exportMeasure exports data from the specified workbook (wbname). A valid workbook name must be used.
-intx *intxString*	Specifies the export intersection. Measures will be exported at this intersection. If measure's baseint is higher than export intx, measure's defaff method will be used for aggregation. If measure's baseintx is lower than export intersection, Replication will be used to spread measure down to export intersection.
-mask *measureName*	Specifies a mask measure which must be a valid Boolean measure registered. In current measure store, its baseintx must be at same export intx.
-range *start:end*	Used in conjunction with –wide parameter. Specifies a range along the innermost dimension. Only values in the range are considered for export.
-processes *max*	Defines the maximum number of processes to run in parallel.
-append	Appends new output to current output file. If not specified, current output file will be erased and replaced with new data.

Argument	Description
-nomerge	If run in a global domain environment and exporting intersection below partition dimension, and have processes set greater than 1, specifying nomerge will stop exportMeasure from merging multiple output files created from each local domain to the master output file. Output files created from local domain will be stored at masterdomain/output/exportMeasure[TS] folder, where [TS] represents a timestamp. Files are named as out000X.txt, where 000X is the index of the local domain.
-compress	Specifies the output file should be in the compressed CSV format.

Exporting Measure Data – exportData

Use exportData to export measure data from RPAS into text files. Each line that is exported contains the position name for the exported dimension followed by the value in the cell for each array being exported.

> **Note:** More than one array may be exported and more than one dimension in each array can be exported.

The utility may be invoked by specifying all parameters on the command line or by specifying an array that contains a list of the parameters.

When running this utility in a Global Domain environment, the utility should only be called to export data from the master domain. The utility will extract the data from either the local domains or the master domain depending on where the data resides, which depends on the level at which the Global Domain environment is partitioned.

The parameters specify what arrays and dimensions are exported and how to format the data. It is best to specify the arrays first. An array specification begins with –array followed by the array information. This includes the array name, formatting string, NA cell value, and NA cell value formatting string. The formatting string for both the cell value and NA value is based on the C language printf function formats. See the documentation on the printf for more information on the possible values. The –array parameter can be repeated as needed to export more than one array into the same export file. Remember that the order the arrays appear in the –array parameter is the order that they will appear in the export file.

After the arrays have been specified, the administration must specify the dimensions to be exported within the arrays. The –dim parameter is used to specify a dimension in an array. The –dim parameter is followed by the dimension name, a convert option, the formatting string (just like an array), and the order the dimension appears in the export file. Because arrays are not required to contain identical dimensions, it is important to list all dimensions in all arrays with the –dim parameter. This makes it possible to track dimensions across arrays and line the data up correctly. If a dimension in an array is not to be in the export file, set the last value of this parameter to 0. The conversion option specifies either the number of characters to be removed from the position name or it specifies an array that contains the real position name. If an array name is given, this array must be a vector. The function will go to this array and use the original position name to jump to the cell of the same position name. It will then get the cell value and use that as the position name in the export.

It is possible to specify the number of decimal places when exporting numeric measures of data type "real." This setting is defined in the specifications for measures, arrays, and dimensions (measSpec, arraySpec, and dimSpec). The format is %[.*precision*]*type* where [.*precision*] is the number of decimal places and *type* is the letter "f" (without quotes). For example, the setting *%.2f* would export numbers with two decimal places. Other settings are provided below.

If all parameters are contained in an array, after the export file name and source database name, the **-**-params parameter is used to specify the database name and array name that contains all of the parameters needed for the export.

> **Note:** Either the –array, –meas, or –params parameters must be specified when using this utility.

Usage

```
exportData –d domainPath –out outputFile –params db array
exportData –d domainPath –out outputFile –meas \"measSpec\"
{–wb wbName} {options}
exportData –d domainPath –out outputFile –array \"arraySpec\" {options}
```

The following table provides descriptions of the arguments used by the exportData utility.

Argument	Description
–d *domainPath*	Specifies the domain that contains the data that to export.
–out *outputFile*	Specifies the file that will contain the exported data. The *outputFile* is relative to the domain unless the full path is specified.
–meas \"*measSpec*\"	Specifies the measure(s) to export. *measSpec* must be quoted, and the format is \"measName cellFormat naValue naFormat\" The –meas argument can be repeated to export multiple measure arrays to the same output file. Measures are exported at the base intersection.
–array \"*arraySpec*\"	Specifies the array to export. *arraySpec* must be quoted, and the format is \"dbName|arrayName cellFormat naValue naFormat\" ▪ dbName can be a path to the database (relative paths are relative to the domain root). ▪ Both cellFormat and naFormat use printf format commands. See the printf function for more information on the possible values. The –array argument can be repeated to export multiple arrays to the same output file. The order in which arrays are listed is the order in which they will be exported. **Note:** This argument cannot be used in a Global Domain environment and can only be used in simple domains. This argument cannot be used with –useLoadFormat.

Argument	Description
–params *db array*	Instead of specifying all parameters on the command line, this parameter allows the parameters to be read from an array. ▪ *db* specifies the name of a Gem file where the array of parameters is stored. ▪ *array* specifies the name of an array in the specified database that has the above parameters.
–dim \"*dimSpec*\"	Specifies the dimension to be exported. ▪ dimSpec must be quoted, and the format is \"dimName *conversion format order*\" ▪ *conversion* is either a count of the number of characters to strip from the start of the position name or the name of an array to be used to translate the position name before writing to the output file. ▪ format is a printf-style format for the position names. See the printf function for more information on the possible values. ▪ order indicates the order the dimension is listed in the output file. If the value is 0, then the dimension is not exported. The –dim parameter can be repeated. The -dim parameter is not allowed with the -useLoadFormat.
–skipNA *always|allna|anyna|arrayna*	Controls whether a line of data is exported based on having NAs in a cell. ▪ *always* exports data regardless of whether or not it contains NAs. ▪ *allna* does not export a row of data if all columns are NA (default). ▪ *anyna* does not export a row of data if any cell contains a NA value. ▪ *arrayna* does not export a row of data if the value in the given array name is NA (requires –naArray).
–naArray *arrayName*	When *arrayna* is specified using the –skipNA parameter, this option specifies the export array that is checked to determine if data is exported.
–index *arrayName*	Controls whether arrays are indexed by looking at a specified array. Only export the non-NA cells in the given array and each cell in the other arrays that have the same position names. If another array is at a higher dimension level, translate the given arrays cell index to the other arrays.
–append	Specifies that output is appended at end of output file. The default is to overwrite output file.

Argument	Description
–wide	This parameter causes the data to be exported wide, which means the innermost dimension will go across the row instead of each cell on a separate line. This is most useful when the innermost dimension is time. This format puts all time data on one row of output with breaks for each of the other exported dimensions. The –range parameter can be used in conjunction with wide format (–wide) to specify a range along the innermost dimension. Only values in the range will be considered for export. In future versions ranges could be expanded to include the column format.
–range *start:end*	Used in conjunction with the –wide parameter, specifies a range along the innermost dimension. Only values in the range are considered for export.
–time	Specifies the YYYYMMDD format for dates.
–precision *precisionValue*	This parameter causes the utility to avoid exporting values that differ from the NA value by the specified value. Any values smaller than the precision value are not exported. For example, consider a measure with the NA value of zero and a precision value of 0.01. A value of 0.0034 would not be exported while a value of 0.34 would be exported. The precision value must be less than one. If a value greater than one is provided the utility returns a warning.
-processes *max*	Defines the maximum number of processes to run in parallel.
-useArrayNaValue	This argument will enable the use the NA value of the array instead of the NA value specified in measSpec or arraySpec.
-useLoadFormat	This argument enables the use of the format as specified by the measure property. The level at which the data is stored in the domain will be used. The -dim parameter is not allowed with the -useLoadFormat.

-useLoadFormat Parameter

Use the format specified by the measure's loading format to export the measure. This loading format includes Start and Width, which defines the column that corresponds to this measure's data in the measure load file. The measure will be exported into the same column in the output file. If the full measure export specs are not provided including the cellFormat, naValue and naFormat, the default format will be used. The default export format for each type of measure is listed below:

- Integer: %<width>.0f
- Real: %<width>f
- String: %<width>s
- Date: %Y%m%d
- Boolean: TRUE or FALSE as string

All value will be exported right aligned as in the measure loading file.

If users give a full measure specs, user-specified cellFormat, naValue, and naFormat will be used rather than the default format.

Users can either use the default format by specify the measure name only, or give the full specs. It is not allowed to give a partial measure spec.

If users specify multiple measures to be exported into the same file, these measures will each occupy a column in the file defined by its start and width attributes. If two measures occupy the same column, `exportData` will throw an exception with an error message saying "overlapping measures in the output file" and exit. If a measure's column is overlapping with the columns occupied by the position names, `exportData` will throw an exception with an error message saying "measure column is overlapping with position columns" then exit. Basically, if the measure cannot be exported correctly, `exportData` will not try to export it but simply exit and alert user with a proper exception.

The `-dim` and `–array` parameters are not allowed if `-useLoadFormat` is used. All dimensions in the measure's base intersection will be exported by default. The external position name will be exported to the export file, in the order specified by the hierarchy's order attribute, usually in the order of CLND, PROD, and LOC. The position names will be left aligned in the export file.

Mapping Data Between Domains – mapData

The `mapData` utility is used to move data from one domain to another. Specifically, it copies data from an existing domain, database, or array to a new domain, database or array.

Before running this utility, the new hierarchy must be loaded in the destination domain. After `mapData` has copied data, administrators can purge the source domain by calling `loadHier` with a purge age of 0. Tasks such as hierarchy loading, hierarchy purging, and the validation of source and destination domains are performed outside of this utility.

Note: This utility does not update buffer positions.

Usage

```
mapdata -d SrcPath -dest destPath [-db dbName [-array arrayName]]
        {-db dbName {-array arrayName}} {-loglevel}
```

The following table provides descriptions of the arguments used by the `mapData` utility.

Argument	Description
`–d SrcPath`	Specifies the path to the source domain.
`-dest DestPath`	Specifies the path to the destination domain.
`–db dbName`	Apply `mapdata` only on the given database. Must be a valid .gem file. If this argument is not specified the entire domain will be included in the operation.
`–array arrayName`	Apply `mapdata` only on the given array. The database in which the array resides must be specified with the `–db` argument.

Moving Data between Arrays – updateArray

The `updateArray` utility moves data from a source array to a destination array. The destination array must contain the superset of dimensions in both source arrays. The source array's dimensions may be at the same or higher level as mapped by the dimension dictionary. If a dimension in the source array is at a higher level, the results are spread across the lower level dimension in the destination. If there are extra dimensions in the destination array, the results are replicated across these extra dimensions. The NA value of the destination array remains unchanged.

To limit the scope of the update, a mask array and an innermost range may be specified. If a mask array is given, the update is limited to cells in the source array for which the corresponding mask cell is on. If an innermost range is given for source or destination array, the update is limited to cells that are within the start and end of this range on the innermost dimension. If the source and destination arrays are not in the same domain, the measure store associated with the source domain is used to find hierarchy information.

Note: This utility does not update buffer positions.

Usage

```
updateArray –destArray dbPath.arrayName {–srcArray dbPath.arrayName}
{–destDomain domainPath {–srcDomain domainPath} {–maskDomain domainPath} {–
maskArray dbPath.arrayName} {–updateMethod method} {–srcRange first:last} {–
destRange first:last} {–srcScalar scalarCell} {–version} {–loglevel level}
updateArray –argFile filename {–version} {–loglevel level}
```

The following table provides descriptions of the arguments used by the `updateArray` utility.

Argument	Description
`–destArray dbPath.arrayName`	Required argument to specify the destination array where the data will be copied. *dbPath* is relative to `destDomain`.
`–srcArray dbPath.arrayName`	Optional argument. Default is no source array. **Note:** This parameter cannot be used with `–srcScalar scalarCell`.
`–destDomain domainPath`	Optional argument. Default is current working directory.
`–srcDomain domainPath`	Optional argument. Default is current working directory.
`–maskDomain domainPath`	Optional argument. Default is current working directory.
`–updateMethod method`	Optional argument. Default is OVERLAY. The following update methods are available: ▪ SKIPNA – Omit NA cells in source. ▪ SKIPPOP – Omit populated cells in source. OVERLAYNA – Update NA cells in destination. ▪ OVERLAYPOP – Update populated cells in destination. ▪ OVERLAY – Update all cells in destination with source.
`–srcRange first:last`	Optional argument. Default is no range. Defines range along innermost dimension of source array.
`–destRange first:last`	Optional argument. Default is no range. Defines range along innermost dimension of destination array.

Argument	Description
`–srcScalar "TYPE:VALUE"`	Optional argument. Default is NA cell. Format for scalar cell is one of: ▪ NUMERIC: numeric value ▪ STRING: literal value ▪ BOOL: Boolean value ▪ NA **Note:** This parameter cannot be used with `–srcArray dbPath.arrayName`.

12

Operational Utilities

Find Alerts – alertmgr

Alerts are an exception management tool for users. An alert is a measure that evaluates a business rule (returning a value of true or false). RPAS then notifies users of the "true" conditions and allows users to build workbooks to resolve the scenario that drove the alert.

Alert measures are first defined in the domain using the Configuration Tools. These measures are of type Boolean, which means they have a value of true or false. Next, rules (expressions) are registered in the domain for the alert measures to define the business rules used to evaluate the alert.

Once the registration process is complete, the alert utility is run to "find" the alerts in the domain. After the alert finder has been run, the identified alerts can be viewed in the Alert Manager window in the RPAS Client.

The following is a summary of the process for defining and finding an alert:

1. Create an alert measure – This must be a Boolean measure (values are true-false, or yes-no) and must be defined in the RPAS Configuration Tools.
2. Create the alert (the expression) for which the alert should be evaluated using the Configuration Tools. This flags the registered measure as an alert so that it is recognized when the "alert finder" is run.
3. Run the "alert finder" on the domain to evaluate the number of instances when one or more alert expressions are true. This operation is completed using the RPAS utility `alertmgr`.

Usage

```
alertmgr -d domainPath [COMMAND [parameters]]
alertmgr –d pathtodomain –findAlerts {-alerts "a1 a2 …" | -categories "cat1 cat2 …"}
```

> **Note:** This utility includes arguments that are not documented in this guide as it is recommended that those operations be configured using the Configuration Tools to ensure consistency between the configuration and the domain.

The table below provides descriptions of the arguments used by the `alertmgr` utility.

Argument	Description
`–d pathToDomain`	Specifies the directory in which to run the utility. All commands except `–version` require `–d domainPath`.
`–findAlerts`	Finds alerts in the specified domain. The utility will find all alerts in the domain if neither the `–alerts` or `–categories` arguments are specified. If `–alerts` or `–categories` list is not specified, `findAlerts` is run on all alerts. `findAlerts` can be run from either Master or Local Domains.

Argument	Description
–alerts *a1 a2 …*	Evaluate specific alerts in the domain. *a1 a2 …* must be valid names of alerts that are defined in the domain.
–categories *cat1 cat2 …*	Evaluate all alerts in the domain that are associated with specific categories of alerts. *cat1 cat2 …* must be valid names of alert categories that are defined in the domain.
-sumAlerts	-sumAlerts sums up the hit counts of alerts across local domains. It can be run based on a list of alerts or alert categories. If none are provided, then the respective hit count of each alert across all local domains is summed. -sumAlerts can be used only from Master Domain. **Note:** -findAlerts must be run first to generate hit counts of alerts.

Note: alertmgr can be run on the local domains individually. The administrator may spawn several processes in parallel, and when needed, run alertmgr -sumAlerts again to aggregate the results to the global domain. If parallelization is desired, the administrator should create a script to spawn the parallel processes.

Copying Domains – copyDomain

The copyDomain utility is used to copy a simple domain, all domains included in a global domain environment, or a subset of domains in a global domain environment. Domains are often copied before upgrading the domains after receiving a patch to RPAS.

For a standard, simple domain (in other words, not a global domain environment), copyDomain copies the domain directory recursively from one location to another.

For a global domain environment copyDomain copies the master domain to the specified destination, and then it copies each local domain into corresponding subdirectories of the new location. As part of this particular replication process, the utility also updates all relevant data structures so that the domains are properly connected together.

Relative paths are supported with this utility and are used when creating the new copies of all the underlying data structures (arrays). Relative paths are based on the full pathname of the domain's root directory.

Usage

```
copyDomain –d  pathToSrcDomain –dest pathToDestDomain { -f }
copyDomain –xmlConfigFile pathToXmlConfigFile.xml
copyDomain –version
```

The following table provides descriptions of the arguments used by the copyDomain utility.

Argument	Description
–d pathToSrcDomain	Specifies the path of the domain to be copied. This argument and –dest should not be used with –xmlConfigFile.

Argument	Description
–dest pathToDestDomain	Specifies the path to where the domain is to be copied. The copied domain can also be renamed in this step by providing a name different than the source domain. This argument must be provided when using any other option (other than –xmlConfigFile) of the utility. If this argument is not provided then the domain is updated to have relative paths.
-xmlConfigFile pathToXmlConfigFile.xml	This argument will allow copyDomain to copy each sub-domain into user-instructed specific locations. This argument should not be used with –d OR –dest.
–force	Deletes the existing domain at the specified destination path before copying the source domain.
-clone dimposlist	Use this argument to copy a subset of a domain environment. Copies only positions specified in a format as dim1,pos1,..,posn:dim2,pos1,..,posn where the sequence dim1,pos1,..,posn specifies the selected positions along dim1. Multiple dimensions may be specified, but only one dimension per each hierarchy is allowed.
–partitionPositions *positions*	Limits the copying of local domains to the specified positions at the partition level; positions is a comma-separated list of positions at the partition level.
–copyWorkbooks *workbookList*	Copies only the specified workbooks to the destination location. *workbookList* is either a comma-separated list of the workbooks to copy, or the value “none” such that no workbooks are copied. If this argument is not specified all workbooks in the environment will be copied.
–skipInput	Do not copy the “input” directory located in the source domain.
–skipConfig	Do not copy the “config” directory located in the source domains.
–skipEmptyDir	Do not copy the empty directory located in the source domain.
-maxProcesses count	If this argument is specified, some parts of copyDomain will run in parallel, utilizing up to the given number of processes.
–noSubDomains	Do not copy any local domains in the source domain.
–export	Export each database (.gem file) from the source domain into a format that can be used on a UNIX platform. This argument cannot be used when specifying an –xmlConfigFile.

Argument	Description
-gzip	Compress the copied domain into a gzip format. This argument cannot be used when specifying an -xmlConfigFile.

Note: If the pathToSrcDomain argument is provided but the pathToDestDomain is not, then the utility will update the source domain environment to have relative paths to all the local domains. This is commonly used to update a global domain to have a recursive directory structure which is useful when copying a domain environment.

Format of the XML configuration file

The XML configuration file contains source and destination fields for the location of the master domain and each of the sub-domains. Here is a basic example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<rpas>
    <globaldomain>
        <srcPath>C:\usr\Rpas\Domains\GlobalDomain</srcPath>
        <dstPath>C:\usr\Rpas\Domains\GlobalDomain2</dstPath>
        <subdomain>
            <srcPath>C:\usr\Rpas\Domains\GlobalDomain\ldom0</srcPath>
            <dstPath>C:\usr\Rpas\Domains\GlobalDomain2\ldom0</dstPath>
        </subdomain>
        <subdomain>
            <srcPath>C:\usr\Rpas\Domains\GlobalDomain\ldom1</srcPath>
            <dstPath>C:\usr\Rpas\Domains\GlobalDomain2\ldom1</dstPath>
        </subdomain>
        <subdomain>
            <srcPath>C:\usr\Rpas\Domains\GlobalDomain\ldom2</srcPath>
            <dstPath>C:\usr\Rpas\Domains\ldom2</dstPath>
        </subdomain>
        <subdomain>
            <srcPath>C:\usr\Rpas\Domains\GlobalDomain\ldom3</srcPath>
            <dstPath>C:\usr\Rpas\Domains\ldom3</dstPath>
        </subdomain>
    </globaldomain>
</rpas>
```

The "globaldomain" tag should contain one "srcPath" tag, one "dstPath" tag, and a "subdomain" tag for each sub-domain. Each "subdomain" tag should contain one "srcPath" tag and one "dstpath" tag. Each "srcPath" tag should be a path to either the master or sub-domain begins copied. Each matching "dstPath" tag should be a path to where to copy that part of the domain.

The copyDomain utility will validate the configuration xml file first before any files are copied. If any of the sub-domain source paths do not match a sub-domain path of the global domain being copied, a "can't find source sub-domain 'sub-domain' error will be report. If the global domain being copied contains any sub-domain that doesn't have a matching "srcPath" tag, a "sub-domain 'sub-domain' doesn't have a subdomain xml tag" error will be reported. If the global domain "srcPath" tag doesn't contain the path of a valid global domain then an "invalid source path 'srcPath' to global domain" will be reported.

The destination paths in all cases will be validated when that part of the global domain is being copied. Unless the switch -force is provided, the destination must not exist and must be writable.

There are two options that control the number of sub-domains to be copied. These options will still limit the number of sub-domains that are copied; however the configuration file must still contain entries for all domains.

Move a Domain – moveDomain

The `moveDomain` utility provides the flexibility to move elements of global domains such as individual local domains and the master domain to pre-specified locations based on a given XML configuration file. The utility automatically updates RPAS metadata to reflect the modified directory paths in local and master domains. This utility also ensures that the **globalDomainConfig.xml** is updated as domains are moved.

The XML configuration being used will be simple and designed to fit the required task. It will contain fields for the locations of the source master domain and destination master domain as well as source and destination fields for each of the sub-domains that need to be moved.

Usage

```
movedomain -version
movedomain -xmlConfigFile filename
movedomain -d master -srcSubDomain src -dstSubDomain dst
```

The following table provides descriptions of the arguments used by the `moveDomain` utility.

Argument	Description
`-xmlConfigFile pathToXmlConfigFile.xml`	This argument will allow movedomain to move a sub-domain into user-instructed specific locations based paths specified in an xml file. This argument should not be used with the `–d`, `-srcSubDomain`, and `–dstSubDomain` parameters.
`–d pathTomaster`	This argument will allow `movedomain` to move each sub-domain based on the user-specified paths. Enter the path to the master domain.
`–srcSubDomain src`	Path of the sub-domain to be moved.
`–destSubDomain src`	Path where the sub-domain is to be moved.

Format of the XML Configuration File

The XML configuration being used will be simple and designed to fit the required task. It will contain fields for the locations of the source master domain and destination master domain as well as source and destination fields for each of the sub-domains that need to be moved. Here is a basic example of the XML configuration file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<rpas>
    <globaldomain>
        <srcPath>C:\usr\Rpas\Domains\GlobalDomain</srcPath>
        <dstPath>C:\usr\Rpas\Domains\GlobalDomain2</dstPath>
        <subdomain>
            <srcPath>C:\usr\Rpas\Domains\GlobalDomain\ldom0</srcPath>
            <dstPath>C:\usr\Rpas\Domains\GlobalDomain2\ldom0</dstPath>
        </subdomain>
        <subdomain>
            <srcPath>C:\usr\Rpas\Domains\GlobalDomain\ldom1</srcPath>
            <dstPath>C:\usr\Rpas\Domains\GlobalDomain2\ldom1</dstPath>
```

```
        </subdomain>
        <subdomain>
            <srcPath>C:\usr\Rpas\Domains\GlobalDomain\ldom2</srcPath>
            <dstPath>C:\usr\Rpas\Domains\ldom2</dstPath>
        </subdomain>
        <subdomain>
            <srcPath>C:\usr\Rpas\Domains\GlobalDomain\ldom3</srcPath>
            <dstPath>C:\usr\Rpas\Domains\ldom3</dstPath>
        </subdomain>
    </globaldomain>
</rpas>
```

The "globaldomain" tag must contain a "srcPath" tag and "dstPath" tag for the master domain. Master domain will not be moved if "srcPath" and "dstPath" are the same. It is essential to specify "srcPath" and "dstPath" for the master domain even if the master is not intended to be moved; otherwise an error condition will be incurred.

"srcPath" and "dstPath" tags for local domains are required ONLY if the local domain is intended to be moved; otherwise, the lack of tags for a specific local domains indicates that the local domain will not be relocated. If "srcPath" and "dstPath" are identical for a given local domain, it will not be moved.

When global domain "srcPath" and "dstPath" are different; i.e. when moving global domains, all local domains that reside under global domain folder and are not included in the XML file, will be moved to the destination global domain folder. Other local domains with a specified destination location will be moved according to the configuration.

Assumptions and Requirements

The following rules apply to the XML configuration settings:

- All source and destination paths must be absolute.
- All source paths must correspond to existing directories.
- All destination paths must be valid, in the sense that:
- The parent of the destination directory must exist
- The parent directory must be writable by the user.
- The destination directory itself must not exist.
- The source and destination master domain paths are required.
- The source and destination sub domain paths are required only for the domains that need to be moved.
- If the xmlConfigFile contents do not abide by the above mentioned rules, the utility does not clear the validation phase and terminates with the appropriate error message.

Minimum Space Requirement

Minimum space requirement for moving a global domain is the size of (just) master domain plus the size of the largest local domain.

Setting Miscellaneous Domain Properties – domainprop

Use the domainprop utility to manipulate the properties of a domain. Specify password properties, lock user accounts, and determine whether or not a daemon is currently managing a domain. The domainprop utility can be run on a global domain master to set values in all subdomains.

Usage

```
domainprop –d pathToDomain –expirePassword {days} {–passwordHistory
{oldPasswordCount}} {–property propertyname=value} {–lockAccount {failedLogins}}
{–daemonPort}
```

The following table provides descriptions of the arguments used by the domainprop utility.

Argument	Description
–d *pathToDomain*	Specifies the domain path.
–expirePassword *days*	Used to set or view the number of days a password is valid. Specify a valid integer after the argument to set the number of days for which the password will be valid. If no number is provided the utility prints the current setting.
–passwordHistory *oldPasswordCount*	Used to set or view the number of previous passwords that are kept to ensure that a user does not repeat a password too often. If a number follows –passwordHistory, the property is set to that number. Otherwise, the current setting is printed.
–property *propertyname=value*	Used to specify the property to be changed. See the list of properties below that can be set with this utility. To view the current property setting, use the property command with no value.
–lockAccount *{#failedLogins}*	Used to set or view the number of failed login attempts that can occur before the account is locked out. If a number follows –lockAccount, the property is set to this value. Otherwise, the current setting is printed.
–daemonPort	Prints a message that indicates whether or not the domain is currently being managed by a domain daemon. If the domain is not currently being managed, the port of the last daemon to manage the domain is printed.
-reportSubDomains	Use this option to show property values for subdomains (should all match).

Available Properties

- disable_commit_later (value is Boolean) – Setting this property to "true" (or "t") disables the ability to use "Commit Later" in the File menu of the RPAS Client. This property is set to false by default.
- help_path – This property is set when using custom help files for the RPAS Client instead of the default help files that are provided. These files need to be in "WebHelp" format and this path would normally be a path to a network server where the common help files would be located.

Calculation Engine – mace

The mace utility (Multi-dimensional Array Calculation Engine) allows the administrator to evaluate rule groups or expressions in order to manipulate measures. mace supports the use of the RPAS calculation engine in batch.

The mace utility is most commonly used to run a rule group or an expression, but can also be used to:

- create rules and rule groups
- add rules to rule groups
- add expressions to rules
- delete rules not contained in a rule group
- remove any or all rule groups
- validate expressions
- print a list of rules or rule groups

Usage

```
mace -version
mace -d domainPath -find  string
mace -d domainPath -newRule {-ruleName ruleName} {-label ruleLabel}
mace -d domainPath -delRule ruleName
mace -d domainPath -addRule     groupName:ruleName
mace -d domainPath -removeRule  groupName:ruleName
mace -d domainPath -newGroup    groupName {-label groupLabel}
mace -d domainPath -removeGroup groupName
mace -d domainPath -addExpression ruleName -expression exprString
mace -d domainPath -check -expression exprString
mace -d domainPath -run -group groupName      {-debugRuleEngine}
mace -d domainPath -run -expression expString {-debugRuleEngine}
mace -d domainPath -resolve groupName -measures measList {-debugRuleEngine}
mace -d domainPath -transit workbookName -group groupList {-debugRuleEngine}
mace -d domainPath -print -rule  ruleList
mace -d domainPath -print -group groupList
mace -d domainPath -print -allGroups
mace -d domainPath -purgeRules
mace -d domainPath -removeAllRuleData
mace -d domainPath -validate calc    -ruleGroup groupName
mace -d domainPath -validate general -ruleGroup groupName
mace -d domainPath -validate refresh -ruleGroup groupName -calcRuleGroup calc
```

The following table provides descriptions of the arguments used by the mace utility.

Argument	Description
–d *domainPath*	Specifies the domain in which to load the measure.
–find *string*	Use this argument to search all expressions for the specified string, printing all the rules and rule groups that have these expressions.
–newRule {–ruleName *ruleName*	Use this argument to create a new empty rule. If desired use the *–ruleName* parameter to specify a name for the rule.
-delRule *ruleName*	Use this argument to remove a specified rule.
–addRule *groupName:ruleName*	Use this argument to add the specified rule to the specified rule group.
-removeRule *groupName:ruleName*	Use this argument to remove the specified rule from the specified group.
–newGroup *groupName*	Use this argument to create a new rule group with the specified name.
–removeGroup *groupName*	Use this argument to remove specified group and non-shared rules in it.
–addExpression *ruleName*	Use this argument to add an expression to the specified rule. –expression should be used with this argument.
–check	Use this argument to validate the specified expression. –expression should be used with this argument.
–run	Use this argument to evaluate the specified expression or rule group. –expression should be used with this argument.
–resolve *groupName*	Use this argument to order (does not evaluate) expressions within rule group. Requires a comma-separated list of edited measures.
–transit *workbookName*	Use this argument to run a calc engine by transitioning over a list of rule groups. Requires the name of an existing workbook and a comma-separated list of rule-group names.
–print *{ruleList | groupList | true}*	Use this argument to print all the specified rules and rule groups. The *ruleList* is a comma-separated list of rule names. The *groupList* is a comma-separated list of group names. If "true" is supplied for either *ruleList* or *groupList*, all rules or rule groups are printed.
–purgeRules	Use this argument to remove all rules not contained in any rule groups.

Argument	Description
–removeAllRuleData	Use this argument to remove all rule groups and all rules.
–validate {calc | general | refresh}	Use this argument to validate rule groups. Use calc to validate a calc rule group. To validate a refresh rule group, use refresh parameter along with –calcRuleGroup to specify the corresponding calc rule group. For all other types of rule groups, use general.
–debugRuleEngine	Use this argument to generate a file "mace.log" in the working directory for logging RuleEngine specific debug information.
–expression *exprString*	Use the argument to specify the expression. This argument is used in conjunction with the –addExpression, –check, and –run arguments.
–group *groupName*	Use this argument to specify the rule group to evaluate using the –run argument.
–measures *measureList*	Use this argument to specify the measures to resolve.
–group *groupList*	Use this argument to specify a list of group names, separated by commas. Use this argument in conjunction with the –transit and –print arguments.
–rule *ruleList*	Use this argument to specify a list of rule names, separated by commas. Use this argument in conjunction with the –print argument.
–allGroups	Use this argument in conjunction with the –print argument to print all rule groups.
–addGroup	Use this argument to create a new rule group with the specified name

Managing Users – usermgr

Use the usermgr utility to add a user, remove a user, or print information about a user in a specified domain.

Usage

```
usermgr -d domainPath -add userName -label label -password psw -group grp {-ad
min}
usermgr -d domainPath -remove username
usermgr -d domainPath -removeLabel label
usermgr -d domainPath -list
usermgr -d domainPath -print -user username
usermgr -d domainPath -print -group groupname
usermgr -d domainPath -addUsers userFormat -label label -group grp -begin begin -
end end {-admin}
```

The following table provides descriptions of the arguments used by the usermgr utility.

Argument	Description
–d *domainPath*	Specifies the path to a domain to add, remove, or get information about a user.
–add *userName*	Use this argument to add a user with a specified name. Use the other arguments specified in the usage to add those attributes for that user.
–label *label*	Use this argument to specify the label of the user to add to the domain.
–password *psw*	Use this argument to specify the password of the user to add to the domain.
–group *grp*	Use this argument to specify the user group of the user to add to the domain.
–admin	Use this argument to specify that the user to add to the domain has administrative rights.
–remove *userName*	Use this argument to remove the user with the specified name from the domain.
-removeLabel *label*	Use this argument to Remove all users with this label.
–list	Use this argument to list all the users registered to the specified domain.
–print	Use this argument to print the specified user or group information.
–user *username*	Use this argument to specify the user name in the specified domain to print. This argument is only applicable to –print option.
–group *groupname*	Use this argument to specify the group in the specified domain name to print. This argument is only applicable to –print option.

Argument	Description
-addUsers	Adds users from begin to end using userFormat argument as a printf format string to form both the user name and the password. For example the arguments: "-addUsers 'usr$02d'-begin 1 -end 3" would result in usr01, usr02, and usr03 created with the password matching the userName.

Managing the Workbook Batch Queue – wbbatch

The wbbatch utility is used to manage workbooks in the workbook batch queue. The workbook batch queue is updated by using the standard RPAS wizard Auto-Workbook Build or using various options of the wbbatch utility.

The most common use of this utility is to build workbooks that have been scheduled to be automatically built using the Auto-Workbook Build wizard in the RPAS Client.

When a user defers a workbook commit (using Commit Later), that workbook commit process is added to the Commit Later queue which is committed using this utility. An administrator can also add a workbook to the commit later queue with this utility.

RPAS provides the ability to update workbook data with domain data without having to rebuild the workbook; this refreshing process is completed using a workbook's default refresh rule group. Workbooks are added to the queue to be refreshed and refreshed using this utility.

The build and refresh operations can be executed in multiple, parallel processes using the –processes argument.

Usage

```
wbbatch -version
wbbatch -d pathToDomain -build workbookName
wbbatch -d pathToDomain –refresh workbookName
wbbatch -d pathToDomain –commit workbookName
wbbatch -d pathToDomain –scheduleRefresh workbookName
wbbatch -d pathToDomain –unscheduleRefresh workbookName
wbbatch -d pathToDomain –scheduleCommit workbookName
wbbatch -d pathToDomain –unscheduleCommit workbookName
wbbatch -d pathToDomain -startQueue [all|build|refresh|commit] [-processes max]
wbbatch -d pathToDomain -printQueue [all|build|refresh|commit]
```

The following table provides descriptions of the arguments used by the wbbatch utility.

Argument	Description
–d pathToDomain	Specifies the domain containing the workbooks.
–build *workbookName*	Runs workbook build for provided *workbookName*.
–refresh *workbookName*	Refreshes workbooks scheduled to be refreshed using this utility. To refresh a single workbook in the queue specify the name of the workbook. If no name is provided all workbooks scheduled to be refreshed will be completed.

Argument	Description
–commit *workbookName*	Commits workbooks with deferred commits. To commit a single workbook in the commit later queue specify the name of a workbook. If no name is provided all workbooks in the commit later queue will be committed.
–processes *count*	Used with either –build or –refresh to build or refresh workbooks in the auto-workbook queue in parallel using the specified number of parallel processes.
–scheduleRefresh	Schedules a workbook to be refreshed later by adding it to the workbook batch queue.
–unscheduleRefresh *workbookName*	Removes a workbook from the workbook batch queue.
–scheduleCommit *workbookName*	Schedules a workbook to be committed later by adding it to the workbook batch queue.
–unscheduleCommit *workbookName*	Removes a workbook from the workbook batch queue.
-startQueue	Runs all workbooks in provided queue.
-printQueue	Prints the contents of the queue argument.

Workbook Manager – wbmgr

Use the Workbook Manager utility to inspect or remove the existing workbooks. It is recommended that administrators use this utility to remove workbooks rather than doing so manually.

Usage

```
wbmgr -version
wbmgr -d pathToDomain -list -all
wbmgr -d pathToDomain -list -user userName
wbmgr -d pathToDomain -print  -wbList wb1,wb2,...
wbmgr -d pathToDomain -remove -all
wbmgr -d pathToDomain -remove -user userName
wbmgr -d pathToDomain -remove -user userName -wbList wb1,wb2,...
```

The following table provides descriptions of the arguments used by the wbmgr utility.

Argument	Description
–d *pathToDomain*	Specifies the domain that contains the workbooks.
–list –all	Lists all workbooks in the domain.
–list –user *userName*	Lists all workbooks belonging to the user.
–print –wbList wb1,wb2,...	Prints detailed information about workbooks in the list.
–remove –all	Removes all workbooks from the domain.
–remove –user *userName*	Removes all workbooks from the domain belonging to the specified user.

Argument	Description
–remove –user *userName* –*wbList wb1,wb2*	Removes all the workbooks in the specified list for the specified user.

13

Informational Utilities

There are numerous RPAS utilities that can be used for finding information about many of the different components of a domain or domain data. The following utilities are solely for retrieving information and to not make any changes to a domain or data in a domain.

Retrieving Domain Information – domaininfo

The `domaininfo` utility is used to provide miscellaneous details about a domain, such as the type of domain (simple, master, or sub/local), and the upgrade/version history of the domain.

The domain path (`-d`) is required for all commands except `–expectedversion`.

Usage

```
domaininfo –d pathToDomain [Command]
domaininfo –expectedversion
```

The following table provides descriptions of the arguments used by the `domaininfo` utility.

Argument	Description
`-d`	Path to the domain. Required for all options except `–expectedversion`.
`–domainversion`	Display the RPAS version of the specified domain.
`–expectedversion`	Displays the expected RPAS version of the domain that the utility expects to find.
`-apptag`	Displays the application associated with domain.
`–history`	Displays the version history of the domain, specifically when the domain was upgraded to new versions of RPAS (patches or releases).
`–xnames`	Lists dimensions which use external names.
`–type`	Command to display the type of the domain. Possible values are Simple, Global, and Sub. A Simple domain is a traditional, non-partitioned (non-global) domain. A Global domain is the central/master domain of a global domain environment. A Sub domain is one local domain in a global domain environment that can contain one or more partitions.
`–listsubdomains`	Displays a list of all the local domains in a global domain environment, and indicates which positions at the partition level are in each local domain. This argument is only valid when run on a global domain.
`-showrelativepaths`	When listing subdomains, indicates if paths are relative. Only relevant in combination with `–listsubdomains` or `–all`.

Argument	Description
-masterdomaininfo	Lists the master domain path and partition dims for subdomains.
–subdomain *dim,pos*	Indicates to which local domain the specified position belongs. The position can be at or below the partition level.
–all	Displays all of the above information about the domain.
-version	Displays the version of this utility.

Checking the Validity of a Domain – checkDomain

This utility is used to check the validity of an existing domain. Its primary purpose is to verify that a master domain matches its respective local domains and report all discrepancies to the administrator.

Usage

checkDomain –d *pathToDomain* –type *expectedType* {-q}

The following table provides descriptions of the arguments used by the checkDomain utility.

Argument	Description
–d *pathToDomain*	Path to the domain that needs to be validated.
–type *expectedType*	Expected type of domain: simple, master, or sub.
–q	Quiet mode. Do not display progress messages.

When checkDomain is run on a **simple** domain the following two items get validated:

- The domain directory exists
- It is of type "simple"

If checkDomain is run on a **Global Domain**, it verifies the following:

- The global domain exists
- The global domain is of type "master"
- The global domain checks all of the sub-domains for:
  - The sub-domain directory exists and is of type "sub"
  - If the master domain and the sub-domain have a repos directory
  - The measures, rules, rule groups, templates, and functions are the same in the global and sub-domain

If it is run on a **sub-domain**, it checks all of the items listed above for the global domain, but the validation is only performed between the global domain and the specified sub-domain.

Determining RPAS Server Version – rpasversion

Use the rpasversion utility to determine which version of the RPAS Server is running in a particular location.

Usage

```
rpasversion –l pathToLibrary
```

List Contents of a Database – listDb

Use listDB to list the basic information of all arrays contained in the databases provided.

Usage

```
listdb pathToDb*
listdb -row -db pathToDb*
listdb -row -pageUsage -db pathToDb*
listdb -row -standardOptions -db pathToDb*
listdb -standardOptions -db pathToDb*
listdb –version
```

The following table provides descriptions of the arguments used by the listDb utility.

Argument	Description
–db *pathToDb*	Specifies the database to list the contents.
-row	List array information in a row format.
-pageUsage	Show btree page usage. Requires –row switch to be active.
-standardOptions	List only standard options.

Printing Data from Arrays – printArray

Use printArray to print the contents of an array.

Usage

```
printArray –array db.array –specs {–maxpos num}
printArray –array db.array {–cell "dim1:pos1,dim2:pos2,..."
{–format "formatString"}
printArray –array db.array –slice "dim1:pos1,dim2:pos2,..."
{–format "formatString"} {–cellsperrow num} {–noposnames}
printArray –array db.array –allpopulatedcells {–format "formatString"}
{–cellsperrow num} {–noposnames}
```

The following table provides descriptions of the arguments used by the printArray utility.

Argument	Description
–array *db.array*	Specifies the array to print. Specify the full path to the database containing the array. Required for all commands except –version. *db* is a full or relative path to a database. Do not specify the .gem suffix. If no other commands are included, the array defaults to -allpopulatedcells with cells per row 1. The -allpopulatedcells command is still available, but now functions as a useful default action. The -noposnames, -cellsperrow, and -format parameters may still be specified when relying on the implicit -allpopulatedcells behavior.
–specs	Prints the specifications of the array and positions along each dimension.
-popcount	Outputs only the popcount of the specified array. Useful to shell script writers to get the popcount value into a shell script variable. For example, export POPCOUNT=`printArray -array hmaint.dim_year -popcount`
–cell *CELLSPEC*	Prints a specific cell value from the array. Must not contain spaces. Must identify a single of 1-D slice. Specify using the format "dim1:pos1,dim2:pos2,…"
-cellplain *CELLSPEC*	Outputs a specific cell value with no space padding. Useful for scripts when capturing cell values into shell variables. Must not contain spaces. Must identify a single of 1-D slice. Specify using the format "dim1:pos1,dim2:pos2,…"
–slice *CELLSPEC*	Prints a one-dimensional slice from the array. Must not contain spaces. Must identify a single of 1-D slice. Specify using the format "dim1:pos1,dim2:pos2,…"
–allpopulatedcells	Print all populated cells including the navalue of the array.
–format *"fmtstr"*	If –format is specified, any cells with numeric values are interpreted as dates. *fmtstr* (formatString) determines how dates are interpreted, and can include: ▪ %Y - 4 digit year ▪ %m - month number (01 to 12) ▪ %d - numeric day of month (01 to 31) ▪ %H - 24 hour clock (00 to 23) ▪ %M - minute (00 to 59) ▪ %S - seconds (00 to 61) ▪ %s – milliseconds
–cellsperrow *num*	For multi-cell output commands (-slice and --allpopulatedcells), indicates how many cells should be printed on each line.
–noposnames	Suppresses the output of position names, only cell values are shown.

Printing Data from Measures – printMeasure

Use the printMeasure utility to print measure information.

Usage

```
printmeasure -d domainPath {-wb wbName} {-m measure} [COMMAND]
```

The following table provides descriptions of the arguments used by the printMeasure utility.

Argument	Description
–d *pathToDomain*	Specifies the domain that contains the measure to print. Requires the -m parameter.
–m *measure*	Specifies the measure to print.
–wb *workbookName*	Specifies the workbook associated with the measure to print. If -wb is not used, the domain measure information is printed. Requires the -m parameter.
–list	printMeasure will return a list of all registered measures in the domain. This argument does not require –d domainPath.
-listHBIMeasures	In a global domain, printMeasure will return a list of all measures registered at or above the partition dimension.
–specs	printMeasure returns the list of measure properties. Requires the -m parameter.
–listDataIntersections	printMeasure will return the base intersection of the measure
–printData *aggType.intersection*	Prints out the nobs and nods format of the measure array at the specified intersection and agg type.

14

RPAS Reporter Tool User Guide

The purpose of the RPAS Reporter Tool is to provide an SQL interface to the Oracle RPAS embedded database (OREDB), which includes both domain data and workbook data. This tool presents OREDB as a relational database to ODBC and JDBC client applications. The RPAS Reporter Tool was designed to connect OREDB to ODBC 3.51 or JDBC 2.0 compatible applications. Connectivity has been verified thus far against the following applications:

- Oracle Business Intelligence Enterprise Edition
- Interactive SQL (ISQL) Utility
- JDeveloper

The RPAS Reporter Tool enables system users in reading measure data for stored measures in an RPAS domain.

- In a global domain environment, connection to local domains is not supported. Access to local domain data will be possible through queries in global domains.
- The Reporter Tool does not provide support for Forced Non-HBI (FNHBI) and non-materialized measures.
- The Reporter Tool reports only external position names in both dimension tables and "fact" tables. Internal position names will not be reported.
- Limited support is provided for conditional queries on measure data.
- The tool is not intended to replace the `exportData` utility, which is used for high-speed data export to ASCII files.

> **Note:** See the *RPAS Installation Guide* for more information on installing the RPAS Reporter Client and Server (ODBC Client and Server).

Technical Information about the ODBC Environment

Required Files

Both the ODBC Server and the ODBC Client configurations require the presence of certain files. In a Windows environment, these files will be located with the ODBC Server or ODBC Client installations. In the UNIX environment, the location of this file is stored in the OPENRDA_INI environment variable. This file is created during installation.

For UNIX systems running both the ODBC Server and ODBC Client, Oracle recommends maintaining separate environments for each application to prevent conflicts in the configuration files.

Server Files:

- oaserver.lic – License file for the ODBC Server.
- oaserverkey_oracle.lic – Oracle-specific key file for the ODBC Server.
- openrda.ini – This file is generated by the installation process. It stores the configuration parameters used by the server.
- oadrd.ini – This file lists all the domains and domain parameters used by the ODBC Server. It is generated by the configuration tools.

Client Files:

- oaisql.lic – License file for using oaisql application.
- oaclientkey_oracle.lic – Oracle-specific key file for the ODBC client
- openrda.ini – This file is generated by the installation process. It stores the configuration parameters used by the Client.
- oadrd.ini – This file lists all the domains and domain parameters used by the ODBC Server. It is generated by the configuration tools.

ODBC Configuration for Windows

Overview

Configuring the system to connect the ODBC drivers and a domain environment consists of the following procedures.

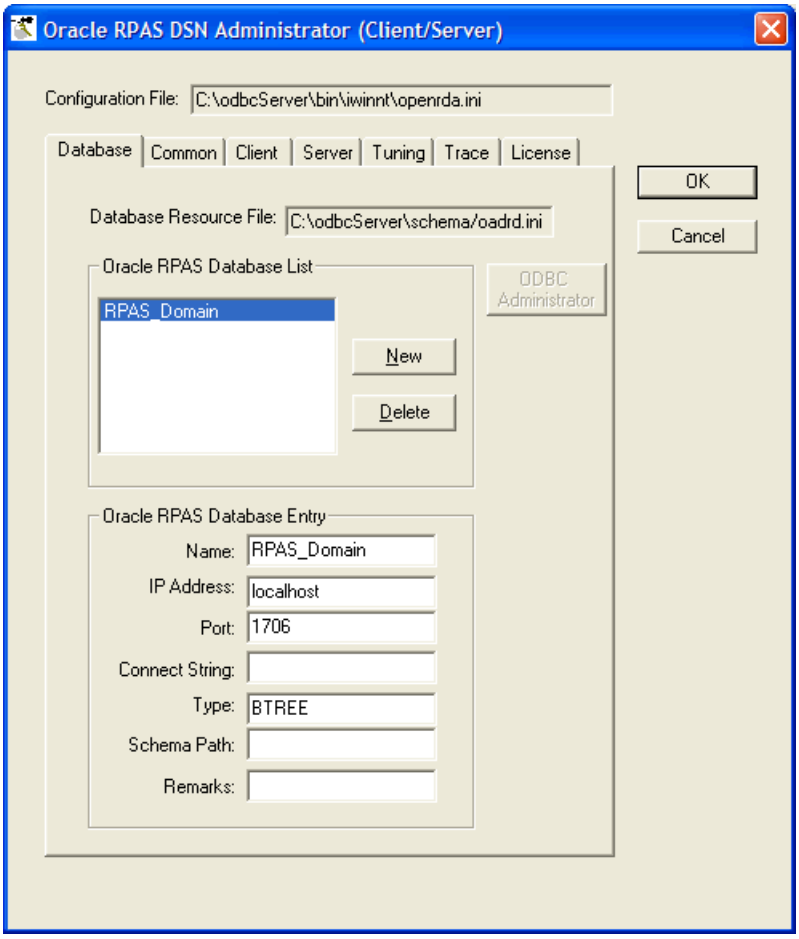
1. Install the ODBC server components. Refer to the *RPAS Installation Guide*.
2. Install the ODBC client components. Refer to the *RPAS Installation Guide*.
3. Configure the ODBC server components.
4. Configure the ODBC client components.
5. Create the ODBC data source name (DSN) for Windows. This enables tools such as OBIEE used on the local machine to connect to the domain environments configured in the ODBC server and client configuration.
6. Start the RPAS ODBC server process (ODBCMeasureModeServer.exe).
7. Test the connection using Interactive SQL.

Defining the ODBC Server Configuration Settings

To define the ODBC Server configuration settings, perform the following procedures:

1. From the **Start** menu, select **Oracle RPAS ODBC Server – Administration Tool**. The Oracle RPAS DSN Administrator (Client/Server) dialog appears which contains several tabbed panels.

   This ODBC Server Administration Tool (wrdadmin.exe, located in *{odbcserver_root}*\bin\iwinnt folder) should appear in the Oracle RPAS ODBC Server program folder (on Windows XP using the default installation settings) on the Start menu.



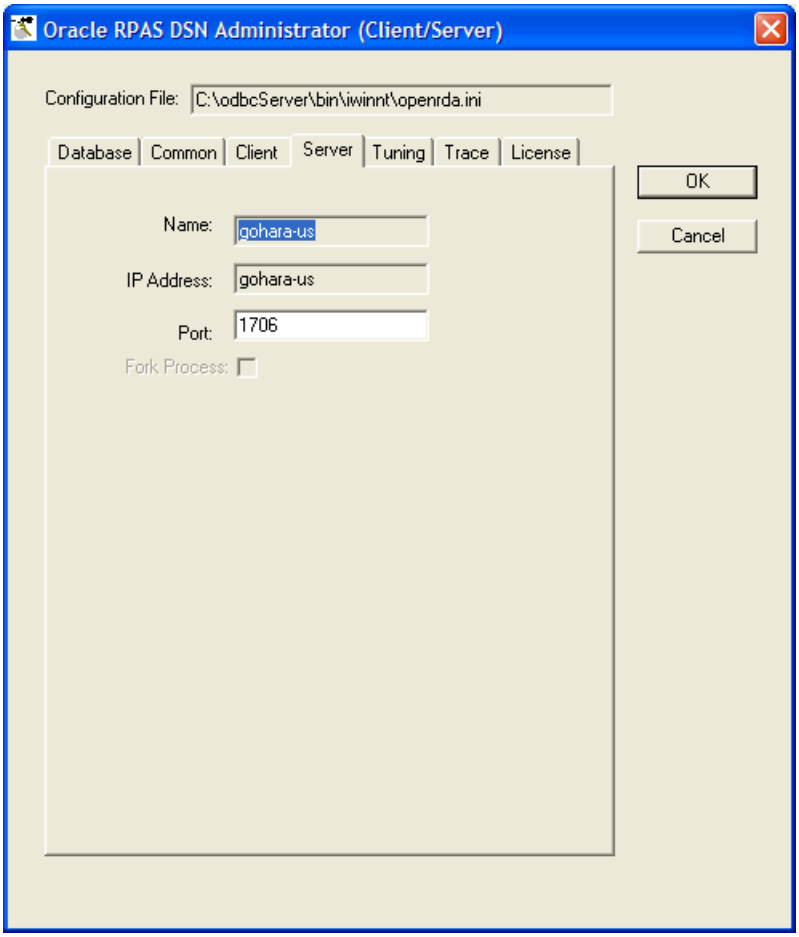
Oracle RPAS DSN Administrator Dialog Box– Database Tab

2. Use the **Database** tab to view and edit currently defined databases (domains), as well as add or remove existing ones.

   For each database, you can configure the following fields:

   - **Name –** Enter the name of the database connection; this can be any name.
   - **IP Address –** Enter the IP Address of ODBC Server. This should be set to **localhost** to designate use of the drivers on the local computer.
   - **Port –** Enter the port to which the database is assigned. This is NOT the port that the server process will use.
   - **Connect String –** Enter the path to the root directory of the domain.
   - **Type –** Enter **BTREE** in this field. This field indicates the type of database.
   - **Schema Path –** Enter the path to custom workbook schema (optional).
   - **Remarks –**Enter a description of the connection (optional).

3. Select the **Server** tab to specify which port the server process will use.



Oracle RPAS DSN Administrator Dialog Box– Server Tab

When the process starts, only databases assigned to that port will be accessible to the connecting clients.

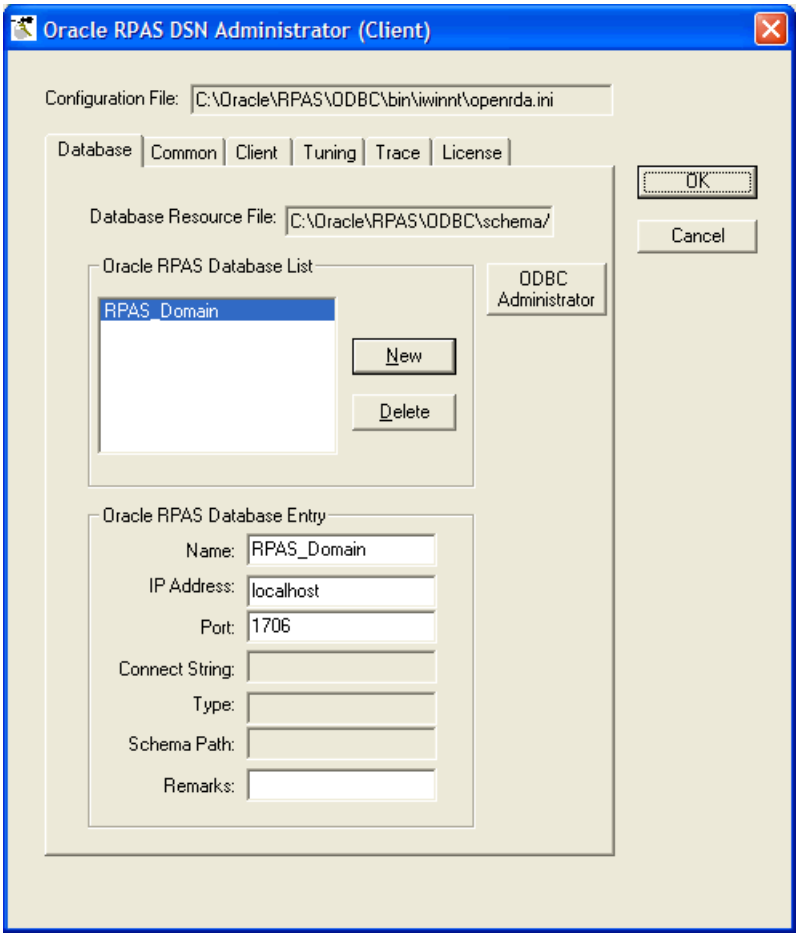
4. Click **OK** to save your changes and close the dialog box.

Defining the ODBC Client Configuration

To define the ODBC Client configuration settings, perform the following procedures:

1. From the **Start** menu, select **All Programs – Oracle RPAS ODBC Client – Administration Tool**. The Oracle RPAS DSN Administrator (Client) dialog appears which contains several tabbed panels.

   This ODBC Client Administration Tool (wrdadmin.exe, located in *{odbcclient_root}*\bin\iwinnt folder) should appear in the Oracle RPAS ODBC Client program folder (on Windows XP using the default installation settings).



Example of Oracle RPAS DSN Administrator (Client) – Database Tab

2. Use the **Database** tab to view and edit currently defined databases (domains), as well as add new databases or remove existing ones.

   For each database, you can configure the following fields:

   - **Name –** Enter the name of the database connection. This should be the same name used for the ODBC Server configuration.
   - **IP Address –** Enter the IP Address of ODBC Server. This should be set to **localhost** to designate use of the drivers on the local computer.
   - **Port –** Enter the port the server uses to set up connections. This value should match the value defined for the ODBC Server configuration.
   - **Remarks –** Enter a description of the connection (optional).

3. Click **OK** to save your changes and close the dialog box.

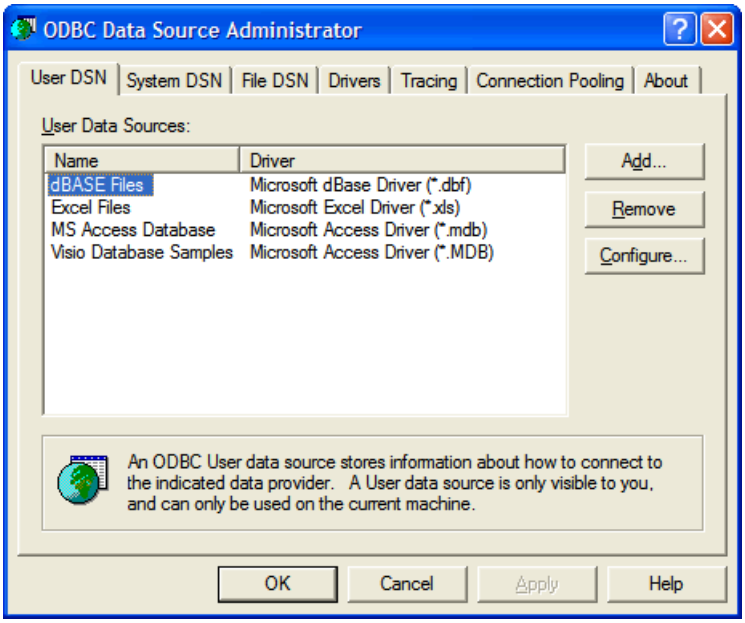
Creating the ODBC Windows Data Source

Before the ODBC Client can access a domain using a tool such as OBI, you must create a Windows data source. Perform the following procedure to create the data source:

1. Open the ODBC Data Source Administrator.

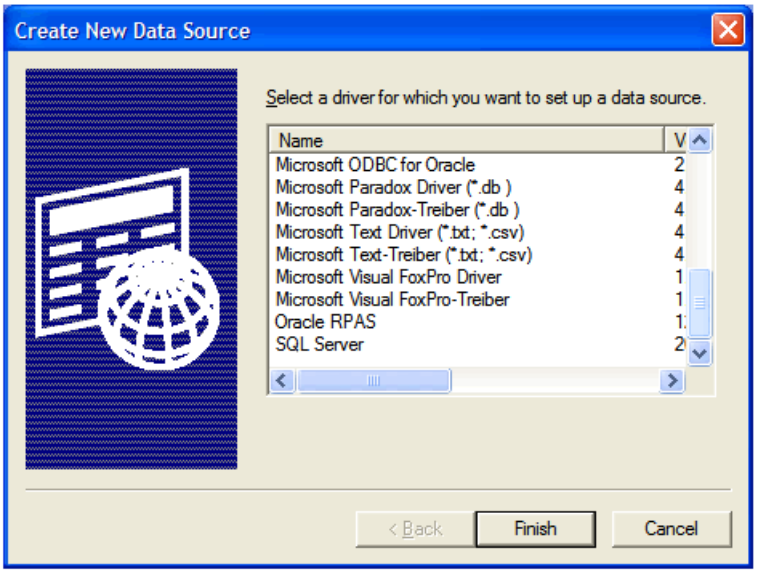
   This can be performed b selecting the **ODBC Administration** button the **Database** tab of the Oracle RPAS DSN Administrator (Client). It may also be accessed from the Windows Control Panel by opening **Administrative Tool**, and then opening **Data Sources (ODBC)**.

   The ODBC Data Source Administrator window appears.



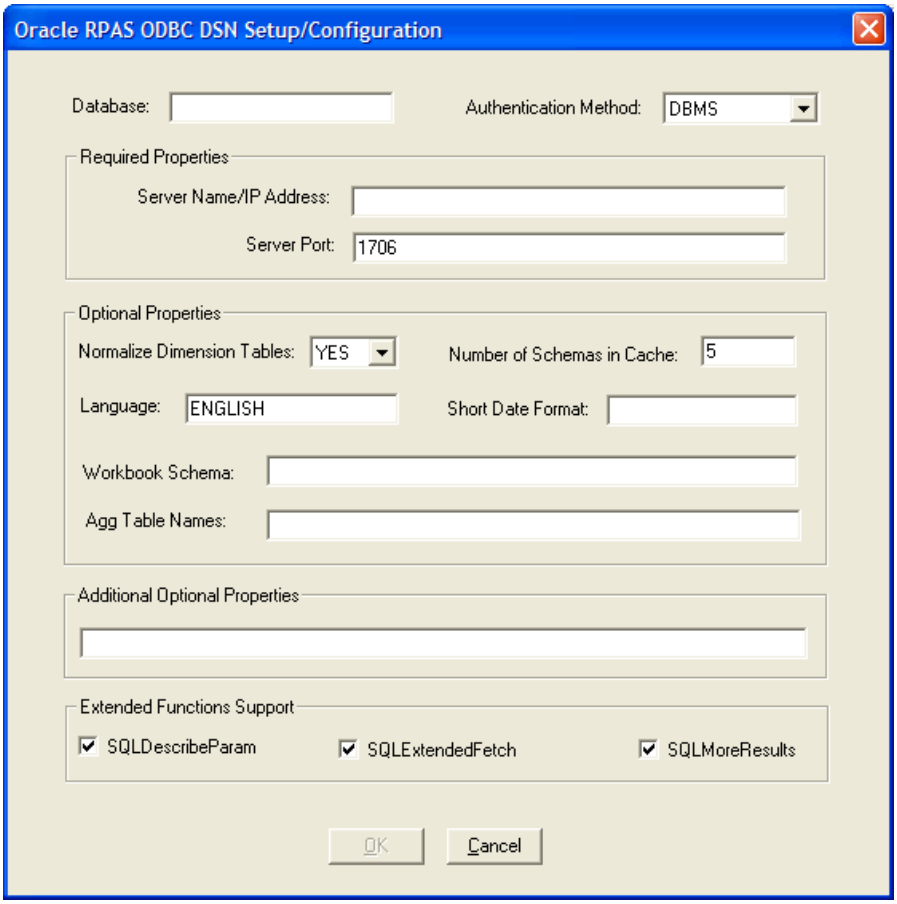
ODBC Data Source Administrator Window – User DNS Tab

2. To create a new system data source name, select the **System DSN** tab and click **Add**. The Create New Data Source dialog box appears.



Create New Data Source Dialog Box

3. Select **Oracle RPAS** from the list of drivers and click **Finish**. The Oracle RPAS ODBC DSN Setup/Configuration dialog box appears.



Oracle RPAS ODBC DNS Setup/Configuration Dialog Box

4. In the Configuration dialog box, specify the following settings, and then click **OK**:
   - **Database –** Enter the name of the domain that was defined in the ODBC Client and Server configuration processes.
   - **Authentication Method –** Set to **DBMS**.

   Required Properties:
   - **Server Name/IP Address –** Enter the server DNS name or IP address. Use **localhost** if on the same machine.
   - **Server Port –** Make sure the port is set to the same value defined in the ODBC Client and ODBC Server configuration processes.

   Optional Properties:
   - **Normalize Dimension Tables –** Determines whether the dimension tables are stored with the complete hierarchy or normalized with their immediate parent only. Selecting **YES** will result in many small tables as opposed to one big table containing the full hierarchy.
   - **Number of Schemas in Cache –** Number of schemas to cache on the local system. There is one schema for the domain itself, as well as an additional schema for each workbook.
   - **Language –** The language in which the Label fields and error messages will be returned.

- **Short Date Format –** Format that the date should be returned in.
- **Workbook Schema –** Name of a workbook. This allows the user to connect directly to the workbook rather than the entire domain.
- **Agg Table Names –** Comma separated list of specific fact tables to include in the domain. By default, only the base level intersections are created.
- **Additional Optional Properties –** Comma-separated list of additional properties, such as IgnoreUnicodeFunctions. For a list of properties accepted by ODBC, consult the ODBC documentation.

  The Addition Optional Properties field can be used to specify a value for RPAS_LOG_LEVEL. The syntax for this parameter is `RPAS_LOG_LEVEL=<loglevel>` where `<loglevel>` is one of the following: `all`, `profile`, `warning`, `information`, `error` or `none`.

  The log file is similar to the rpas.log generated when connecting to the domain with the RPAS Client. The log file file generate will be named as rpas_odbc_<timestamp>.log, where <timestamp> is the date and time the file was created. Just as rpas.log, the rpas_odbc.log file will exist in each user's directory in the domain. Each user connection will generate a new rpas_odbc.log file. Ten log files are maintained. When an eleventh log file is generated, the oldest log file is automatically deleted.
- **Extended Functions Support –** These options determine if the ODBC driver will make calls to specific performance enhancing ODBC APIs. The RPAS ODBC implementation supports each of the APIs, so they should always remain checked.

Starting the RPAS ODBC Server Process

The RPAS ODBC Server process must be started prior to using the clients.

To start this process, run the RPAS executable ODBCMeasureModeServer.exe from a Command Prompt window or using MKS Toolkit.

Testing the Connection

Once the ODBC Server and ODBC Client have been configured, you can test the connection using Interactive SQL. The RPAS ODBC Server process must be running.

1. Select **Start – All Programs – Oracle RPAS ODBC Client – Interactive SQL (ODBC)**. The Interactive SQL command window appears.
2. Type `'connect <user Name>/<password>@<dsn_name>'` where `<dsn_name>` is the name of the connection defined in the ODBC Server and ODBC Client configuration process.

   Example:

   `'connect adm/adm@myDb'`

   If the configuration is defined correctly, no errors are displayed.

ODBC Configuration for UNIX

Overview

Configuring the UNIX system to connect the ODBC/JDBC drivers and a domain environment consists of the following procedures.

1. Install the ODBC Server Components. Refer to the *RPAS Installation Guide*.
2. Install the ODBC Client Components. Refer to the *RPAS Installation Guide*.
3. Configure the ODBC server components.
4. Configure the ODBC client components.
5. Start the RPAS ODBC Server Process (ODBCMeasureModeServer).
6. Test the connection using Interactive SQL.

Server Configuration

The rpasodbcadmin tool allows you edit server parameters, such as logging levels and the port that the server will use. In addition, you can add, edit, or remove DSN connections as well as set the logging levels of the ODBC programs. When editing or adding DSN connections, you can modify the following fields:

- **DATABASE –** The name the client will use to connect to the domain.
- **ADDRESS –** The IP address of the ODBC Server
- **PORT –** The port number used to connect to this domain
- **CONNECT_STRING –** The path to the domain.
- **TYPE –** The type of database. Should be set to **BTREE**.
- **SCHEMA PATH –** Path to workbook schemas.
- **DESCRIPTION –** Optional description of the domain.

Client Configuration

The rpasodbcadmin tool is also used to configure the DSN connections used by the client. Although the application is also used to configure the server, you only need to fill in the following fields for the client:

- **DATABASE –** The name the client will use to connect to the domain.
- **ADDRESS –** The IP address of the ODBC Server.
- **PORT –** The port number used to connect to this domain.
- **CONNECT_STRING –** Ignored for client.
- **TYPE –** Ignored for client.
- **SCHEMA PATH –** Ignored for client.
- **DESCRIPTION –** Optional description of the domain.

Starting the RPAS ODBC Server Process

The RPAS Server executable to run the ODBC drivers is called ODBCMeasureModeServer. It can be found in $RPAS_HOME/bin.

To start the process, you can either run it in its own shell, or kick it off as a background process in the current shell.

Testing the Connection

Once the ODBC Server and ODBC Client have been configured, you can test the connection using Interactive SQL. The RPAS ODBC Server process must be running.

1. From the command prompt, run the oaisql executable.
2. Type `'connect <user Name>/<password>@<dsn_name>'` where `<dsn_name>` is the name of the connection defined in the ODBC Server and ODBC Client configuration process.

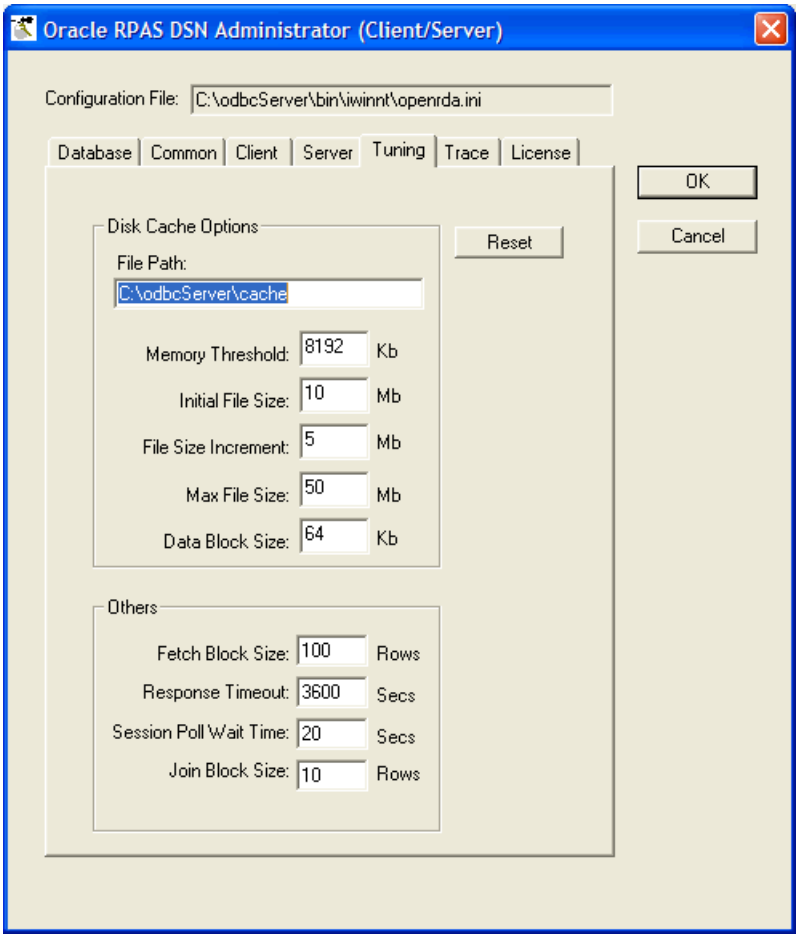
   Example:

   `'connect adm/adm@myDb'`

   If the configuration is correctly defined, no errors are displayed.

Tuning the ODBC Server

There are several ODBC Server parameters that can be tuned to optimize ODBC for the production environment. In Windows, these parameters can be found on the Tuning tab from the **Oracle RPAS ODBC Server – Administration Tool**.



Example of Tuning Tab for Oracle RPAS ODBC Server

- **Memory Threshold –** Specifies the maximum amount of memory the driver will use before it starts to use disk cache file. For optimal performance, the driver should be given as much memory as is available on the system.
- **Initial Files Size –** Initial size of the disk cache file.

- **File Size Increment –** Size by which the disk cache file will be incremented when it is full. This value should be large enough to avoid frequent disk space allocation.
- **Max File Size –** Maximum size of the disk cache file. If a query requires more space than the Max Size, the driver will simply stop processing the query and throw a "max disk cache size reached" message. Ideally this should not occur because the driver should be given enough memory (specified by Memory Threshold) in the first place. In case this does occur, try to increase Memory Threshold and /or Max File Size.
- **Data Block Size –** Block size of the disk cache file. It is recommended to keep this size tune with operating system's file buffer size. The maximum value for this parameter is 64 KB.
- **Fetch Block Size –** Specifies number of rows each time the driver will return to the client. This is also the number of rows the driver fetches each time in its internal processing. Larger Fetch Block Size will result in less network round trips (in client-server configuration) but requires the driver to use more memory.
- **Join Block Size –** Specifies the number of rows from outer table the driver use to join inner table when processing join queries. The larger this number, the less number of sub-join queries the driver will need to process. The maximum value is limited by amount of available memory (values between 100 to 1000 are reasonable).

In UNIX, these same parameters can be changed in the openrda.ini file.

Installing and Using the RPAS JDBC Driver

The following specifies how you can install, set up, and use the RPAS JDBC driver.

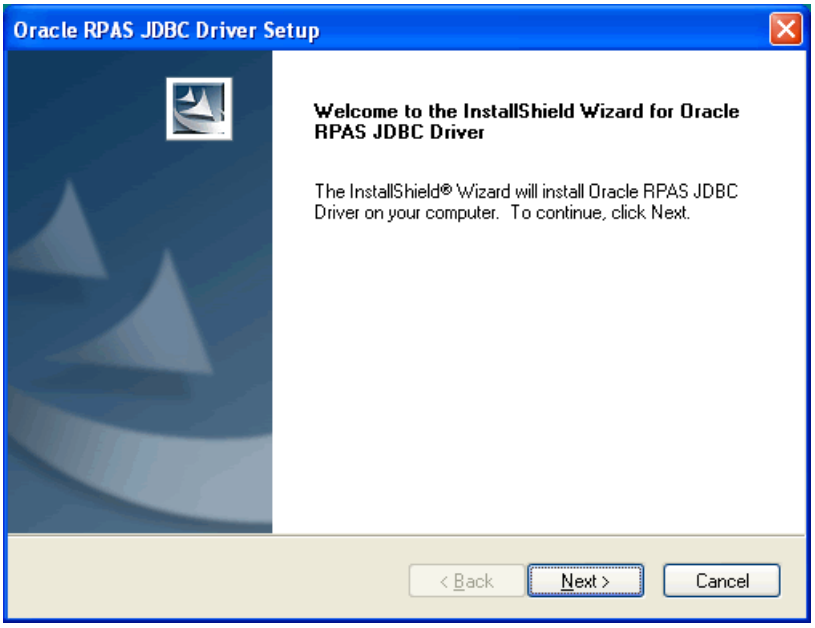
Installing and Setting Up the Oracle RPAS JDBC Driver on Windows

Installing the RPAS JDBC driver not only provides JDBC connectivity, but it also provides the following options in the program folder created during installation:

- **Administration Tool –** Opens Oracle RPAS DSN Administrator (Client/Server) utility.
- **Configuration File –** Opens the openrda.ini file.
- **Interactive SQL (ISQL) –** A utility to perform SQL commands on a database.
- **JDBC Readme –** Opens the driver Readme file.
- **Uninstall –** Opens the uninstall utility to remove the Oracle RPAS JDBC driver and all of its components from your system.

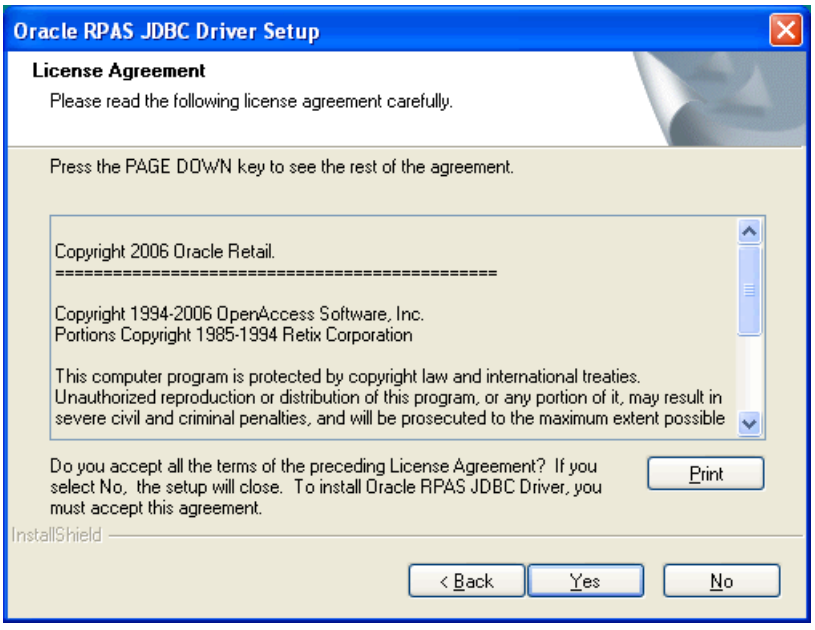
To install the Oracle RPAS JDBC driver, perform the following procedure:

1. Double-click the setup.exe file in the JDBC Client folder. The Welcome screen for the Oracle RPAS JDBC Driver Setup appears.



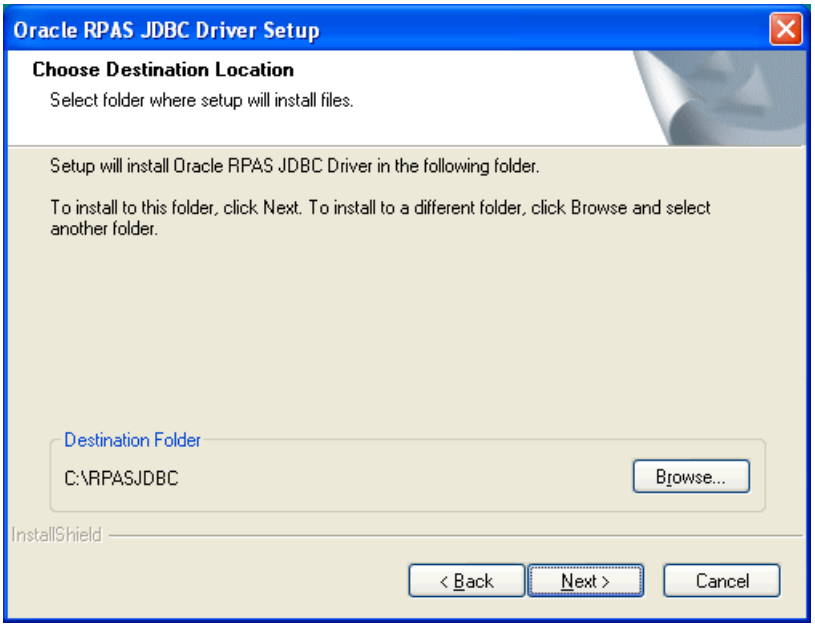
Welcome Screen

2. Click **Next**. The License Agreement screen appears.



License Agreement Screen

3. Select **Yes** to accept the agreement and proceed. The Choose Destination Location screen appears. This screen defines where the RPAS JBDC driver is installed on your system.



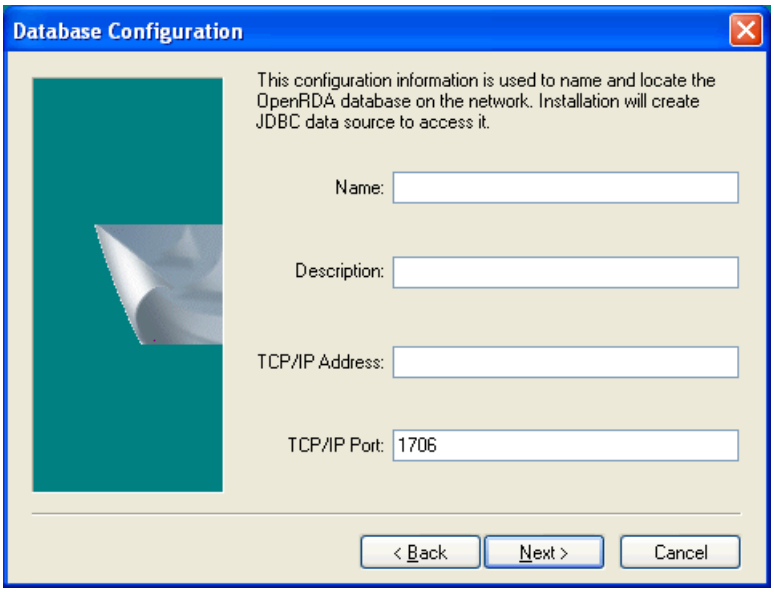
Choose Destination Location Screen

4. Click **Next** to accept the default location, C:\RPASJDBC.

   If you want to define a different location where the JDBC driver will be installed, click **Browse** to navigate to the appropriate location and click **OK**. Once you have selected the location, click **Next** to proceed.

   > **Note:** This folder is your `driver_home` path, which will be required later to update the *Path* and *classpath* environment variables. If you change the default folder, make sure to note this path when updating the environment variables.

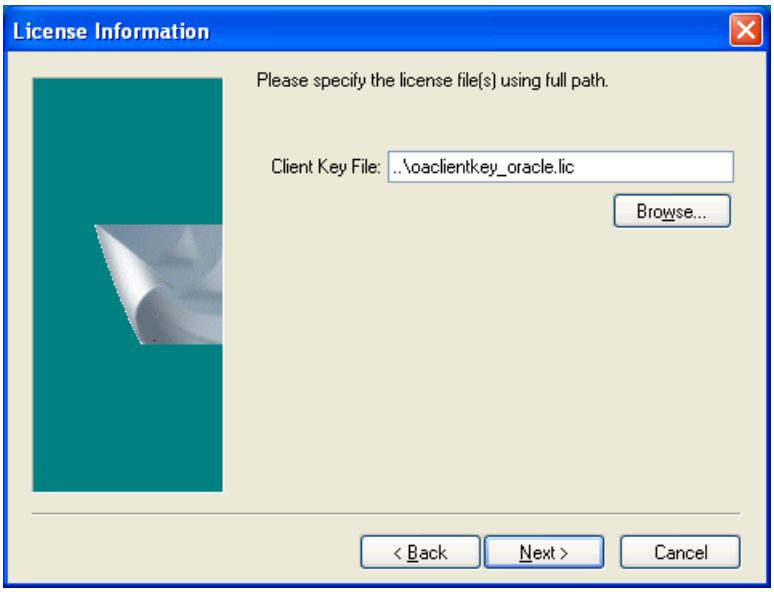
   The Database Configuration screen appears.



Database Configuration Screen

5. Enter the following information to define the database configuration and click **Next**:
   - **Name –** Enter the database identifier for the RPAS database that you want to connect to. This name should be the same as the database identifier on the server. This name should be provided to you by the RPAS ODBC/JDBC database administrator. This is not the path to the database.
   - **Description –** Enter a description for the connection. This field is optional.
   - **TCP/IP Address –** Enter the network address ODBC/JDBC server. This field can be an alias.
   - **TCP/IP Port –** Enter the port number that the RPAS ODBC/JDBC server is polling on the server. This information should be provided by the RPAS ODBC/JDBC database administrator. A default setting of 1706 appears in this field.

   The License Information screen appears. The path to the license file, oaclientkey_oracle.lic, should appear in the **Client Key File** field.



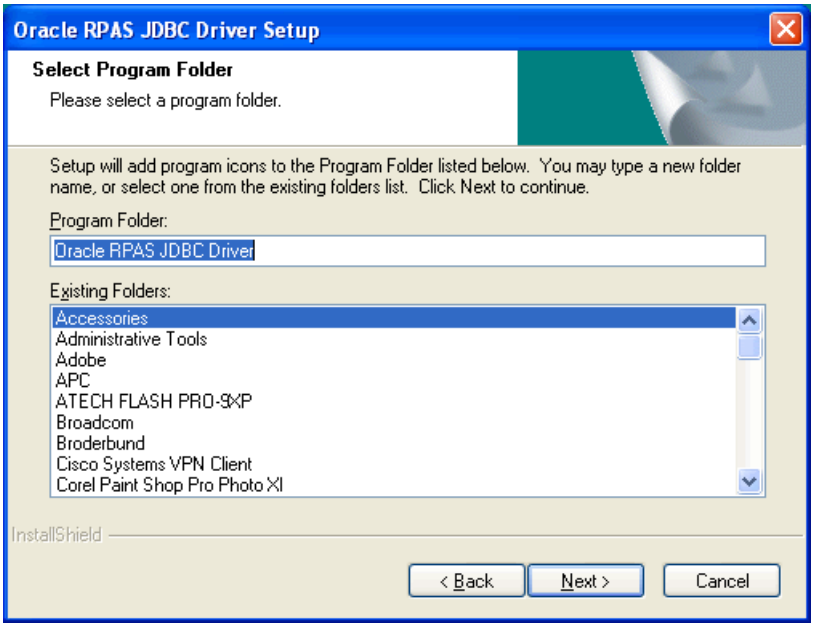
License Information Screen

6. If the correct path and file appear in the Client Key File field, click **Next** to proceed.

   If the path to the client key file is incorrect, click **Browse** to locate and select the file. Click **Next** to proceed.

   > **Note:** The path must include the oaclientkey_oracle.lic file name and extension.

   The Select Program Folder screen appears.

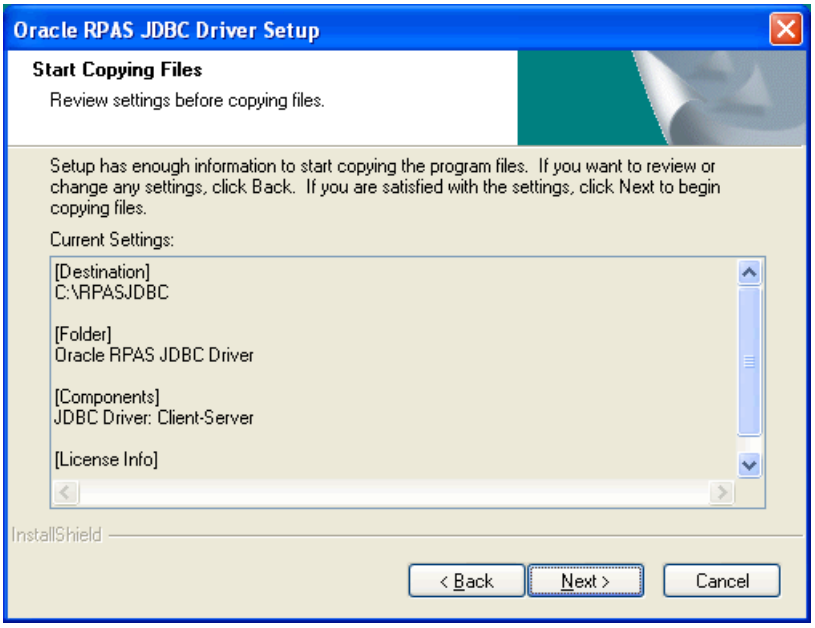


Select Program Folder Screen

7. Click **Next** to install the driver in the defined Program Folder.

   If you want to install the driver to an existing folder, select the appropriate folder from the Existing Folders list. To define a specific folder, you can enter the appropriate folder name in the Program Folder field, and then click **Next**.

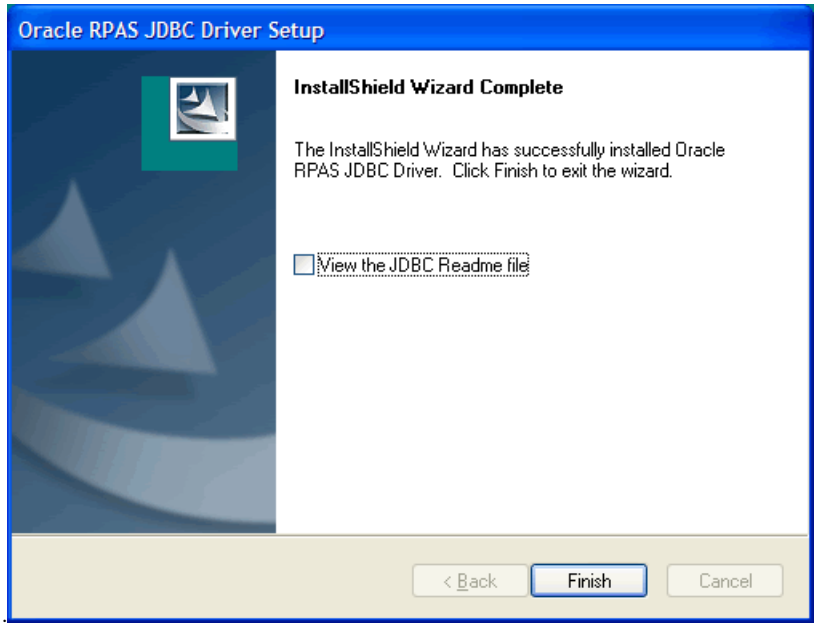
   The Start Copying Files screen appears.



Start Copying Files Screen

8. Click **Next** to install the JDBC driver.

   When the installation is complete, the Wizard Complete screen appears.



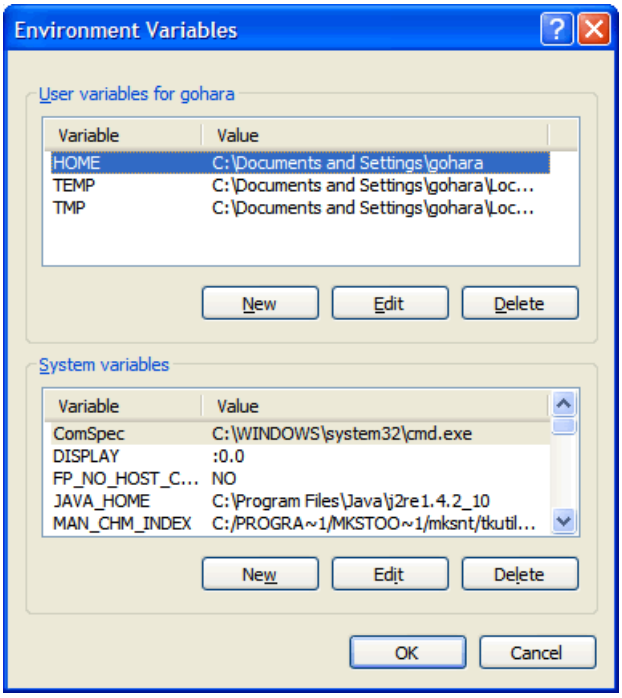
Wizard Complete Screen

9. Click **Finish**. To view the JDBC Readme file, select the option on the screen and then click **Finish**.
10. Update the environment variables so your system can access the JDBC driver. See Updating Environment Variables for JDBC Driver for more information.

Updating Environment Variables for JDBC Driver

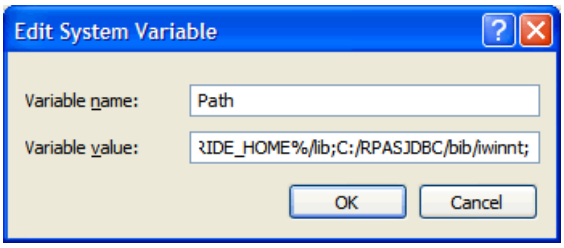
Once the JDBC driver is installed on your system, you need to update the Path and classpath environment variables. These variables ensure that the JDBC client can access the appropriate library and resource files needed to connect o the database.

1. Open **System** in the Control Panel. The System Properties window appears.
2. On the **Advanced** tab, click **Environment Variables**. The Environment Variables dialog appears.



Environment Variables Dialog Box

3. Select the **Path** from the **System variables** list and click **Edit**. The Edit System Variable dialog box appears.



Edit System Variable Dialog Box

4. Add *driver_home*`/bin/iwinnt` to the Path environment variable and click **OK**.

   `driver_home` is the location where the RPAS JDBC was installed. If you accepted the default location, C:/RPASJDBC, this path would be as follows:

   `C:/RPASJDBC/bin/iwinnt`

   Note: Remember to separate paths with semi-colons (;).

   By adding this to the Path environment, you ensure that the oajdbc.dll is in the path so that the Java driver class can link to the native driver libraries. The RPAS JDBC is a type III JDBC driver.

5. From the Environment Variables window, select classpath from the **System variables** list and click **Edit**. The Edit System Variable dialog box appears.

6. Add `diver_home/jdbc;driver_home/jdbc/oadriver.jar` to the classpath environment variable and click **OK**.

   As stated earlier, `driver_home` is the location where the RPAS JDBC was installed. If you accepted the default location, C:/RPASJDBC, this path would be as follows:

   `C:/RPASJDBC/jdbc;C:/RPASJDBC/jdbc/oadriver.jar`

Adding this path ensures that JDBC clients can find the driver Java classes. The RPAS JDBC driver is packaged in the oadriver.jar file.

7. After updating the environment variable, restart your PC.

   Once you have updated the environment variables and restarted your PC, you are ready to use the RPAS JDBC driver with any JDBC client.

Installing and Setting Up the JDBC Client on UNIX

The JDBC client is distributed with the RPAS installation.

To configure your environment, run the jdbclogin.ksh script, which is located in $RPAS_HOME/jdbcclient directory.

Using the RPAS JDBC Driver

Any JDBC client needs the following information to use a JDBC driver to connect to a database:

- A driver class
- A URL to the database specified in a form that the particular JDBC driver understands

For the RPAS JDBC driver, this information is specified as follows:

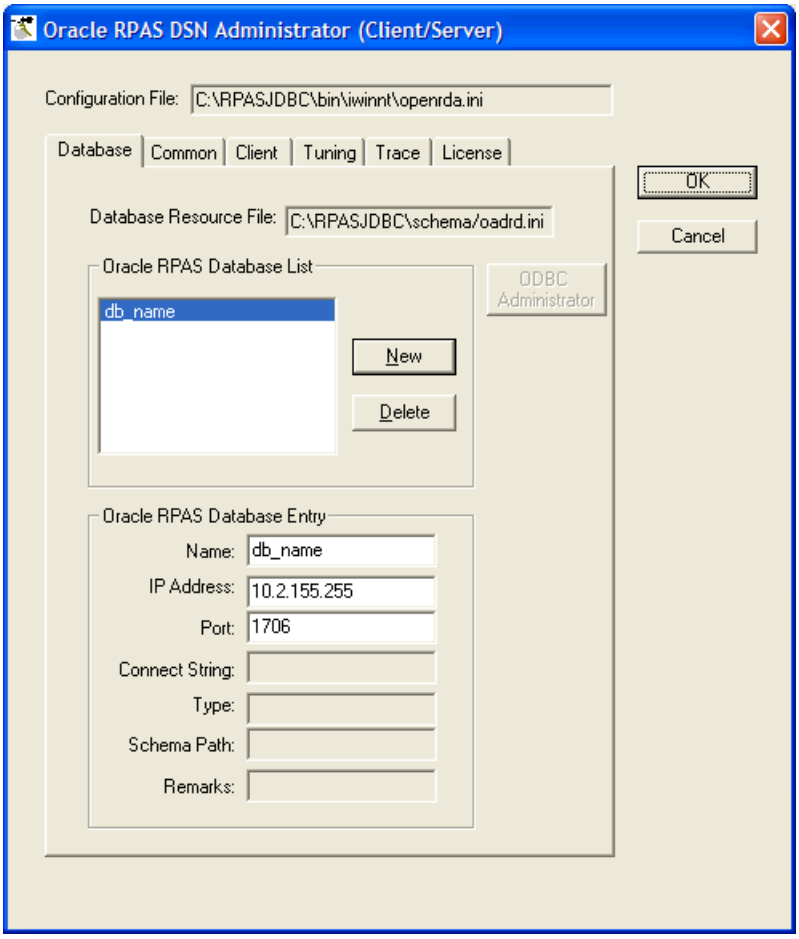
- Driver Class: jdbc.sql.oadriver
- URL: jdbc:RPASJDBC:<db_identifier>

<db_identifier> stands for the database identifier you specified during your installation or one that you specified using the Oracle JDBC Driver Administration Tool.

Creating a New Database Using the JDBC Client Administration Tool

To create a new database identifier using the Oracle JDBC Client Administration Tool, perform the following procedure:

Navigate: Select **Start – All Programs – Oracle RPAS JDBC Client – Administration Tool**. The Oracle RPAS DSN Administrator (Client) dialog appears.



Example of Oracle RPAS DNS Administrator Dialog Box (JDBC Client)

1. On the **Database** tab, click **New**. A new entry appears in the Oracle RPAS Database List.
2. Perform the following to define the new database:
   a. Enter a **Name** for the database, which is the database identifier. This name should be given to you from your RPAS ODBC/JDBC database administrator.
   b. Enter the TCP/IP address and the port number of the ODBC/JDBC server.
3. Click **OK** and you are ready to use a JDBC client application to connect to the database.

Using the jdbcisql Utility Provided with RPAS JDBC Driver

From the command line, type the following:

```
java jdbcisql -d jdbc.sql.oadriver -u jdbc:RPASJDBC:rpasdb
```

When the application starts, type the following to log in and make the connection (user name = adm; password = adm) as shown below.

```
Connect adm*adm@rpasdb
```

Using Oracle SQL Developer

Create an XML file with the following content:

```
<?xml version = '1.0'?>
<!DOCTYPE connections>
<connections>
   <connection>
      <URL>jdbc:RPASJDBC:JDBCSrc</URL>
      <ConnectionName>MyConnection</ConnectionName>
      <user>adm</user>
      <ConnectionType>OTHER_JDBC</ConnectionType>
      <JdbcDriver>jdbc.sql.oadriver</JdbcDriver>
   </connection>
</connections>
```

The URL should correspond to the URL specification required by the RPAS JDBC Driver as specified in the preceding sections. `ConnectionName` can be anything the user desires, and this field can be changed using the client application later. Enter a user name for the connection; this too can be changed later using the application. Leave the remaining information as shown in the code sample above. Save this XML file with any name you like.

In SQL Developer, using the Tools/Preferences/Database/Third Party Drivers, add oadriver.jar file to the list of third party drivers used by SQL Developer (SQL Developer does not look in the classpath for drivers). Then go to the Connection Navigator and right-click on Connections and then select Import Connections. Browse to the XML file, upon selection of which the dialog will display the list of connections you specified in the file. Choose your connection, in this case MyConnection, and you are ready to go.

Using Oracle JDeveloper

Perform the following procedure to use Oracle JDeveloper with the JDBC driver:

1. Start JDeveloper.
2. From the JDeveloper left panel, select the **Connections** tab.
3. Right-click on **Databases**, and select **New Database Connection**. The Create New Database Connection wizard appears.
4. On the first screen of Create New Database Connection wizard, enter a connection name, and choose **Third Party JDBC driver** for **Connection** Type.
5. On the second screen, enter user name and password, and then click **Next**.
6. On the third screen, perform the following:
   a. Click **New** to add the driver.
   b. Locate the library Oadriver.jar file and its path. Oadriver.jar is made available from Installation/Configuration of the Reporter Tool.
   c. Enter the RPAS JDBC Driver connection URL as specified at the beginning of this section.
   d. In the **Driver Class** field, enter **jdbc.sql.oadriver**.
7. Follow the instructions to finish creating the connection.

Using a Java Program

You can instantiate oadriver in your application by one of following method:

- `new oadriver();`
- `Class.forName ("jdbc.sql.oadriver").newInstance();`

Make sure *driver_home*/jdbc/oadriver.jar is included in the CLASSPATH. On Windows make sure oajdbc.dll path is defined in the system PATH environment variable. On Solaris, Linux & Digital Unix make sure oajdbc.so is defined in the LD_LIBRARY_PATH environment variable. On RS6000 make sure oajdbc.a is defined in the LIBPATH environment variable. On HP-UX make sure oajdbc.sl is defined in the SHLIB_PATH environment variable.

The Java code snippet below shows you how you can write a program that uses the driver.

Java Code Sample:

```
import java.sql.*;
import jdbc.sql.*;

public class RPASDriverTest {
    public RPASDriverTest() {
    }

    public static void main(String[] args) {
        try {

            if (args.length != 3) {
                System.out.println("Format:\n" +
                                   "java RPASDriverTest <Database> <UID>
<PWD>\n");
                return;
            }

            Connection conn = null;
            //Class.forName ("jdbc.sql.oadriver");
            new oadriver();

            String url      = "jdbc:RPASJDBC:";
            String database = args[0];
            String uid      = args[1];
            String pwd      = args[2];

            url += database;
            conn = DriverManager.getConnection(url, uid, pwd);

            DatabaseMetaData dma = conn.getMetaData();

            System.out.println("\nConnected to " + dma.getURL());
            System.out.println("Driver       " +
                    dma.getDriverName());
            System.out.println("Version      " +
                    dma.getDriverVersion());
            System.out.println("");

            // sample query
            String query = "SELECT * FROM DIM_ITEM";
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            rs.close();
            stmt.close();
```

```
            } catch (SQLException ex) {
                System.out.println ("\n*** SQLException caught ***\n");
                while (ex != null) {
                    System.out.println ("SQLState: " +
                                        ex.getSQLState ());
                    System.out.println ("Message:  " + ex.getMessage ());
                    System.out.println ("Vendor:   " +
                                        ex.getErrorCode ());
                    ex = ex.getNextException ();
                    System.out.println ("");
                }
            }
            catch (java.lang.Exception ex) {
                //Got some other type of exception.  Dump it.
                ex.printStackTrace ();
            }
        }
    }
```

Data Query

It is important to note the following limitations when performing data queries:

- All domain data queries are allowed only in the batch mode. This means that one cannot expect the driver to be querying a domain while users are connected to the domain using the RPAS Client. This limitation is due to the fact that the RPAS platform implements an exclusive lock policy that allows either a single write process to a measure or 1 or more read processes to a measure. This lock policy prohibits querying from a domain during online operations. The queries will starve all workbook commits and other domain updates. In order to query data during periods of online operations, a copy of the queried data must be established, and all queries must be made against the copy. The copy of data should be refreshed during a nightly batch.
- All workbook queries must be performed against closed workbooks (not open in RPAS Client). This limitation is required to avoid lock contention between the query process and workbook operations such as calculations and refreshes.

Re-using Aggregate RPAS Reporter Tool Queries

Aggregated results from previous RPAS Reporter Tool (ODBC driver) queries against a global domain or workbook are stored and can be reused. Aggregation results are stored by the user in the master domain in a folder called AggregateDatastore. Database and array names are constructed from measure names and the corresponding intersection at which data is stored. Aggregations are maintained throughout the life of a connection. At its termination, each connection cleans up all the aggregate data generated during its life time.

Metadata

Figure 1 shows the metadata tables available in a domain or a workbook. These tables can be used to examine the structure of the domain, such as:

- which measures and dimensions exist within the database
- which hierarchies exist and what is their rollup structure
- which fact tables are available
- which measures exist at the intersections that 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.

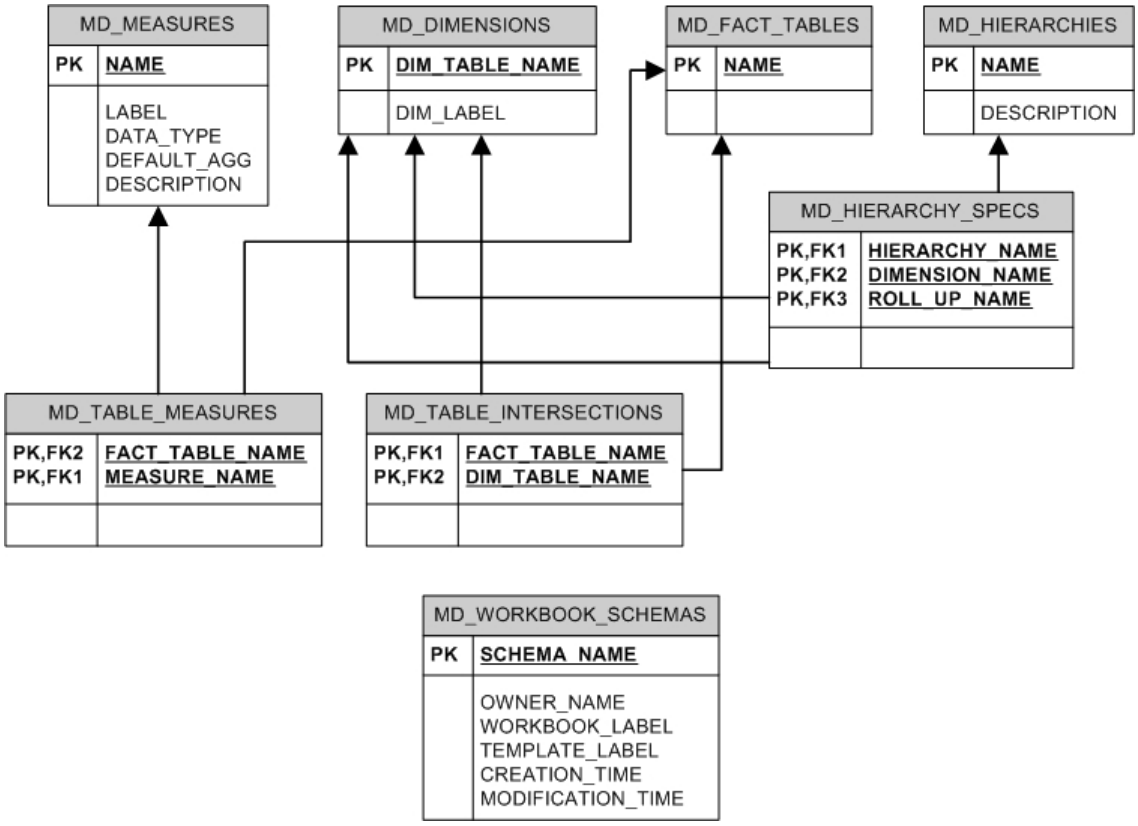


Figure 1: Database diagram for all metadata tables in a domain or in each workbook.

Note: MD_WORKBOOK_SCHEMAS table is not included in workbooks.

Fact and Dimension Tables

Figure 2 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 to maintain data integrity between the dimensions and the facts. Dimension tables have foreign key relationships with other dimension tables to represent hierarchical relationships between them.

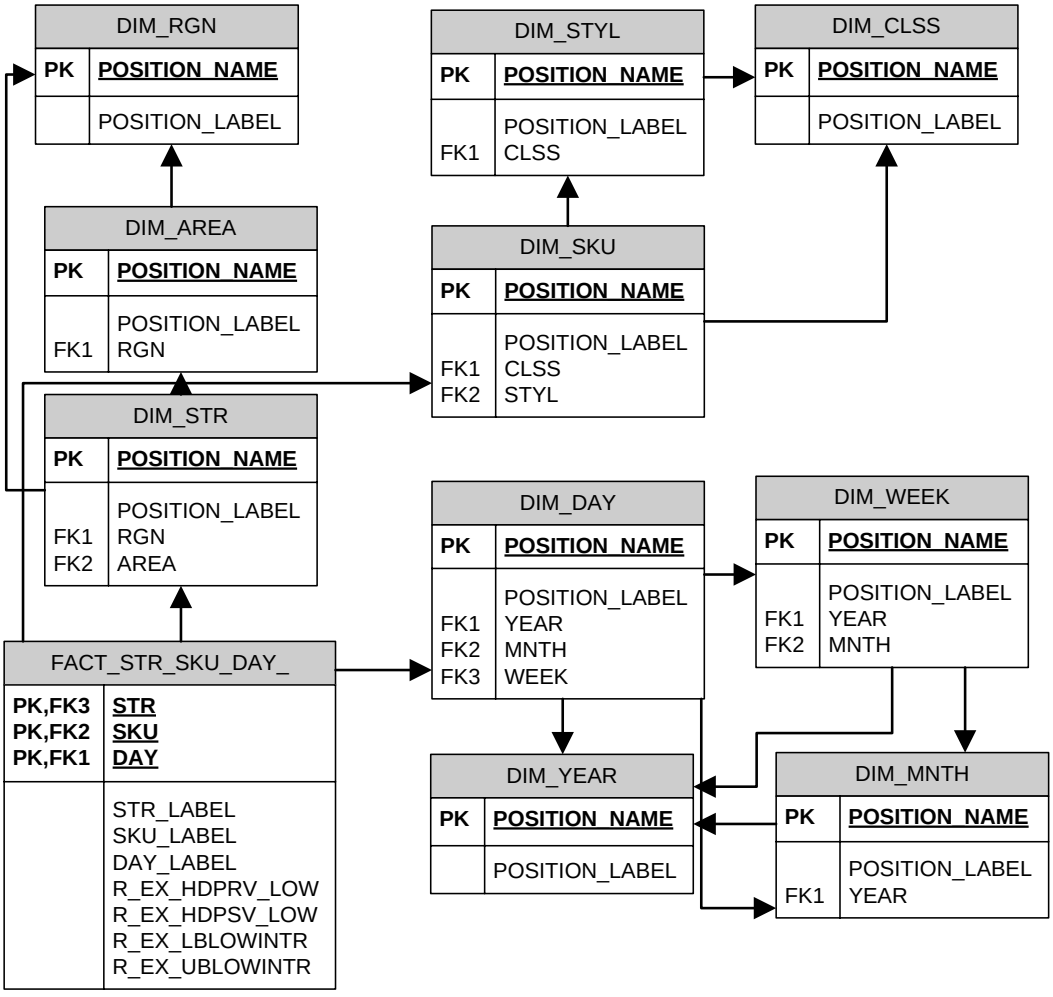


Figure 2: Example of Star – De-normalized Schema to Represent Facts and Dimensions in RPAS

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 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 will 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 some one 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 smaller than four characters, it is padded with '_' characters to make it four characters long.

Limitations

Queries with Many Conditions

Currently, there is a known issue with the driver processing queries with `where` clause that expands to large number of condition lists. An example of such queries would be as follows:

```
Select * from fact_str_sku_week
Where str in (select position_name from dim_str)
    and sku in (select position_name from dim_sku)
    and week in (select position_name from dim_week);
```

In order to process this query, the driver needs to process the subqueries first, then expand the `where` clause into lists of conditions, and the number of condition lists is the product of number of rows returned from the sub-queries. When the product is too large, the driver may not have enough resources to create the complete set of condition lists, and it will fail in that case. The driver SDK vendor DataDirect is currently in the process of resolving this issue.

Fortunately, the above example is not a common query reports developer will write (it's equivalent to "`select * from fact_str_sku_week`"). From our testing, we found the driver works fine if the number of condition lists does not exceed 10,000, which means the driver will work for queries similar to the following:

```
Select * from fact_str_sku_week
Where str in ('str1','str2','str3',…)
    and sku in ('sku1','sku2','sku3','sku4',…)
    and week in ('week1','week2','week3','week4','week5'…);
```

Such a query is valid as long as n1 x n2 x n3 does not exceed 10,000, where n1, n2, and n3 are the size of value sets of str, sku, week respectively.

Contention

All domain data queries are allowed only in the batch mode. This means that one cannot expect the driver to be querying a domain while users are connected to the domain using the RPAS Client. This limitation is due to the fact that the RPAS platform implements an exclusive lock policy that allows either a single write process to a measure or 1 or more read processes to a measure. This lock policy prohibits querying from a domain during

online operations. The queries will starve all workbook commits and other domain updates. In order to query data during periods of online operations, a copy of the queried data must be established, and all queries must be made against the copy. The copy of data should be refreshed during a nightly batch.

Workbook Queries

All workbook queries must be against saved workbooks. This limitation is required to avoid lock contention issues between the query process and workbook operations, such as calculations and refreshes.

Measure Security in the ODBC Driver

Before the existence of the ODBC/JDBC driver, an RPAS user could only use RPAS workbooks to access measures. Consequently, the ODBC/JDBC driver emulates the RPAS workbook security model to manage access rights to RPAS measures. It allows users to view all measures that they could view using the templates that they have access to.

This means that when they connect to a domain, they can see all measures that they could insert into a workbook. These include all measures that their templates have access rights too (managed through the use of the Workbook Template Measure Rights worksheet in the Security Administration workbook) and all measures to which the users have explicitly been given access rights using the Measure Rights worksheet in the Security Administration workbook. All other measures are not accessible to the users.

When they connect to a workbook, they can access all measures in the workbook, irrespective of how those measures were brought into the workbook and irrespective of whether access rights to some of those measures were removed after the workbook was created. Since those measures will exist in the workbook that the users can access, those measures (their workbook copies) will be accessible to the users.

Use Cases

Using Metadata Tables to Explore the Structure of a Domain or a Workbook

1. Fetch the list of workbook schema names given that the workbook is owned by 'USER01', is built using the template 'TestTemplate' and is labeled 'MyWorkbook':

```
Select
    SCHEMA_NAME, CREATION_TIME, MODIFICATION_TIME
From
    MD_WORKBOOK_SCHEMAS
Where
    OWNER_NAME = 'USER01' and
    WORKBOOK_LABEL = 'MyWorkbook' and
    WORKBOOK_TEMPLATE = 'TestTemplate'
```

   The SCHEMA_NAME obtained using this query can be directly used in custom properties of the driver configuration to enable direct connection to a workbook instead of a domain.

2. List all measures in the domain or workbook (default schema).

```
Select
    *
From
    MD_MEASURES
```

3. List all measures in a specific schema (for example, 'DOMAIN_T0').

```
Select
    *
From
    DOMAIN_T0.MD_MEASURES
```

4. List all dimensions in the domain or workbook (default schema).

```
Select
    *
From
    MD_DIMENSIONS
```

5. List all fact tables in the domain or workbook (default schema).

```
Select
    *
From
    MD_FACT_TABLES
```

6. List all hierarchies in the domain or workbook (default schema).

```
Select
    *
From
    MD_HIERARCHIES
```

7. List all fact tables with the measures that are represented in those tables (default schema).

```
Select
    *
From
    MD_TABLE_MEASURES
```

   List all fact tables with the dimension table names that intersect in the fact table (default schema).

```
Select
    *
From
    MD_TABLE_INTERSECTIONS
```

8. To understand the structure of a particular hierarchy (for example: CLND) the following table will list the hierarchy with each of its dimensions and the roll up dimension name for each one of them (default schema).

```
Select
    *
From
    MD_HIERARCHY_SPECS
Where
    HIERARCHY_NAME = 'CLND'
```

Querying Fact Data

1. Query fact data for all measures at the STR-SKU-DAY intersection with the unique position names for these dimensions

```
Select
    *
From
    FACT_STR_SKU_DAY_
```

2. Query fact data for specific measures at the STR-SKU-DAY intersection and list them with the position labels for each dimension

```
Select
DS.POSITION_LABEL, DU.POSITION_LABEL, DD.POSITION_LABEL, R_EX_LBLOWINTR,
R_EX_UBLOWINTR
From
    FACT_STR_SKU_DAY_ F,
    DIM_STR DS,
    DIM_SKU DU,
    DIM_DAY DD
Where
    DS.POSITION_NAME = F.STR and
    DU.POSITION_NAME = F.SKU and
    DD.POSITION_NAME = F.DAY
Hint Join (FACT_STR_SKU_DAY_, DIM_STR, DIM_SKU, DIM_DAY);
```

> **Note:** The optional "`Hint`" clause in the above SQL statement is not ANSI SQL standard, but the Reporter Tool supports it. This "`Hint`" tells the driver to process the join tables in the specified order (fact table first, and then dimension tables).

Connecting to a Workbook

1. Go to **Start – Settings – Control Panel – Administrative Tools – Data Sources (ODBC)**.
2. Select the **System DSN** tab. Select the appropriate DSN and click **Configure**.
3. In the **Options** frame enter WORKBOOK_SCHEMA=<workbook schema name>. Replace '<workbook schema name>' with the workbook schema name for the workbook you want to connect to. The workbook schema names can be obtained by first connecting to the domain and then examining the MD_WORKBOOK_SCHEMAS table to obtain the schema name for the appropriate workbook (may be identified by owner name, template, creation and last modification time). For example: 'WORKBOOK_SCHEMA=DOMAIN_T0' or 'WORKBOOK_SCHEMA=SD0_T0'
4. Click **OK**.

Requesting Additional Aggregate Tables

1. Select **Start – Settings – Control Panel – Administrative Tools – Data Sources (ODBC)**.
2. Select the **System DSN** tab. Select the appropriate DSN and click **Configure**.
3. In the **Options** frame enter AGG_TABLE_NAMES=<comma separated list of any additional aggregate fact table names>.

   By default, the database includes every fact table (a fact table represents an intersection) that one or more measures have as their base intersection. Any other fact tables can be specifically requested by adding a comma-separated list as the value for this custom property. For example, if someone wants to see a fact table for the intersections 'DEPT' and 'DEPT_YEAR', the value of this custom property would be 'AGG_TABLE_NAMES=FACT_DEPT, FACT_DEPT_YEAR'.
4. Click **OK**.

   If entering more than one connection property (that is, both the WORKBOOK_SCHEMA and AGG_TABLE_NAMES properties), the property key value pairs must be separated by a semicolon. Using examples above, the content of the custom properties input box would appear as follows:

   WORKBOOK_SCHEMA=DOMAIN_T0**;**AGG_TABLE_NAMES= FACT_DEPT, FACT_DEPT_YEAR

Clients

This section lists some sample ODBC/JDBC client applications that can connect to the RPAS data store through the RPAS Reporter Tool. The examples in this section do not include all client applications that can connect to the RPAS Reporter Tool.

> **Note:** In client/server configuration, the server (executable) must be started before a client can connect to it.

Oracle Business Intelligence Enterprise Edition (OBIEE)

In this section, we only outline how to connect to the defined DSN using the OBIEE Administration Tool, and how to import data from the DSN. For more information about OBIEE, please refer to OBIEE documentation.

To connect OBIEE to a predefined DSN:

1. Make sure the following Windows services are running:
   - Oracle BI Java Host
   - Oracle BI Server
2. Start the OBIEE Administration Tool (**Start – All Programs – Oracle Business Intelligence – Administration**).
3. From File menu, choose **open – online**. A window appears to enter log in credentials.
4. Enter administrator's user name and password, and then click **open**.

   Three panels now appear in the Admin Tool window: **Presentation**, **Business Model and Mapping**, and **Physical**.
5. From File menu, choose **Import–From database**. A window appears select the connection type and RPAS user information.
6. Select **ODBC 3.5** for Connection Type, enter the RPAS user name and password, and then click **OK**.

   The RPAS schemas and tables appear in a new window.
7. Select the objects you want to import, and then click **import**. Once the import is complete, click **close**.

   A new physical model is created and listed in the **Physical** panel of the Admin Tool window.
8. Expand the physical model, double-click on **connection** to open "connection" properties, make sure the **Connection Type** is set to **ODBC 3.5**, and then click **OK** to exit.
9. In the Admin Tool window, click **Save** to save your physical model.

   Now you have a basic physical model, you can build the business model and presentation layer on top of it. For more information on the business model, the presentation layer, and the OBIEE Web interface, refer to OBIEE documentation.

Configuring the ODBC Client for OBIEE

The following example is provided as a sample of configuring the ODBC client for OBIEE. This example was developed for OBIEE on AIX, but the process is the same for other environments.

1. Open $BIEE_HOME/setup/odbc.ini file where $BIEE_HOME is the directory where OBIEE is installed.
2. In the `[ODBC Data Sources]` section, insert an entry for RPAS domain.

   Example:

   ```
   rpas_domain=This is the name of the data source for RPAS.
   ```

   The name here (`rpas_domain`) should be the same as the data source name configured in the RPAS ODBC Server.

3. Create a section in the file for the `rpas_domain`.

   Example:

   ```
   [rpas_domain]
   Driver=absolute_path_to_odbc_client/oaodbc.so
   DriverUnicodeType=1
   Description=Rpas domain data source
   ```

4. Save your changes to the file.

Microsoft Access

To connect using Microsoft Access:

1. Start Microsoft Access.
2. Create a new (or open an existing) Access file (.mdb file).
3. From File menu, select **Get External Data– Link Tables** (or **Import** if you really want to import the data from RPAS data store to Access). A dialog box appears.
4. In the Files of type box, select **ODBC Databases()**.
5. Click **Machine Data Source** tab, and then double-click the pre-configured ODBC data source that you want to link from.
6. At the logon prompt, type your user ID and password, and then click **OK**.

   At this point, MS Access connects to the RPAS data source and displays the list of schemas/tables that you can import or link.
7. Click each table that you want to import or link, and then click **OK**. If you're linking a table and it doesn't have an index that uniquely identifies each record, then Microsoft Access displays a list of the fields in the linked table. Select a field, or a combination of fields, that will uniquely identify each record, and then click **OK**.

JDeveloper

JDeveloper works best with a native JDBC driver, which is included in the RPAS Reporter Tool package.

To connect using JDeveloper:

1. Start JDeveloper.
2. On the left panel JDeveloper, select the **Connections** tab.
3. Right-click on **Databases**, and select **New Database Connection**.
4. On the first screen of Create New Database Connection wizard, enter a connection name, choose **Third Party JDBC driver** for Connection Type.
5. On the second screen, enter user name and password, and then click **Next**.
6. On the third screen, click **New** to add the driver. You will need to find the library Oadriver.jar and its path (Oadriver.jar is made available from Installation/Configuration of the Reporter Tool); enter **jdbc:OpenRDA:testDomainJ** for URL.
7. Follow the instructions to finish creating the connection.

   Once the connection is established, the user can expand the Connection (and the nodes under the Connection) to browse the objects in the RPAS data store. The user can also open a SQL worksheet (by selecting **SQL Worksheet** from the Tools of the menu) to write/execute SQL statements.

XML Publisher

This section briefly describes how to make the connection from XML Publisher to RPAS using JDBC driver (XML only supports JDBC).

1. Install and configure JDBC client driver and ODBC/JDBC server. Start the server.
2. Copy JDBC client driver oadriver.jar (from JDBC client installation) to D:\OraHome_1\oc4j\j2ee\home\applib, where D:\OraHome_1 is the root directory where XML Publisher was installed.
3. Start XML Publisher server (**Start – All Programs – Oracle XML Publisher Server – OUIHome1 – Oracle XML Publisher Enterprise Start**).
4. Start a Web browser, go to the URL: http://localhost:15101/xmlpserver/

   This URL is only an example. Contact the XML Publisher administrator/installer for the actual URL. The actual URL is recorded in D:\OraHome_1\xmlpserver\setupinfo.txt file of the XML Publisher server machine.
5. Login as admin/admin.
6. Select **Admin** tab, then select **JDBC Connection** under Data Sources to create a JDBC connection.
7. Click **Add Connection** to create a new connection and provide the following information:
   - Enter a display name for Data Source Name.
   - Enter **jdbc:OpenRDA:BTree_CS** for the URL, where **Btree_CS** is the DSN name set up in JDBC client configuration.
   - Enter **adm** for user name and password.
   - Enter "**jdbc.sql.oadriver**" for Database Driver Class.
8. Click **Test Connection**. The confirmation message: "connect established successfully" should appear.
9. Click **apply** to save the connection.

Interactive SQL (ISQL) Utility

ISQL is an interactive SQL tool that is provided by the OpenAccess ODBC/JDBC SDK. There are different versions of ISQL:

- To connect to the Local ODBC version of the Reporter Tool, use odbcisql.exe by selecting **Start – All Programs – Oracle RPAS ODBC Client – Interactive SQL (ODBC)**.
- To connect to Local JDBC version of the Reporter Tool, use jdbcisql.class.
- To connect the Reporter Tool Server located on the same machine without installing ODBC client, use oaisql.exe.
- To connect to remote Reporter Tool server, use odbcisql.exe (with ODBC Client installed) or jdbcisql.class (with JDBC client installed).

Note: Users are expected to know basic SQL to use ISQL.

To connect to the Reporter Tool using odbcisql, start odbcisql then, at the SQL prompt, issue the connect command as follows:

```
connect john/doe@rpasDomain
```

Where `john/doe` is a predefined administrator account in RPAS, and `rpasDomain` is a pre-configured Data Source Name.

Issue the connect command for jdbcisql.class as follows:

```
connect john*doe@rpasDomain
```

Once connected, users can issue various SQL DML and DDL statements to inspect and modify the data in RPAS data store.

A

Appendix: Integration Guide

RPAS, RDF and Planning Integration with RMS and Price

Summary of Integration Approach with RMS

The strategy for the extraction of foundation data from RMS is for the extract programs (RMSE) to provide flat files in a generic format. For each solution that will use this data, transformation scripts are used to reformat the data as needed to produce a file suitable for loading into the application. For the instances of data coming from RPAS to non-RPAS applications, extract programs are specific to the application in need of the data. Other scripting languages are then used (Perl or AWK) to perform additional data formatting.

This appendix summarizes the following:

- RMS 11.0/12.0 to RDF 12.1 and Merchandise Financial Planning 12.1 transformation programs
  - Merchandise Hierarchy
  - Organization Hierarchy
  - Calendar Hierarchy
  - Store Close Dates
  - Store Open Dates
  - Daily Sales and Issues
  - Weekly Sales and Issues
  - Out of Stock Indicator
- RDF 12.1 to RMS 11.0/12.0 extract programs
  - Approved Forecasts and Standard Deviations (Cumulative Intervals)
- Grade (RPAS 12.1) to RMS 11.0/12.0 extract programs
  - Store Grades
- Curve (RPAS 12.1) to RMS 11.0/12.0 extract programs
  - Differentiator Profiles
- Merchandise Financial Planning 12.1 to RMS 11.0/12.0 extract programs
  - Receipt Plan
- Merchandise Financial Planning 12.1 to Price 4.5 extract programs
  - Markdown Budget
- Merchandise Financial Planning 12.1 to RDW 12.0 extract programs
  - Current Plan
  - Original Plan

Specifics on the usage of RMS extract programs (RMSE's) within the RDF transformation programs are beyond the scope of this document. See the *RMS Operations Guide* for more information on the RMS extract programs.

Environment Variable Setup

In addition to any variables identified in the RMS integration documentation, the transformation and/or extract programs require the following environment variables:

- $RPAS_INTEGRATION_HOME: Identifies the location of the integration scripts when /common/header.ksh is run. This variable is used for all integration scripts packaged with the ARPOPlatform EXCEPT those included in 'rfx' (see $RDF_HOME below).
- $TO_RPAS: The staging area for the data to be loaded into RPAS. This directory should be located at the same level as the root of the RPAS domain. For example, if the domain RDF is located in Domains directory (example: /Domains/RDF), then $TO_RPAS should located at the same level as RDF (example: /Domains/to_rpas).
- $FROM_RPAS: The staging area for the data extract out of RPAS. This directory should be located at the same level as the root of the RPAS domain. For example, if the domain RDF is located in Domains directory (example: /Domains/RDF), then $FROM_RPAS should located at the same level as RDF (example: /Domains/from_rpas).
- $RDF_HOME: Identifies the location of the root of the 'rfx' directory. The 'rfx' directory packaged with the ARPOPlatform should be added to the location 'rfx' directory packaged with the RMS RETL programs.
- $RI_RMSVERSION: Identifies the version of RMS. If this variable is not set, the integration scripts assume an RMS version of 12. Set the value of this environment variable to 11 or 12, depending on your RMS version.

  If integrating with RMS 11, the $RI_RMSVERSION is set to 11 and the integration scripts expect a 2 digit Domain ID. If integrating with RMS 12, a 3 digit Domain ID is expected.

RDF and Merchandise Financial Planning Transformation Programs

Common Program for All Transformations

The **rdft.ksh** script runs all of the necessary data extraction and transformation scripts (**rmse_*.ksh** and **rdft_*.ksh**, respectively) that are needed to produce the files to be loaded into RPAS/RDF/Planning. Most of these scripts are run in parallel (as background jobs).

Usage

```
rdft.ksh  [-x]  [-c]  [-d dir]
```

Arguments:

- `-x`: This option will cause the execution of the RMS data extraction wrapper (rmse.ksh) to be skipped.
- `-c`: This option will cause FILE_DATE in rmse_config.env to be set to the current date instead of using VDATE.
- `-d`: This option will cause all programs executed by rdft.ksh to be obtained from the "dir" directory.

Transformations of Merchandise Hierarchy Data

rdft_merchhier.ksh is the primary script used to build the data for RPAS from the RMS Merchandise Hierarchy tables. The schema used to produce the output file depends on the attributes and differentiator settings in RMS:

- Case 1: If PROD_ATTRIBUTES_ACTIVE = False and DIFFS_ACTIVE = False, then rdft_merchhier.base.schema will be used to produce the file. In this case attributes and diff fields will not be included in the merchandise hierarchy file.
- Case 2: If PROD_ATTRIBUTES_ACTIVE = True and DIFFS_ACTIVE = False, then rdft_merchhier.attributes.schema will be used to produce the file. This schema must be manually edited to support a specific attribute model and must be kept in sync with rmse_attributes.schema and rmse_attributes.ksh (see the RMSE end user documentation).
- Case 3: If PROD_ATTRIBUTES_ACTIVE = False and DIFFS_ACTIVE = True, then rdft_merchhier.schema will be used to produce the file. In this case diff fields will be included in the merchandise hierarchy file.
- Case 4: If PROD_ATTRIBUTES_ACTIVE = True and DIFFS_ACTIVE = True, then an error will result. In this release, the combination of diffs and attributes is not supported.

Intermediate schema and scripts which may be used (depending on configuration options) to produce the merchandise hierarchy file:

- **rdft_diff.domain.schema**
- **rdft_merchdiff.domain.schema**
- **rdft_merchhier_diff_trans.ksh**
- **rdft_merchhier_split_by_domain.ksh**
- **rdft_clean_partition.ksh**

Additional merchandise hierarchy support for issue domains is provided in **rdft_item_loc.ksh**. This script is designed to produce a full item list for issues domains, only containing items that exist in the warehouses.

> **Note:** Issues-specific data transformation functionality is triggered based on the issues setting in RMS (ISSUES_ACTIVE must be set to True).

Transformations of Location Hierarchy Data

rdft_orghier.ksh is the primary script used to build the location data file needed for RPAS from the RMS Organizational Hierarchy Table.

The following five constants may be modified in the script based on location hierarchy data requirements:

- **COMPANY_NAME** – The label for the company position to be populated in the file.
- **COMPANY_ID** – The name for the company position to be populated in the file.
- **STORE_CLASS_CONCAT** – When set to "True", causes the STORE_CLASS to be concatenated on the left of the STORE_CLASS_DESCRIPTION field in the final Store data output file.
- **ADD_AT_SIGN_TO_WH_DESC** – When set to "True", will cause the WHSE_NAME field in the Warehouse ouput file to have an "@" prefix.
- **LONG_WAREHOUSE_RECORDS** – When set to "True", the Warehouse output records will consist of 16 fields. If it is "False", the records will contain only four fields, WH, WHSE_NAME, COMPANY and CO_NAME.

Intermediate schemas which may be used (depending on configuration options) to produce the location hierarchy file:

- rdft_issues.schema
- rdft_issues_long.schema
- rdft_orghier_store.schema

> **Note:** Issues-specific data transformation functionality is triggered based on the issues setting in RMS (ISSUES_ACTIVE must be set to True).

Transformations of Calendar Hierarchy Data

rdft_calhier.ksh transforms the Calendar Hierarchy data extracted from RMS for loading into RPAS.

Configuration inputs to the script include:

- **DATE_PREF** – The path to the file that contains text indicating whether the format of the Date Description field will be mm/dd/yyyy or dd/mm/yyyy. See the *RMS Operations Guide* for date format options.
- **LAST_DOW** – The path to the file that contains a day of week name or abbreviation indicating which day of the week is considered to be the end of the week for the fiscal calendar being used at this installation.

Transformations of Daily Sales and Issues Data

rdft_daily_sales.ksh produces the daily sales and issues data files based on regular, promotion, clearance, and issues.

The following constant may be modified in the script based on data requirements:

- **DOM_START_COL** – Defines the starting column position of the Domain ID in the RETL output schema. This is needed by **rdft_merchhier_split_by_domain.ksh** to split the files by domain ID. If the OUTPUT_SCHEMA file is modified, the value of DOM_START_COL may also require modification from the default value.

Intermediate schemas which may be used (depending on configuration options) to produce the sales and/or issues data file:

- rdft_daily_sales.schema

> **Note:** Issues-specific data transformation functionality is triggered based on the issues setting in RMS (ISSUES_ACTIVE must be set to True).

Transformations of Weekly Sales and Issues Data

rdft_weekly_sales.ksh produces the weekly sales and issues data files based on regular, promotion, clearance and issues.

The following constant may be modified in the script based on data requirements

- **DOM_START_COL** – Defines the starting column position of the Domain ID in the RETL output schema. This is needed by rdft_merchhier_split_by_domain.ksh to split the files by domain ID. If the OUTPUT_SCHEMA file is modified, the value of DOM_START_COL may also require modification from the default value.

Intermediate schemas which may be used (depending on configuration options) to produce the sales and/or issues data files:

- rdft_weekly_sales.schema

> **Note:** Issues-specific data transformation functionality is triggered based on the issues setting in RMS (ISSUES_ACTIVE must be set to True).

Transformations of Store Open Date Data

rdft_open_date.ksh produces the Store/Warehouse Opening Date data file.

Intermediate schema used to produce the store open date data files:

- rdft_open_date.schema

Transformations of Store Close Date Data

rdft_close_date.ksh produces the Store/Warehouse Closing Date data file.

Intermediate schema used to produce the store closing date data files:

- rdft_close_date.schema

Transformations of Out of Stock Indicator Data

rdft_outofstock.ksh produces the Store and Warehouse (issues) Out of Stock Indicator data extracted from RMS.

Intermediate schema and scripts which may be used (depending on configuration options) to produce the Out of Stock Indicator data file:

- rdft_outstock_split_by_domain.awk
- rdft_outofstock.schema
- rdft_outofstock_issues.schema
- rdft_outofstock_sales.schema

RDF and Merchandise Financial Planning Transformation Matrix

The following matrix identifies the transformation scripts and schemas used for each the hierarchy and data files produced for RDF 12.1 and Merchandise Financial Planning 12.0:

Directory	Script or Schema Name	Merchandise Hierarchy	Location Hierarchy	Calendar	Daily Sales & Issues	Weekly Sales & Issues	Out of Stock Indicator	Store Open Dates	Store Close Dates
rfx/lib	rdft_merchhier_diff_trans.ksh	X							
	rdft_merchhier_split_by_domain.ksh	X							
	rdft_outofstock_split_by_domain.ksh						X		
rfx/schema	rdft_close_date.schema								X
	rdft_daily_sales.schema				X				
	rdft_diff.domain.schema	X							
	rdft_merchierdiff.domain.schema	X							
	rdft_merchier.attributes.schema	X							
	rdft_merchhier.base.schema	X							
	rdft_merchhier.domain.schema	X							
	rdft_merchhier.schema	X							
	rdft_open_date.schema							X	
	rdft_orghier_issues.schema		X						
	rdft_orghier_issues_long.schema		X						
	rdft_orghier_strore.schema		X						
	rdft_outofstock.schema						X		
	rdft_outofstock_issues.schema						X		

Directory	Script or Schema Name	Merchandise Hierarchy	Location Hierarchy	Calendar	Daily Sales & Issues	Weekly Sales & Issues	Out of Stock Indicator	Store Open Dates	Store Close Dates
	rdft_outofstock_sales.schema						X		
	rdft_weekly_sales.schema					X			
rfx/src	rdft_ksh	X	X	X	X	X	X	X	X
	rdft_calhier.ksh			X					
	rdft_clean_partition.ksh	X							
	rdft_close_date.ksh								X
	rdft_daily_sales.ksh				X				
	rdft_item_loc.ksh	X							
	rdft_merchhier.ksh	X							
	rdft_open_date.ksh							X	
	rdft_orghier.ksh		X						
	rdft_outofstock.ksh						X		
	rdft_weekly_sales.ksh					X			

Common Programs for Extracts

config.ksh is a configuration directory that requires both the RMS version being integrated and the backup action to be defined.

The following OPTIONAL arguments are available:

- Name of the domain: Defaults to directory name
- Number of the domain: Defaults to the 2 last digits of the directory name
- Format of timestamp attached to logs and processed input files: Defaults to: (date +"%b%d%a%I%M%p") (example: Aug02Thu0111PM)
- Data Drop: Defaults to ../../to_rpas
- Data Export: Defaults to ../../from_rpas
- Log Drop: Defaults to ./logs
- Error Drop: Defaults to ./err
- Reclass Data: Defaults to ../reclass_data

functions.ksh

This script file contains ksh functions that are used by scripts in [DOM]/scripts. It should be sourced, not executed in order to preserve environment variables.

header.ksh

This script should be run at the beginning of any implementation-specific script to setup function libraries, environment, and platform-specific routines.

Extract of Forecast Data for RMS

rdf_e_rms.ksh extracts forecast demand value and standard deviation (cumulative interval) at both day and week aggregations from an RDF domain.

Arguments:

- -t: <Domain Type> (S for sales, I for issues)
- -w: <Data Width> ([7...18], defaults to 12)
- -d: <Domain> (defaults to current directory)
- -n: <Domain Number> (defaults to last two digits of domain)

Output files:

${RPAS_EXPORT}/d<s|i>demand.<Domain Number> (demand at day)

${RPAS_EXPORT}/w<s|i>demand.<Domain Number> (demand at week)

The following table provides information about the output file data format.

Field	Start	Width	Format
Day | EOW Day	1	8	Alpha
Product ID	1	25	Alpha
Location ID	26	20	Alpha
Demand	46	12	Alpha
Std. Dev. Demand	68*	12*	Numeric (floating point, 4 decimal digits with decimal)

* Width of Demand and Std. Dev. Demand may be overridden with the -w parameter; stated values Demand width and Std. Dev. Demand start and width are based on default width of 12.

Note: the following must be defined in the shell environment prior to calling this script:

– RPAS_HOME

– RPAS_INTEGRATION_HOME

Load of Extracted Forecast Data and Standard Deviations to RMS

rmsl_forecast.ksh pulls the daily/weekly forecast items into RMS.

During the loading of each domain file the following steps are performed:

1. Truncate the partition in the RMS forecast table which corresponds to the domain ID.

   Note: Partition names should always be in the format: [tablename]_[domainID]

2. Append a domain field and insert the domain_id into each record.
3. Load the forecast data into the RMS forecast table.

   Example: `rmsl_rpas_forecast.ksh daily | weekly`

Intermediate schemas which may be used (depending on configuration options) to produce the forecast data files:

- rmsl_forecast_daily.schema
- rmsl_forecast_weekly.schema

Extract of Diff Profile Data for RMS

profile_e_alloc.ksh extracts Curve diff profiles for use by Allocation.

Arguments:

- `-p`: <Profile Number>
- `-m`: <Mask Measure> (Optional mask; only positions for which the mask value is non-NA will be exported.)
- `-w`: <Data Width> ([7...18], defaults to 12)
- `-d`: <Domain> (defaults to current directory)
- `-n`: <Domain Number> (defaults to last two digits of domain)

Output file: ${RPAS_EXPORT}/d1<Product Level>.<Domain Number>, where Product Level is the Aggregation intersection's Prod dimension

The following table provides information about the output file data format.

Field	Start	Width	Format
Product ID	1	25	Alpha
Location ID	26	20	Alpha
Diff ID (optional)	46	36	Alpha
Quantity	82	12*	Numeric (floating point, 4 decimal digits, no decimal)*
Std. Dev. Demand	68*	12*	Numeric (floating point, 4 decimal digits with decimal)

* Quantity width may be overridden with the `-w` parameter.

Note: The following must be defined in the shell environment prior to calling this script:

– RPAS_HOME

– RPAS_INTEGRATION_HOME

Extract of Store Grade Data for RMS

grade_e_rms.ksh extracts store grades for use by RMS.

Arguments:

- `-t` <Timestamp> (YYMMDDTTTT). This value corresponds to the timestamp of the Cluster Membership measure (clpm+<Timestamp>) to be extracted
- `-d` <Domain> (defaults to current directory)
- `-n` <Domain Number> (defaults to last two digits of domain)

Output file: ${RPAS_EXPORT}/gr<Timestamp>.<Domain Number>

Output file data format:

- (a thru c) constitutes the header records
- (d thru j) constitutes the detail records
- (k thu l) constitutes the footer records
  - **a.** 'FHEAD'
  - **b.** Line ID Number
  - **c.** 'GRADU'
  - **d.** 'FDETL' (Record Identifier)
  - **e.** Line Sequence Identifier
  - **f.** Grade Group ID Number: This value corresponds to the first 8 characters of the Cluster Run Name measure (clnam+<user-defined name>) set by the user in the Generate Cluster wizard in Grade. For integration with RMS, the Cluster Run Name must be populated with only numeric characters.
  - **g.** Grade Group: This value corresponds to the first "**N**" characters of the Cluster Run Name measure (clnam+<user-defined name>) set by the user in the Generate Cluster wizard in Grade. "**N**" is 20 for RMS version 11.0, and "**N**" is 120 for RMS version 12. The script determines the RMS version from the environment variable RI_RMSVERSION. RMS version 12 is assigned by default.
  - **h.** Store ID
  - **i.** Grade Member Name
  - **j.** 'FTAIL' (Record Identifier)
  - **k.** Line ID Number
  - **l.** FDETL Line Total Number

Extract of Receipt Plan for RMS

plan_e_alloc.ksh extracts Merchandise Financial Planning measures for use by RMS

Arguments:

- `-m`: <Measure Name>
- `-f`: <Dif Dimension>
- `-w`: <Data Width>] ([7...18], defaults to 12)
- `-d`: <Domain> (defaults to current directory)
- `-n`: <Domain Number> (defaults to last two digits of domain)

Output file: ${RPAS_EXPORT}/p1<Prod Dimension>.<Domain Number>

The following table provides information about the output file data format.

Field	Start	Width	Format
Product ID	1	25	Alpha
Location ID	26	20	Alpha
Diff ID	46	36	Alpha (For future use; blank now)
EOW Date	82	8	Alpha
Quantity	90	12*	Numeric (floating point, 4 decimal digits, no decimal)*

* Quantity width may be overridden with the `-w` parameter.

Note: The following must be defined in the shell environment prior to calling this script:

– RPAS_HOME

– RPAS_INTEGRATION_HOME

Extract of Markdown Budget Data for Price

plan_e_price.ksh extracts Markdown Planning measures for use by Price.

Arguments:

- -a: <measure name for Markdown Budget>
- -b: <measure name for Planned GM Dollars>
- -c: <measure name for Planned GM %>
- -e: <month format> (The month position format in the domain, must be one of the following:
  1. "PYYYYMM", "PMMYYYY", "PMM_YYYY", "PYYYY_MM", "PYYYYMMM", "MMMYYYY","PYYYY_MMM", "MMM_YYYY"
  2. P represents a prefix letter required by RPAS for certain calendar formats. MM represents 01, 02, 03, ...12
  3. MMM represents JAN, FEB, MAR,..DEC. YYYY represents 2005, 2006, ...
- -d: <Domain> (The domain path has to be a master domain or a simple domain. A subdomain path will fail)
- -o: <outputfile name> (including file path)
- -m: <Optional: space separated list of additional measures to include> ("measure_x measure_y measure_z")

Example:

```
sh plan_e_price.ksh -a mdbudget -b plgmbudget -c plgmperc -e MMM_YYYY -d . -m rsal psal csal -o ASH_BUDGET_TBL.dat
```

Data Assumptions:

1. All measures to be extracted include month in their base intersection
2. All measures to be extracted are configured at the same base intersection
3. All measure to be extracted are aggregated to Price Zone

The extract produces the following pipe (|) delimited data fields, each row having the format described in the following table.

Field	Max Width	Format
Product ID	25	Alpha
Location ID	25	Alpha
Year ID	4	Alpha
Month ID	2	Alpha
MARKDOWN_BUDGET	22	Numeric (floating point, 3 decimal digits)
PLANNED_GM_DOLLARS	22	Numeric (floating point, 3 decimal digits)
PLANNED_GM_PERC	22	Numeric (floating point, 3 decimal digits)
Measure Name1	22	Numeric (floating point, 3 decimal digits)
Measure Name2	22	Numeric (floating point, 3 decimal digits)

Field	Max Width	Format
Measure Name3	22	Numeric (floating point, 3 decimal digits)
Measure Name4	22	Numeric (floating point, 3 decimal digits)
Measure Name5	22	Numeric (floating point, 3 decimal digits)
Measure Name6	22	Numeric (floating point, 3 decimal digits)
Measure Name7	22	Numeric (floating point, 3 decimal digits)
Measure Name8	22	Numeric (floating point, 3 decimal digits)
Measure Name9	22	Numeric (floating point, 3 decimal digits)
Measure Name10	22	Numeric (floating point, 3 decimal digits)

Note: The following must be defined in the shell environment prior to calling this script:

– RPAS_HOME

– RPAS_INTEGRATION_HOME

Extract of Current Plan Data for RDW

plan_e_plcblwdm.ksh extracts Current Plan data from Merchandise Financial Planning for use by RDW.

The script expects an inputfile as an argument. The inputfile specifies the hierarchy labels and measure names that need to be extracted by the script. The user is expected to create this input file and this is used by the script to extract the appropriate measures.

Arguments

- `-d` <Domain> (The domain path has to be a masterdomain or a simple domain. A subdomain path will fail)
- `-o` <outputfile> (Including file path)
- `-i` <inputfile with field to measure mapping> (Including file path)

  Input file should contain all the fields below followed by a space and the associated measure name.

  Example of input file data format:

```
DEPT_IDNT DEPARTMENT_LABEL
CLASS_IDNT CLASS_LABEL
SBCLASS_IDNT SBCLASS_LABEL
LOC_IDNT LOCATION_LABEL
F_PLN_CURR_CLRC_SLS_QTY <MEASURE_NAME>
F_PLN_CURR_PRMTN_SLS_QTY <MEASURE_NAME>
F_PLN_CURR_RGLR_SLS_QTY <MEASURE_NAME>
F_PLN_CURR_CLRC_SLS_AMT <MEASURE_NAME>
F_PLN_CURR_PRMTN_SLS_AMT <MEASURE_NAME>
```

F_PLN_CURR_RGLR_SLS_AMT <MEASURE_NAME>
F_PLN_CURR_GRS_PRFT_AMT <MEASURE_NAME>
F_PLN_CURR_RGLR_MKDN_AMT <MEASURE_NAME>
F_PLN_CURR_CLRC_MKDN_AMT <MEASURE_NAME>
F_PLN_CURR_PRMTN_MKDN_AMT <MEASURE_NAME>
F_PLN_CURR_SHRK_QTY <MEASURE_NAME>
F_PLN_CURR_SHRK_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_BOP_QTY <MEASURE_NAME>
F_PLN_CURR_BOP_COST_AMT <MEASURE_NAME>
F_PLN_CURR_BOP_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_OTB_QTY <MEASURE_NAME>
F_PLN_CURR_OTB_COST_AMT <MEASURE_NAME>
F_PLN_CURR_OTB_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_RCPTS_QTY <MEASURE_NAME>
F_PLN_CURR_RCPTS_COST_AMT <MEASURE_NAME>
F_PLN_CURR_RCPTS_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_CMTS_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_ORD_CNCLLD_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_ORD_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_RECL_IN_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_RECL_OUT_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_RTV_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_CMTS_QTY <MEASURE_NAME>
F_PLN_CURR_ORD_CNCLLD_QTY <MEASURE_NAME>
F_PLN_CURR_ORD_QTY <MEASURE_NAME>
F_PLN_CURR_RECL_IN_QTY <MEASURE_NAME>
F_PLN_CURR_RECL_OUT_QTY <MEASURE_NAME>
F_PLN_CURR_RTV_QTY <MEASURE_NAME>
F_PLN_CURR_EOP_RTL_AMT <MEASURE_NAME>
F_PLN_CURR_WOS_AMT <MEASURE_NAME>
F_PLN_CURR_EOP_COST_AMT <MEASURE_NAME>
F_PLN_CURR_ORD_CNCLLD_COST_AMT <MEASURE_NAME>
F_PLN_CURR_ORD_COST_AMT <MEASURE_NAME>
F_PLN_CURR_CMTS_COST_AMT <MEASURE_NAME>
F_PLN_CURR_CUM_MKUP_PCT <MEASURE_NAME>
F_PLN_CURR_EOP_QTY <MEASURE_NAME>
F_PLN_CURR_WOS_QTY <MEASURE_NAME>
F_PLN_CURR_COGS_AMT <MEASURE_NAME>
F_PLN_CURR_EXCL_SLS_VAT_AMT <MEASURE_NAME>
F_PLN_CURR_EMPLY_DISC_AMT <MEASURE_NAME>
F_PLN_CURR_FRGHT_COST_AMT <MEASURE_NAME>
F_PLN_CURR_WRKRM_COST_AMT <MEASURE_NAME>
F_PLN_CURR_RTRNS_SLS_AMT <MEASURE_NAME>

The following table provides information about the output file data format.

Field	Start	Width	Format
DAY_DT EOW Day	0	8	Date
DEPT_IDNT	6	4	String
CLASS_IDNT	10	4	String
SBCLASS_IDNT	14	4	String
LOC_IDNT LOCATION_LABEL	18	10	String
MEASURE	28		Float

Note: The following must be defined in the shell environment prior to calling this script:

– RPAS_HOME

– RPAS_INTEGRATION_HOME

Extract of Original Plan Data for RDW

plan_e_ploblwdm.ksh extracts Original Plan data from Merchandise Financial Planning for use by RDW.

The script expects an inputfile as an argument. The inputfile specifies the hierarchy labels and measure names that need to be extracted by the script. The user is expected to create this input file and this is used by the script to extract the appropriate measures.

Arguments:

- `-d` <Domain> (The domain path has to be a masterdomain or a simple domain. A subdomain path will fail)
- `-o` <outputfile> (Including file path)
- `-i` <inputfile with field to measure mapping> (Including file path)

  Input file should contain all the fields below followed by a space and the associated measure name.

  Example of the input file data format:

```
DEPT_IDNT DEPARTMENT_LABEL
CLASS_IDNT CLASS_LABEL
SBCLASS_IDNT SBCLASS_LABEL
LOC_IDNT LOCATION_LABEL
F_PLN_ORIG_CLRC_SLS_QTY MEASURE_NAME
F_PLN_ORIG_PRMTN_SLS_QTY MEASURE_NAME
F_PLN_ORIG_RGLR_SLS_QTY MEASURE_NAME
F_PLN_ORIG_CLRC_SLS_AMT MEASURE_NAME
F_PLN_ORIG_PRMTN_SLS_AMT MEASURE_NAME
F_PLN_ORIG_RGLR_SLS_AMT MEASURE_NAME
F_PLN_ORIG_GRS_PRFT_AMT MEASURE_NAME
F_PLN_ORIG_RGLR_MKDN_AMT MEASURE_NAME
F_PLN_ORIG_CLRC_MKDN_AMT MEASURE_NAME
F_PLN_ORIG_PRMTN_MKDN_AMT MEASURE_NAME
F_PLN_ORIG_SHRK_QTY MEASURE_NAME
F_PLN_ORIG_SHRK_RTL_AMT MEASURE_NAME
F_PLN_ORIG_BOP_QTY MEASURE_NAME
F_PLN_ORIG_BOP_COST_AMT MEASURE_NAME
F_PLN_ORIG_BOP_RTL_AMT MEASURE_NAME
F_PLN_ORIG_RCPTS_QTY MEASURE_NAME
```

```
F_PLN_ORIG_RCPTS_COST_AMT MEASURE_NAME
F_PLN_ORIG_RCPTS_RTL_AMT MEASURE_NAME
F_PLN_ORIG_CMTS_RTL_AMT MEASURE_NAME
F_PLN_ORIG_ORD_CNCLLD_RTL_AMT MEASURE_NAME
F_PLN_ORIG_ORD_RTL_AMT MEASURE_NAME
F_PLN_ORIG_RECL_IN_RTL_AMT MEASURE_NAME
F_PLN_ORIG_RECL_OUT_RTL_AMT MEASURE_NAME
F_PLN_ORIG_RTV_RTL_AMT MEASURE_NAME
F_PLN_ORIG_CMTS_QTY MEASURE_NAME
F_PLN_ORIG_ORD_CNCLLD_QTY MEASURE_NAME
F_PLN_ORIG_ORD_QTY MEASURE_NAME
F_PLN_ORIG_RECL_IN_QTY MEASURE_NAME
F_PLN_ORIG_RECL_OUT_QTY MEASURE_NAME
F_PLN_ORIG_RTV_QTY MEASURE_NAME
F_PLN_ORIG_EOP_RTL_AMT MEASURE_NAME
F_PLN_ORIG_EOP_QTY MEASURE_NAME
F_PLN_ORIG_ORD_COST_AMT MEASURE_NAME
F_PLN_ORIG_ORD_CNCLLD_COST_AMT MEASURE_NAME
F_PLN_ORIG_CMTS_COST_AMT MEASURE_NAME
F_PLN_ORIG_CUM_MKUP_PCT MEASURE_NAME
F_PLN_ORIG_COGS_AMT MEASURE_NAME
F_PLN_ORIG_EXCL_SLS_VAT_AMT MEASURE_NAME
F_PLN_ORIG_EMPLY_DISC_AMT MEASURE_NAME
F_PLN_ORIG_FRGHT_COST_AMT MEASURE_NAME
F_PLN_ORIG_WRKRM_COST_AMT MEASURE_NAME
F_PLN_ORIG_RTRNS_SLS_AMT MEASURE_NAME
F_PLN_ORIG_EOP_COST_AMT MEASURE_NAME
```

The following table provides information about the output file data format.

Field	Start	Width	Format
DAY_DT EOW Day	0	8	Date
DEPT_IDNT	6	4	String
CLASS_IDNT	10	4	String
SBCLASS_IDNT	14	4	String
LOC_IDNT LOCATION_LABEL	18	10	String
MEASURE	28		Float

Note: The following must be defined in the shell environment prior to calling this script:

– RPAS_HOME

– RPAS_INTEGRATION_HOME

RDF and Merchandise Financial Planning Extract Matrix

The following matrix identifies the extract scripts and schemas used for each the data files produced for either RMS or Price:

Directory	Script or Schema Name	Forecasts and Standard Deviations	Diff Profiles	Receipt Plan	Markdown Budget	Store Grades	Current Plan	Original Plan
common	config.ksh							
	functions.ksh	X						
	header.ksh	X	X	X		X		
curve	profile_e_alloc.ksh		X					
grade	grade_e_rms.ksh					X		
plan	Plan_e_alloc.ksh			X				
	Plan_e_price.ksh				X			
	Plan_e_plcblwdm.ksh						X	
	Plan_e_ploblwdm.ksh							X
rdf	rdf_e_rms.ksh	X						
	rmsl_forecast.ksh	X						
	rmsl_forecast_daily.schema	X						
	rmsl_forecast_weekly.schema	X						

Integration with Oracle Retail Workspace

The Oracle Retail Workspace installer prompts you to enter the URL for your supported Oracle Retail applications. However, if a client installs a new application after Oracle Retail Workspace is installed, the retail-workspace-page-config.xml file needs to be edited to reflect the new application.

The file as supplied comes with all appropriate products configured, but the configurations of non-installed products have been "turned off". Therefore, when "turning on" a product, locate the appropriate entry, set "rendered" to "true", and enter the correct URL and parameters for the new application.

The entry consists of the main URL string plus one parameter named "config". The value of the config parameter is inserted by the installer. Somewhere in the installer property files there is a value for the properties "deploy.retail.product.rms.url" and "deploy.retail.product.rms.config".

The entry consists of the main URL string plus one parameter named "config". The value of the config parameter will be inserted by the installer. Somewhere in the installer property files there will be a value for the properties "deploy.retail.product.rms.url" and "deploy.retail.product.rms.config".

For example, suppose RMS was installed on mycomputer.mycompany.com, port 7777, using a standard install and rms configured with the application name of "rms121sedevhpsso". If you were to access RMS directly from your browser, you would type in:

```
http://mycomputer.mycompany.com:7777/forms/frmservlet?config=rms121sedevhpsso
```

The entry in the retail-workspace-page-config.xml after installation would resemble the following:

```
<url>http://mycomputer.mycompany.com:7777/forms/frmservlet</url>
    <parameters>
        <parameter name="config">
            <value>rms121sedevhpsso</value>
        </parameter>
    </parameters>
```

Oracle Single Sign-on Overview

What is Single Sign-On?

Single Sign-On (SSO) is a term for the ability to sign onto multiple Web applications via a single user ID/Password. There are many implementations of SSO – Oracle currently provides three different implementations: Oracle Single Sign-On (OSSO), Java SSO (with the 10.1.3.1 release of OC4J) and Oracle Access Manager (provides more comprehensive user access capabilities).

Most, if not all, SSO technologies use a session cookie to hold encrypted data passed to each application. The SSO infrastructure has the responsibility to validate these cookies and, possibly, update this information. The user is directed to log on only if the cookie is not present or has become invalid. These session cookies are restricted to a single browser session and are never written to a file.

Another facet of SSO is how these technologies redirect a user's Web browser to various servlets. The SSO implementation determines when and where these redirects occur and what the final screen shown to the user is.

Most SSO implementations are performed in an application's infrastructure and not in the application logic itself. Applications that leverage infrastructure managed authentication (such as deploying specifying "Basic" or "Form" authentication) typically have little or no code changes when adapted to work in an SSO environment.

What Do I Need for Oracle Single Sign-On?

The nexus of an Oracle Single Sign-On system is the Oracle Identity Management Infrastructure installation. This consists of the following components:

- An Oracle Internet Directory (OID) LDAP server, used to store user, role, security, and other information. OID uses an Oracle database as the back-end storage of this information.
- An Oracle Single Sign-On servlet, used to authenticate the user and create the OSSO session cookie. This servlet is deployed within the infrastructure Oracle Application Server (OAS).
- The Delegated Administration Services (DAS) application, used to administer users and group information. This information may also be loaded or modified via standard LDAP Data Interchange Format (LDIF) scripts.
- Additional administrative scripts for configuring the OSSO system and registering HTTP servers.

Additional OAS servers will be needed to deploy the business applications leveraging the OSSO technology.

Can Oracle Single Sign-On Work with Other SSO Implementations?

Yes, OSSO has the ability to interoperate with many other SSO implementations, but some restrictions exist.

Oracle Single Sign-on Terms and Definitions

Authentication

Authentication is the process of establishing a user's identity. There are many types of authentication. The most common authentication process involves a user ID and password.

Dynamically Protected URLs

A "Dynamically Protected URL" is a URL whose implementing application is aware of the OSSO environment. The application may allow a user limited access when the user has not been authenticated. Applications that implement dynamic OSSO protection typically display a "Login" link to provide user authentication and gain greater access to the application's resources.

Identity Management Infrastructure

The Identity Management Infrastructure is the collection of product and services which provide Oracle Single Sign-on functionality. This includes the Oracle Internet Directory, an Oracle HTTP server, and the Oracle Single Sign-On services. The Oracle Application Server deployed with these components is typically referred as the "Infrastructure" instance.

MOD_OSSO

mod_osso is an Apache Web Server module an Oracle HTTP Server uses to function as a partner application within an Oracle Single Sign-On environment. The Oracle HTTP Server is based on the Apache HTTP Server.

Oracle Internet Directory

Oracle Internet Directory (OID) is an LDAP-compliant directory service. It contains user ids, passwords, group membership, privileges, and other attributes for users who are authenticated using Oracle Single Sign-On.

Partner Application

A partner application is an application that delegates authentication to the Oracle Identity Management Infrastructure. One such partner application is the Oracle HTTP Server (OHS) supplied with the Oracle Application Server. OHS uses the MOD_OSSO module to configure this functionality.

All partner applications must be registered with the Oracle Single Sign-On server. An output product of this registration is a configuration file the partner application uses to verify a user has been previously authenticated.

Realm

A Realm is a collection users and groups (roles) managed by a single password policy. This policy controls what may be used for authentication (for example, passwords, X.509 certificates, and biometric devices). A Realm also contains an authorization policy used for controlling access to applications or resources used by one or more applications.

A single OID can contain multiple Realms. This feature can consolidate security for retailers with multiple banners or to consolidate security for multiple development and test environments.

Statically Protected URLs

A URL is considered to be "Statically Protected" when an Oracle HTTP server is configured to limit access to this URL to only SSO authenticated users. Any attempt to access a "Statically Protected URL" results in the display of a login page or an error page to the user.

Servlets, static HTML pages, and JSP pages may be statically protected.

What Single Sign-On is not

Single Sign-On is NOT a user ID/password mapping technology.

However, some applications can store and retrieve user IDs and passwords for non-SSO applications within an OID LDAP server. An example of this is the Oracle Forms Web Application framework, which maps OSSO user IDs to a database logins on a per-application basis.

How Oracle Single Sign-On Works

Oracle Single Sign-On involves a couple of different components. These are:

- The Oracle Single Sign-On (OSSO) servlet, which is responsible for the back-end authentication of the user.
- The Oracle Internet Directory LDAP server, which stores user IDs, passwords, and group (role) membership.
- The Oracle HTTP Server associated with the Web application, which verifies and controls browser redirection to the OSSO servlet.
- If the Web application implements dynamic protection, then the Web application itself is involved with the OSSO system.

Statically Protected URLs

When an unauthenticated user accesses a statically protected URL, the following occurs:

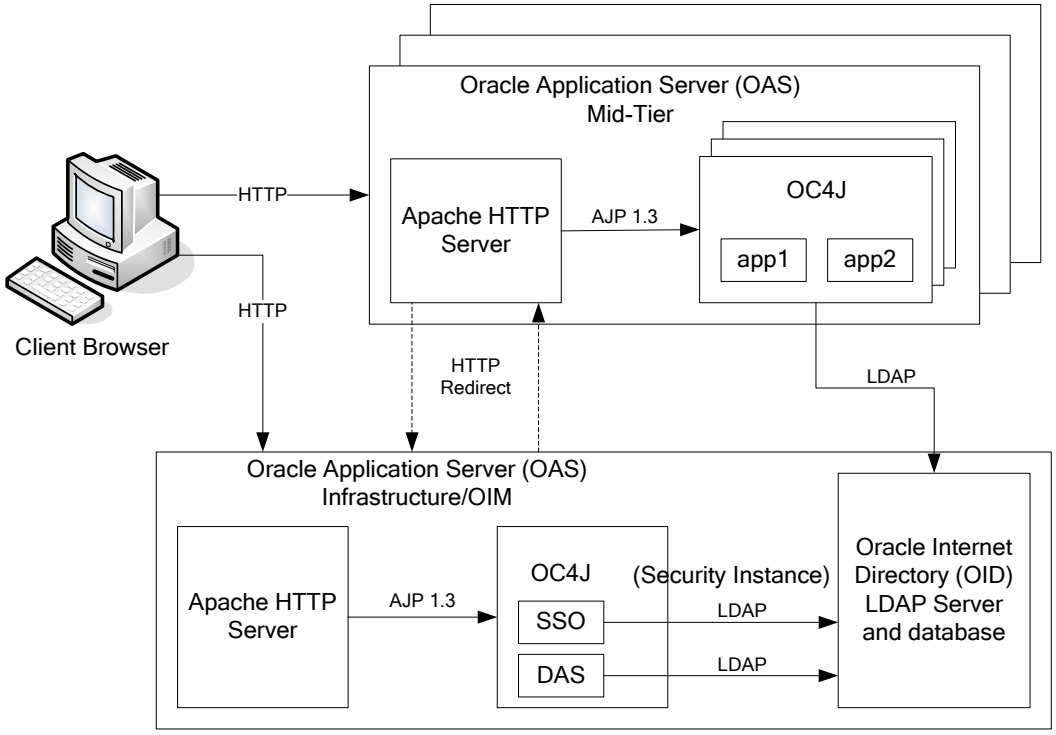
1. The Oracle HTTP server recognizes the user has not been authenticated and redirects the browser to the Oracle Single Sign-On servlet.
2. The OSSO servlet determines the user must authenticate, and displays the OSSO login page.
3. The user must sign in via a valid user ID and password. If the OSSO servlet has been configured to support multiple Realms, a valid realm must also be entered. The user ID, password, and realm information is validated against the Oracle Internet Directory LDAP server.
4. The OSSO servlet creates and sends the user's browser an OSSO session cookie. This cookie is never persisted to disk and is specific only to the current browser session. This cookie contains the user's authenticated identity. It does NOT contain the user's password.
5. The OSSO servlet redirects the user back to the Oracle HTTP Server, along with OSSO specific information.
6. The Oracle HTTP Server decodes the OSSO information, stores it with the user's session, and allows the user access to the original URL.

Dynamically Protected URLs

When an unauthenticated user accesses a dynamically protected URL, the following occurs:

1. The Oracle HTTP server recognizes the user has not been authenticated, but allows the user to access the URL.
2. The application determines the user must be authenticated and sends the Oracle HTTP server a specific status to begin the authentication process.
3. The Oracle HTTP Server redirects the user's browser session to the OSSO Servlet.
4. The OSSO servlet determines the user must authenticate, and displays the OSSO login page.
5. The user must sign in via a valid user ID and password. If the OSSO servlet has been configured to support multiple Realms, a valid realm must also be entered. The user ID, password, and realm information is validated against the Oracle Internet Directory LDAP server.
6. The OSSO servlet creates and sends the user's browser an OSSO session cookie. This cookie is never persisted to disk and is specific only to the current browser session. This cookie contains the user's authenticated identity. It does NOT contain the user's password.
7. The OSSO servlet redirects the user back to the Oracle HTTP Server, along with OSSO specific information.
8. The Oracle HTTP Server decodes the OSSO information, stores it with the user's session, and allows the user access to the original URL.

Single Sign-on Topology



Installation Overview

Installing Oracle Single Sign-On consists of installing the following components:

1. Installing the Oracle Internet Directory (OID) LDAP server and the Infrastructure Oracle Application Server (OAS). These are typically performed using a single session of the Oracle Universal Installer and are performed at the same time. OID requires an Oracle relational database and if one is not available, the installer will also install this as well.

   The Infrastructure OAS includes the Delegated Administration Services (DAS) application as well as the OSSO servlet. The DAS application can be used for user and realm management within OID.
2. Installing additional OAS 10.1.2 midtier instances for the Oracle Retail applications, such as RMS, that are based on Oracle Forms technologies. These instances must be registered with the Infrastructure OAS installed in step 1).
3. Installing additional application servers to deploy other Oracle Retail applications and performing application specific initialization and deployment activities.

Infrastructure Installation and Configuration

The Infrastructure installation for OSSO is dependent on the environment and requirements for its use. Deploying an Infrastructure OAS to be used in a test environment does not have the same availability requirements as for a production environment. Similarly, the Oracle Internet Directory (OID) LDAP server can be deployed in a variety of different configurations. See the *Oracle Application Server Installation Guide* and the *Oracle Internet Directory Installation Guide* for more details.

OID User Data

Oracle Internet Directory is an LDAP v3 compliant directory server. It provides standards-based user definitions out of the box.

The current version of Oracle Single Sign-On only supports OID as its user storage facility. Customers with existing corporate LDAP implementations may need to synchronize user information between their existing LDAP directory servers and OID. OID supports standard LDIF file formats and provides a JNDI compliant set of Java classes as well. Moreover, OID provides additional synchronization and replication facilities to integrate with other corporate LDAP implementations.

Each user ID stored in OID has a specific record containing user specific information. For role-based access, groups of users can be defined and managed within OID. Applications can thus grant access based on group (role) membership saving administration time and providing a more secure implementation.

OID with Multiple Realms

OID and OSSO can be configured to support multiple user Realms. Each realm is independent from each other and contains its own set of user IDs. As such, creating a new realm is an alternative to installing multiple OID and Infrastructure instances. Hence, a single Infrastructure OAS can be used to support many development and test environments by defining one realm for each environment.

Realms may also be used to support multiple groups of external users, such as those from partner companies. For more information on Realms, see the *Oracle Internet Directory Administrators Guide*.

User Management

User Management consists of displaying, creating, updating or removing user information. There are two basic methods of performing user management: LDIF scripts and the Delegate Administration Services (DAS) application.

OID DAS

The DAS application is a Web-based application designed for both administrators and users. A user may update their password, change their telephone number of record, or modify other user information. Users may search for other users based on partial strings of the user's name or ID. An administrator may create new users, unlock passwords, or delete users.

The DAS application is fully customizable. Administrators may define what user attributes are required, optional or even prompted for when a new user is created.

Furthermore, the DAS application is secure. Administrators may also what user attributes are displayed to other users. Administration is based on permission grants, so different users may have different capabilities for user management based on their roles within their organization.

LDIF Scripts

Script based user management can be used to synchronize data between multiple LDAP servers. The standard format for these scripts is the LDAP Data Interchange Format (LDIF). OID supports LDIF script for importing and exporting user information. LDIF scripts may also be used for bulk user load operations.

User Data Synchronization

The user store for Oracle Single Sign-On resides within the Oracle Internet Directory (OID) LDAP server. Oracle Retail applications may require additional information attached to a user name for application-specific purposes and may be stored in an application-specific database. Currently, there are no Oracle Retail tools for synchronizing changes in OID stored information with application-specific user stores. Implementers should plan appropriate time and resources for this process. Oracle Retail strongly suggests that you configure any Oracle Retail application using an LDAP for its user store to point to the same OID server used with Oracle Single Sign-On.

B

Appendix: Curve Administration Guide

curvevalidate

`Curvevalidate` automatically executes during the domain install, and it can also be run at any time against a Master or one subdomain. If run against the Master Domain, it checks the master and all subdomains. If run against a subdomain, it checks the Master and only the subdomain (not all other subdomains). This function verifies that:

- Profile and Source intersections and source data are properly defined
- Profile intersections respect the partition dimension

Usage

```
curvevalidate -d domainpath [-s]
```

The following table provides descriptions of the arguments used by the `curvevalidate` utility.

Argument	Description
`–d` *`domainpath`*	Path to the domain.
`-s`	Set defaults.
`-debug`	This argument causes temporary measures to be retained for debugging purposes.
`-h`	Standard information argument to display information about `curvebatch` in the terminal screen.
`-version`	Standard argument that displays version information.
`-loglevel`	Used to set the logger verbosity level. Available verbosity levels are as follows: ▪ `all` ▪ `profile` ▪ `information` ▪ `warning` ▪ `error` ▪ `none`
`-noheader`	Standard argument to disable the use of timestamp header.

1. Each Profile must have at least one Source Level.
2. For each Profile:
   a. For global domains, ALL intersections {Data Intersection, Profile Intersection, Stored Intersection, Aggregation Intersection, and Approval Intersection} must be below the partition (NOT HBI).
   b. Data Intersection (if a data source is specified) must conform to X in {Profile Intersection, Stored Intersection, Aggregation Intersection, and Approval Intersection}.

c. Profile Intersection must conform to the Stored Intersection.
d. Aggregation Intersection must conform to the Approval Intersection.
e. Aggregation Intersection must not be below the Approval Intersection.
f. Aggregation Intersection must be above the Data Intersection (if data source specified).
g. If the Aggregation Intersection conforms to Profile Intersection:
    i. The Profile Type must NOT be diff(8).
    ii. The Aggregation Intersection must be above the Profile Intersection.
    iii. The Aggregation Intersection must be above the Stored Intersection.
h. If Aggregation Intersection does not conform to Profile Intersection:
    i. The Profile Type must be Diff (8).
    ii. There must be at least one common hierarchy between the Aggregation Intersection and X in {Profile Intersection, Stored Intersection}.
    iii. For each common non-PROD hierarchy H of Aggregation Intersection and X in {Profile Intersection, Stored Intersection}:
    Aggregation Intersection's H dimension must not be below X's H dimension.

3. For each Source Level:
    a. For global domains, ALL intersections {Profile Intersection, Stored Intersection, and Aggregation Intersection} must be below the partition (NOT HBI).
    b. Parent Profile's Data Intersection (if data source specified) must conform to X in {Profile Intersection, Stored Intersection, and Aggregation Intersection}.
    c. Profile Intersection must conform to Stored Intersection.
    d. Aggregation Intersection must be above parent Profile's Data Intersection (if data source specified).
    e. If Aggregation Intersection conforms to Profile Intersection:
        i. The Profile Type must NOT be diff(8).
        ii. The Aggregation Intersection must be above the Profile Intersection.
        iii. The Aggregation Intersection must be above the Stored Intersection.
    f. If Aggregation Intersection does not conform to Profile Intersection:
        i. The Parent Profile Type must be Diff (8).
        ii. There must be at least one common hierarchy between the Aggregation Intersection and X in {Profile Intersection and Stored Intersection}.
        iii. For each common non-PROD hierarchy H of Aggregation Intersection and X in {Profile Intersection and Stored Intersection}:
        Aggregation Intersection's H dimension must not be below X's H dimension.

curvebatch

Usage

```
curvebatch -d domainpath [-level # ] [-debug] | -h | -version
```

The following table provides descriptions of the arguments used by the curvebatch utility.

Argument	Description
–d *domainpath*	Path to the domain.
-level #	The # signifies the profile ID. When using level, a valid profile ID must be provided.
-debug	This argument causes temporary measures to be retained for debugging purposes.
-h	Standard information argument to display information about curvebatch in the terminal screen.
-version	Standard argument that displays version information.
-loglevel	Used to set the logger verbosity level. Available verbosity levels are as follows: ▪ all ▪ profile ▪ information ▪ warning ▪ error ▪ none