Oracle® Retail Predictive Application Server

Administration Guide for the Fusion Client

Release 14.0

E50799-03

December 2013

Oracle® Retail Predictive Application Server Administration Guide for the Fusion Client, Release 14.0

E50799-03

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Primary Author: Bernadette Goodman

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

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

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

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

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

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

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

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

Contents

Send Us Your Comments xix

Preface xxi

1 Introduction

Administrator Overview 1-1
Basic RPAS Concepts 1-1
  Multidimensionality 1-2
  Dimensions 1-2
  Measures 1-2
  Domains and Workbooks 1-2
    Measure Data 1-3
Overview of the RPAS Fusion Client 1-4
  Terminology Differences Between Clients 1-4
Administrative Workbooks and Wizards 1-5
A Note about RPAS Configurability and Extensibility 1-5

2 Building and Upgrading Domains

Building a Domain 2-1
  Prerequisites 2-1
  Client-Side Procedures 2-1
    Zip the Configuration Project Folder 2-2
  Server-Side Procedures 2-2
    Unzip the Configuration File 2-2
    Verify the Environment Variable Settings 2-3
    Preparing the Input Files 2-3
Building the Domain Manually or Through the Command-Line Interface 2-3
  rpasInstall 2-3
    rpasInstall Usage 2-3
  Validate Domain Build Results 2-5
After Building the Domain 2-5
  Upgrading and Patching Domains 2-6
  Upgrading From Any Release of RPAS Prior to 13.3 2-6
  Domain Conversion 2-6
    Domain Preparation Before Conversion 2-7

Setting up Domain Conversion Parameters 2-9
Running the Domain Conversion 2-10
Post-Conversion Processing 2-10
Upgrading From RPAS 13.3 or Later 2-11
Updating Domain Content 2-12
Hierarchies 2-13
Measure Properties 2-13
Rule Groups 2-13
Workbooks 2-13
RPAS 14.0 Cryptography Upgrades 2-14
Upgrade Configurations 2-14
Command-line Support for Configuration Upgrading 2-15
Upgrade Application Configurations 2-16
convertDomain 2-16
convertDomain Usage 2-17
convertDomain -suggestDimBitSize 2-19
Usage for -suggestDimBitSize 2-20
upgradeDomain 2-20
upgradeDomain Usage 2-20
fixDomain 2-21
fixDomain Usage 2-22

3 Domain Administration

DomainDaemon 3-1
DomainDaemon Usage 3-1
Starting the DomainDaemon 3-3
Monitoring the DomainDaemon 3-4
Stopping the DomainDaemon 3-4
Losing a Client-Server Connection 3-5
Graceful Termination of the Existing Session 3-5
Autosaved Workbooks 3-5

4 SSL

Introduction 4-1
Setting Up SSL 4-1
Two-Way SSL 4-2
Configuring the RPAS Server 4-2
Setting Up the Key Store 4-3
Setting Up a Fusion Client Trust Store 4-4
Importing the Client Certificates 4-6
Starting the Domain Daemon 4-6
Configuring the Fusion Client 4-6
Setting Us a Key Store and a Trust Store 4-6
Importing the Client Certificates 4-8
Two-Way SSL with Authentication Only 4-8
Creating SSL Certificates Automatically 4-8

5 Setting Up the RPAS Fusion Client

Maintaining Layout and Formatting 5-1
Setting Up the Configuration Properties File 5-1
  Latency 5-2
  Other Properties Files 5-2
    Files in the Config Folder 5-3
    Files in the Config/rpas Folder 5-3
  Updating the Session Reminder 5-3
  Configuring Announcements 5-4
Configuring Additional Domains 5-5
  Sample Foundation.xml 5-7
Logging into the Fusion Client 5-7

6 User Maintenance

User Administration Activity 6-1
  Adding a User 6-2
  Adding a User Group 6-3
  Deleting a User 6-3
  Deleting a User Group 6-4
  Editing a User 6-4
Managing Users Using usermgr 6-4
  usermgr Usage 6-5
  XML Schema 6-7
  Use Cases 6-9
    Exporting from an Existing Domain 6-9
    Importing into a Domain 6-9
    Converting Between XML and Database 6-9

7 Single Sign-On

SSO Authentication 7-1
  Identity Assertion 7-3
  Fetching Roles and Groups 7-3
  Security Provider Order 7-4
Manually Configuring SSO Within the Fusion Client 7-4

8 System Administration

Auto Workbook Maintenance Wizard 8-2
  Adding a Workbook to the Auto Build Queue 8-2
  Deleting a Workbook from the Auto Build Queue 8-3
  Editing a Workbook in the Auto Build Queue 8-3
Workbook Batch Category Management 8-4
  Workbook Batch Category Management Wizard 8-4
  Adding a Category 8-4
  Deleting a Category 8-5
  Changing a Category Label 8-6

Hierarchy Maintenance Workbook 8-7
  Hierarchy Maintenance Example 8-8
  Hierarchy Maintenance Wizard 8-8
  Hierarchy Maintenance View 8-8
  Maintaining a User-Defined Dimension Within a Hierarchy 8-10
Security Administration Workbook 8-11
  Security Overview 8-11
    User and Group Security 8-11
    Measure Level Security 8-12
    Position Level Security 8-12
    Workbook Security 8-14
  Security Administration Workbook 8-14
    Workbook Template Rights View 8-15
    Workbook Template Measure Rights View 8-16
    Measure Rights View 8-16
    Dimension Modification Rights View 8-17
    Position Level Security Views 8-17
    Workbook Template Limits Views 8-17
    Max Domain Session Limit View 8-17
    Max User Session Limit View 8-17
  Using the Security Administration Workbook 8-18
    Accessing Security Administration 8-18
    Setting or Modifying Users' Access to Workbook Templates 8-18
    Setting Measure Availability for Workbook Templates 8-18
    Assigning or Restricting User Access to Measures 8-19
    Changing a User's Ability to Modify Dimensions 8-19
    Setting or Modifying Access to Positions (if Position Level Security has been Enabled) 8-19
    Limiting the Number of Workbooks that a User Can Save 8-20
    Limiting the Number of Workbooks Per Template 8-20
    Limiting the Number of Workbooks that a Group Can Save 8-21
Measure Analysis Workbook 8-21
  Measure Analysis View 8-21
  Accessing Measure Analysis 8-22
  Reviewing and Editing Sales or Other Registered Measure Data 8-22

9 Hierarchy Management

Loading Hierarchies Using loadHier 9-2
  loadHier Usage 9-3
  loadHier Notes 9-4
  Position Label Translation 9-5
Integer Indexing 9-6
  Changing the Bit Size 9-9
Reindexing Domains Using reindexDomain 9-9
  reindexDomain Usage 9-10
    ReindexDomain Option: Analysis 9-11
    ReindexDomain Option: Domain Properties 9-11

- When to Reindex 9-12
  - Run Reports 9-14
- How to Reindex a Dimension 9-15
  - Reindex Entire Domain as Needed 9-15
  - Reindex Entire Domain with Force Option 9-15
  - Reindex Dimensions 9-15
  - Prepending Calendar Dimensions 9-16
- Reindex Domain Examples 9-17
  - Condition 1: Change of Bit Size 9-17
  - Condition 2: Using the -force Option 9-17
  - Condition 3: Fragmentation Checks 9-18

Optimizing Domains Using optimizeDomain 9-19
- optimizeDomain Usage 9-20
  - Examples 9-21

Adding New Dimensions to Hierarchies 9-22
- Exporting Measure Data 9-23
- Exporting Hierarchy Data 9-23
- Purging Hierarchy Data 9-23
- Adding New Dimensions 9-24
- Reloading Formal Hierarchy Data 9-24
- Informalizing DPM Positions 9-24
- Reloading Measure Data 9-25
- Adding New Hierarchy Dimensions Sample Script 9-25
- Transferring Data 9-27
  - Masks 9-28
  - transferData Usage 9-29
  - Scenarios 9-29

FilterHier Utility 9-30
- filterHier Usage 9-30
- Notes 9-31
- Error Conditions 9-31

Position Repartitioning 9-31
- Loading RDF and Curve Parameters after Repartitioning 9-32
  - Syntax 9-32

Reconfiguring the Partitions of a Global Domain Using reconfigGlobalDomainPartitions 9-32
- reconfigGlobalDomainPartitions Usage 9-33
- Using an Input File 9-34
- Notes, Assumptions, and Limitations 9-35

Renaming Positions Using renamePositions 9-35
- renamePositions Usage 9-36

Setting Properties for Dimensions Using dimensionMgr 9-37
- Changing the Label of a Dimension 9-37
- dimensionMgr Usage 9-37

Exporting Hierarchy Data Using exportHier 9-38
- exportHier Usage 9-38

Informal Position Manager 9-39
- InformalPositionMgr Usage 9-39

    Basic Operations: Informalize, Formalize, and Remove 9-39
        Error Handling 9-40
    Creating Informal Positions in Bulk 9-41
        Example 1: Creating Informal Positions on the STYL Dimension 9-42
        Example 2: Creating Informal Positions on the SZ12 Dimension 9-43
        Position Naming Convention 9-44
    Data Slice Copying 9-44
        Merging Informal Positions to Formal Positions 9-46
Managing Position Lists as PQDs Using pqdMgr 9-46
    pqdMgr Usage 9-47
    pqdMgr Notes 9-47

10 Data Management

Loading Measure Data Using loadmeasure 10-1
    Load File Names and Load Behavior 10-2
    Loading Multiple Measures from One File 10-3
        CSV Files 10-4
        Fixed Width Files 10-4
    Loading Data from Below the Base Intersection of the Measures 10-4
    Staging Measure Loads 10-4
    Running Pre-Load or Post-Load Scripts 10-5
    Purging Old Measure Data 10-5
    Behavior in a Global Domain Environment 10-5
    loadmeasure Usage 10-6
    Loading Image Paths for Positions 10-8
Exporting Measure Data Using exportMeasure 10-8
    exportMeasure Usage 10-9
Exporting Measure Data Using exportData 10-11
    exportData Usage 10-12
        The -useLoadFormat Parameter 10-15
Mapping Data Between Domains Using mapData 10-16
    mapData Usage 10-16
Moving Data between Arrays Using updateArray 10-16
    updateArray Usage 10-17
Scan Domain Data Using scanDomain 10-18
    scanDomain Usage 10-18
Repair Domain Metadata Using fixDomain 10-19

11 Operational Utilities

Finding Alerts Using alertmgr 11-1
    alertmgr Usage 11-2
Copying Domains Using copyDomain 11-3
    copyDomain Usage 11-3
        copyDomain: Format of the XML Configuration File 11-6
Moving a Domain Using moveDomain 11-7
    moveDomain Usage 11-7
        moveDomain: Format of the XML Configuration File 11-7

Assumptions and Requirements 11-8
Minimum Space Requirement 11-8
Setting Miscellaneous Domain Properties Using domainprop 11-8
domainprop Usage 11-8
Available Properties 11-9
Calculation Engine Using mace 11-10
mace Usage 11-11
mace Restrictions in Global Domains 11-13
Managing the Workbook Batch Queue Using wbbatch 11-14
wbbatch Usage 11-14
Workbook Manager with wbmgr 11-16
wbmgr Usage 11-16
Registering Measures Using regmeasure 11-17
regmeasure Usage 11-17
Registering Token Measures Using regTokenMeasure 11-23
regTokenMeasure Usage 11-24
Batch Plug-In Tasks: execPluginTask.sh 11-24
execPluginTask.sh usage 11-24

12 Informational Utilities

Retrieving Domain Information Using domaininfo 12-1
domaininfo Usage 12-1
Checking the Validity of a Domain Using checkDomain 12-3
checkDomain Usage 12-3
Determining RPAS Server Version Using rpasversion 12-3
rpasversion Usage 12-4
List Contents of a Database Using listDb 12-4
listDb Usage 12-4
Printing Data from Arrays Using printArray 12-4
printArray Usage 12-4
Printing Data from Measures Using printMeasure 12-5
printMeasure Usage 12-6

13 Internationalization

Translation 13-1
Translation Administration 13-2
Translation Administration Workbook 13-3
Hierarchy Labels View 13-3
Dimension Labels View 13-3
Workbook Template Group Labels View 13-3
Workbook Template Labels View 13-3
Measure Labels View 13-4
Measure Descriptions View 13-4
User Group Labels View 13-4
Message Labels View 13-4
RGRP Labels View 13-4

Taskflow Components Configuration 13-4

14 Commit as Soon as Possible

Managing the Workbook Queue Using showWorkbookQueues 14-1
showWorkbookQueues Usage 14-1
Commit ASAP Settings Using configCommitAsap 14-2
configCommitAsap Usage 14-2
Logging and Technical Information 14-3

15 Batch Processes and RPAS Utilities

CSV File Format 15-1
RPAS Utilities Logging Options 15-2
Log Levels 15-2
Utilities with Standard Logging 15-3
Scripts 15-3
Utilities with Multi-Process Logging 15-3
Utilities with Special Logging 15-4
Utilities and Database Locks 15-6
Using Shell Scripts to Run Batch Processes 15-6
A Sample Shell Script 15-7
Common Information and Parameters for RPAS Utilities 15-7
Configuration Tools Log Files 15-8
RPAS Intraday Enabler 15-8
ride Usage 15-10
Scenarios 15-10
Scenario 1 15-11
Scenario 2 15-11
Scenario 3 15-12
Scenario 4 15-12
Scenario 5 15-13
Scenario 6 15-13
Domain Lock Status Using domainStatus 15-14
domainStatus Usage 15-14

16 RPAS ODBC/JDBC Driver

ODBC Configuration 16-2
Defining the ODBC Server Configuration Settings 16-2
Add RPAS ODBC Manager in the Management Console 16-2
Configure Using the ODBC Manager 16-7
Import the configuration changes 16-13
Defining the ODBC Client Configuration for Windows 16-13
Starting the RPAS ODBC Server Process 16-16
Testing the Connection Using Interactive SQL 16-17
ODBC Client Configuration for UNIX 16-17
Client Configuration 16-17
Testing the Connection 16-19

Installing and Using the RPAS JDBC Driver 16-19
- Updating Environment Variables for the JDBC Driver on Windows 16-20
- Updating Environment Variables for JDBC Driver on UNIX/Linux 16-21
- Using the RPAS JDBC Driver 16-21
- Enabling Spy for RPAS JDBC Driver 16-21
- Using the jdbcisql Utility Provided with RPAS JDBC Driver 16-22
- Using Oracle SQL Developer 16-23
- Using Oracle JDeveloper 16-23
- Using a Java Program 16-24
  - Running the Program 16-25

Data Query 16-25
- Limitations 16-25
  - Contention 16-25
  - Workbook Queries 16-26
- Metadata 16-26
- Fact and Dimension Tables 16-26

Measure Security in the ODBC Driver 16-28

Use Cases 16-28
- Using Metadata Tables to Explore the Structure of a Domain or a Workbook 16-28
- Querying Fact Data 16-29
- Connecting to a Workbook 16-30
- Requesting Additional Aggregate Tables 16-30

Clients 16-31
- Oracle Business Intelligence Enterprise Edition (OBIEE) 16-31
  - Configuring the ODBC Client for OBIEE 16-31
  - Connecting OBIEE to an RPAS Domain 16-32
- Microsoft Access 16-33
- JDeveloper 16-33
- XML Publisher 16-34
- Interactive SQL (ISQL) Utility 16-35

Supported and Unsupported SQL Functions 16-35
- Supported SQL Functions 16-35
  - Numeric Functions 16-36
  - String Functions 16-38
  - Time and Date Functions 16-42
  - System Functions 16-46
  - Aggregate Functions 16-46
  - Other Functions 16-47
- Unsupported SQL Functions 16-52
  - Handling of NULLS 16-52
  - Schema Information 16-53
  - Data Definition Language (DDL) 16-53
  - Insert 16-53
  - SELECT Syntax 16-53
  - Value Expressions 16-53
  - Value Functions 16-53
  - Date and Time Functions 16-53

Advanced Value Expressions 16-54
Row Value Constructor 16-54
Predicates 16-54
Join Operators 16-55

17 Publishing Measure Change Events

Configuring Subjects and Measures for Monitoring 17-2
Configuring the RPAS JMS Publisher 17-3
Command Line 17-4
Configuration Settings 17-5
General Settings 17-5
Vendor-Specific Settings 17-6
File-Based JNDI Object Store 17-6
LDAP-Based JNDI Object Store 17-6
Configuring the RPAS JMS Subscriber 17-7
Command Line 17-7
Configuration Settings 17-7

18 Guided Launch

Launching the RPAS Fusion Client 18-1
Issuing a Launch Request from a Web Page 18-2
User Authentication 18-3
Deploying Guided Launch 18-3
Use Cases 18-4
RPAS Guided Launch Format 18-5
Workbook Settings 18-5
XML Schema 18-5
Open Workbook 18-9
Specifications 18-9
Examples 18-10
Building a Workbook 18-10
Specifications 18-11
Example 18-14
buildWorkbookContext Utility 18-15
Command Line Usage 18-15

A Appendix: Curve Administration Guide

curvevalidate A-1
curvevalidate Usage A-1
curvebatch A-3
curvebatch Usage A-3

B Appendix: RPAS Test Automation

Introduction B-1
Usage B-1
Writing Test Cases B-2

Example B-3
Output B-6
Schema B-7
  1. Top Level Parent Tag B-7
    1.1 testscript B-7
    1.2 testsuite B-7
    1.3 testcase B-7
    1.4 setup B-7
    1.5 teardown B-7
  2. Generic Tags B-8
    2.1 Attribute B-8
    2.2 Description B-8
    2.3 Fail B-8
    2.4 key B-8
    2.5 rpas-today B-8
    2.6 shell B-8
    2.7 register-measure B-8
    2.8 unregister-measure B-9
  3. Workbook Operation Tags B-9
    3.1 Workbook Wizard Operations B-9
    3.2 Workbook Operations B-11
    3.3 Workbook Assertions B-13
    3.4 Workbook Data Edit Operations B-22
  4. Domain Operation Tags B-24
    4.1 Assertions B-24
    4.2 Domain Data Edit Operations B-32

C Appendix: Fusion Client Properties Files

Editing Property Files C-1
rpasConfig.properties File C-1
PivotTableStyles.properties File C-5
GraphConfig.xml File C-5
  Global Settings C-5
  Individual Graph Settings C-6
cdtConfig.properties File C-6

D Appendix: Environment Variables

Required Settings D-3
Database Settings D-3
  RPAS_ARRAY_TAG D-3
  RPAS_CACHE D-3
  RPAS_PAGE_SIZE D-3
  RPAS_PAGE_SPLIT_PERCENTAGE D-4
  RPAS_LOCK_TIMEOUT D-4
  RPAS_REQUEST_TIMEOUT D-5
  RPAS_USER_MODE D-5

Java Special Expression Settings D-5
RPAS_JAVA_CLASSPATH D-5
RPAS_JAVA_DEBUG_PORT D-5
RPAS_JAVA_MAXHEAPSIZE D-6
Log Settings D-6
RPAS_CACHE_LM_MULTIPILER / RPAS_DISABLE_LM_DIRECT D-6
RPAS_CLIENT_TIMEOUT / RPAS_SERVER_TIMEOUT D-6
RPAS_COMMIT_FAILED_MAX / RPAS_COMMIT_SUCCESS_MAX D-6
RPAS_JAVA_MAXHEAPSIZE D-6
RPAS_LOCK_DETAIL_PATH D-6
RPAS_LOCK_LOGGING D-7
RPAS_LOCK_LOG_PATH D-7
RPAS_LOCK_TRACK_MODE D-7
RPAS_LOCK_TRACK_PATH D-7
RPAS_LOCK_TRACK_DATABASES D-7
RPAS_LOCK_ACCESS_LIMIT D-7
RPAS_LOCK_MSG_PATH D-8
RPAS_LOG_BACKUPS D-8
RPAS_LOG_FD D-8
RPAS_LOG_FLUSH D-8
RPAS_LOG_LEVEL D-8
RPAS_LOG_PATH D-8
RPAS_REQUEST_TIMEOUT D-9
RPAS_SHELL D-9
RPAS_SPLIT_BUF_MB D-9
RPAS_STATS_PATH D-9
RPAS_TCPNODELAY D-9
RPAS_TRACE_SSL D-10
TMPDIR D-10
TODAY_RETEK D-10
Parallelization Settings D-10
RPAS_EXPORT_MEASURES_MAX D-10
RPAS_MIN_MEAS_PER_PROC D-10
RPAS_PROCESSES D-10
Profiling Logging D-11
RPAS_PROFILING_ENABLE D-11
RPAS_PROFILING_PATH D-11
Date and Time Setting D-11
RPAS_TODAY D-11
RPAS_TODAY_STATIC D-11
Numeric Precision D-12
RPAS_INCAGGPRC D-12
epsilon D-12
RPAS_SSL_WALLET D-13
LDR_CNTRL=NAMEDSHLIB=RPASZONE D-13
Nagle's Algorithm D-13

E Appendix: RPAS Diagnostic Tool

Diagnostic Tool Description E-1
  Using Recorder E-1
  Using Replayer E-2
    Configuring Realplayer for Use E-2
    Replayer Use E-2
  Using the Transaction Log as Data Store E-3
  Working with Oracle Support E-3
Design E-3
  Data Stored in Log File E-3
  Replay E-4
Replaying - Real Time Scenarios E-4
  Example 1: Percent of parent measure creation error E-4
  Example 2: Dimension Exception when Overwriting Level Split on Dimension E-7
  Example 3: Performance Issues E-8

Send Us Your Comments

Oracle Retail Predictive Application Server Administration Guide for the Fusion Client, Release 14.0

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

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

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

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

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

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

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

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

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

Preface

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.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Retail Predictive Application Server Release 14.0 documentation set:

- *Oracle Retail Predictive Application Server Administration Guide for the Classic Client*
- *Oracle Retail Predictive Application Server Configuration Tools User Guide*
- *Oracle Retail Predictive Application Server Installation Guide*
- *Oracle Retail Predictive Application Server Release Notes*
- *Oracle Retail Predictive Application Server and Applications Security Guide*

- *Oracle Retail Predictive Application Server User Guide for the Classic Client*
- *Oracle Retail Predictive Application Server User Guide for the Fusion Client*

Customer Support

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

`https://support.oracle.com`

When contacting Customer Support, please provide the following:

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

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 14.0) or a later patch release (for example, 14.0.1). If you are installing the base release or additional patches, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

`http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html`

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

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

`http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html`

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

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

Conventions

The following text conventions are used in this document:

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

1

Introduction

This guide is for administrators of the RPAS Server and the RPAS Fusion Client.

Administrator Overview

After the RPAS Server and Fusion Client have been installed, administrators must set up the RPAS Fusion Client and complete several administration activities before they can begin using RPAS and RPAS applications. The activities include the following:

- Domain Administration
- SSL
- User Maintenance
- Single Sign-On
- System Administration
- Hierarchy Management
- Data Management
- Translation Administration
- Guided Launch

Before you start any of these activities, you should understand the basics of RPAS: domains, workbooks, views, hierarchies, and measures.

Basic RPAS Concepts

Retail Predictive Application Server (RPAS) is a configurable platform with a proven scalability for developing multidimensional forecasting-based and planning-based solutions. This platform provides capabilities such as a multidimensional database structure, batch and online processing, a configurable slice-and-dice user interface, a sophisticated configurable calculation engine, user security and utility functions such as importing and exporting, all on a highly scalable technical environment that can be deployed on a variety of hardware platforms.

This section introduces you to the following RPAS concepts:

- Multidimensionality
- Dimensions
- Measures
- Domains and Workbooks

Multidimensionality

In RPAS, information is stored and represented based on the multidimensional framework. In a multidimensional database system, data is presented as a multidimensional array, where each individual data value is contained within a cell accessible by multiple indexes.

Multidimensional database systems are a complementary technology to entity relational systems and achieve performance levels above the relational database systems. Applications that run on RPAS identify data through dimensional relationships. Dimensions are qualities of an item (such as a product, location, or time) or components of a dimension that define the structure and roll up within the dimension.

Dimensions

Dimensions describe the top-to-bottom relationship between the levels or positions of the dimensions in RPAS. They reflect the dimensions set up at your business and being used by the merchandising solutions.

RPAS supports many alternative dimensions that provide different roll ups and help analyze the data from a different perspective.

Measures

Measures represent the events or measurements that are recorded, while the positions in the dimensions provide a context for the measurement. Measures are defined based on the business rules set in the application. The dimensionality of a measure is configured through the definition of its base intersection, which is the collection of levels (one per appropriate dimension) defining the lowest level at which the information is stored for the measure.

Measure names are completely configurable and typically named using a convention that identifies each component and the meaning of the measure.

Domains and Workbooks

RPAS stores information in a persistent multidimensional data cache that is optimized for large volumes and dimensional or time series data access requirements, typically required by multidimensional solutions. This central repository is called a domain. The domain also includes central definitions of metadata for the solution and provides a single update point.

When you use an RPAS solution, you interact with the solution through a personal data repository called a workbook. A workbook contains the subset of the data (and metadata) from the domain and its scope is constrained by the access rights available to a user. Workbooks are stored on the RPAS server, and can be built using an online wizard process or scheduled to be built in a batch process automatically. Workbooks are made up of one or more views. These views display the dimension and measure data of the domain.

Although the data and metadata in the workbook are copied from the domain, the data remains independent of the domain.

Domains can be built in one of two methods:

- Simple domain: This is the traditional, standalone domain that has no visibility to other domains.

- Global domain: This is a domain environment that contains two or more local domains (or subdomains) and a master domain that has visibility to all local domains that are part of that environment.

A 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.

Using a global domain environment provides 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, or partitioned, along organizational structures that reflect user roles and responsibilities. Most users only work within the local domains that contain their area of responsibilities, and they may not need to be know about 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:

- Across the local domains
- 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 is 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 reside in the master domain and are accessed from there when these measures are required in workbooks.

All measures are 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 locations of the measure data are invisible to the user after the measure has been registered.

Overview of the RPAS Fusion Client

The RPAS Fusion Client is the Web-based Rich Client for the Retail Predictive Application Server (RPAS) platform developed using the latest Oracle Application Development Framework (ADF). It includes all the features available in the RPAS Windows-based Client and delivers an enhanced user experience that meets the performance and scalability requirements set for the RPAS platform.

Planning is one of the most important and complex processes in a retail business. It typically involves a complex set of activities that need to be followed as part of a workflow. Unlike the RPAS Windows-based client, the RPAS Fusion Client includes an Activity Task Flow feature that provides a robust workflow in a single taskflow representing a number of RPAS solutions. This feature can make each planning activity easier to track and maintain.

The RPAS Fusion Client uses the same RPAS server as the current RPAS Windows-based client. You can continue to configure an RPAS solution and connect to it using the Web-based interface or the RPAS Windows-based client. Apart from the enhanced user experience, the Fusion Client provides access to a larger number of users and a certain degree of platform independence. In addition, it can leverage enterprise infrastructure such as LDAP and SSO.

Terminology Differences Between Clients

There are some key terminology differences between the Fusion Client and Classic Client. Understanding these differences is useful if you use both clients. The following table describes the differences.

Table 1–1 Terminology in Fusion Client and Classic Client

Fusion Client Term	Term Description	Classic Client Term
Dimension	Grouping of a particular type of information. Typical dimensions are for products, locations, time, and measures. For instance, a Product dimension could contain information about items, item groups, departments, and divisions.	Hierarchy
Level	A subdivision of a dimension. Levels group information of the same type. For instance, a level within the Product dimension could be Department. For example, the Department level might contain all the departments (men's shoes, women's shoes, children's shoes) that exist.	Dimension

Table 1–1 (Cont.) Terminology in Fusion Client and Classic Client

Fusion Client Term	Term Description	Classic Client Term
View	Multidimensional spreadsheets that are used to display information from the workbook. Workbooks can include one view or multiple views, which can present data in the form of numbers in a grid. These numeric data values can easily be converted to a graphical chart. Data can be viewed at a very high level of detail, or data values can be quickly aggregated and viewed at summary levels. You can display the information in a view in a variety of formats, generally by rotating, changing the data rollup, showing and hiding measures, and drilling up or down.	Worksheet

Administrative Workbooks and Wizards

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 tasks:

- Set up and maintain users and user groups. Note that authentication is handled externally through SSO or WebLogic/LDAP. The users in the external authentication system must exist in RPAS so that user access can be managed.
- 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.

> **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. See each workbook or workbook wizard section in this guide for details about the domain access.
>
> The taskflow automatically brings you to the master domain when necessary. You do not have to log into a specific master or local domain; you log into the global domain environment.

A Note about RPAS Configurability and Extensibility

RPAS is a configurable and extensible platform providing a significant amount of flexibility in design and implementation. Lack of specific verbiage around a platform component does not guarantee a commitment to support its use in a non-documented way. In case of any ambiguity in the documentation, the customer is expected to ask for clarification with Oracle Customer Support.

2

Building and Upgrading Domains

This chapter describes how to build and upgrade domains. It contains the following sections:

- Building a Domain
- Building the Domain Manually or Through the Command-Line Interface
- After Building the Domain
- fixDomain

Building a Domain

After a fully defined configuration is created, an RPAS domain can be installed. Since building an RPAS domain is a manual process, it is expected that this process is supported by UNIX administration if the domain is installed on a UNIX platform. If the domain is being installed on Windows servers for prototyping and demonstration purposes, it can be built using the RPAS Configuration Tools GUI installer.

Prerequisites

The following are the prerequisites for building a domain:

- Installation of RPAS on the server that will store the domains.
- Installation of the Configuration Tools on the server that will store the domains.
- A configuration built using the Configuration Tools.
- A collection of hierarchy input files that contain positions for the domain. A hierarchy data file (name.dat) is required for each defined hierarchy.
- Cygwin installed (for prototyping and demonstration on a Windows server only).

Client-Side Procedures

The following client-side procedures must be completed to build a domain.

To begin the domain build process, a configuration project built using the Configuration Tools is required. This can be a packaged template or a configuration created with the customer's specific hierarchies, measures, and workbooks. If using a new configuration, be sure to note the path where the configuration is saved on the local disk.

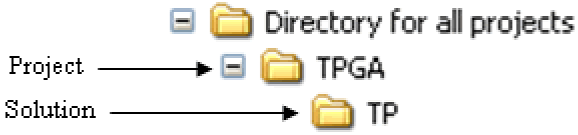
Note: The remainder of this section assumes that the domains are being built on a different server than Windows while the configuration is created on the Windows platform. If the domain is being built on the same server as the configuration, the steps for adding the configuration to the zip file and transferring to a different server can be eliminated.

Zip the Configuration Project Folder

1. Find the location where the configuration project is saved.
2. Using Windows Explorer, go to the path of the configuration project.
3. Right-click the **Configuration** folder and select **Add to Zip.** Package the entire contents of the project beginning with the configuration project root folder so that the zip file includes all solutions. It is important to zip the entire configuration project for the entire directory structure and not just the specific .xml files. Do not change the name of the configuration project folder or alter the contents of the folder in any way.

   In the following example, TPGA is the configuration selected to create the TPGA.zip.

Figure 2–1 Zipping the Configuration Project Folder



4. Using FTP, transfer the .zip file over to the server in binary mode. This can be placed in the home directory for now.

Server-Side Procedures

The following server-side procedures must be completed to build a domain.

Note: Though the RPAS Configuration Tools are supported only on the Windows platforms, the installation tools are supported on all platforms. However, they require Java 1.6. Make sure that the server being used for the domain installation has the correct version of Java.

Unzip the Configuration File

1. Find the location where you are going to save the configuration project file.

Note: Always put an updated configuration project in a new directory path. Do not overwrite an existing configuration project.

---

2. Move the <Configuration Project>.zip file to this location.
3. Unzip the <Configuration Project>.zip file using the UNIX command:

```
unzip -a <Configuration Project>.zip
```

4. Do not change the directory name for the configuration project or alter the contents in any way.

Verify the Environment Variable Settings

Prior to this step, RPAS and the Configuration Tools must be installed on the server that will store the domains. During that process, the necessary environment variables for RPAS and the Configuration Tools must be defined. Refer to the "Appendix: Environment Variables" if the environment variables below have not been set up.

Log in to the server. Use the commands below to verify the environment settings:

```
echo $RPAS_HOME
echo $RIDE_HOME
echo $JAVA_HOME
echo $PATH
```

> **Note:** The path for the RPAS_HOME variable may change with each new RPAS release.

If any changes are made to the environment variable settings, remember to exit and restart the UNIX session in order to execute the corrected .profile. This step must be done before continuing with the remaining steps.

Preparing the Input Files

1. Designate a directory for the location of the input files and move the files into this directory.

> **Note:** As a recommendation, use the directory name to_rpas as a standard for the location of input files. At a minimum, the hierarchy files (product, location, and calendar hierarchy files) are needed to build the domain. At this time, a calendar file must be loaded.

2. If necessary, rename the hierarchy files to match the name of the configured hierarchies. The files must end in either .dat or .csv.dat. For example, a file for a configured product hierarchy named prod in the Configuration Tools should be either prod.dat or prod.csv.dat. When using the .dat extension, the format of the files must match the hierarchy configuration specified using the Configuration Tools. If using the .csv.dat extension, the files will contain fields that are comma separated. See "Loading Hierarchies Using loadHier" for more details.

Building the Domain Manually or Through the Command-Line Interface

Run the rpasInstall utility to build the domain. This executable is located in the path to $RIDE_HOME/bin.

rpasInstall

This section provides the details required for the upgrade of rpasInstall.

rpasInstall Usage

```
rpasInstall <arguments>
```

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

Table 2–1 Arguments Used by the rpasInstall Utility

Argument	Description
`[-fullinstall | -patchinstall | -testinstall]`	Indicates the type of installation to be performed, where: - `-fullinstall`: Builds a full domain and loads the hierarchy data files. - `-patch install`: Patches an existing domain. Updates or unregisters and registers measures that have changed (as necessary). - `-testinstall`: Used for testing only. Only generates configuration files. Required.
`-ch` *config_home*	Indicates the path to the directory containing the configuration file. Required.
`-cn` *config_name*	Indicates the name of the configuration. Required.
-in *input_home*	Indicates the path that includes the directory containing the input files for the domain to be created. If this directory includes a users.db file, then the users and groups within that file are added upon domain creation. For more information, see "Managing Users Using usermgr". Required.
`-log` *log_name*	Indicates the path that includes the name of the log file to be created or updated. Required.
`-configdir` *config_directory*	Indicates 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*	Indicates 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*	Indicates the partitioning dimension. Use if and only if the global domain is being implemented without the use of globaldomainconfig.xml.
`-rf` *function_name*	Indicates the filename of the function to be registered. This pairing may be repeated for multiple functions. This argument is not required if no functions are to be registered.
`-updatestyles`	Imports configured style information into the domain. This option is automatically set in a full install. If you do not use this flag in a patch installation, changes to configured styles will not be imported into the domain.

Notes: When a domain is built for the first time, an installation directory is created inside the domain. The installation directory is essential for the patch process and must not be removed, moved, or renamed.

When submitting an issue to Oracle Retail Customer Support, if you are asked to provide a domain, be sure to provide the installation directory as well. The following information must be provided in order to help Customer Support better diagnose the issue:

- The configuration
- The script used to run the rpasInstall script
- The domain
- The log output file

Validate Domain Build Results

When the domain build process is complete, the logfile should be reviewed to verify that the process executed successfully. Search for the words "ERROR," "FAILURE," and "exception" inside the logfile. The end of the logfile should look similar to the following output example:

```
Time: 58.451
COMPLETE
```

After Building the Domain

After you build the domain, it can be accessed by the RPAS Fusion Client.

Note: Building a domain creates the shell of the domain. All measures, rules, and workbooks are created, but the measures are not populated. Measures are populated with the loadmeasure utility. For more information, see "Loading Measure Data Using loadmeasure".

In order to connect to a domain, the domain information must be set with the Fusion Application Server and the RPAS DomainDaemon must be running. The taskflow must include tasks for the solution (either a single taskflow or a combined taskflow), and Foundation.xml must include the solution ID. See the *RPAS Installation Guide* for details on setting up the domain information. See "DomainDaemon" for information.

Before you can log into the new domain, you must create at least one user with the usermgr utility. Once you have created one user, you can use the "Adding a User" process to create others. Each user must be set up or available for external authentication (through either SSO, WebLogic/LDAP or WebLogic/embedded). For more information about the usermgr utility and creating users, see the "User Maintenance" chapter.

After logging into the domain and creating the appropriate users, ensure that the appropriate permissions to workbooks and measures are set. See "Security Administration Workbook" for more information.

Upgrading and Patching Domains

RPAS supports the upgrade of RPAS 11.0.x, 11.1.x, 12.x, and 13.x environments to RPAS 14.x.

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

For details on upgrading to this release of RPAS from any release of RPAS prior to 13.3, see "Upgrading From Any Release of RPAS Prior to 13.3".

For details on upgrading to this release of RPAS from RPAS 13.3 or later, see "Upgrading From RPAS 13.3 or Later".

For details on managing the 14.0 Cryptography Update process, see "RPAS 14.0 Cryptography Upgrades"

Upgrading From Any Release of RPAS Prior to 13.3

If you are upgrading to the current release of RPAS from any release prior to RPAS 13.3, complete the following steps.

You must first upgrade the domain to the latest 13.2.3 hotfix prior to starting this upgrade process.

> **Note:** For information on upgrading to the latest RPAS 13.2.3 hot fix, see the *13.2.3 RPAS Administration Guide for the Fusion Client*, "Upgrading and Patching Domains". Note that it is optional to perform Step 8 in that section. Step 8 states that you should run the `rpasInstall` utility with the `-patchinstall` argument. This is not necessary if you are upgrading to this release because you will run that utility and argument at the end of this section.

Domain Conversion

In RPAS 13.3, a new methodology called Integer Indexing was introduced to simplify the management of multidimensional data in an RPAS domain. Integer indexing aims at eliminating a lot of inefficiencies in the addition, removal, and reclassification of positions in a hierarchy and improving the performance of hierarchy operations.

Since integer indexing uses different internal data structures to manage the hierarchy, a pre-13.3 domain must be converted to the new format before it can be recognized by the newer version of RPAS. This conversion is accomplished by using the convertDomain utility.

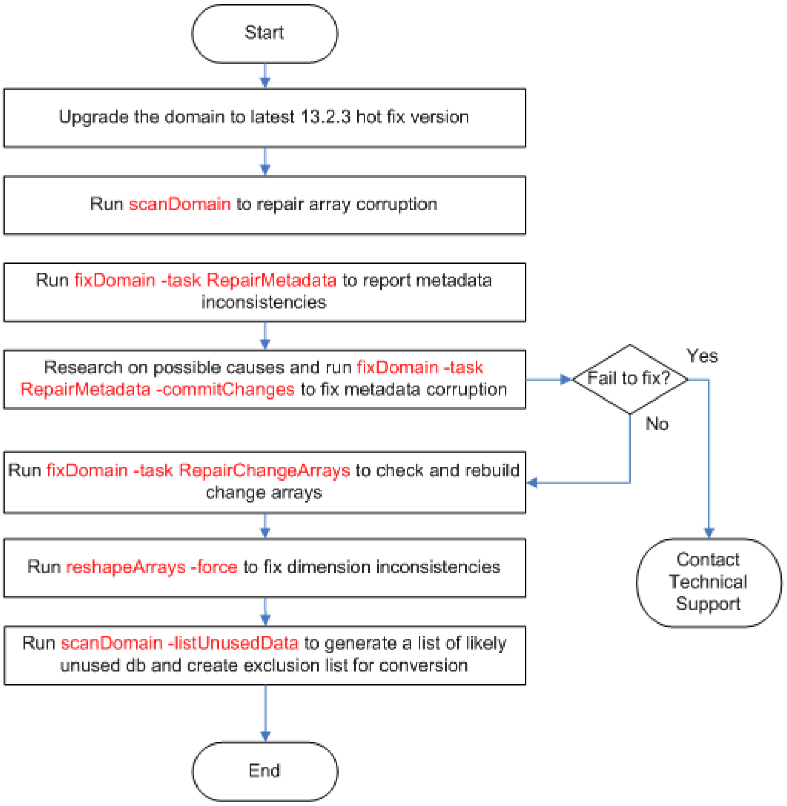
Domain conversion is an elaborate and complex process that runs on the assumption that the source domain is in a good state free of system level inconsistencies. In reality, an existing domain may be in an undesirable state due to various reasons. This can cause the domain conversion process to fail.

The information in this chapter can help you when upgrading to RPAS 13.3 or later. It provides general guidelines for RPAS domain conversion. It covers the preparation of the domain, setting up the conversion parameters, and carrying out the conversion and post-conversion processing.

Domain Preparation Before Conversion

The diagram below shows an overview of the domain preparation that is necessary before running domain conversion. All these utilities must be run under the RPAS 13.2.3 hotfix version 13.2.3.36 or later.

Figure 2–2 Overview of Domain Preparation


Start
Upgrade the domain to latest 13.2.3 hot fix version
Run scanDomain to repair array corruption
Run fixDomain -task RepairMetadata to report metadata inconsistencies
Research on possible causes and run fixDomain -task RepairMetadata -commitChanges to fix metadata corruption
Fail to fix?
Yes
No
Run fixDomain -task RepairChangeArrays to check and rebuild change arrays
Run reshapeArrays -force to fix dimension inconsistencies
Run scanDomain -listUnusedData to generate a list of likely unused db and create exclusion list for conversion
Contact Technical Support
End

The procedure is as follows:

1. Upgrade the domain to the latest 13.2.3 hot fix.

   RPAS supports the upgrade of RPAS 12.x environments to RPAS 13.x. If the domain is not currently on the latest 13.2.3 hot fix, it must be upgraded to that hot fix first before upgrading to 13.3.

   To upgrade to the latest RPAS 13.2.3 hot fix, see the 13.2.3 version of the *RPAS Administration Guide for the Fusion Client*, section"Upgrading and Patching Domains". Note that it is optional to perform Step 8 in that section. Step 8 states that you should run the rpasInstall utility with the -patchinstall argument. This is not necessary if you are upgrading to 13.3 because you will run that utility and argument at the end of this section.

   In addition, refer to the solution-specific Installation Guide, Implementation Guide, and Operations Guide, as appropriate, for potential solution upgrade limitations.

2. Run scanDomain to repair any array corruption.

   The scanDomain utility can be used for repairing data corruption in an RPAS database. Data corruption can occur if an external program modifies the RPAS database files or an unforeseen defect occurs in the processes using the RPAS database.

To run the utility from the command line, do the following:

```
scanDomain -d domainPath -repairCorruption -processes N
```

For more information about scanDomain, see the 13.2.3 release of the *RPAS Administration Guide for the Fusion Client*.

3. Run fixDomain to detect metadata errors.

   The Fix Domain utility is a tool used to detect and fix data inconsistencies in a domain. It supports a few different types of "tasks" to analyze different areas of data in the domain. Task "RepairMetadata" is used to detect and repair metadata inconsistencies.

   To run the utility from the command line, do the following:

   ```
   fixDomain -d domainPath -task RepairMetadata
   ```

   By default, this utility only reports errors without making changes to the domain. All errors are reported to standard output, which can be directed to a file.

4. Run fixDomain to fix metadata errors.

   Run fixDomain again in the commit mode to actually fix the metadata errors in the domain.

   To run the utility from the command line, do the following:

   ```
   fixDomain -d domainPath -task RepairMetadata -commitChanges
   ```

   Some metadata discrepancies can be too severe to fix. For example, discrepancies between the system arrays dim_hier and hierinfo cannot be fixed by the utility. It is recommended that you run fixDomain in report mode (without the -commitChanges flag) again to detect any remaining errors. Contact RPAS technical support if there are still errors.

   Caution: Note that fixDomain may remove measures from the domain if the metadata for those measures are corrupted. This may prevent some workbooks from being built. If this happens, the missing measures should be re-registered in the domain.

5. Run fixDomain to check and rebuild change arrays.

   Change arrays are system arrays used by the reshapeArrays utility. They should be rebuilt in case they are out of sync with other domain data, which can cause the failure or partial failure of the reshapeArrays.

   To run the utility from the command line, do the following:

   ```
   fixDomain -d domainPath -task RepairChangeArrays -commitChanges
   ```

6. Run reshapeArrays to fix dimension inconsistencies.

   This cleans up potential dimension inconsistencies between data arrays and dimension dictionaries.

   To run the utility from the command line, do the following:

   ```
   reshapeArrays -d domainPath -registered -force -processes N
   ```

7. Run scanDomain to generate a list of likely unused databases.

   An existing RPAS domain can accumulate a lot of unused databases during the course of its history. If those databases are not excluded from the domain conversion process, at a minimum they can slow down the conversion considerably, and, worse yet, they can cause the conversion to fail if any of the data arrays in them is corrupt. The scanDomain utility has been enhanced to list all likely unused databases of the domain. Best efforts have been applied to generate a complete and accurate list. However, some databases may have been created by custom RPAS extension libraries or third party plug-ins and can be listed as unused by mistake. This list should be examined and corrected if necessary.

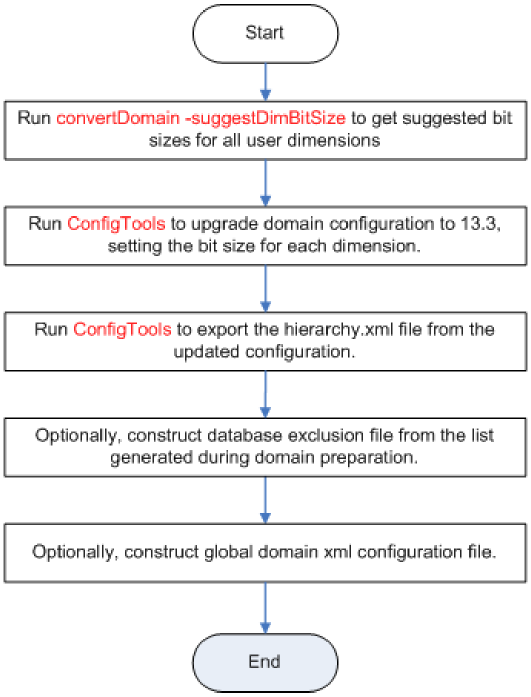
   To run this utility from the command line, do the following:

   ```
   scanDomain -d domainPath -listUnusedData outputFile
   ```

Setting up Domain Conversion Parameters

The convertDomain utility can take three configuration input files: a hierarchy configuration file, an optional global domain configuration file, and an optional database exclusion file. The diagram below shows an overview of setting up these files. All utilities should be run under RPAS 13.3 hotfix version 13.3.0.17 or later, RPAS 13.3.1 hotfix version 13.3.1.5 or later, or RPAS 13.4 or later.

Figure 2–3 Setting Up Domain Conversion Parameters


Start
Run convertDomain -suggestDimBitSize to get suggested bit sizes for all user dimensions
Run ConfigTools to upgrade domain configuration to 13.3, setting the bit size for each dimension.
Run ConfigTools to export the hierarchy.xml file from the updated configuration.
Optionally, construct database exclusion file from the list generated during domain preparation.
Optionally, construct global domain xml configuration file.
End

The procedure is as follows:

1. Run convertDomain to generate a suggested bit size for each user dimension.

   To run the utility from the command line, do the following:

   ```
   convertDomain -src domainPath -suggestDimBitSize outputFile
   ```

See "convertDomain" for more information on using the suggestDimBitSize option.

2. Run ConfigTools to upgrade the domain configuration and to set the bit sizes.

   Use the RPAS Configuration Tools to upgrade the 13.2.3.x domain configuration to 13.3, setting the bit size for each dimension. (Bit size configuration has an impact on performance. See the "Defining Dimension Properties" section in the *RPAS Configuration Tools User Guide* for instructions on setting the bit size and reindex threshold.)

   At this point, no additional changes should be made to the configuration; those changes must be processed separately at a later stage in the process. Solutions that include plug-ins that are normally run as part of the domain upgrade process will also be handled later.

3. Export the hierarchy.xml file from the updated configuration.

   This file can be created by running the hierarchy.xml report from within the Report Generator of the RPAS Configuration Tools.

4. Construct the database exclusion file (optional).

   An existing domain may contain a lot of stale or obsolete databases in it that can hinder the domain conversion progress. During the domain preparation, the scanDomain utility can be used to generate a list of likely unused databases. After confirmation, these databases can be put in an XML file that can be named exclusionList.xml.

   See "convertDomain" for more information on the use of exclusion lists.

5. Construct the global domain XML configuration file (optional).

   If you are converting a global domain, and not all local domains reside under the global domain path, then an XML configuration must be constructed to specify the source and destination path for each local domain.

   See "convertDomain" for more information on domain.xml configuration files.

Running the Domain Conversion

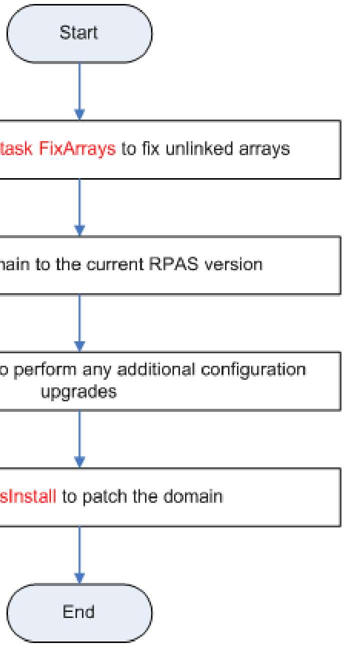
Once the domain has been prepared and all input parameters have been generated, the domain can be converted using the convertDomain utility. This utility will create a new integer indexing domain rather than updating the existing pre-integer indexing domain. This new domain can then be used to finish the upgrade process.

See "convertDomain" for more information on using the convertDomain utility.

Post-Conversion Processing

After the domain has been successfully converted to integer indexing, some post-conversion processing steps, as shown in the following diagram, are required to bring the domain up to date.

Figure 2–4 Post-Conversion Processing


Start
Run fixDomain -task FixArrays to fix unlinked arrays
Upgrade domain to the current RPAS version
Run ConfigTools to perform any additional configuration upgrades
Run rpasInstall to patch the domain
End

To run post-conversion processing, do the following:

1. Run fixDomain to fix unlinked arrays.

   If any arrays are listed in the unlinkListFile during the domain conversion, they must be fixed by running the fixDomain utility. The "FixArrays" task fixes unlinked arrays that contain one or more dimensions that are not linked to the dimension registry.

   See "fixDomain" for more information on fixing unlinked arrays.

2. Upgrade the domain to the current RPAS version.

   The upgradeDomain utility is used to upgrade just the RPAS version of the domain. It does not update the configuration or any other aspects of the domain itself. Refer to "upgradeDomain" for more information.

3. Upgrade the domain configuration.

   Run the new version of the Configuration Tools to upgrade existing configurations to the latest version. Those changes should be made to the configuration modified in the Step 2 of the "Setting up Domain Conversion Parameters". Refer to"Upgrade Configurations" for more information.

4. Patch the domain.

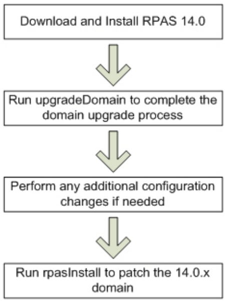
   Running the patch installation through rpasInstall over the converted domain is required even if there are no configuration updates. Refer to "rpasInstall" for more information.

Upgrading From RPAS 13.3 or Later

If you are upgrading to the current release of RPAS from RPAS 13.3 or later, complete the following steps.

Figure 2–5 shows an overview of the upgrade and patching process. The detailed steps are provided after the figure.

Figure 2–5 Upgrade Process from RPAS 13.3.x


Download and Install RPAS 14.0
Run upgradeDomain to complete the domain upgrade process
Perform any additional configuration changes if needed
Run rpasInstall to patch the 14.0.x domain

1. Acquire the latest 14.0.x RPAS release and install the following components:
   - RPAS Server
   - RPAS Client (Classic or Fusion)
   - RPAS Configuration Tools

   > **Note:** If you are using `rsp_manager` to install an RPAS 14.0.x patch, use the `-no_domain` option.

2. Run the `upgradeDomain` utility to complete the conversion of the domain to the current release. See "upgradeDomain" for more information.
3. Make any additional configuration changes to the domain (such as the re-execution of plug-ins for applications that require the plug-ins to be run and applied through a patch install as part of an upgrade).
4. If any additional changes were made to the domain in step 3, those changes should now be applied to the upgraded domain. To apply these changes, run the `rpasInstall` utility with the `-patchinstall` argument. See "rpasInstall" for more information.

   > **Notes:** After you run the `upgradeDomain` utility, the patch installation (through `rpasInstall`) must be run over the domain even if there are no configuration updates. This is done to ensure that configuration within the domain is synchronized with any changes made as the result of the RPAS and Configuration Tools upgrade.
   >
   > When using the `-updatestyles` flag, it is not necessary to delete existing styles.

Updating Domain Content

When moving to a new version of RPAS and RPAS solutions, you can also update the content of the domain. For example, the content of workbooks can be modified or the business logic represented by a rule group can be updated.

To update the domain content, make the desired changes in the configuration using the RPAS Configuration Tools. Then, when you run the `-rpasInstall` utility with the `-patchinstall` argument, the changes are applied to the domain.

The sections below specify the restrictions and special cases for updating hierarchies, measures, rule groups, and workbooks.

Hierarchies

When patching a domain, you can add new hierarchies. A hierarchy file must be present in the input directory specified in the call to rpasInstall for that hierarchy to be added.

> **Note:** Although a new hierarchy can be added, the order of the existing hierarchies cannot be changed. However, the order of the new hierarchy can be between or after existing hierarchies.

Patching a domain upgrades some, but not all, hierarchy and dimension attributes. For existing hierarchies, only the security dimension is updated. For dimensions, the following attributes are updated: user-defined dimensions, labels, and the state of image support (enabled or disabled). In addition, a dimension can have DPM support and translation support enabled but not disabled. You can also change the bit size and reindex threshold of an existing dimension (see "Reindexing Domains Using reindexDomain") as well as add or rename dimensions (see "Adding New Dimensions to Hierarchies").

Patching a domain does not patch changes to the hierarchy purge age or change the multi-language setting for a domain.

> **Note:** If updating the hierarchy purge age inside an existing domain, use the `loadHier` utility in batch mode to update the current settings. See "Loading Hierarchies Using loadHier" for more information.

Measure Properties

Certain measure properties cannot be modified without unregistering and reregistering the measures (which results in the loss of measure data). The measure properties that can be modified without being reregistered are listed in "Registering Measures Using regmeasure". If all of the measure properties that are changed can be modified without reregistration, the measure data will not be lost in the domain during the patch process.

If even one of the measure properties that change cannot be modified without re-registration, the patch process results in the un-registration and re-registration of the measure. Therefore, the data that was in the domain for that measure before the patch process will not be there after the patch process is complete. For more information, see "Registering Measures Using regmeasure".

Rule Groups

As part of the patch installation process, rule groups can be deleted, created, or modified with no restrictions. During the patching process, the rule sets, rule groups, and rules are completely rebuilt.

Workbooks

As part of the patch installation process, workbooks can be deleted, created, or modified with no restrictions.

Note: If you remove a template from the configuration and then patch the domain, the formatting for that template will be deleted.

RPAS 14.0 Cryptography Upgrades

As a part of the 14.0 release, RPAS incorporated a number of improvements to its cryptography functionality. These changes involved the ability to select a desired cipher suite and hash iteration count for the encryption of user credentials. As a result of these upgrades, the user dictionary of any domain will be automatically updated by the upgradeDomain utility when it converts a pre-14.x domain to version 14.x or later.

upgradeDomain can perform the conversion using a default cipher suite and iteration. However, it is possible to specify to upgradeDomain to use values other than the default when performing the upgrade. This is accomplished by setting two domain properties prior to running upgradeDomain. These properties are:

Table 2–2 Domain properties for Cryptography upgrade

Domain Property	Description
r_upgrade_hash_alg	This property sets the value for r_crypt_hash_alg in the upgraded domain. Because the list of algorithms can change with each release, it is not feasible to provide a list here. To get the list of values that can be used here, go to the rpasHome of the version to which you are upgrading and run: usermgr -listHashAlgs
r_upgrade_hash_iter	This property sets the value for r_crypto_hash_iter in the upgrade process. It is the number of iterations of hashing to perform on the password.

Note: The default cipher suite and hash iteration count will vary depending on the version of RPAS the domain us being upgraded to. In every release, RPAS will use the suite and iteration count deemed most secure at the time of that release.

For details on setting domain properties, see "Setting Miscellaneous Domain Properties using domainprop"

Upgrade Configurations

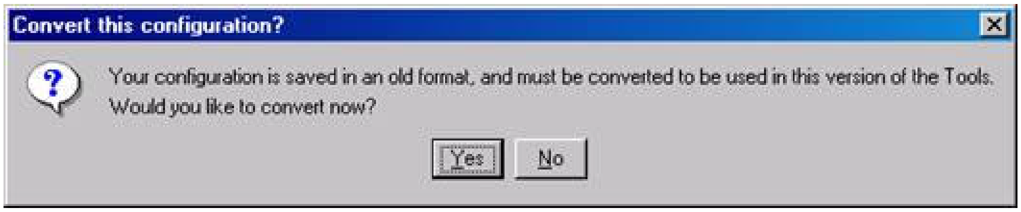
After the RPAS Configuration Tools have been installed and the domains have been upgraded, you may need to upgrade existing configurations to version 13 or the latest version 13 patch.

Note: Configuration upgrade is not always required. When it is required, the Configuration Tools automatically notifies you that a configuration upgrade is needed.

1. Launch the new version of the Configuration Tools.
2. Open the existing configuration by selecting **Open** from the File menu.
3. Select the configuration and click **OK**.

4. A message box appears prompting you to convert the configuration. Click **Yes**.

Figure 2–6 Converting the Configuration

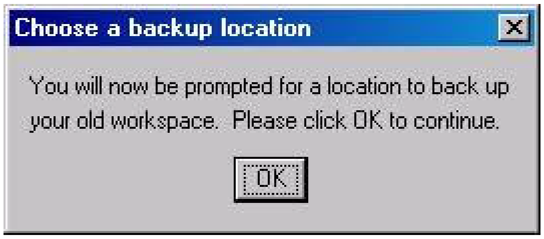


Convert this configuration?

Your configuration is saved in an old format, and must be converted to be used in this version of the Tools. Would you like to convert now?

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

Figure 2–7 Choosing the Backup Location



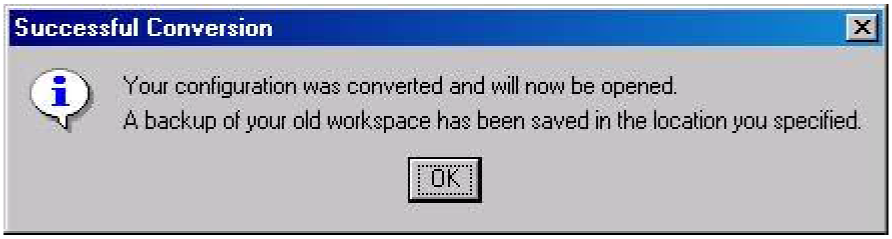
Choose a backup location

You will now be prompted for a location to back up your old workspace. Please click OK to continue.

6. 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.

Figure 2–8 Confirmation of Conversion



Successful Conversion

Your configuration was converted and will now be opened.
A backup of your old workspace has been saved in the location you specified.

7. The upgraded configuration now opens 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 installation 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.

Update the solution configuration for each application in the following order:

1. Curve
2. RDF
3. RDF Clone
4. Promote
5. Grade

See the RDF, Curve, or Grade Configuration Guides for more information on the auto-generation process for each application.

convertDomain

The convertDomain utility creates a new domain that has all the dimensions and measure arrays converted to integer-indexed arrays.

The conversion process creates a new domain based on the source domain provided. Since a new domain is created from the domain and does not replace the source domain, the source domain remains unaffected.

Note: Because you are creating a second domain, you must have enough space for the original source domain and the new converted domain. Since workbooks are not converted, you only need the amount of space the source domain uses, minus the space that the workbooks in the source domain use.

The convertDomain utility copies all information specific to the domain at the time of conversion. It does not copy any custom information such as shell scripts, which exist under the domain root. Such information must be manually copied.

The convertDomain utility blindly copies the following:

- config (to master domain only)
- input
- installs
- output
- repos
- scripts
- fusionClient
- WizardPQD

The data directory is recreated base on the source domain. Only databases (.db) are converted, excluding the following:

- changearray.db
- shadow.db
- styles.db
- hmaint.db (rebuilt separately)
- measdata.db (rebuilt separately)
- master_staging.db (use only for loadhier)
- alitemp*.db

You can also use an exclusion list to exclude more databases from the conversion.

If files exist in the data directory but are not a database, they are not copied.

The users directory is recreated without all workbooks. Therefore, for all users' directory structures will only contain the <username>.db and <username>.db.lck files.

The styles directory is copied conditionally. Only style files listed in #workbook.xml are copied. Any arrays that exist in that directory are ignored.

If your domain contains files or directories that are not copied (based on the information above), you must manually copy those files and directories to the converted domain.

Note: The convertDomain utility can be run multiple times if needed. Data already converted will be skipped during each subsequent run.

convertDomain Usage

```
convertDomain   [-src srcDomain -dest destDomain | -xmlConfigFile XMLConfigFile]
-xmlHierFile PathToHierarchy.xml [-exclude exclusionList.xml] -genUnlinkedList
unlinkedListFile -maxProcesses count
```

To use the convertDomain utility, perform the following steps:

1. Provide a pair of source/destination domain paths using the `-src` and `-dest` arguments. Or, supply an XMLConfig file.
    - If you use `-src` and `-dest` arguments to specify the source and destination domain paths, the local domain path must be under the global domain path.

- If converting a global domain, and not all local domains reside under the global domain path, then you must use the `-xmlConfigFile` argument to specify the source and destination path for each local domain.
- The `XMLConfigFile` should follow this syntax:

```
<rpas>
    <globaldomain>
       <srcPath>/path/to/13.2.3/domain</srcPath>
       <dstPath>/path/to/13.3.0/destination</dstPath>
       <subdomain>
          <srcPath>/path/to/13.2.3/local/domain/0</srcPath>
          <dstPath>/path/to/13.3.0/destination/local/domain/0</dstPath>
       </subdomain>
       ...
    </globaldomain>
</rpas>
```

2. Use the `-xmlHierFile` to specify the path to the hierarchy.xml file that is exported from the upgraded configuration, including a BitSize attribute for each dimension.
3. If it is desired, the -exclude argument may be used to specify databases that should be excluded from the conversion process. The databases specified in the exclusionList.xml document will not be recreated in the new integer indexing domain. The exclusionList.xml should use the following format. Note that if the path is an absolute path (starting with a slash "/"), it refers to only one directory. If it is a relative path, it applies to the master domain and all local domains in a global domain environment.

```
<rpas>
    <convertDomain option="exclude">
        <path>relative/path/to/database1.db</path>
        <path>relative/path/to/database2.db</path>
        <path>/absolute/path/to/database.db</path>
        …
    </convertDomain>
</rpas>
```

4. Under certain conditions, an array in a pre-integer indexing domain can become corrupted such that the embedded dimension information of the array does not match the dimension information contained in the dimension dictionary of the domain. These arrays are called unlinked arrays. When convertDomain is run with the -genUnlinkedList argument, it creates a file in the specified location containing any arrays that are corrupted. The fixDomain utility can then be used to bring these unlinked arrays into conformity with the domain. See "fixDomain" for more information on restoring unlinked arrays.
5. Use the `-maxProcesses` argument to specify the number of processes that can be used.

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

Table 2–3 Arguments Used by the convertdomain Utility

Argument	Description
`-src` *srcDomain*	Specifies the path to the source domain being converted.
`-dest` *destDomain*	Specifies the path to the destination domain.

Table 2–3 (Cont.) Arguments Used by the convertdomain Utility

Argument	Description
`-xmlConfigFile` *XMLConfigFile*	If you are converting a global domain and not all local domains reside under the global domain path, use this argument to specify the source and destination paths for each local domain.
`-xmlHierFile` *PathToHierarchy.xml*	Specifies the path to the hierarchy.xml file that is exported from the upgraded configuration. This file should include the new BitSize attribute for each dimension.
-exclude *exclusionList.xml*	Excludes the databases listed in the specified file from the conversion process.
-genUnlinkedList *unlinkedListFile*	Creates a file in the specified location containing the names of arrays whose embedded dimension information does not match the dimension dictionary of the domain. These arrays can then be corrected using the fixDomain utility. Works with an absolute path.
`-maxProcesses` *count*	If specified, some parts of convertDomain will run in parallel. This means that it will use a maximum of the defined processes, which are specified by *count*.

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.

convertDomain -suggestDimBitSize

The convertDomain utility can also be used to analyze a pre-integer indexing domain in order to suggest appropriate values for the bitSize attributes of the dimensions in that domain.

The convertDomain utility enumerates all non-system dimensions in the source domain, calculates a suitable bit size based on the current number of positions (including unused buffer positions) within the domain, and prints them out in a table format. It then calculates the key size for each of the data arrays for all measures, based on those dimension bit sizes. If the key size is above 64 bit for any measure, it will print out the measure with a warning message so that the user can consider a lower bit size for any of the dimensions in the array.

The algorithm for suggesting the bit size is to find the number of bits that can fit twice the number of positions currently in the dimension, starting with 2 bits. The results of the analysis are output to a file. Below is an example of this output:

Table 2–4 Example of Analysis Output

Hierarchy	Dimension	Size	BitSize	Capacity	DPM-Enabled
clnd	day	3647	13	8192	No
clnd	week	522	11	2048	No
loc	str	22	6	64	Yes
loc	dstr	12	5	32	No
prod	sku	488	10	1024	Yes
prod	stco	258	10	1024	Yes

Note: The following measures have a key size greater than 64 bits. For better performance, it is recommended that you reduce the bit size of one or more dimensions in the base intersection of these measures so that the key size is within 64 bits.

Measure Name	Base Intersection	Key Size
ACT	www_xxx_yyy_zzzz	67

Usage for -suggestDimBitSize

```
convertDomain -src domainPath -suggestDimBitSize outputFile
```

The following table provides descriptions of the arguments used by the convertDomain utility with -suggestDimBitSize.

Table 2–5 Arguments Used by the convertDomain Utility with -suggestDimBitSize

Argument	Description
-src *domainPath*	Indicate the path to the domain being analyzed.
-suggestDimBitSize outputFile	Indicates the location in which to create output file containing bitSize recommendations.

upgradeDomain

The upgradeDomain utility is used to upgrade only the RPAS version of the domain. It does not update the configuration or any other aspects of the domain itself.

upgradeDomain Usage

```
upgradeDomain -d domainPath [OPTIONS]
```

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

Table 2–6 Arguments Used by the upgradedomain Utility

Argument	Description
`-d` *path*	Indicates the path to the domain being upgraded.
`-verbose`	Shows the detail about each change that is applied to the domain. Optional.
`-n`	Reports which changes would be applied without applying the changes. Optional.
`-purgeWorkbooks`	Purges all existing workbooks and clears the workbook batch queue. Optional.
`-ignoreSharedNames`	Allows upgrade even if dimensions and hierarchies share names.
`-apptag`	Indicates the application and version associated with this upgrade. Parameter must be APP:VERSION.

Note: An administrator can also run the `domaininfo` utility to verify the upgrade process, as shown below.

```
domaininfo -d pathtodomain -domainversion
```

fixDomain

The fixDomain utility is used to repair many types of corruption that can occur within an RPAS domain. This utility can be used to perform a number of tasks, which are described below. In all cases, fixDomain can execute in two modes. The default mode is to perform an examination to detect problems. The second mode, which is performed if the -commitChanges argument is passed in the call to fixDomain, is to take action to correct problems that are detected.

Remove inconsistent measure attributes

It is possible for inconsistencies in measure attributes between the master and local domains of a global domain to interfere with operations performed by RPAS. By executing fixDomain with the CleanMeasureAttrs task, users can analyze a domain for such discrepancies and correct them by removing attributes that appear in only a subset of the domains.

Remove partially unregistered measures

It is possible for the process of measure un-registration to exit without completely removing the measure from all local domains of a global domain. By executing fixDomain with the CleanMeasure task, users can analyze a domain for partially unregistered measures and correct them by completing the un-registration process and completely removing these measures from the domain.

Repair discrepancies in hierarchy information

It is possible, in integer-indexing domains, for the hierarchy and dimension information of an array to become out of synch with the information of the domain. It is also possible for this information to become out of synch between the local domains of a global domain. By executing fixDomain with the RepairHiearchy task, users can analyze a domain for desynchronized dimension information and correct it when found by updating arrays or local domains with the dimension information of the master domain.

Repair unlinked arrays

In pre-integer indexing domains, there may be discrepancies between the dimension information in an array and the dimension information of the domain. When these corrupted domains are converted to integer indexing, these arrays are converted to rely on internal dimension information and cannot be resolved against the dimension information of the domain. By executing fixDomain with the FixArrays task, users can analyze a domain to detect these unlinked arrays and correct them by rebuilding the array using the dimension information of the domain.

Repair metadata inconsistencies

System dimensions and related metadata define the basic structure of a domain. Any discrepancies within them can cause serious problems during domain operations. By executing fixDomain with the RepairMetadata task, users can analyze a domain for any discrepancies in system dimensions. Depending on the type of information that contains a discrepancy, fixDomain may or may not be able to resolve the issue.

Hierarchy and Dimension Information: In general, this type of discrepancy is a result of a domain that did not build correctly. Due to the severity of this type of problem, fixDomain cannot resolve such issues.

Measure Information: It is possible for measure information to appear in some but not all the metadata arrays of a domain. Depending on which arrays do contain information about a measure, fixDomain may be able to correct the problem. Otherwise, the measures with incomplete information will be removed from the domain and must be re-registered.

Workbook Information: It is possible for information about workbooks, workbook templates, and workbook template groups to appear in some but not all of the metadata arrays of the domain. Depending on which arrays do contain information about the workbook, template or group, fixDomain may be able to correct the problem. Otherwise, the workbooks, templates or groups with incomplete information will be removed from the domain and must be re-registered.

User Information: It is possible for information about RPAS users and user groups to appear in some but not all of the metadata arrays of the domain. Depending on which arrays do contain information about the user or group, fixDomain may be able to correct the problem. Otherwise, the user or group will be removed and must be re-registered.

Note: The RepairMetadata task is only supported in versions of 13.2.3.x after 13.2.3.36. It cannot be used in RPAS version 13.3 and later. It is described here to support the upgrade and conversion process. See"Upgrading From Any Release of RPAS Prior to 13.3" for more information on the convertDomain process.

Repair corrupted change arrays

In pre-integer indexing domains, it is possible for change arrays to become out of synch with the measure metadata of a domain. By executing fixDomain with the RepairChangeArrays task, users can analyze a domain for any discrepancies between change arrays and measure metadata and repair them, based on the measure metadata of the domain.

Note: The RepairChangeArrays task is only supported in versions of 13.2.3.x after 13.2.3.36. It cannot be used in RPAS version 13.3 and later. It is described here to support the upgrade and conversion process. See "Upgrading From Any Release of RPAS Prior to 13.3" for more information on the convertDomain process.

fixDomain Usage

The following arguments, described inTable 2–7, are used with fixDomain:

```
fixDomain -d pathToDomain -task CleanMeasure [-commitChanges] [-loglevel level]
fixDomain -d pathToDomain -task RepairHierarchy -hier hierName [-commitChanges]
[-loglevel level]
fixDomain -d pathToDomain -task FixArrays [-db dbName | -array arrayName]
[-commitChanges] [-loglevel level]
fixDomain -d pathToDomain -task RepairMetadata [-dim dimName] [-commitChanges]
[-loglevel level]
fixDomain -d pathToDomain -task RepairChangeArrays [-commitChanges] [-loglevel
level]
```

Table 2–7 Arguments Used by the fixDomain Utility

Argument	Description
-d pathToDomain	Indicates the location of the domain begin repaired.
-d CleanMeasureAttrsthe	Detects and removes measure attributes that do not exist in the master and all local domains of a global domain.
-task CleanMeasure	Detects partially unregistered measures and completes their un-registration.
-task RepairHierarchy	Detects and repairs inconsistencies in hierarchy information in arrays within the master and local domains of a global domain.
-hier *hierName*	Used in conjunction with the Repair Hierarchy task. Specifies the hierarchy to correct within arrays.
-task FixArrays	Detects and corrects unlinked arrays.
-db *dbName*	Used in conjunction with the FixArrays task. Specifies the database to examine or repair. The FixArrays task requires the use of either the -db or the -array argument, but not both.
-array *arrayName*	Used in conjunction with the FixArrays task. Specifies the array to examine or repair. The FixArrays task requires the use of either the -db or the -array argument, but not both.
-task RepairMetadata	Detects and repairs metadata discrepancies in system dimensions. **Note:** the RepairMetadata task is supported only in versions of 13.2.3.x after 13.2.3.36. It cannot be used in RPAS version 13.3 and later.
-dim *dimName*	Used in conjunction with the RepairMetadata task. Specifies the system dimension to examine or repair. This is an optional argument. If not supplied, fixDomain will examine all system dimensions.
-task RepairChangeArrays	Detects problems in change arrays within the domain and repairs them by relying on the measure metadata of the domain. **Note:** the RepairChangeArrays task is supported only in versions of 13.2.3.x after 13.2.3.36. It cannot be used in RPAS versions 13.3 and later.
-commitChanges	If the -commitChanges argument is not supplied to any call of fixDomain, it will only examine the domain and report detected problems. The fixDomain utility only modifies the domain to repair changes if the -commitChanges argument is specified.
-loglevel *level*	Sets the logging verbosity level. Supports use of all, profile, debug, audit, information, warning, error and none as values of level.

3

Domain Administration

This chapter describes domain administration tasks such as using the DomainDaemon and setting environment variables. It contains the following sections:

- DomainDaemon
- Losing a Client-Server Connection

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 the 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] represents the full path to the directory where the RPAS Server is installed.

Note: On rare occasions, the DomainDaemon can throw the following exception when SSL is enabled: Failed to initialize Oracle SSL environment (nzerror 43120: NZERROR_LX_ERROR).

This is caused by an unmatched language setting of the operating system. Setting the environment variable NLS_LANG using the following command before starting the DomainDaemon will resolve this problem.

```
export NLS_LANG=AMERICAN
```

DomainDaemon Usage

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

Note: DomainDaemon commands that talk to a running DomainDaemon (stop, ping, stopActiveServers, etc) now need to specify the wallet information. This can be set in the environment variable RPAS_SSL_WALLET to avoid having to pass it on the command line.

Table 3–1 DomainDaemon Utility Arguments

Argument	Description
-version	Prints the RPAS version, revision, and build information of the utility.
-start	Starts a DomainDaemon on the specified port.
-ipaddr	Specifies which local IP address on a multi-homed server that the DomainDaemon and its spawned RpasDbServer processes will listen to. If this argument is omitted, the DomainDaemon and RpasDbServer processes will listen to all local IP addresses.
-port *portNum*	Defines the *portNum,* which must be between 1025 and 65535 (inclusive). If *portNum* is set to auto, it will find any free port.
-loglevel	Enables additional logging. **Note**: The -debug option has been deprecated. Instead of using -debug, use -loglevel to add 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.
-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 or 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.

Table 3–1 (Cont.) DomainDaemon Utility Arguments

Argument	Description
-ssl	Specifies whether the messages exchanged between the Fusion Client and the RPAS database server are encrypted using a password generated during the initial SSL handshake. -ssl 2: two-way authentication with encryption. -ssl 4: two-way authentication without encryption. The ssl 4 option is useful for customers who have ensured that an unauthorized third party cannot intercept the messages exchanged between the Fusion Client and the RPAS database server. Encryption is not be valuable in this case. By avoiding the encryption overhead they obtain a slight improvement in performance. For more information on SSL, see Chapter 4, "SSL."
-CipherSuite	Specifies the encryption algorithms used by SSL with the -cipherSuite field. A cipher suite consists of three components: - Asynchronous: The asynchronous algorithm is used in the SSL handshake to validate the client/server certificates and exchange the session key. - Synchronous: The synchronous algorithm uses the session key to encrypt data transfers between the client and server. - MAC: The MAC is a hash of the message contents appended to the end of the message to validate that the contents of the message were not altered. Because cryptography requirements evolve over time, this list of supported algorithms will change from release to release. To see which cipher suites are supported by your version of RPAS, use the -listCipherSuites option. If -cipherSuite is not specified, a default suite will be provided. The default suite will be whichever one is considered the most secure at the time of release. Conversely, if at the time of release a supported cipher suite is considered insecure, RPAS will deprecate it. Using a deprecated cipher suite will result in a warning being logged when starting DomainDaemon
-listCipherSuites	Lists the cipher suites supported by the current version of RPAS. Deprecated cipher suites are marked with a (X), and the default suite is marked with a (*). Because the list of available cipher suites can vary based what type of SSL you are using, you must specify the -ssl argument.
-wallet	Specifies the location of the server wallet folder. The wallet contains the RPAS server's private key and certificate and the client's certificate. Format: -wallet file:<path to the server wallet folder> Example: -wallet file:d:\wallets\server [on Windows] on Unix: -wallet file:/u00/admin/wallets/server

Starting the DomainDaemon

To start the DomainDaemon, execute the DomainDaemon utility. 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.

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

```
DomainDaemon -port 55278 -start -ssl 2 -wallet file:/usr/adm/wallets/server
```

If the command is successful, the DomainDaemon returns the following message:

```
Domain Daemon started with two-way SSL.
Daemon listening on port '55278'.
```

A server may be configured with multiple network interfaces and each network interface is assigned one or more unique IP addresses. A server with multiple local IP addresses is known as "multi-homed". On such a server, the user may use the -ipaddr argument to specify which local IP address the DomainDaemon and its spawned RpasDbServer processes will listen to. If the -ipaddr argument is not specified, then DomainDaemon and RpasDbServer will listen to all of the local IP addresses on the server. If the DomainDaemon is listening to only one local IP address, then the DomainDaemon's port is still available on the remaining local IP addresses.

Note: The proper environment variables must be set in order to execute the DomainDaemon. See the Verify the Environment Variable Settings section for more information about setting RPAS_HOME and updating the PATH variable.

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.

```
DomainDaemon -port 55278 -ping
DomainDaemon on port 55278 is alive.
```

It is possible to have multiple instances of DomainDaemon listening to the same port but different local IP addresses. Thus, the 'port' parameter by itself is no longer sufficient to identify a DomainDaemon instance. Therefore, when sending commands to the DomainDaemon, the 'ipaddr' parameter will need to be specified along with the 'port' parameter to properly identify which instance of the DomainDaemon must be stopped. If the 'ipaddr' parameter is omitted, it will behave as it currently does and send the stop request to the DomainDaemon instance listen-ing to the specified port and the host's primary local IP address (determined by 'localhost').

Stopping the DomainDaemon

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

```
DomainDaemon -port 55278 -stop
```

It is possible to have multiple instances of DomainDaemon listening to the same port but different local IP addresses. Thus, the 'port' parameter by itself is no longer sufficient to identify a DomainDaemon instance. Therefore, when sending commands to the DomainDaemon, the 'ipaddr' parameter will need to be specified along with the 'port' parameter to properly identify which instance of the DomainDaemon must be stopped. If the 'ipaddr' parameter is omitted, it will behave as it currently does and send the stop request to the DomainDaemon instance listen-ing to the specified port and the host's primary local IP address (determined by 'localhost').

Losing a Client-Server Connection

The connection between the RPAS Client and the RPAS Server can be lost for any number of reasons, but occurs most commonly when the user's computer crashes or the network connection is lost.

If an RPAS Client-Server connection is lost, the user's work is guaranteed to be saved up to the last calculation; all deferred calculations are lost. Upon subsequent login, the user can access either the last version of the original workbook explicitly saved by the user or the auto-saved version of the workbook that was being worked on when the connection was lost.

If the user tries to log in after recently losing the connection or from a different instance of the RPAS client, the user is prompted to either terminate all existing sessions and start a new session, or start a new concurrent user session. If the user chooses to terminate the existing session, the RPAS server gracefully terminates the existing session and then logs the user in and starts up a new session.

Graceful Termination of the Existing Session

If a connection has been lost and the previous session has not timed out or a session for the user is running from a different instance of the RPAS client, the user can use the RPAS client to gracefully terminate the existing session and log into a new session. Graceful termination includes the completion of any pending processes and custom menu or rule group processing in the existing session, followed by an auto-save of the workbook and subsequent termination of the server process.

Graceful termination of an existing session could take an arbitrarily long time. This time includes both the sum of the time taken to complete any running calculations and the time to save the workbook.

When the RPAS server times out waiting on a request from the RPAS Client, it gracefully terminates, irrespective of whether the user tried to log in again.

If for any reason the client server connection is lost before the new login occurs, RPAS asks users if they want to resume the previous session or terminate it and start a new one. However, it will not start another session for any user until that user regains connectivity and tries to log in again.

There is a limit on the number of login sessions per user. By default, the maximum number of concurrent login sessions per user is five. This limit can be altered by an administrator using the Security Administration workbook.

Autosaved Workbooks

When an RPAS Server session terminates automatically, either due to re-login or a server timeout, it auto-saves the workbook that is currently open by the session. An auto-saved workbook includes all of the user's work up to the last calculation. If the user had some pending calculations, that is, edits were made in the client but the calculation was not performed, all the edits are lost.

When a workbook is auto-saved, the original workbook is kept in the same state it was in the last time it was explicitly saved by the user. A new workbook is created for the user with the name of the original workbook suffixed with '_autosave' and added to the user's Most Recently Used list. Upon subsequent login, the user can view both the original workbook and the auto-saved workbook in the workbook list.

RPAS administrators can impose a limit on the number of workbooks a user can create for a particular workbook template. When such a limit has been imposed, the auto-save feature allows the user to exceed that limit by one. This allows the user to operate at the limit without the fear of losing any work because of a connection failure or computer crash. However, when the limit is exceeded due to an auto-save, the auto-save feature is disabled for the user for the workbook template on which the limit has been imposed. The auto-save feature is disabled until the user deletes one or more of the workbooks for that template in order to bring the user at or below the limit. It is not required that the user delete the original or the auto-saved workbook. Once the user is back at or below the limit, the auto-save feature is automatically enabled. Note that if a user exceeds the limit, the RPAS Client will inform the user of this situation every time the user attempts to open a workbook for the given template.

4

SSL

Secure Socket Layer (SSL), a protocol for securing network connections, is used by RPAS to provide secure communication between RPAS client and server processes. RPAS Fusion Client supports two-way SSL and two-way SSL with authentication only.

This chapter contains the following sections:

- Introduction
- Setting Up SSL
- Two-Way SSL
- Two-Way SSL with Authentication Only
- Creating SSL Certificates Automatically

Introduction

In order to use SSL with RPAS, you must configure both the server and the client. For two-way SSL, this involves setting up key stores and trust stores used to manage certificates. The key store contains a private key and its corresponding public certificate chain. The trust store contains trusted public certificates and certificate chains.

In order to enable the use of SSL on RPAS, you must use the Domain Daemon to set up the SSL type and the Oracle wallet location.

Oracle Wallet is managed by a command line utility called orapki, which is included in the RPAS installation. JKS Keystore is managed by the keytool utility that is included in the standard JDK. Oracle Wallet and JKS are in different formats and cannot be used interchangeably. However, the orapki utility can be used to convert between them.

A Certificate Authority (CA) certificate is the public certificate from the CA that issues the server certificate and the client certificate.

Setting Up SSL

As part of the installation of the RPAS Fusion Client, the following SSL-related setup occurs:

- The user-selected password for accessing the client-side key store and trust store files is saved.
- The Fusion Client code has access to the password at run time.

After installation, the client-side key store and trust store files and the server-side wallet must be created.

Either before or after installation:

- The client-side key store must be copied as fckey.jks into the <configroot>/idstores directory.
- The client-side trust store files must be copied as fctrust.jks into the <configroot>/idstores directory.

Two-Way SSL

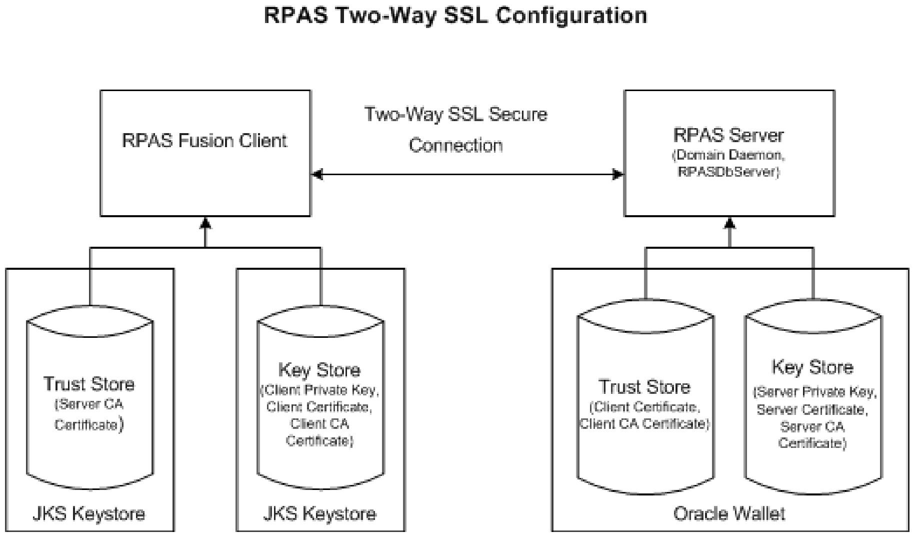
In two-way SSL, the client and the server each request to verify the identity of the other. In addition, the server requires that the client certificate be pre-registered in its trust store.

Two-way SSL authenticates both the server and the client. Both sides must have a key store and a trust store. The same wallet may be used with multiple domain daemon instances. This effectively means that all RPAS servers within a single customer site can use the same server certificate. The customer therefore needs to obtain two certificates. The RPAS server uses one Oracle Wallet for both stores. Both the trust store and the key store are in the JKS format.

On the Fusion Client side, the key store and the trust store are two files that use the JKS format for storing the keys and the certificates. They cannot be read by the orapki utility. Orapki can only handle wallet files created in the proprietary orapki format. This format is used on the RPAS server side. In other words, the server side has one single wallet file that has the keys and the certificates, written in the orapki format.

The trust store must contain the CA certificate in order to enable mutual authentication. In addition, the trust store of the server must contain the client certificate so that it can recognize and authorize that particular client. Both the client certificate and the server certificate typically have the same CA certificate since there is no reason to apply to different CAs for client and server.

Figure 4–1 RPAS Fusion Client Two-Way SSL


RPAS Two-Way SSL Configuration
RPAS Fusion Client
Two-Way SSL Secure Connection
RPAS Server (Domain Daemon, RPASDbServer)
Trust Store (Server CA Certificate)
JKS Keystore
Key Store (Client Private Key, Client Certificate, Client CA Certificate)
JKS Keystore
Trust Store (Client Certificate, Client CA Certificate)
Key Store (Server Private Key, Server Certificate, Server CA Certificate)
Oracle Wallet

Configuring the RPAS Server

To configure the RPAS Server for two-way SSL, you must set up a key store and a trust store, import the client certificate into the server wallet as trusted certificates, and then start the Domain Daemon. One Oracle wallet can act as both the key store and the trust store.

Setting Up the Key Store

To set up the key store, complete the following steps:

1. Create an Oracle wallet with auto login. This type of wallet can be read without a password. The default file permissions for the wallet are 0300 (accessible by owner only).

   ```
   orapki wallet create -wallet {server_wallet_directory} -auto_login -pwd
   {server_wallet_password}
   ```

   Example:

   ```
   orapki wallet create -wallet C:/wallets/server -auto_login -pwd serverpass1
   ```

2. Generate a private key.

   ```
   orapki wallet add -wallet {server_wallet_directory} -keysize {key_len} -dn
   {server_dn} -pwd {server_wallet_password}
   ```

   Example:

   ```
   orapki wallet add -wallet C:/wallets/server -keysize 2048 -dn "cn=rpas_qa_
   server,dc=us,dc=oracle,dc=com" -pwd serverpass1
   ```

3. Export a certificate request.

   ```
   orapki wallet export -wallet {server_wallet_directory} -dn {server_dn} -request
   {server_cert_req_file} -pwd {server_wallet_password}
   ```

   Example:

   ```
   orapki wallet export -wallet C:/wallets/server -dn "cn=rpas_qa_
   server,dc=us,dc=oracle,dc=com" -request C:/wallets/server_cert_req.txt -pwd
   serverpass1
   ```

4. The certificate request must be digitally signed before it can be used. If the deployment of the RPAS Fusion Client and the RPAS server are both under the control of the customer, then a self-signed SSL certificate may be used as the signer. Otherwise, the certificate request must be sent out for signing by a well-known certificate authority.

   A self-signed SSL certificate used in this manner may be referred to as a self-signed root certificate or as a private certificate authority.

   This is the method for creating a self-signed root certificate:

   1. Generate a private key for the root certificate:

      ```
      orapki wallet add -wallet {root_wallet_directory} -keysize {key_len} -dn
      {root_dn} -self_signed -validity {validity_days} -pwd {root_wallet_
      password} -addext_ski
      ```

      Example:

      ```
      orapki wallet add -wallet C:/wallets/root -keysize 1024 -dn
      "cn=root,dc=us,dc=example,dc=com" -self_signed -validity 3650 -pwd
      rootpass1 -addext_ski
      ```

   2. Export the root certificate chain to a file:

```
orapki wallet export_trust_chain -wallet {root_wallet_directory} -certchain
{root_cert_chain_file} -dn {root_dn} -pwd {root_wallet_password}
```

Example:

```
orapki wallet export_trust_chain -wallet C:/wallets/root -certchain
C:/wallets/root_chain.txt -dn "cn=root,dc=us,dc=example,dc=com" -pwd
rootpass1
```

In the above example, the file containing the root certificate chain is c:/wallets/root_chain.txt.

5. If using an external root certificate, create the server certificate signed by a third-party CA. (Alternatively, see step 6.) To do this, send the server certificate request file ({server_cert_req_file}) to the CA. The CA sends back a certificate ({server_certificate_file}) for the server, along with the public certificate ({root_cert_chain_file}) of the CA.
6. If using a self-signed root certificate, the server certificate file is signed as follows:

   ```
   orapki cert create -wallet {root_wallet_directory} -request {server_cert_req_
   file} -cert {server_certificate_file} -validity {validity_days} -pwd {root_
   wallet_password}
   ```

   Example:

   ```
   orapki cert create -wallet C:/wallets/root -request C:/wallets/server_cert_
   req.txt -cert C:/wallets/server_cert.txt -validity 3650 -pwd rootpass1
   ```

   The file c:/wallets/server_cert.txt is the signed server certificate, in the form of a file.
7. Import the CA into the server wallet. Note that the public certificate chain may contain more than one certificate. In such cases, the certificates must be imported one by one, starting at the top of the chain.

   ```
   orapki wallet add -wallet {server_wallet_directory} -trusted_cert -cert {root_
   cert_chain_file} -pwd {server_wallet_password}
   ```

   Example:

   ```
   orapki wallet add -wallet C:/wallets/server -trusted_cert -cert
   C:/wallets/root_chain.txt -pwd serverpass1
   ```

8. Import the server certificate into the wallet.

   ```
   orapki wallet add -wallet {server_wallet_directory} -user_cert -cert {server_
   certificate_file} -pwd {server_wallet_password}
   ```

   Example:

   ```
   orapki wallet add -wallet C:/wallets/server -user_cert -cert C:/wallets/server_
   cert.txt -pwd serverpass1
   ```

Setting Up a Fusion Client Trust Store

To set up a trust store, complete the following steps:

1. Create an Oracle wallet with auto login.

   ```
   orapki wallet create -wallet {client_wallet_directory} -auto_login -pwd
   {client_wallet_password}
   ```

   Example:

```
orapki wallet create -wallet C:/wallets/client -auto_login -pwd clientpass1
```

2. Generate a private key.

```
orapki wallet add -wallet {client_wallet_directory} -keysize {key_len} -dn
{client_dn} -pwd {client_wallet_password}
```

   Example:

```
orapki wallet add -wallet C:/wallets/client -keysize 1024 -dn
"cn=fusionclient,dc=us,dc=example,dc=com" -pwd clientpass1
```

3. Export a certificate request.

```
orapki wallet export -wallet {client_wallet_directory} -dn {client_dn} -request
{client_cert_req_file} -pwd {client_wallet_password}
```

   Example:

```
orapki wallet export -wallet C:/wallets/client -dn
"cn=fusionclient,dc=us,dc=example,dc=com" -request C:/wallets/client_cert_
req.txt -pwd clientpass1
```

4. Create the client certificate in one of the two following ways:

   If a third-party CA is used, send the client certificate request file ({client_cert_req_file}) to the CA. The CA will send back a certificate for the client (referred to as {client_certificate_file}) along with the public certificate of the CA (referred to as {root_cert_chain_file}).

   If the self-signed root certificate is used (use the same one that was used for signing the server certificate), run the following command:

```
orapki cert create -wallet {root_wallet_directory} -request {client_cert_req_
file} -cert {client_certificate_file} -validity {validity_days} -pwd {root_
wallet_password}
```

   Example:

```
orapki cert create -wallet C:/wallets/root -request C:/wallets/client_cert_
req.txt -cert C:/wallets/client_cert.txt -validity 3650 -pwd clientpass1
```

5. Import the CA certificate from the server into the wallet. Note that the public certificate chain may contain more than one certificate. In such cases, the certificates must be imported one by one, starting at the top of the chain.

   Typically the CA certificates of both client and server will be one and the same. There is no reason for them to be different at the same customer site.

```
orapki wallet add -wallet {client_wallet_directory} -trusted_cert -cert {root_
cert_chain_file} -pwd {client_wallet_password}
```

   Example:

```
orapki wallet add -wallet C:/wallets/client -trusted_cert -cert
C:/wallets/root_chain.txt -pwd clientpass1
```

6. Convert the Oracle wallet to JKS format:

```
orapki wallet pkcs12_to_jks -wallet {client_wallet_directory} -pwd {client_
wallet_password} -jksKeyStoreLoc {key_store_file} -jksKeyStorepwd {key_store_
password} -jksTrustStoreLoc {trust_store_file} -jksTrustStorepwd {trust_store_
password}
```

Example:

```
orapki wallet pkcs12_to_jks -wallet C:/wallets/client -pwd clientpass1
-jksKeyStoreLoc testkey.jks -jksKeyStorepwd welcome1 -jksTrustStoreLoc
testtrust.jks -jksTrustStorepwd welcome1
```

Importing the Client Certificates

For every client that connects to the RPAS server using two-way SSL, you must import its certificates into the server wallet as trusted certificates. If the client certificates are from a different CA than the certificates that are already in the server wallet, you must first import the new CA certificate.

Use the following command to import the certificates:

```
orapki wallet add -wallet {server_wallet_directory} -trusted_cert -cert {client_
cert_file} -pwd {server_wallet_password}
```

Example:

```
orapki wallet add -wallet C:/wallets/server -user_cert -cert C:/wallets/server_
cert.txt -pwd serverpass1
```

Starting the Domain Daemon

To start the Domain Daemon using two-way SSL, execute the following command, where <walletLocation> is the absolute path to the directory of the Oracle wallet.

```
DomainDaemon -port portNum -ssl 2 -wallet file:<walletLocation> start
```

Configuring the Fusion Client

To configure the Fusion Client for two-way SSL, you must set up a key store and a trust store in the JKS format. Then, import the CA server certificate into the trust store.

Setting Us a Key Store and a Trust Store

To set up a key store and a trust store, complete the following steps:

1. Create an Oracle wallet with auto login.

   ```
   orapki wallet create -wallet {client_wallet_directory} -auto_login -pwd
   {client_wallet_password}
   ```

   Example:

   ```
   orapki wallet create -wallet C:/wallets/client -auto_login -pwd clientpass1
   ```

2. Generate a private key.

   ```
   orapki wallet add -wallet {client_wallet_directory} -keysize {key_len} -dn
   {client_dn} -pwd {client_wallet_password}
   ```

   Example:

   ```
   orapki wallet add -wallet C:/wallets/client -keysize 1024 -dn "cn=rpas_qa_
   client,dc=us,dc=oracle,dc=com" -pwd clientpass1
   ```

3. Export the certificate request.

   ```
   orapki wallet export -wallet {client_wallet_directory} -dn {client_dn} -request
   {client_cert_req_file} -pwd {client_wallet_password}
   ```

Example:

```
orapki wallet export -wallet C:/wallets/client -dn "cn=rpas_qa_
client,dc=us,dc=oracle,dc=com" -request C:/wallets/client_cert_req.txt -pwd
clientpass1
```

4. Create the client certificate for a third-party CA. To do this, send the client certificate request file ({client_cert_req_file}) to the CA. The CA sends back a certificate ({client_certificate_file}) for the client, along with the public certificate ({root_cert_chain_file}) of the CA.

5. Import the CA into the wallet. Note that the public certificate chain may contain more than one certificate. In such cases, the certificates must be imported one by one, starting at the top of the chain.

   ```
   orapki wallet add -wallet {client_wallet_directory} -trusted_cert -cert {root_
   cert_chain_file} -pwd {client_wallet_password}
   ```

   Example:

   ```
   orapki wallet add -wallet C:/wallets/client -trusted_cert -cert
   C:/wallets/root_chain.txt -pwd clientpass1
   ```

6. Import the client certificate into the client wallet.

   ```
   orapki wallet add -wallet {client_wallet_directory} -user_cert -cert {client_
   certificate_file} -pwd {client_wallet_password}
   ```

   Example:

   ```
   orapki wallet add -wallet C:/wallets/client -user_cert -cert C:/wallets/client_
   cert.txt -pwd clientpass1
   ```

7. Import the CA for the server certificate into the wallet. You can skip this step if both the client and the server certificates are issues by the same CA.

   ```
   orapki wallet add -wallet {client_wallet_directory} -trusted_cert -cert {root_
   cert_chain_file_for_the_server} -pwd {client_wallet_password}
   ```

   Example:

   ```
   orapki wallet add -wallet C:/wallets/client -trusted_cert -cert
   C:/wallets/root_chain.txt -pwd clientpass1
   ```

8. Convert the Oracle wallet to the JKS format.

   ```
   orapki wallet pkcs12_to_jks -wallet {client_wallet_directory} -pwd {client_
   wallet_password} -jksKeyStoreLoc {key_store_file} -jksKeyStorepwd {key_store_
   password} -jksTrustStoreLoc {trust_store_file} -jksTrustStorepwd {trust_store_
   password}
   ```

   Example:

   ```
   orapki wallet pkcs12_to_jks -wallet C:/wallets/client -pwd clientpass1
   -jksKeyStoreLoc testkey.jks -jksKeyStorepwd welcome1 -jksTrustStoreLoc
   testtrust.jks -jksTrustStorepwd welcome1
   ```

9. Copy the client key and trust files into <fusion_client_install_dir>/config/idstores. The password must be kept inside the WebLogic credential store, which will be handled by the Fusion Client installer.

Importing the Client Certificates

For every client that connects to the RPAS server using two-way SSL, you must import its certificates into the server wallet as trusted certificates. If the client certificates are from a different CA than the certificates that are already in the server wallet, you must first import the new CA certificate.

Use the following command to import the certificates:

```
orapki wallet add -wallet {server_wallet_directory} -trusted_cert -cert {client_
cert_file} -pwd {server_wallet_password}
```

Two-Way SSL with Authentication Only

It is possible to run two-way SSL without encryption. To accomplish this, set up two-way SSL as described in the previous section for both the RPAS Server and the Fusion Client. Then, to start the Domain Daemon with two-way SSL authentication-only, execute the following command, where the <walletLocation> is the absolute path for the directory of the Oracle wallet:

```
DomainDaemon -port portNum -ssl 4 -wallet file:<walletLocation> start
```

You will see the following warning message:

```
Warning: The client/server network communication is using two-way SSL
authentication only. The data communication is not encrypted.
```

Creating SSL Certificates Automatically

The input.userSelfSignedSSLCertificates property can be used to create SSL certificates automatically using the Fusion Client Installer.

Usage

If this property is set to yes, it triggers the automatic creation of SSL certificates signed by a private certificate authority. At the end of the process, the server wallet directory must be copied to its correct destinations. The server wallet directory is the "walletdir/server" subdirectory of the installer's current directory.

Use this option only if deploying to a trusted hardware and software environment that is under the control of a single business entity. If the RPAS client or the server is not in the customer's control, externally signed SSL certificates are recommended. See the Installation Guide for details.

5

Setting Up the RPAS Fusion Client

After you install the RPAS Fusion Client, before letting users access the application, you must configure it based on your business needs. This chapter describes the configuration options available with the RPAS Fusion Client. It also includes information on how you can configure additional domains that are created after installing the RPAS Fusion Client as well as information about logging into the Fusion Client. This chapter includes the following sections:

- Maintaining Layout and Formatting
- Setting Up the Configuration Properties File
- Configuring Additional Domains
- Logging into the Fusion Client

Maintaining Layout and Formatting

The RPAS Fusion Client layout and formatting capabilities are used along with Save Format/All (and sometimes Group) for each template. Mass, consistent formatting is important in this case.

Periodically, the layout and formatting of an installed application will need to be revised, and these capabilities will be used once again.

With RPAS, when the domain's configuration is patched in a way that includes structural changes, or sometimes when the RPAS software is upgraded, the format database in the domain can be invalidated. In these cases, all saved formatting (Template All, Group, and User, plus the workbooks themselves) must be cleared out and recreated. Mass, consistent formatting is especially important in these cases.

Setting Up the Configuration Properties File

The Configuration Properties (config.properties) file enables you to set up the session timeout properties.

To set up the config.properties file:

1. Navigate to the following location where the RPAS Fusion Client is installed:

   ```
   <RPAS_Fusion_Client_Installation>/config/
   ```

2. In a text editor, open the config.properties file and set the following parameters:

   - session.timeout: set a time limit (in minutes) after which the current user session will expire. The default value is 35 minutes. Do not use a value that is less than 5 minutes. A reminder dialog is presented to users at a defined number of minutes prior the expiration. The defined number of minutes is set by the session.remindin parameter in the web.xml file. See "Updating the Session Reminder" for more information about updating the web.xml file.

- sso.logout.url=<logoutURL>: use this URL to log out from both SSO and RPAS.

  If this property is not specified, then the following occurs. The logout link on the Fusion Client page states "end session". Clicking the link only terminates the RPAS session. The user is taken to a page that states "session has ended". A login link is provided. If the user clicks the login link, the user is taken back to the Fusion Client home page.

  For Oracle 11g SSO, set this property to /oam/pages/logout.jsp.

- isSSO: set the value to true to indicate that the application is running in an Single Sign-On environment.

  Note: For information about Single Sign-On, see the *Oracle Retail Predictive Application Server Installation Guide*.

3. Save the file and, for the changes to take effect, restart the WebLogic server.

Latency

In 14.0 the Fusion Client introduces a basic latency measurement capability. This will help you to get an idea of the latency on their network and analyze any slowness of their application. An estimate of the latency is obtained and displayed in the bottom of the home page and the workbook page.

Latency is the time taken for data to move between the browser and the application server, in any direction. We measure latency as the time taken for a small image file to be fetched from the application server into the browser. The request and response are processed in several layers of the network stack on both ends. Hence it should be treated as an approximation of the latency.

The feature can be disabled by setting the property "latency.measurement.enabled" to false. This is the default value. If you are not concerned about latency might want to leave this feature disabled.

The frequency of measurement is specified by a property named "latency.measurement.interval". The value is the number of seconds between measurements. The default value is 30 seconds. Higher values cause less of a performance hit on the ongoing user session, but the latency measurement is less recent.

These two properties are to be set in "config.properties".

Other Properties Files

In addition to the config.properties file, there are several more properties files located in the config folder of the Fusion Client installation. The majority of these files should not be altered. For information about the ones you can alter, see "Appendix: Fusion Client Properties Files".

Files in the Config Folder

- Config.properties: Overall application configuration settings such as timeout and SSO configuration.
- Log4jconfig.xml: Logging settings: detail levels and locations for log files.

  The log file that is configured here contains configuration errors from the taskflow and the foundation. For example, if the taskflow and foundation do not have any solution IDs in common, you will see a "NO CONNECTABLE SOLUTIONS from Taskflow and Foundation" error in this log and nobody will be able to log in. In addition, end users cannot in general see why they are not able to connect to a particular solution. An administrator may want to inspect these logs to diagnose such problems.
- Jps-config.xml: Internal configuration file used for application security.

Files in the Config/rpas Folder

- FormatConfig.xml: Configuration information used to determine what styles or formats to use for specific types.
- FormatMapping.xml: Internal file used to read FormatConfig.xml.
- GraphConfig.xml: Configuration information used to determine default styles for graphing and graph types. For more information, see "GraphConfig.xml File".
- GraphMapping.xml: Internal file used to read GraphConfig.xml.
- LanguageMap.properties: Used to map Browser locales to RPAS Supported languages.
- PivotTableStyles.properties: Used to set some default styles for pivot table cell text and backgrounds. For more information, see "PivotTableStyles.properties File".
- Foundation.xml: Specifies connection and domain information for RPAS domains. For more information, see "Configuring Additional Domains".
- rpasConfig.properties: Configuration properties for RPAS applications deployed on the Fusion Client. For more information, see "rpasConfig.properties File".

Updating the Session Reminder

Before a Fusion Client session times out, a reminder message appears, informing the user that the session will expire. The number of seconds before expiration can be configured in the web.xml file.

By default, sessions will expire after 35 minutes of inactivity. The default reminder message appears two minutes prior to session expiration. This means that after 33 minutes of inactivity, the default reminder appears. After two addition minutes of inactivity, the session expires.

To change the amount of time before session expiration when the reminder message appears, perform the following steps:

1. Locate the Fusion Client ear file called `rpas.ear`.
2. Make a temporary folder called tmp. In that folder, extract the `rpas.war` file from the `rpas.ear` file.

```
mkdir tmp
cd tmp
jar xvf ../rpas.ear rpas.war
```

3. Extract the `web.xml` file from the `rpas.war` file:

   ```
   jar xvf rpas.war WEB-INF/web.xml
   ```

4. Edit the `web.xml` file as follows:

   a. From the beginning of the file, search for the string WARNING_BEFORE_TIMEOUT.

   b. Update the param-value property that is directly below the WARNING_BEFORE_TIMEOUT line by entering the number of seconds you want the reminder message to appear before session expiration.

      For example, if you want the reminder to appear 180 seconds before session expiration, enter the following:

      ```
      <context-param>
           <description>Number of seconds before timeout when warning must be
            displayed</description>
           <param-name>oracle.adf.view.rich.sessionHandling.WARNING_BEFORE_
            TIMEOUT</param-name>
           <param-value>180</param-value>
      </context-param>
      ```

   c. Save the web.xml file.

5. Update rpas.war while in tmp folder:

   ```
   jar uvf rpas.war WEB-INF/web.xml
   ```

6. Update rpas.ear while in the tmp folder. Delete the tmp folder after you are finished.

   ```
   jar uvf ../rpas.ear rpas.war
   cd ..
   rm -rf tmp
   ```

Configuring Announcements

Announcements can be created and broadcast to Fusion Client users. These messages appear on the Login screen, a user’s home page, and the workbook.

The properties file is called announcements_<locale-abbreviation>.properties and is located in <configroot>/announcements. The text of a message can be customized for different locales and countries. For example, To display a message in US English (the default), in German for all locales where that language is spoken, and Brazilian Portuguese:

```
announcements.properties
announcements_de.properties
announcements_pt_BR.properties
```

The properties file contains three types of announcement messages:

```
loginpage.message=<Message to be shown on login page>
homepage.message=<Message to be shown on home page>
workbookpage.message=<Message to be shown on workbook page>
```

For example:

```
loginpage.message=System will be shutting down in ten minutes.
```

By default, the latest message appears when a use logs in; however, this is configurable. A new message overwrites an existing one. Set announcements.update.delivery located in config.properties to configure this:

Table 5–1 Settings for announcements.update.delivery

Setting	Description
onaction	Enables dynamic loading. The latest announcement appears whenever a user performs an action in the UI that requires an interaction with the server.
never	Disables dynamic loading. Announcement messages are only loaded into a user's session when the user logs in. No other messages will appear until after a user logs out and then logs in again.

Messages can be formatted using HTML tags such as <ul> and <li> for bulleted lists, <br> for line breaks, <p> for paragraphs, as well as <a> for hyperlinks.

Configuring Additional Domains

During the RPAS Fusion Client installation, the installer automatically sets up the RPAS domain connection configuration in the Foundation.xml file. If you choose to set up the domain connection after the installation or set up an additional domain, you must manually set up the connection.

> **Note:** Beginning with the 13.3.1 release, ProfileList.xml is no longer used by the Fusion Client. Foundation.xml is used to maintain the required domain-connection information.

To set up a domain connection manually, perform the following steps:

1. After you have the additional domain set up, navigate to the following location in the RPAS Fusion Client installation directory:

   ```
   <RPAS_Fusion_Client_Installation>/config/rpas/
   ```

2. In a text editor, edit the Foundation.xml file, and append connection details for your solution using the following syntax:

   ```
   <solution>
      <name>{Solution Id}</name>
      <description>{Solution Description}</description>
      <server-name>{RPASServerName}</server-name>
      <daemon-port>{Port Number}</daemon-port>
      <daemon-name>{Domain Name}</daemon-name>
      <domain-path>{location where simple or global domain is
      installed}</domain-path>
   </solution>
   ```

   > **Note:** In the example above, placeholders have been used as values for the parameters. Ensure that you replace the placeholders with the relevant values. The Solution ID should be the same as the one specified in Taskflow_MultiSolution.xml.

   Where,

- Solution ID: The name of the solution that is used in the MultiSolution taskflow file. The ID must match the values that have been set up in that file.
- Solution Description: The description for the solution that is displayed in the Fusion Client.
- RPAS Server Name: The host where the RPAS Server is installed and the Domain Daemon process is running.
- Port Number: The port to specify when starting the Domain Daemon.
- Domain Name: The name used for the master domain or the simple domain.
- Domain Path: The location where the domain is installed. It must be either a simple domain or a global domain environment.

> **Note:** The connection details for every solution ID in the MultiSolution taskflow can appear in the same Foundation.xml file.

3. Save and close the Foundation.xml file.
4. In the `<RPAS_Fusion_Client_Installation>/config` directory, create a new subdirectory and specify `MultiSolution` as the name. In addition, create a resources subdirectory within the MultiSolution directory.
5. Navigate to the `fusionClient` directory located within the directory where the domain is installed. This subdirectory contains the following files that define the Taskflow configuration:
   - taskflowBundle.properties
   - taskflow.xml

   These files exist in the domain's `fusionClient` directory after the domain is created with the rpasInstall script. These files can also be created directly from the report generation tool in the RPAS Configuration Tools. For more information, see the *RPAS Configuration Tool User Guide*.

   > **Note:** If using characters outside the "Latin 1" ascii range, you must run the native2ascii command after editing the taskflowBundle.properties file. This ensures that the characters in the taskflow are displayed properly in the Fusion Client. The native2ascii command can be found in any Java JDK $JAVA_HOME/bin folder.

6. Copy the two files to the `MultiSolution` directory you created in step 4.
7. Obtain the taskflow file from the domain.
8. If the Fusion Client installation supports only this single solution, then copy the file to the MultiSolution directory and rename the file to Taskflow_MultiSolution.xml.

   > **Note:** Ensure that the Solution ID in this file matches the Solution ID specified in Foundation.xml.

9. If the Fusion Client installation supports multiple solutions, you must combine and arrange them. For more information about taskflow formats and combining

taskflows, see the *Oracle Retail Predictive Application Server Configuration Tools User Guide*.

10. If the Fusion Client installation supports only this single solution, then copy the file to the MultiSolution/resources directory and rename the file to MultiSolutionBundle.properties.

> **Note:** Ensure that the Solution ID in this file matches the Solution ID specified in Foundation.xml.

11. If the Fusion Client installation supports multiple solutions, you must combine and arrange them. For more information about taskflow formats and combining taskflows, see the *Oracle Retail Predictive Application Server Configuration Tools User Guide*.
12. For the changes to take effect, restart the RPAS and WebLogic servers.

Sample Foundation.xml

The following example illustrates a sample configuration of Foundation.xml with two solutions.

```
<?xml version="1.0" encoding="UTF-8" ?>
<solutions>
    <solution>
        <name>ap</name>
        <description>Assortment Planning</description>
        <server-name>RPAS_host</server-name>
        <daemon-port>12345</daemon-port>
        <domain-path>/folder/APGlobalDomain</domain-path>
    </solution>
    <solution>
        <name>ip</name>
        <description>Item Planning</description>
        <server-name>RPAS_host</server-name>
        <daemon-port>12345</daemon-port>
        <domain-path>/folder/IPGlobalDomain</domain-path>
    </solution>
</solutions>
```

Logging into the Fusion Client

Before you log into the Fusion Client, ensure that your system meets the recommended configuration. For more information, see the *RPAS Installation Guide.*

Once you check the configuration, obtain the following information:

- Uniform Resource Locator, URL: You must enter the URL or the web address of the application in the web browser to access the application. For example, http://<fullyqualifieddomainname>:<port>/<applicationcontextroot>
- User name and password.

> **Note:** Although you must provide a user name and password, you are authenticated externally either by SSO or LDAP.

To log into the Fusion Client:

1. Start a supported web browser.
2. In the Address bar, enter the Fusion Client URL, and press **Enter**. The Fusion Client Login page appears.
3. On the Login page, enter the username and password.
4. Click **Login**.

6

User Maintenance

User administration is the process by which administrators add or delete authorized system users, create 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 chapter:

- Adding a User
- Adding a User Group
- Deleting a User
- Deleting a User Group
- Editing a User

These procedures can be performed through the RPAS Fusion Client by accessing the User Administration task in the Administration activity in the taskflow (Figure 6–1).

In addition, the usermgr utility is also described in this chapter. This utility allows you to manage users and users groups through a command line interface. For more information, see "Managing Users Using usermgr".

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 Admin status associated with individual users.

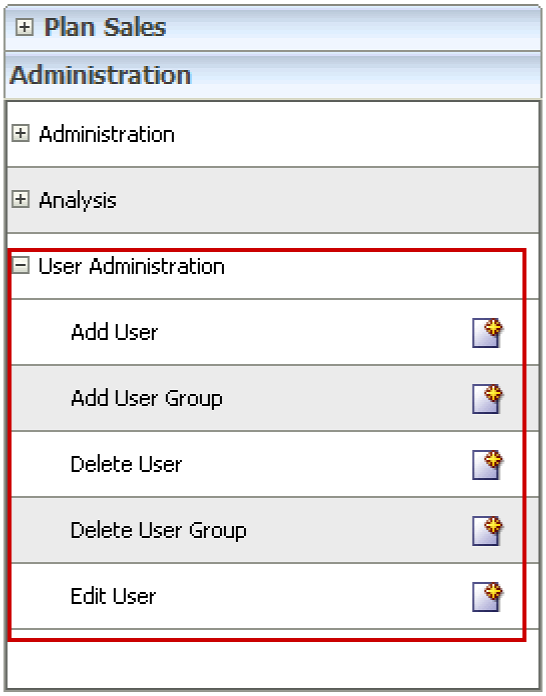
Note: Users must be set up in the directory for the external authorization provider as well. The users must also exist here to define access to and within each RPAS solution.

User Administration Activity

To access the User Administration workbooks:

1. In the taskflow, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, select **User Administration** for the solution you want to work on. The steps within the User Administration task appear.

Figure 6–1 User Administration



Adding a User

To add a user:

1. Under User Administration, click the **Create New Workbook** icon next to **Add User**. A Workbook Wizard window appears.
2. In the **Workbook Wizard** window, enter relevant information in the following fields:

   - **User name**: Type the user name that the user uses for logging on.

     Note: Each user name must begin with a letter and contain only alphanumeric characters and underscores. It cannot have spaces. User names are case sensitive.

   - **User label**: 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 some views to specify which user performed a task.
   - **Default group**: Select the user group to which the user belongs.
   - **Other groups**: If a user belongs to more than one group, select the additional groups from the list in the **Other groups** field.

     Note: Passwords are not used by the Fusion Client. The password is defined and managed by your external authentication provider.

     ---

3. If the user should have Admin status, check the **Administrator** box.

Note: Administrator privilege enables users to access the Save Format option on the View menu, which creates new system-wide default styles for workbook templates.

Administrative users have special privileges and the read-only status may not apply to them.

If you are not sure whether a user should be granted this ability, you can modify the Admin status later in the Users view (in 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.

---

4. Select the **Lock user account** check box to temporarily disable the user's account.
5. 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 this, access the Security Administration workbook. For more information, see "Accessing Security Administration".

Adding 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.

To add a user group:

1. Under User Administration, click the **Create New Workbook** icon next to **Add User Group**. A Workbook Wizard window appears.
2. In the **Workbook Wizard** window, enter relevant information in the following fields:
   - In the **Group Name** field, type a name for the group.

     Note: Each group name must begin with a letter and contain only alphanumeric characters and underscores. It cannot have spaces. User group names are case sensitive.

   - In the **Group Label** field, type a descriptive label for the group. This label is displayed when referring to the group throughout RPAS.
3. Click **Finish** to add the user group to the database.

Deleting a User

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

To delete a user:

1. Under User Administration, click the **Create New Workbook** icon next to **Delete User**. A **Workbook Wizard** window appears.
2. The **Workbook Wizard** window appears and displays the user names and labels for all users. Select the name of the user to delete.
3. Click **Finish** to delete the user from the system.

Deleting 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: Before you can delete a user group, all users must be removed from the group. For each user in the group, you must either delete the user or change the default user group assignment for the user.

To delete a user group:

1. Under User Administration, click the **Create New Workbook** icon next to **Delete User Group**. A **Workbook Wizard** window appears.
2. In the **Workbook Wizard** window, select the user group you want to delete.
3. Click **Finish** to delete the user group from the system.

Editing a User

To edit a user's profile:

1. Under User Administration, click the **Create New Workbook** icon next to **Edit User**. A **Workbook Wizard** window appears.
2. The **Workbook Wizard** window appears, displaying the user names and labels for all users. Select the user you want to edit and click **Next**.
3. Make the necessary changes to the user's profile. You can change anything except the User name. For more information on the fields, see "Adding a User".
4. Click **Finish** to save the changes.

Managing Users Using usermgr

Use the usermgr utility to add and remove users and groups, copy user and group information to other domains, edit user and group information, and convert that information from XML files to database files and vice versa.

With the usermgr utility, you can create both user and administrator accounts using the -addGroup command. This allows you to add many accounts at once.

To create users and groups, you must create a database file called users.db, which contains user and group information. Once you have created and imported the users.db file into RPAS, it contains all user, administrator, and group information. This file is an encrypted binary file that you cannot edit.

To edit any account information, you must convert the database file to an XML file using the -convertDbToXml command. This creates an editable XML file that contains all the information in the user.db file. Again, this is to ensure the security and safety of the account information. Once you edit the XML file with the changes you need to make, you must convert it back to a database file in order to import it into RPAS. To convert it, use the -convertXmlToDb command.

Once the users.db file is created, it can be shared across multiple domains. To automatically import the user, administrator, and group information every time a domain is created, place the users.db file in the rpasInstall input directory of the domain.

usermgr Usage

```
usermgr -d domainPath -add [userName] -label [label] -group [groupName] {-admin}
usermgr -d domainPath -addGroup [groupName] -label [label]
usermgr -d domainPath -remove [userName]
usermgr -d domainPath -removeLabel [label]
usermgr -d domainPath -removeGroup [groupName]
usermgr -d domainPath -list
usermgr -d domainPath -listHashAlgs
usermgr -d domainPath -showHashSettings
usermgr -d domainPath -hashAlg [algorithmid]
usermgr -d domainPath -hashIter [iterations]
usermgr -d domainPath -forcePasswordChange {true|false} [-user username]
usermgr -d domainPath -changePassword [username]
usermgr -d domainPath -lock [userName]
usermgr -d domainPath -unlock [userName]
usermgr -d domainPath -print -user [userName]
usermgr -d domainPath -print -group [groupName]
usermgr -d domainPath -importXml [path]
usermgr -d domainPath -exportXml [path]
usermgr -d domainPath -importDb {-replace}
usermgr -d domainPath -exportDb [path]
usermgr -convertXmlToDb -src [pathToXml] -dest [pathToDb]
usermgr -convertDbToXml -src [pathToDb] -dest [pathToXml]
```

Note: -convertDbToXml, -convertXmlToDb, and -listHashAlgs do not require a domain.

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

Table 6–1 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*	Adds a user with a specified name. Use the other arguments specified in the usage to add those attributes for that user. If -noPassword is specified, then this user can only be logged in via the Fusion Client. Otherwise, this command will prompt for a temporary password for use with the Classic Client only
-label *label*	Specifies the label of the user or group to add to the domain. Use this argument with -add and -addGroup.
-group *grp*	Specifies the user group of the user to add to the domain.
-admin	Specifies that the user to add to the domain has administrative rights.

Table 6–1 (Cont.) Arguments Used by the usermgr Utility

Argument	**Description**
-addGroup *groupName*	Adds a group with a specified name. Use -label to specify the label for the group.
-remove *userName*	Removes the user with the specified name from the domain.
-removeLabel *label*	Removes all users with this label.
-removeGroup groupName	Removes a group with this groupName.
-list	Lists all the users registered to the specified domain.
-listHashAlgs	Lists the different hashing algorithms supported in this release. The list will show an integer id, the details of the algorithm, and a status field indicating whether the algorithm has been deprecated 'X' or is the default '*'. This option is only used with the Classic Client.
-showHashSettings	Shows the selected hashing algorithm and the selectednumber of hash iterations. This option is only used with the Classic Client.
-hashAlg	The given algorithm id parameter defines the algorithm to use for new users or users whom update their password. Use the string 'default' to always chose the default algorithm. This option is only used with the Classic Client.
-hashIter	The given iterations parameter is used to set the number of iterations the hash algorithm will used for new users or users whom update their password. This option is only used with the Classic Client.
-forcePasswordChange	The given boolean parameter either 'true' or 'false' determine the setting of the password must change variable. If 'true' then the next login will request the password to be updated. If the optional -user parameter is provided the effect of this value will only affect that user. This option is only used with the Classic Client.
-changePassword	For the given user application will prompt for the new password. This option is only used with the Classic Client.
-print	Use this argument to print the specified user or group information.
-user *username*	Specifies the user name in the specified domain to print. This argument is only applicable to the -print option.
-group *groupname*	Specifies the group in the specified domain name to print. This argument is only applicable to the -print option.
-importDb	Imports the database. The database must be located in the domain's input directory. The database is time stamped and moved to the processed directory upon successful completion. Existing user are skipped unless -replace is used.
-replace	Updates existing users when you are using -importDb. The user label, the groups that user belongs to, the admin status, and the account lock status are updated. Password information is not affected by the update.

Table 6–1 (Cont.) Arguments Used by the usermgr Utility

Argument	Description
`-lock` *userName*	Locks the specified user. This prevents the user from logging into the domain.
`-unlock` *userName*	Unlocks the specified user. This allows the user to log into the domain.
-importXml path	Add or update users from 'users.xml' placed in the input directory of the domain. By default, existing users are ignored. However, if the -replace option is selected, their information (excluding password) are replaced. If -noPassword is used, then any new users added can only log in via the Fusion Client. Otherwise, this command will prompt for temporary passwords for admin and non-admin users.
-exportXml *path*	Creates an XML file that contains all users and groups in the selected domain. Passwords and password histories are not exported.
-importDb path	Add or update users from 'users.db' placed in the input directory of the domain. By default, existing users are ignored. However, if the -replace option is selected, their information (excluding password) are replaced.
-exportDb *path*	Creates a database that contains all users and groups in the selected domain. This prompts you for new temporary passwords for admin and user accounts.
-convertDbToXml	Converts a user database to a user XML file. Passwords are not included in the conversion.
-convertXmlToDb	Converts a user XML file to a user database. This prompts you for temporary passwords for admin and user accounts.
-src *path*	Specifies the source file that is used in the arguments -convertDbToXml and -convertXmlToDb.
-dest *path*	Specifies the destination file that is used in the arguments -convertDbToXml and -convertXmlToDb.

Note: The arguments userName and groupName must conform to the standards for position names in RPAS. They may contain only the characters a-z, A-z, 0-9, _, &, $ or %. User and group names cannot begin with an underscore. Any supplied upper case letters will be converted to lower case by the application. User and group names cannot be an empty string.

XML Schema

The XML schema contains information for all groups and users that are imported into the domain. Since this file can be edited, it does not contain any password information. Each group and user contains an XML attribute with the group or user name as well as the following inner tags:

Table 6–2 XML Schema

Outer Tag	Inner Tag	Description
GROUP	LABEL	The group's label.
USER	LABEL	The user's label.

Table 6–2 (Cont.) XML Schema

Outer Tag	Inner Tag	Description
USER	DFLT_GRP	The user's default group.
USER	OTHER_GRPS	A comma-separated list that contains all other groups that the user is associated with.
USER	ADMIN	If this value contains **T**, this user is an admin.
USER	LOCKED	If this value contains **T**, the user is locked when the file is imported.

Note: All XML tags must be in all caps.

Additionally, the schema for users.xml has been updated. The 14.0 version of usermgr will accept either schema, but only outputs the new version.

Here is a sample users.xml file.

```
<?xml version=”1.0” ?>
<VERSION>1.0</VERSION>
<USER_LIST version="2.0">
<GROUPS>
     <GROUP NAME=”grp1”>
          <LABEL>Administrators</LABEL>
     </GROUP>
     <GROUP NAME=”grp1”>
          <LABEL>Group 1</LABEL>
     <?GROUP>
     <GROUP NAME=”grp2”>
          <LABEL>Group 2</LABEL>
     </GROUP>
     <GROUP NAME=”grp3”>
          <LABEL>Group 3</LABEL>
     </GROUP>
</GROUPS>
<USERS>
     <USER NAME=”adm1”>
          <LABEL>admin 1</LABEL>
          <DFLT_GRP>adms</DFLT_GRP>
          <ADMIN>T</ADMIN>
     </USER>
     <USER NAME>=”adm2”>
          <LABEL>admin 2</LABEL>
          <DFLT_GRP>adms</DFLT_GRP>
          <ADMIN>T</ADMIN>
          <LOCKED>T</LOCKED>
     </USER>
     <USER NAME=”usr1”>
          <LABEL>user_1</LABEL>
          <DFLT_GRP>grp1</DFLT_GRP>
          <OTHER_GRPS>grp2</OTHER_GRPS>
     </USER>
     <USER NAME=”usr2”>
```

```
          <LABEL>user_2</LABEL>
          <DFLT_GRP>grp2</DFLT_GRP>
          <OTHER_GRPS>grp1,grp3</OTHER_GRPS>
     </USER>
     <USER NAME=”usr3”>
          <LABEL>user_3</LABEL>
          <DFLT_GRP>grp1</DFLT_GRP>
          <OTHER_GRPS>grp1</OTHER_GRPS>
     </USER>
</USERS>
</USER_LIST>
```

Use Cases

Here is a list of common use cases for the XML and database files.

Exporting from an Existing Domain

You can export from an existing domain using -exportDb or -exportXml. Exporting is useful for sharing users with another domain or for creating backups.

Use -exportXml if you need to edit the users or groups. This can be used when you are making bulk updates that apply to many users or groups. Once you have made changes to the file, you must convert it back to a database by using the -convertXmlToDb and then import the updated file using -importDb -replace.

Use -exportDb if you do not need to edit the users or groups. This method produces a binary file that is ready for import.

Importing into a Domain

You can import existing users.db by using -importDb. Importing is useful for bulk insertion or updates of users and groups.

By default, existing users and groups are skipped. However, if -replace is used, existing users are updated with the following information: user label, groups that the users belong to, admin status, and account lock status. Group labels are updated for existing groups.

Password information for existing users is not affected by the update. New users receive temporary passwords that are stored in users.db.

Note that new users may not be created if the following conditions exist:

- A group already exists with the user name.
- The user's default group does not exist.

New groups are skipped if the following conditions exist:

- A user already exists with the group name.

Note: If a users.db is placed inside the rpasInstall input folder, the users within that file are automatically imported upon domain creation.

Converting Between XML and Database

Since only users.xml can be edited and only users.db can be imported, it is necessary to convert between the two formats when you need to edit and import users and groups. For example, if you are creating users for the first time, you must first create an XML file for those users. Then you must convert that XML file to a database file using -convertXmlToDb in order import the file. If you have an existing users.db that you need to edit, convert it to xml using -convertDbToXml.

When converting from an XML to a database file, the following validation rules apply:

- The file must be a properly formatted XML file; otherwise, the operation fails.
- Groups and users without a NAME attribute are skipped.
- Duplicate groups and users are skipped. Duplicate groups and users are those that share the same NAME attribute as an existing group or user within the same XML file. For instance, a new group called Sales cannot be created if a user or group named Sales already exists.
- Users without a DFLT_GRP field are skipped.

Note: If a user or group is skipped, a warning is logged. Since warnings are not included in the default log level, you should run this utility with warnings visible.

7

Single Sign-On

Oracle Single Sign-On (SSO) provides users with the ability to sign on to multiple Web-based applications using a single user ID and password.

This chapter includes the following sections:

- SSO Authentication
- Manually Configuring SSO Within the Fusion Client

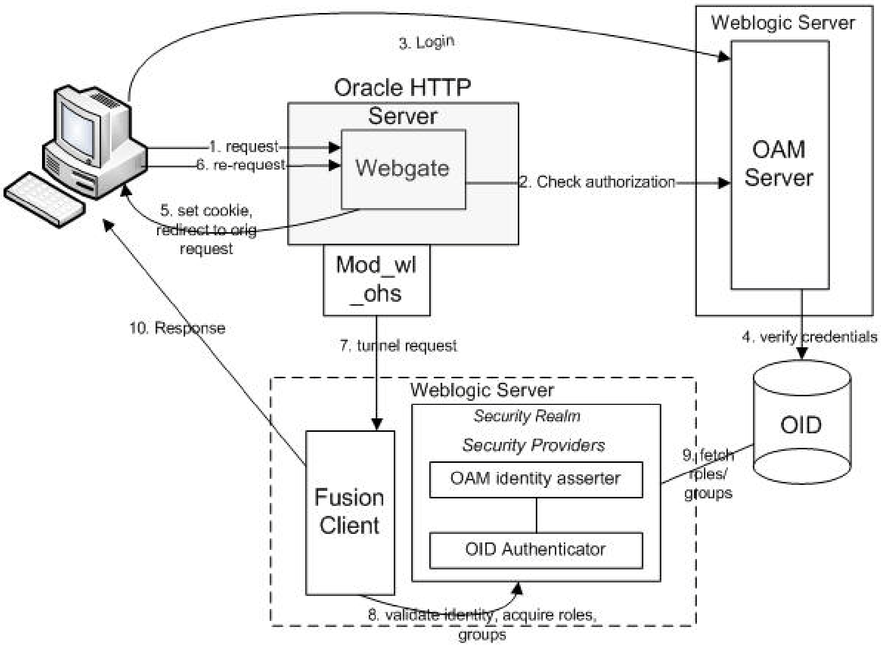
For more information on SSO, see the *Oracle Retail Predictive Application Server Installation Guide.*

SSO Authentication

The RPAS Fusion Client requires perimeter authentication for SSO deployments. User requests are authenticated at a front-end web server that is configured as a reverse proxy server. The authentication function is usually performed by a plug-in component known as a "gateway".The requests are then passed to the Fusion Client. The first time a user request is received, a vendor-specific Identity Assertion component, which is plugged into the WebLogic server, is used to validate the identity of that user. In addition, a custom authentication provider must be configured and plugged into the WebLogic server. The authentication provider retrieves the user’s roles and groups from the SSO environment’s identity store (for example, an LDAP server such as Oracle Internet Directory).

Figure 7–1 illustrates the SSO topology:

Figure 7–1 SSO Topology


3. Login
Weblogic Server
Oracle HTTP Server
1. request
6. re-request
Webgate
2. Check authorization
OAM Server
5. set cookie, redirect to orig request
Mod_wl_ohs
10. Response
7. tunnel request
4. verify credentials
Weblogic Server
Security Realm
Security Providers
OAM identity asserter
OID
9. fetch roles/ groups
Fusion Client
OID Authenticator
8. validate identity, acquire roles, groups

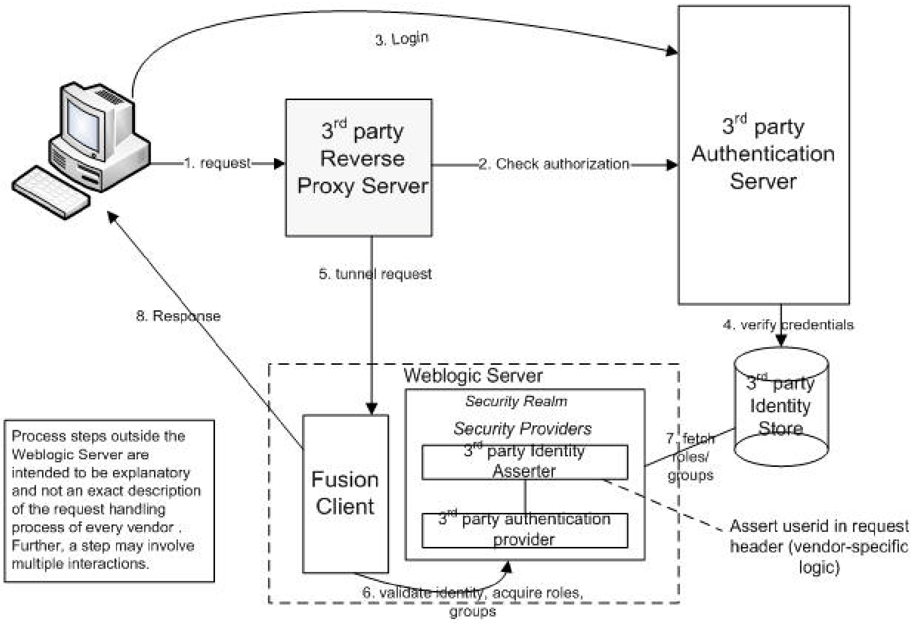
The Fusion Client is certified to work with Oracle 11g SSO; however, it can also work with any vendor-supplied SSO infrastructure that provides the same functionality as described above.

To set up SSO authentication, you must do the following:

- Configure a reverse proxy web server between the user and the WebLogic server to be able to:
  - intercept requests
  - perform authentication and create an identity asserter in the request
  - tunnel requests to the WebLogic server
- Configure a matching Identity Assertion component on the WebLogic server to read and validate the Identity Assertion token
- Configure a matching authentication provider on the WebLogic server to fetch roles and groups from your identity store.

Figure 7–2 illustrates a sample SSO deployment:

Figure 7–2 SSO Sample Deployment


3. Login
1. request
3rd party Reverse Proxy Server
2. Check authorization
3rd party Authentication Server
5. tunnel request
8. Response
4. verify credentials
3rd party Identity Store
Weblogic Server
Security Realm
Security Providers
3rd party Identity Asserter
3rd party authentication provider
Fusion Client
7. fetch roles/ groups
Assert userid in request header (vendor-specific logic)
6. validate identity, acquire roles, groups
Process steps outside the Weblogic Server are intended to be explanatory and not an exact description of the request handling process of every vendor . Further, a step may involve multiple interactions.

The following sections provides general instructions for setting up the Fusion Client to work with SSO authentication.

Identity Assertion

The Identity Assertion component is responsible for extracting the vendor-specific user identity token from the user request, validating the token, and extracting the user identity from the token. This identity token is placed in the request by the front-end web server after the user is authenticated against the SSO identity management system. Note that the Identity Assertion component must be aligned with the SSO token generation process.

WebLogic provides vendor-specific Identity Asserters. To see the list of available ones, in the WebLogic Admin console, go to Security Realms > Default Realm > Providers and click New. You see the Type drop-down list, from which you can select the appropriate vendor-specific Identity Asserter.

To add an Identity Asserter, you must implement the Identity Asserter API and place the jar file in the appropriate location. For more information, see http://docs.oracle.com/cd/E21764_01/web.1111/e13718/ia.htm#i1115766.

For more information about any vendor-specific Identity Asserter, consult the appropriate vendor documentation.

Fetching Roles and Groups

You must install and configure the appropriate vendor-specific authentication provider that is supported by WebLogic. If your SSO vendor is not supported, you can find instructions on how to create a custom authentication provider here: http://docs.oracle.com/cd/E21764_01/web.1111/e13718/atn.htm#i1188880.

Security Provider Order

You must observe the following order when setting up the Identity Asserters and authentication providers on WebLogic:

1. Third-party Identity Asserter. Control flag setting = OPTIONAL.
2. Third-party authentication provider. Control flag setting = OPTIONAL.
3. Default authentication provider (already exists). control flag setting = SUFFICIENT.

Note that if you set the control flag to REQUIRED for either the third-party Identity Asserter or the third-party authentication provider, then WebLogic users will not be able to access the WebLogic Admin console.

Manually Configuring SSO Within the Fusion Client

Once you have set up the SSO environment, you must configure the Fusion Client application to use the SSO environment. Oracle Single Sign-On (SSO) is configured for the RPAS Fusion Client via the RPAS installer. However, you can also manually configure SSO for the Fusion Client.

Note: Manual configuration is only recommended for advanced users.

Note: To avoid authentication issues with SSO users, select the Use Retrieved User Name as Principal check box and then restart the WebLogic server. This option is available in the OID provider on the Provider Specific tab in the Users section.

For information about using the RPAS installer to configure SSO, see the *Oracle Retail Predictive Application Server Installation Guide*.

To convert a non-SSO deployment of the RPAS Fusion Client to use SSO, perform the following steps:

1. Locate the Fusion Client ear file called rpas.ear.
2. Make a temporary folder called tmp. In that folder, extract the rpas.war file from the rpas.ear file.

   ```
   mkdir tmp
   cd tmp
   jar xvf ../rpas.ear rpas.war
   ```

3. Extract the web.xml and weblogic.xml files from the rpas.war file:

   ```
   jar xvf rpas.war WEB-INF/web.xml
   jar xvf rpas.war WEB-INF/weblogic.xml
   ```

4. Edit web.xml as follows:
   a. From the beginning of the file, search for the string SSO. The first match is a `security-constraint` XML element. Uncomment this element.

      To uncomment the element, modify the element from this:

      ```
      <!--element start>…</element end-->
      ```

To this:

```
<element start>...</element end>
```

   b. Search for the next occurrence of `SSO`. This match is the `login--config` XML element. Uncomment this element.

   c. Save the web.xml file.

5. Edit weblogic.xml as follows:

   a. From the beginning of the file, search for `SSO`. This locates the `security-role-assignment` XML element. Uncomment this element.

   b. Save the weblogic.xml file.

6. Update rpas.war while in tmp folder:

```
jar uvf rpas.war WEB-INF/web.xml
jar uvf rpas.war WEB-INF/weblogic.xml
```

7. Update rpas.ear while in tmp folder. Delete the tmp folder after you are finished.

```
jar uvf ../rpas.ear rpas.war
cd ..
rm -rf tmp
```

8. Update config.properties in the Fusion Client configroot folder.

   a. Set isSSO=true.

   b. Save.

8

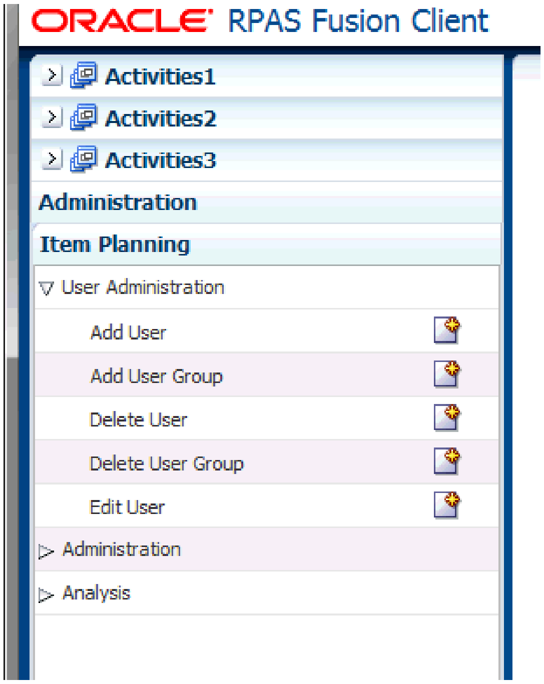
System Administration

This chapter describes the following system administration workbooks and wizards:

- Auto Workbook Maintenance Wizard
- Workbook Batch Category Management
- Hierarchy Maintenance Workbook
- Security Administration Workbook
- Measure Analysis Workbook

These workbooks and wizards are found in the Administration task in the taskflow.

Figure 8–1 Administration Workbooks and Wizards


ORACLE RPAS Fusion Client
Activities1
Activities2
Activities3
Administration
Item Planning
User Administration
Add User
Add User Group
Delete User
Delete User Group
Edit User
Administration
Analysis

Note: The Alert Manager Wizard is described in the *RPAS User Guide for the Fusion Client*.

The Translation Administration workbook is described in the "Internationalization" chapter.

When working with a global domain, the following administration workbooks are available only in the master domain:

- Hierarchy Maintenance
- Password Policy Administration
- Security Administration
- Translation Administration
- Workbook Batch Category Management

All other administration workbooks are available in both the local and master domains.

Auto Workbook Maintenance Wizard

The Workbook Auto Build feature allows users to have workbooks built by the wbmgr utility. Workbooks built in this way are added to the auto build queue. When workbooks are built in this manner, users do not have to spend time making selections in the wizard and waiting for the workbook to build.

Note that for a global domain environment, users are asked to select a domain to work in before the wizard executes.

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

Adding 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. On the Activity Taskflow pane, click the **Expand** icon next to the **Administration** activity. The Administration activity appears.
2. In the Administration activity, click the **Expand** icon next to the **Administration** task.
3. Under Administration task, select the **Create New Workbook** icon next to the **Auto Workbook Maintenance** step. The Workbook Wizard window appears.
4. In the Workbook Wizard window, select **Add a workbook** and click **Next**.
5. Select a workbook template type and then click **Next**.
6. Select an owner for the workbook and then click **Next**.
7. Enter relevant information in the following fields:
   - **Build Label**: Name of the auto build workbook.
   - **Build Frequency (in days)**: Frequency in days with which the workbook should be built.
   - **Next Build Date**: Date (in mm/dd/yyyy format) for the next build.
   - **Category**: The category to which the workbook should belong. See "Workbook Batch Category Management" for more information.
8. In the **Please select the saved access for this workbook** area, select **User**, **Group**, or **World** to specify the saved access for the workbook.

9. In the **Please select the group that owns this workbook** area, select the group that owns the workbook.
10. Click **Next** to initialize the wizard for the workbook template selected in step 5. The choices made are saved under the name specified in the **Build Label** field.

Deleting a Workbook from the Auto Build Queue

To delete a workbook from the Auto Build Queue, perform the following steps:

1. In the taskflow, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Administration**.
3. Under Administration, select the **Create New Workbook** icon next to **Auto Workbook Maintenance**. The Workbook Wizard window appears.
4. In the Workbook Wizard window, select **Delete workbooks**, and click **Next**.
5. Select the workbook or workbooks to delete from the auto build queue.
6. Click **Finish** to delete the workbooks.

Editing a Workbook in the Auto Build Queue

To edit the settings of an auto build workbook, perform the following steps:

1. In the taskflow, click the **Expand** icon next to **Administration** activity. The Administration activity appears.
2. In the Administration activity, click the **Expand** icon next to **Administration** task.
3. Under the Administration task, select the **Create New Workbook** icon next to **Auto Workbook Maintenance** step. The Workbook Wizard window appears.
4. In the Workbook Wizard window, select **Edit workbook settings**, and click **Next**.
5. Ensure that the **OK** radio button is selected and then click **Finish**. The Edit Auto Workbook Settings window appears.
6. In the Edit Auto Workbook Settings window, make the relevant changes in the following columns:
   - **Category**: The category to which the workbook should belong. See "Workbook Batch Category Management" for more information.
   - **Frequency (in days)**: Frequency (in days) with which the workbook should be built.
   - **Save Group**: From the drop-down list, select the group that owns the workbook.
   - **Workbook Name**: Name of the workbook.
   - **Next Build Date**: Date (in mm/dd/yyyy format) for the next build.
   - **World Access**: Select this option to make the workbook accessible to all users.

   Note: The **Owner** and **Template** columns are read-only.

7. After you make your selection, for the changes to take effect, click the **Commit** icon on the global toolbar. You can also choose to click **Commit** in the File menu.

You can choose to save this workbook by clicking the Save icon on the global toolbar or clicking the **Save** option from the File menu.

Workbook Batch Category Management

A category is defined as a group of related workbooks for batch processing purposes. As an administrator, you can create new categories that auto workbook queue entries and batch workbook refresh entries can be assigned to. Each entry can be assigned to only one category. By default, each entry is assigned to a master category called Default. If a category is deleted, the entries assigned to that category are reassigned to the Default category.

When setting up auto workbooks, users can assign a category to an auto workbook and then run the build of a category that a group of entries have been assigned to. This is useful because it allows users to build only the workbooks they need to work with. These categories can also be used when entering workbooks into the refresh queue with the wbbatch utility. Just as with building workbooks, users can refresh select workbooks based on the defined category. These categories are created with the Workbook Batch Category Management wizard or with the `wbbatch` utility. For more information about the assignment of categories, see the "Auto Workbook Maintenance Wizard" and "Managing the Workbook Batch Queue Using wbbatch" sections.

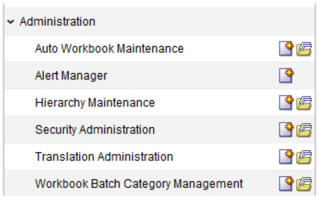
In a global domain, the workbook batch category collection is global and stored in the master domain. It is shared by all local domains.

Workbook Batch Category Management Wizard

The Workbook Batch Category Management wizard allows users to add and delete categories as well as edit the labels of the categories. To access this wizard, perform the following steps:

1. Click the **New Workbook** icon in the Workbook Batch Category Management step.

Figure 8–2 Workbook Batch Category Management Wizard


Administration
Auto Workbook Maintenance
Alert Manager
Hierarchy Maintenance
Security Administration
Translation Administration
Workbook Batch Category Management

The Workbook Batch Category Management Wizard opens.

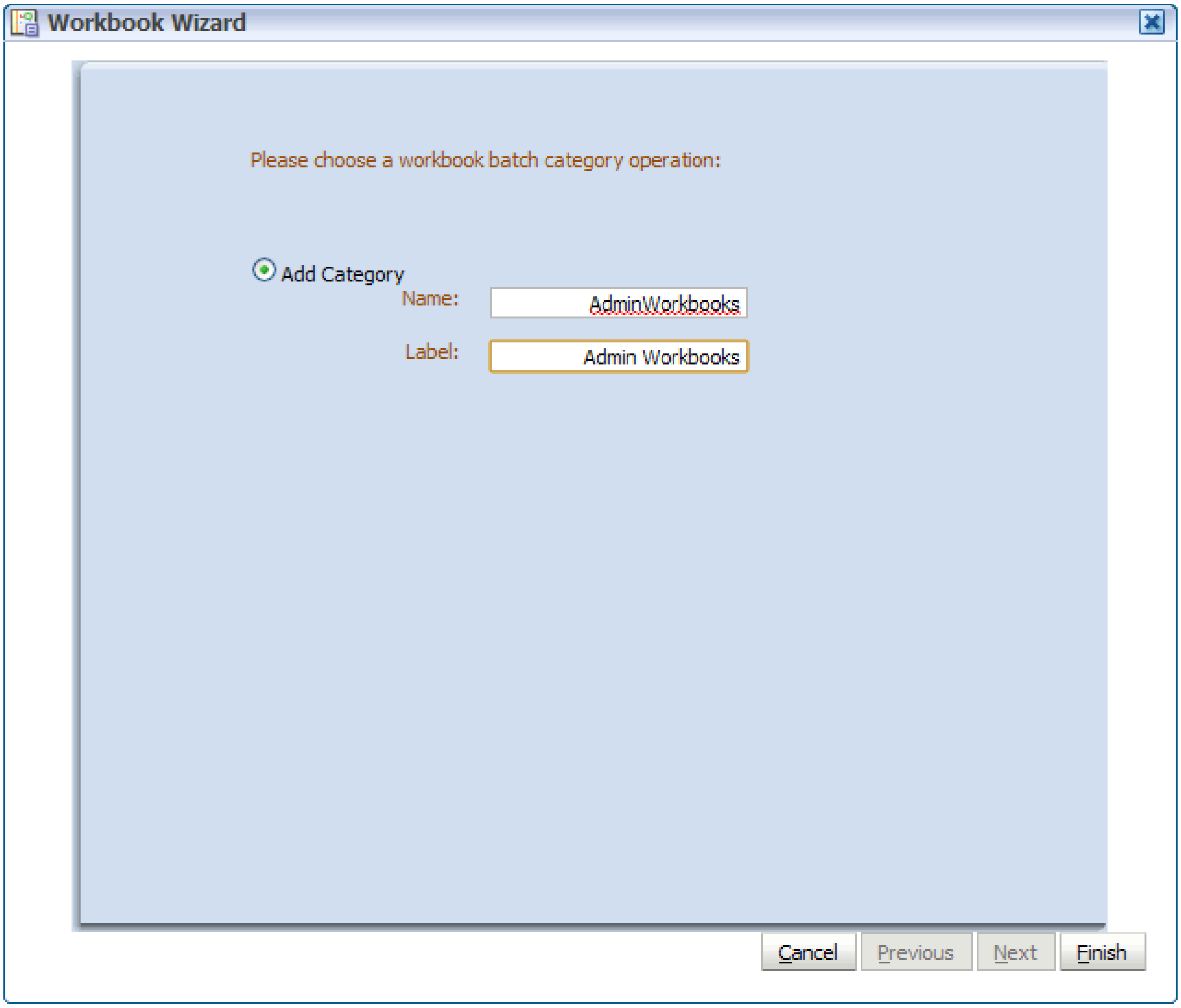
Adding a Category

To add a category, select **Add Category** and enter a name and label for the category.

- **Name**: The category name is restricted to standard alphanumeric characters. It cannot contain spaces. This name is used when specifying the category in the wbbatch utility. After a category has been created, this name cannot be changed.
- **Label**: The category label is displayed on the workbook template wizard pages. It can be in any language and can contain spaces. Category labels are case sensitive and must be unique.

After you have entered a name and label, click **Finish**. The category is created.

Figure 8–3 Adding a Category


Workbook Wizard

Please choose a workbook batch category operation:

● Add Category

Name: AdminWorkbooks

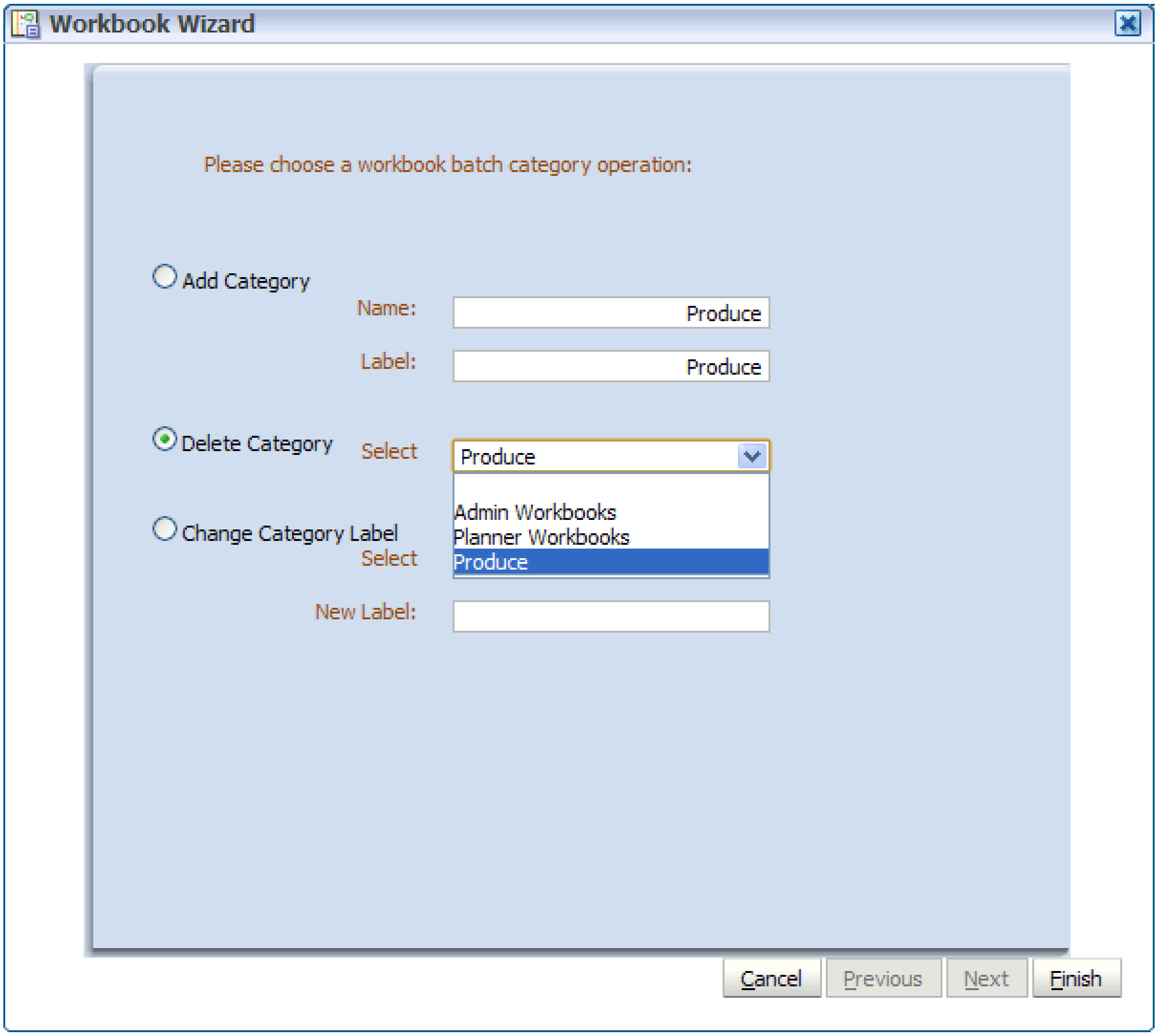
Label: Admin Workbooks

Cancel
Previous
Next
Finish

Deleting a Category

To delete a category, select **Delete Category** and select the category you want to delete from the list. Click **Finish**.

Figure 8–4 Deleting a Category



Please choose a workbook batch category operation:

○ Add Category
Name: Produce
Label: Produce

● Delete Category Select Produce

Admin Workbooks
Planner Workbooks
Produce

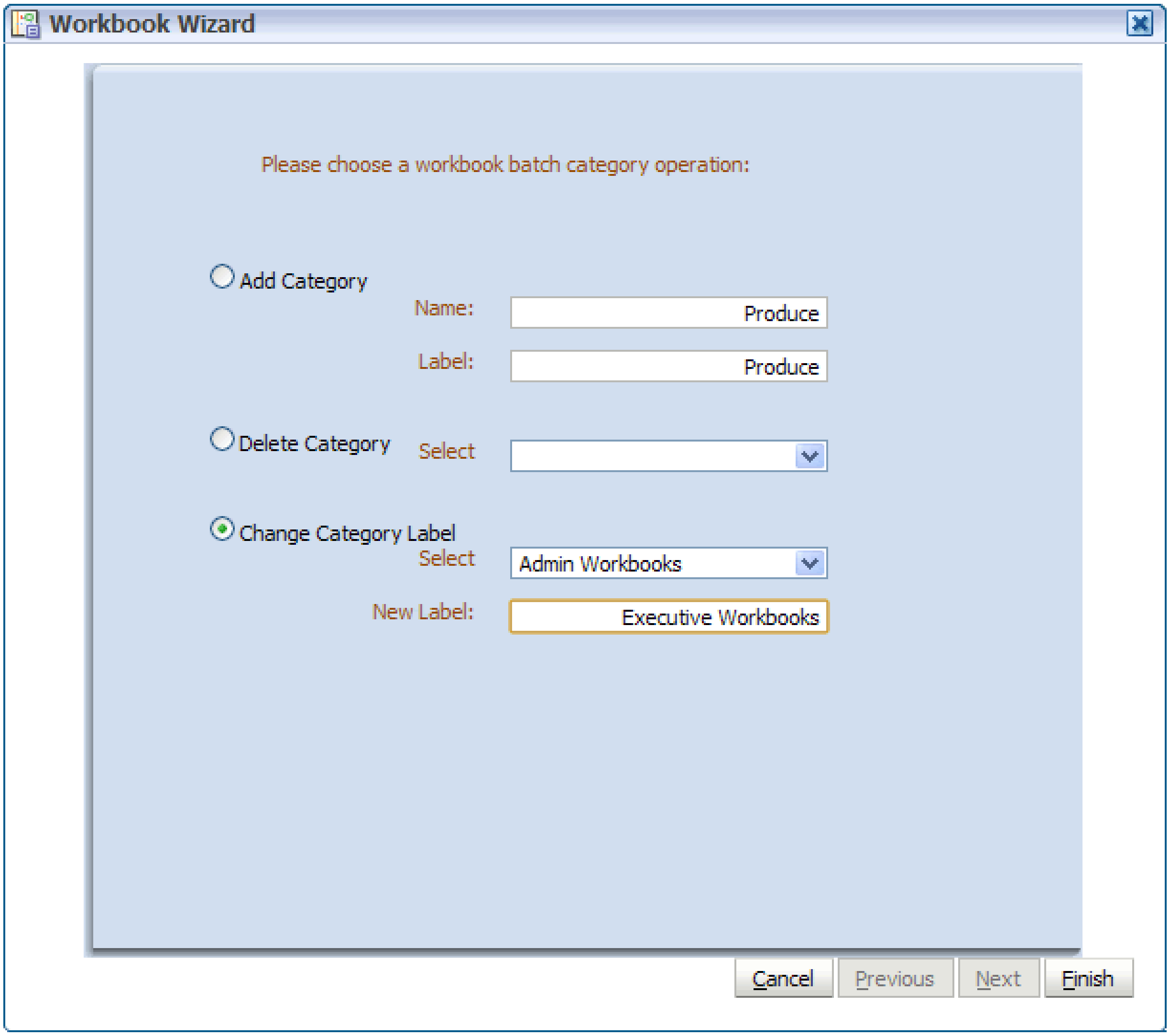
○ Change Category Label
Select
New Label:

Changing a Category Label

To change the category label, select **Change Category Label**. Select the label you want to change and enter the new one. Click **Finish**.

Note: Category names cannot be changed.

Figure 8–5 Changing a Category Label



Please choose a workbook batch category operation:

○ Add Category

Name: Produce

Label: Produce

○ Delete Category Select

● Change Category Label

Select: Admin Workbooks

New Label: Executive Workbooks

Hierarchy Maintenance Workbook

Oracle Retail Predictive Solutions can be used 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 view, 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 can 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.

Remember 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 allows 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 are 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.

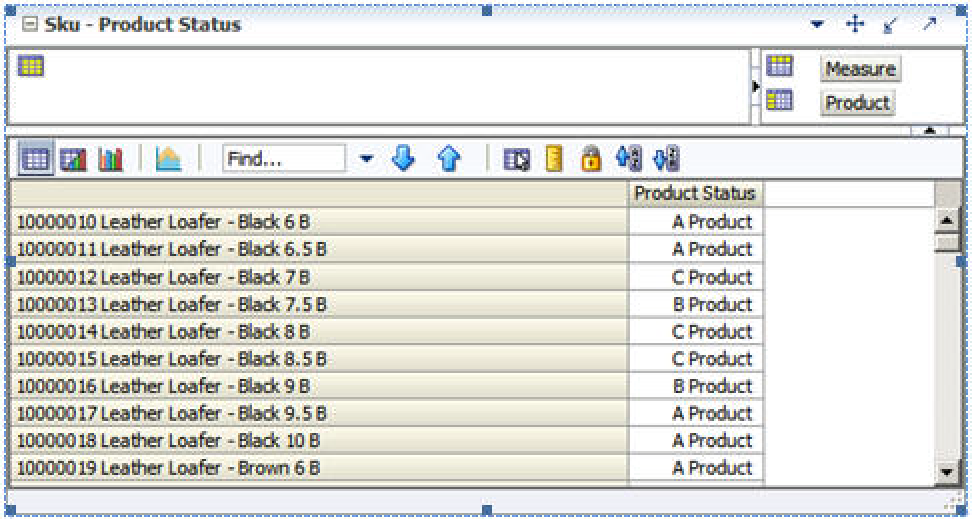
Hierarchy Maintenance View

The Hierarchy Maintenance view 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 either 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 is 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.

Figure 8–6 Hierarchy Maintenance View



	Product Status
10000010 Leather Loafer - Black 6 B	A Product
10000011 Leather Loafer - Black 6.5 B	A Product
10000012 Leather Loafer - Black 7 B	C Product
10000013 Leather Loafer - Black 7.5 B	B Product
10000014 Leather Loafer - Black 8 B	C Product
10000015 Leather Loafer - Black 8.5 B	C Product
10000016 Leather Loafer - Black 9 B	B Product
10000017 Leather Loafer - Black 9.5 B	A Product
10000018 Leather Loafer - Black 10 B	A Product
10000019 Leather Loafer - Brown 6 B	A Product

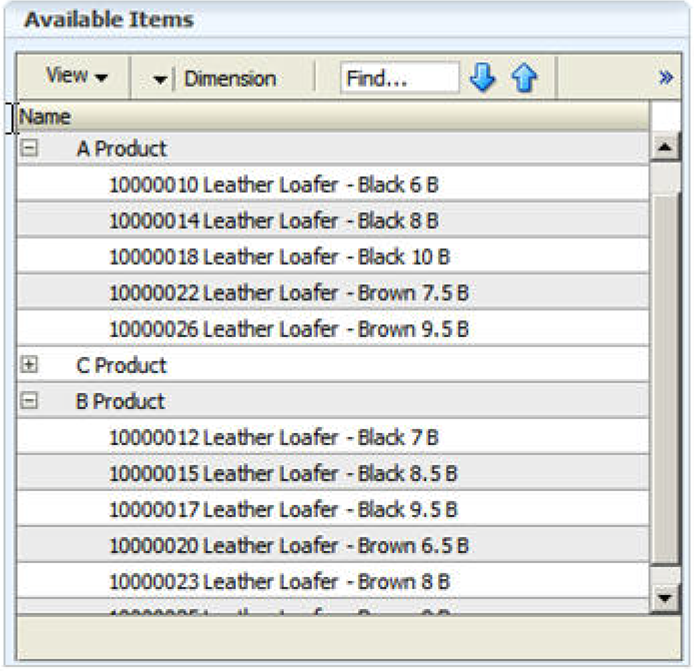
Note: The Oracle Retail Predictive Solutions system is case sensitive when a new position label is entered in the Hierarchy Maintenance workbook. After the workbook is committed, the typing of the group name is not case sensitive. For example, "B Product" can later be entered as "b product" after the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

Figure 8–7 Product Status Dimension Results



Name

- A Product
  - 10000010 Leather Loafer - Black 6 B
  - 10000014 Leather Loafer - Black 8 B
  - 10000018 Leather Loafer - Black 10 B
  - 10000022 Leather Loafer - Brown 7.5 B
  - 10000026 Leather Loafer - Brown 9.5 B
- C Product
- B Product
  - 10000012 Leather Loafer - Black 7 B
  - 10000015 Leather Loafer - Black 8.5 B
  - 10000017 Leather Loafer - Black 9.5 B
  - 10000020 Leather Loafer - Brown 6.5 B
  - 10000023 Leather Loafer - Brown 8 B

Maintaining 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. You can use these dimensions in the same way that you use normal Demand Forecasting levels, aggregating data along these new hierarchical paths.

1. In the taskflow, click the **Expand** icon next to **Administration** activity. The Administration activity appears.
2. In the Administration activity, click the **Expand** icon next to **Administration** task.
3. Under the Administration task, select the **Create New Workbook** icon next to **Hierarchy Maintenance** step. A Workbook Wizard window appears.
4. 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 appear here.
5. Click **Next**.
6. Select the user-defined dimension to be updated. The number and names of available custom dimensions are set at installation.
7. Click **Next**.
8. From the Available Items in the selection wizard, select the items to be mapped to positions within the custom dimension.
9. Use the Add, Add All, Remove, and Remove All buttons or the drag and drop action to move your selection to the **Selected Items** section.
10. After all items to appear in your workbook have been selected, click **Finish**. The Hierarchy Maintenance workbook appears.

11. 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 is treated as a separate user-defined position within the custom dimension.
12. After you make your selection, for the changes to take effect, click the **Commit** icon on the global toolbar. You can also choose to click **Commit** in the File menu.

    You can choose to save this workbook by clicking the **Save** icon on the global toolbar or clicking the **Save** option from the File menu.
13. To close the workbook, click **Home** on the top-right corner of the screen.
14. 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 is treated as a separate user-defined position within the custom dimension.
15. Select **Commit Now** from the File menu to commit the changes to the master database. If desired, you may also save the workbook by selecting Save from the File menu.
16. To close the workbook, select **Close** from the File menu.

Security Administration Workbook

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

Security Overview

This section provides the basic information on the security model in RPAS.

User and Group Security

During Fusion Client installation one or more user groups are specified via the "input.security.group" property. These groups are required to be created in the LDAP directory that the customer has associated with the Weblogic Server for authentication. It is not necessary for these groups to be present before the installation. They can be created after the installation also.

Users created as members of these groups gain access to the Fusion Client UI, as long as they are also present in the RPAS domains. For example, say that the group "rpasusers" is specified during installation. Users A and B are members of rpasusers, whereas user C is not. This means that A and B can access the application whereas C while logging in will receive an "unauthorized" error message. (A and B must also be present in the RPAS domains; otherwise they will receive a "no solutions found" message.)

In course of time it may be necessary to add and remove users. This can be done simply adding and removing them from the LDAP directory. It is not essential to remove them from the RPAS domains as well, but it would be a good housekeeping practice.

An unusual need may arise where a user has to be provided access to the application even though the user is not a member of any of the groups configured for access during installation. Or, a whole new group needs to be provided access. This can be arranged for by one of the following methods:

1. Basic: Re-install the Fusion Client after adding the users and groups to the input.security.user and input.security.group properties respectively. (It is assumed that the customer has kept the ant.install.properties file used in the previous install in a safe place, so that it can be used in the new install.)
2. Complex: Use Enterprise Manager Fusion Middleware Control web application hosted on the Weblogic Server (and accessed via URL http://<host>:<port>/em). Go to the Security Roles section of the Fusion Client application and add the new users and groups as members of the application role "atlantis-users".

   See Fusion Middleware Security Guide for information on how to use the Enterprise Manager Fusion Middleware Control application.
3. Restart application or Weblogic Server after this step.

If using second approach, you must remember to re-do the Enterprise Manager settings after a Fusion Client re-install. Otherwise they will be lost.

Measure Level Security

Measures have access rights; these 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 view allows Read Only, Deny, or Read/Write access to a measure to be specified for each user.

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

Note: Refer to the *RPAS User Guide for the Fusion Client* 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. After the 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. Views 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). Three views are provided in this workbook for each hierarchy with a defined security dimension. The default view controls access to positions for all users (for instance, Prod Security Default); one view controls access to positions by user group (for instance, Prod Security Group); and the last view 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:

Table 8–1 Grant or Denial of Access by a Combination 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.

Note that position-level security, when used for a global domain environment on the same dimension on which it is partitioned, is used to guide a user to the domain or domains that user has access to. If a user only has access to positions within a single local domain, that user will be guided there on New Workbook. If a user has access to more than one, that user will be asked and can choose based on partition-level positions.

Similarly, Open by default only lists workbooks from those domains, and a user is only shown alert counts from those domains.

Workbook Security

Currently, workbook access is either granted or denied. If users have been granted access to a workbook, they 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.

Note that the Open dialog initially shows only workbooks owned by the current user and in domains for which the user has position level security access. This is not the same as workbook access, however, and a user may have access to workbooks saved by others in other domains by using View > Other Domains in the Open dialog by others Word or Group.

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 do not have access to Security template and User Administration template groups even if the administrator inadvertently assigns them access rights.

Security Administration Workbook

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. The user 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 the following views:

- Workbook Template Rights View
- Workbook Template Measure Rights View
- Measure Rights View
- Dimension Modification Rights View
- Position Level Security Views
- Workbook Template Limits Views
- Max Domain Session Limit View
- Max User Session Limit View

Security Template Administration also allows the administrator to modify the label, Admin status, and default workbook template associated with each user. The administrator can 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 view.
- Determine which optional measures are to be accessible through individual workbook templates. Template/measure permissions are set in the Workbook Template Measure Rights view.
- Assign and restrict user access to individual measures. User/measure permissions are established in the Measure Rights view.

Workbook Template Rights View

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

The view contains a drop-down list for each available workbook template and user combination. To grant a user access rights to a workbook template, select one of the following options from the drop-down list for that workbook template:

- Denied
- Read Only
- Full Access

After a user's profile is changed, those 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 does not prevent the user 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 View

The Workbook Template Measure Rights view allows administrators to determine which registered measures are 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 through the Insert Measure command.

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

This view contains a check box for each available workbook template and registered measure combination.

Measure Rights View

The Measure Rights view 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 view to deny users access to measures that they would normally be permitted to edit.

The view 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 datastore.

A measure has the security rights it had when it was inserted in the workbook. The change in measure security rights is only reflected in new workbooks when that measure is inserted.

Note:

- If a measure that has dependent measures is inserted into a view, those dependent measures will also be inserted into the view. If the dependent measures have denied measure access, they are still inserted into the view, but they are hidden by default. These measures can be made visible on the Show/Hide dialog box.
- The Measure Rights view contains only public measures, that is, measures that can be optionally included in a view, depending on choices made in a new workbook wizard. Measures that are registered as private measures do not appear in this view. If there are no public measures available to be displayed in this view, the view will not be built.

Dimension Modification Rights View

The Dimension Modification Rights view allows the administrator to determine which dimensions, if any, a user can modify. The view contains a check box for each available user and dimension combination. A check mark 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 Views

The position-level security views 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 views: 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 Views

The Workbook Template Limit views 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 overrides any values defined at group-template or template. If the above limits are not defined, the default value is one 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.

Max Domain Session Limit View

The Max Domain Session Limit view is used to limit the number of user sessions that can be attached to a single domain by all users of that domain. The limit is set at the domain level. In a global domain environment, the same limit is applied individually to each local domain and the master domain.

This limit is checked during user login. If the limit has been reached, an error message appears to inform the user that the login has failed because this limit has been reached.

Max User Session Limit View

The Max User Session Limit view is used to limit the number of concurrent user sessions that can be attached to a single domain by the same user at the same time. The limit is set per user so that the administrator can control the maximum number of concurrent sessions that are allowed for an individual user. In a global domain environment, the same limit is applied individually to each local domain and the master domain.

This limit is checked during user login. If the limit has been reached, an error message appears to inform the user that the login has failed because this limit has been reached.

Using the Security Administration Workbook

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

Accessing Security Administration

1. On the Activity Taskflow pane, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Administration**.
3. Under Administration, select the **Create New Workbook** icon next to **Security Administration**.

Setting or Modifying Users' Access to Workbook Templates

The Workbook Template Rights view enables you to set one of the following access privileges to each workbook template for a user:

- Denied
- Read-only
- Full Access

To set or modify a user's access to workbook templates:

1. On the Activity Taskflow pane, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Administration**.
3. Under Administration, select the **Create New Workbook** icon next to **Security Administration**.
4. On the Workbook Template Rights view, for each workbook template, select one of the access privileges for the users you want.
5. After you make your selection, for the changes to take effect, click the **Commit** icon on the global toolbar. You can also choose to click **Commit** in the File menu.

You can choose to save this workbook by clicking the **Save** icon on the global toolbar or clicking the **Save** option from the File menu.

To close the workbook, click Home on the top-right corner of the screen.

Setting Measure Availability for Workbook Templates

1. On the Activity Taskflow pane, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Administration**.
3. Under Administration, select the **Create New Workbook** icon next to **Security Administration**.
4. On the Workbook Template Measure Rights view, select each registered measure that must be available for inclusion in the associated workbook template. For measures that must not be included in the associated template, make sure there is no check mark.

5. After you make your selection, for the changes to take effect, click the **Commit** icon on the global toolbar. You can also choose to click **Commit** in the File menu.

You can choose to save this workbook by clicking the **Save** icon on the global toolbar or clicking the **Save** option from the File menu.

To close the workbook, click **Home** on the top-right corner of the screen.

Assigning or Restricting User Access to Measures

1. On the Activity Taskflow pane, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Administration**.
3. Under Administration, select the **Create New Workbook** icon next to **Security Administration**.
4. On the Measure Rights view, for each measure that a user should have access to, select **Read Only** or **Read/Write** from the drop-down list in the cell. For measures to which the user should not have access, make sure **Denied** is selected.

Note: The drop-down list appears when you double-click inside the cell.

5. After you make your selection, for the changes to take effect, click the **Commit** icon on the global toolbar. You can also choose to click **Commit** in the File menu.

You can choose to save this workbook by clicking the **Save** icon on the global toolbar or clicking the **Save** option from the File menu.

To close the workbook, click **Home** on the top-right corner of the screen.

Changing a User's Ability to Modify Dimensions

1. On the Activity Taskflow pane, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Administration**.
3. Under Administration, select the **Create New Workbook** icon next to **Security Administration**.
4. On the Dimension Modification Rights view, 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.
5. After you make your selection, for the changes to take effect, click the **Commit** icon on the global toolbar. You can also choose to click **Commit** in the File menu.

You can choose to save this workbook by clicking the **Save** icon on the global toolbar or clicking the **Save** option from the File menu.

To close the workbook, click **Home** on the top-right corner of the screen.

Setting or Modifying Access to Positions (if Position Level Security has been Enabled)

1. On the Activity Taskflow pane, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Administration**.

3. Under Administration, select the **Create New Workbook** icon next to **Security Administration**.
4. On the Position Level Security view, select the view for which security needs to be set or modified: **User, User Group**, or **World**.
5. By default, the dimension (level) at which position level security is enabled appears. 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.
6. To grant access to a position, select the check box in the relevant cell.

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

7. After you make your selection, for the changes to take effect, click the **Commit** icon on the global toolbar. You can also choose to click **Commit** in the File menu.

You can choose to save this workbook by clicking the **Save** icon on the global toolbar or clicking the **Save** option from the File menu.

To close the workbook, click **Home** on the top-right corner of the screen.

Limiting the Number of Workbooks that a User Can Save

1. On the Activity Taskflow pane, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Administration**.
3. Under Administration, select the **Create New Workbook** icon next to **Security Administration**.
4. On the Workbook Limit Per User/Template view, for a workbook template, set an appropriate value as the limit for a user.
5. After you make your selection, for the changes to take effect, click the **Commit** icon on the global toolbar. You can also choose to click **Commit** in the File menu.

You can choose to save this workbook by clicking the **Save** icon on the global toolbar or clicking the **Save** option from the **File** menu.

To close the workbook, click **Home** on the top-right corner of the screen.

Limiting the Number of Workbooks Per Template

1. On the Activity Taskflow pane, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Administration**.
3. Under Administration, select the **Create New Workbook** icon next to **Security Administration**.
4. On the Workbook Limit Per Template view, for a workbook template, set an appropriate value as the limit.
5. After you make your selection, for the changes to take effect, click the **Commit** icon on the global toolbar. You can also choose to click **Commit** in the File menu.

You can choose to save this workbook by clicking the **Save** icon on the global toolbar or clicking the **Save** option from the File menu.

To close the workbook, click **Home** on the top-right corner of the screen.

Limiting the Number of Workbooks that a Group Can Save

1. On the Activity Taskflow pane, click the Expand icon next to Administration. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Administration**.
3. Under Administration, select the Create New Workbook icon next to Security Administration.
4. On the Workbook Limit Per Group/Template view, for a workbook template, set an appropriate value as the limit for the user group.
5. After you make your selection, for the changes to take effect, click the **Commit** icon on the global toolbar. You can also choose to click **Commit** in the File menu.

You can choose to save this workbook by clicking the **Save** icon on the global toolbar or clicking the **Save** option from the File menu.

To close the workbook, click **Home** on the top-right corner of the screen.

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 writable measures; however, commit capability is only allowed to administrative users.

Note that for a global domain environment, the user may be asked to select a domain before the wizard appears.

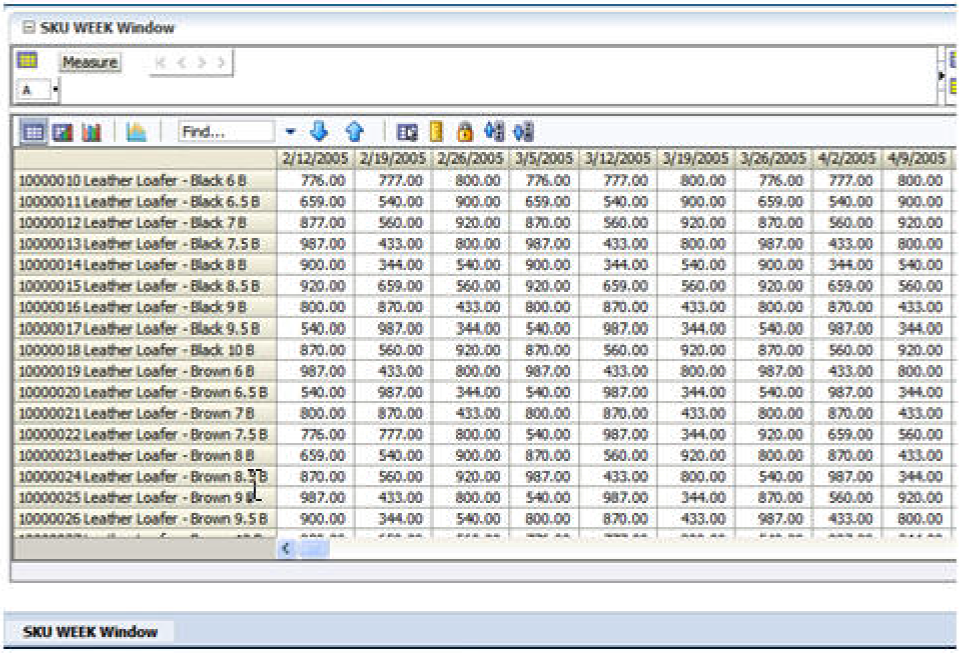
Although a common use of the Measure Analysis workbook is to view actual sales data, the workbook is not restricted to presenting sales data alone. The user can view any data loaded into the Oracle Retail Predictive Solutions master database, such as selling prices, shipments, and orders. The Measure 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.

Note: Due to its dynamic nature, formatting settings cannot be saved in the Measure Analysis workbook.

Measure Analysis View

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

Figure 8–8 Example of Measure Analysis View



	2/12/2005	2/19/2005	2/26/2005	3/5/2005	3/12/2005	3/19/2005	3/26/2005	4/2/2005	4/9/2005
10000010 Leather Loafer - Black 6 B	776.00	777.00	800.00	776.00	777.00	800.00	776.00	777.00	800.00
10000011 Leather Loafer - Black 6.5 B	659.00	540.00	900.00	659.00	540.00	900.00	659.00	540.00	900.00
10000012 Leather Loafer - Black 7 B	877.00	560.00	920.00	870.00	560.00	920.00	870.00	560.00	920.00
10000013 Leather Loafer - Black 7.5 B	987.00	433.00	800.00	987.00	433.00	800.00	987.00	433.00	800.00
10000014 Leather Loafer - Black 8 B	900.00	344.00	540.00	900.00	344.00	540.00	900.00	344.00	540.00
10000015 Leather Loafer - Black 8.5 B	920.00	659.00	560.00	920.00	659.00	560.00	920.00	659.00	560.00
10000016 Leather Loafer - Black 9 B	800.00	870.00	433.00	800.00	870.00	433.00	800.00	870.00	433.00
10000017 Leather Loafer - Black 9.5 B	540.00	987.00	344.00	540.00	987.00	344.00	540.00	987.00	344.00
10000018 Leather Loafer - Black 10 B	870.00	560.00	920.00	870.00	560.00	920.00	870.00	560.00	920.00
10000019 Leather Loafer - Brown 6 B	987.00	433.00	800.00	987.00	433.00	800.00	987.00	433.00	800.00
10000020 Leather Loafer - Brown 6.5 B	540.00	987.00	344.00	540.00	987.00	344.00	540.00	987.00	344.00
10000021 Leather Loafer - Brown 7 B	800.00	870.00	433.00	800.00	870.00	433.00	800.00	870.00	433.00
10000022 Leather Loafer - Brown 7.5 B	776.00	777.00	800.00	540.00	987.00	344.00	920.00	659.00	560.00
10000023 Leather Loafer - Brown 8 B	659.00	540.00	900.00	870.00	560.00	920.00	800.00	870.00	433.00
10000024 Leather Loafer - Brown 8.5 B	870.00	560.00	920.00	987.00	433.00	800.00	540.00	987.00	344.00
10000025 Leather Loafer - Brown 9 B	987.00	433.00	800.00	540.00	987.00	344.00	870.00	560.00	920.00
10000026 Leather Loafer - Brown 9.5 B	900.00	344.00	540.00	800.00	870.00	433.00	987.00	433.00	800.00

Figure 8–8 shows a Measure Analysis view that displays Weekly Sales data for several items in a particular store. The location/product/calendar dimensional intersection of this view, 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 view. However only an administrative user can commit overwrites to writable measures in this workbook.

Accessing Measure Analysis

1. On the Activity Taskflow pane, click the **Expand** icon next to **Administration**. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Analysis**.
3. Under Analysis, select the **Create New Workbook** icon next to **Measure Analysis**.

Reviewing and Editing Sales or Other Registered Measure Data

1. On the Activity Taskflow pane, click the **Expand** icon next to Administration. The Administration panel appears.
2. In the Administration panel, click the **Expand** icon next to **Analysis**.
3. Under Analysis, select the **Create New Workbook** icon next to **Measure Analysis**.

   The Workbook Wizard window appears.

   > **Note:** To open an existing Measure Analysis workbook, click the **Show List of Workbooks** icon next to Measure Analysis, and then go to step 8.

4. In the Workbook Wizard window, select the measures you want to include in the new workbook. Use Ctrl-Click or Shift-Click to select multiple measures.

5. Click **Next**. Based on each hierarchy specified in the base intersection of the selected measures, a hierarchy wizard appears.
6. In the hierarchy wizard, select the relevant positions and then click **Next**. Repeat this step for each hierarchy wizard.
7. Click **Finish** to open the Measure Analysis workbook.
8. On the Measure Analysis View(s), view the stored data associated with the measures and hierarchy positions selected in the wizard. Make any changes as required. As an administrator you can commit the changes.

9

Hierarchy Management

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.
- RPAS uses integer indexing for simplified hierarchy administration. A set number of hierarchy positions, based on bit size, is allocated to each dimension. The pre-allocation of positions reduces the need for updating the measure data structures.
- The length of position names is 24 characters or less by default. RPAS provides the ability to increase this length using the dimensionMgr utility.
- Position names must consist of *only* the following characters: a-z, A-Z, 0-9, _, &, $, and %. Position names cannot start with _. Any uppercase letters are converted to lowercase letters by the application. Position names cannot be an empty string.
- RPAS provides the ability to have placeholder positions in the domain that can be used when loading new hierarchy positions.
- 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 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.

This chapter contains the following sections:

- Loading Hierarchies Using loadHier
- Integer Indexing
- Reindexing Domains Using reindexDomain
- Optimizing Domains Using optimizeDomain
- Adding New Dimensions to Hierarchies
- FilterHier Utility
- Position Repartitioning
- Reconfiguring the Partitions of a Global Domain Using reconfigGlobalDomainPartitions
- Renaming Positions Using renamePositions

- Setting Properties for Dimensions Using dimensionMgr
- Exporting Hierarchy Data Using exportHier
- Informal Position Manager
- Managing Position Lists as PQDs Using pqdMgr

Loading Hierarchies Using loadHier

The loadHier utility is used to load and refresh a hierarchy. The loadHier utility supports comma separated value (CSV) or fixed width flat files for loading. The load file should have a .dat file extension when a flat file is loaded and a .csv.dat file extension when a CSV file is loaded. When using a fixed width file, the width of fields (number of characters) is specified in a configuration file before a domain is built. The width of fields 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. The utility also allows a simple compression method that can skip duplicated values line by line.

Note: The following concern hierarchy loading with intraday:

- Pre-13.2.2: Hierarchy loading is not supported while RPAS users are online unless all users are asked to limit the scope of their activities to workbook calculations only. Other operations such as workbook build, refresh, commit, custom menus, and dynamic position maintenance (DPM) can interfere with the hierarchy loading process. In such scenarios, users and administrators can experience concurrency issues and, in the worst case, hierarchy data corruption can occur.
- 13.2.2 and later: The Ride framework allows the execution of the loadHier process. Users can stay logged in and continue their activities as long as they do not perform activities such as workbook build, refresh, DPM, and custom menu scripts that touch hierarchies. The system restricts users from performing these activities while the batch process is running. Users can submit commit-ASAP requests that are queued and executed at the completion of the batch process.

To manage the addition, removal, and reclassification of positions in a hierarchy, RPAS uses a methodology called integer indexing. It is used to manage multidimensional data at the storage level. For more information, see "Integer Indexing".

The loadHier utility 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 and 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. Multiple files are often used when the hierarchy load data comes from different sources.

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.

The loadHier utility stops with an error if the loadHierBk directory exists in the data directory of the domain, which indicates that a non-recoverable error may have occurred in the previous run. If this occurs, contact My Oracle Support at `http://www.oracle.com/support/contact.html`. The My Oracle Support team can best determine whether to delete the loadHierBk directory or copy the loadHierBk content back to the domain.

To optimize performance while moving or cleaning data during the hierarchy purging or reclassification processes, the `-excludeMeasList` or `-includeMeasList` argument can be specified. Both arguments specify a full path to an xml file in the following format, which contains a list of measures to either be excluded or included:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
</rpas>
     <measures>meas1,meas2,meas3…,measN</measures>
</rpas>
```

Note: It is important to specify the measures wisely, especially with the `-includeMeasList` option. This is because no data moves for those measures not included in the list; that data is lost when loadHier completes.

loadHier Usage

```
loadHier -d domainPath -load hierName -loadAll {-purgeAge purgeage}
{-purgeAll hierarchy1} {-noClean}{-loglevel level} {-defaultDomain ldom#, ldom#}
{-excludeMeasList listName | -includeMeasList listName} {-includeUdd}
```

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

Table 9–1 loadHier Utility Arguments

Argument	Description
-d domainPath	Indicates the domain in which to load the hierarchy data.
-load hierName	Indicates 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.
-purgeAge *purgeage*	Specifies the purgeage during loadHier.If not specified, loadHier gets purgeage from the domain. In global domains, -purgeAge supports the purge of partition positions when the *purgeage* is reached.
-purgeAll *hierarchy1, hierarchy2*	Purges formal, informal, and user-defined positions in the listed hierarchies. It cannot be used on a partition hierarchy or any system hierarchy.
-noClean	Prevents the removal of input files and temporary data files that are generated during the hierarchy load process. Input files remain in the input directory of the domain after the process is completed. This option is often used for debugging or troubleshooting purposes.
-logLoadedPositions	Enables the logging of successfully loaded input file lines into a loaded[HIERNAME].dat file under the processed directory.

Table 9–1 (Cont.) loadHier Utility Arguments

Argument	Description
-maxProcesses *count*	Specifies that some parts of loadHier run in parallel, meaning that it uses a maximum of the defined processes, which are specified by *count*.
-forceInputRollups	Enforces new hierarchy roll-up changes. New roll-up changes override or dominate existing hierarchy roll-ups if 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	Forces NA consistency when the current NA value is different from the originally defined NA value for the measure.
-includeUdd	Loads user-defined positions back to the domain. The data file must be in CSV format with a headerline. The name of the data file should follow the current standard: <hierarchy name>.csv.dat. All user-defined dimensions must be in the data file. Any missing user-defined dimensions cause an error. All loaded positions have formal status after running -includeUdd.
-defaultDomain ldom#, ldom#,	Specifies comma separated default domain paths that are 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`
-excludeMeasList *listName* or -includeMeasList *listName*	Use one of these arguments to optimize performance while moving or cleaning data during the hierarchy purging or reclassification processes: ■ Use -excludeMeasList to optimize performance by excluding the list of measures in *listName*. ■ Use -includeMeasList to optimize performance by including only the list of measures in *listName*.
-headerLine	Specifies the column order base on the header line. If not specified, the order is based on the start property of the dimension. This option is only applicable to the loading of CSV files; it has no effect on the loading of fixed-width files.

loadHier 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 position is added.

Example:

For a global domain that consists of two local domains, ldom0 and ldom1, using the following loadHier command:

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

In this call, three new partition positions (part1, part2, and part3) are in the input file. When the loadHier finishes, two new local domains, ldom2 and ldom3, are available, which include the following new partition positions:

```
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 five partition positions are added (part1, part2, part3, part4 and part5), when the loadHier finishes, two new local domains, ldom2 and ldom3 are available. New partition positions are located as follows:

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

Position Label Translation

To enable translation of position labels for the desired dimensions using the RPAS Configuration Tools, check the box in the Translate column of the Dimension definition tools. Building or patching the domain with this configuration builds the necessary infrastructure in the domain to manage translations for those dimensions. However, label translations must be separately loaded.

Position label translations are loaded in dimension specific translation measures for every language used by the users. If translated labels are not loaded using these measures, workbooks show position names wherever a label has to be shown. Note that for a translatable dimension, RPAS never uses or shows the position labels from the hierarchy load file but always refers to the labels in dimension-specific translation measures. This implies that if a domain were patched to make a dimension translatable but the translation measures were not loaded, RPAS users would see position names instead of position labels from the load file.

Dimension-specific position translation measures are named as r_<dim name>label, where <dim name> must be replaced with the name of the translated dimension. For example, if the sku dimension were to be translated, load the r_skulabel measure with translations. These measures must be loaded after loading the hierarchy because RPAS can only load translations for already loaded positions.

The position label translation measure load files have three columns. The first column has the position names, the second column has the language identifier, and the third column has the translation for the language specified in that row.

For example, a translation measure file for the sku dimension is named r_skulabel.csv.ovr and has the content formatted, as shown below. Note that in the following example, the same file contains labels in four languages.

```
10006782,ENGLISH,White Nike Running Shoe size 11
10006782,CHINESE_SIMPLIF,白色耐克？鞋大小 11
10006782,FRENCH,Taille blanche 11 de chaussure de course de Nike
10006782,ITALIAN,Formato bianco 11 del pattino corrente di Nike
10004523,ENGLISH,Black leather shoe size 8
10004523,CHINESE_SIMPLIF,黑皮鞋大小 8
10004523,FRENCH,Taille noire 8 de chaussure en cuir
10004523,ITALIAN,Formato nero 8 del pattino di cuoio
```

Note: For a list of language identifiers, see Table 13–1, " Supported Languages with Language Identifiers" in the "Internationalization" chapter.

Alternatively, you can manually enter or alter translated labels using the Translations workbook in the Administration tab. In this workbook, a worksheet is available for each dimension that has translations enabled. You can manually enter translated strings for the language of interest. After they are committed, these translations are available for every new workbook.

It is possible that, because of error that may have occurred when translation files were prepared, translated labels for some positions may not be loaded. In a situation where RPAS is unable to look up the label for the locale of the machine on which the RPAS client is being run, RPAS looks for a non-empty label string for the English language. If it fails to find a non-empty label string for the English language, it uses or shows the loaded position name of the position.

Note: For the fixed-width format of translation measure load files, RPAS limits the labels to 80 bytes (RPAS uses UTF-8 encoding). For CSV format files, there is no limit. To avoid the complexity of calculating starting positions for fixed-width format files and the limitation of translation string length, it is recommended that CSV files be used.

Integer Indexing

Integer indexing refers to the RPAS methodology for managing multidimensional data at the storage level. It aims at eliminating some of the inefficiencies around the addition, removal, and reclassification of positions in a hierarchy.

In the earlier versions of RPAS, string identifiers were constructed and used for internal addressing of the multidimensional data. Integer Indexing changes this mechanism in the sense that the data is internally referenced and stored by integer identifiers that are calculated from the dimension size and measure intersection.

Integer index calculation and management is entirely internal to RPAS. From a domain administrator’s point of view, loading and exporting hierarchy and measure data is achieved through external names consistent with earlier versions of RPAS.

One major advantage of integer indexing is its impact on reducing the complexity of operations around managing position placeholders (buffers). The number of positions that a dimension can hold is defined by a bit size that typically provides significant space for including new positions. For instance, a SKU dimension at 24 bits can include in excess of 16 million products. This provides significant growth space for adding new products.

Compared to the earlier versions of RPAS, integer indexing can improve the hierarchy load performance as it eliminates the need for running the frequent re-buffering processes and completely eliminates the reshaping process.

The following definitions provide details about integer indexing:

- **Index**: An identifier that allows for random access to the multidimensional data. In RPAS, it is a numerical value that identifies a position in a given dimension. Prior to integer indexing, RPAS used string-type position identifiers and referred to them as internal names. Internal names were internally generated by RPAS.

They were often different from the Position IDs provided in the hierarchy input files.

- **Position ID**: The name provided in the input hierarchy files. RPAS maps the position ID to an internally generated integer index.

Consider the following example, in which some items are loaded into the Product hierarchy. These items are externally identified by the position names and labels, as shown in the second and third columns in Table 9–2, which represents the SKU dimension in the product hierarchy. Integer indexes, names, and position labels are the three major pieces of information associated with a dimension.

Table 9–2 Example 1

Integer Index	Position Name	Position Label
0	SKU1234	blue sweatpants
1	SKU2345	white sweatpants
2	SKU3456	green sweatpants
3	SKU4567	purple sweatpants
4	SKU5678	yellow sweatpants
5	SKU6789	pink sweatpants
6	SKU7890	orange sweatpants
7		
8		
9		

When the green sweatpants item was loaded into the domain, its position ID, SKU3456, was mapped to an available integer index, which was 2. After these items were loaded, RPAS uses the integer indexes to look up those items instead of their item IDs. Therefore, if you perform a query for green sweatpants, RPAS first identifies that the integer index of the green sweatpants is 2. Then, it searches for instances of 2 in the array. The reason that RPAS searches for 2 instead of SKU3456 is because searching for an integer is faster than searching for a string.

Here is what happens if you delete the blue and white sweatpants from your domain because they are discontinued. Rather than delete these position IDs from the dimension, RPAS marks them as inactive. By keeping these position IDs in the dimension, RPAS maintains the position ID to integer index mapping. As a result, the green sweatpants are still mapped to the integer index 2.

Table 9–3 Example 2

Integer Index	Position Name	Position Label	Position Status
0	SKU1234	blue sweatpants	Inactive
1	SKU2345	white sweatpants	Inactive
2	SKU3456	green sweatpants	
3	SKU4567	purple sweatpants	

Table 9–3 (Cont.) Example 2

Integer Index	Position Name	Position Label	Position Status
4	SKU5678	yellow sweatpants	
5	SKU6789	pink sweatpants	
6	SKU7890	orange sweatpants	
7			
8			
9			

As you delete items from the domain, the dimension begins to have gaps in it where inactive items are using integer indexes. This is not an issue unless you want to add more position IDs than the dimension has room for. When this happens, you will need to reindex the dimension first.

For example, add five more colors of sweatpants to the dimension. This dimension has ten integer indexes, so it can hold ten total position IDs. As the dimension is now, it has only three available integer indexes (7, 8, and 9) even though there are five indexes that are not being used (0, 1, 7, 8 and 9). To use the 0 and 1 integer indexes that the inactive sweatpants are mapped to, you would use the reindexDomain utility to defragment the dimension. This removes the blue and white sweatpants and allows you to use the 0 and 1 integer indexes for the new sweatpants. For more details about reindexDomain and defragging, see "Reindexing Domains Using reindexDomain".

Table 9–4 Example 3

Integer Index	Position Name	Position Label	Position Status
0	SKU3456	green sweatpants	
1	SKU4567	purple sweatpants	
2	SKU5678	yellow sweatpants	
3	SKU6789	pink sweatpants	
4	SKU7890	orange sweatpants	
5			
6			
7			
8			
9			

However, a different result occurs if you add 15 new SKUs instead of five. Even if you defragment the dimension, there is room for only five new SKUs. In this instance, you need to add more room to the dimension by increasing the dimension's bit size. There are two ways to increase the bit size: through the configuration and patching process or through the dimensionMgr utility. (For more information about dimensionMgr, see "Setting Properties for Dimensions Using dimensionMgr").

Changing the Bit Size

To change the bit size after a domain has been built or upgraded to 13.3, perform one of the following sets of steps:

1. Update the configuration and patch the domain. See "Upgrading and Patching Domains" for instructions. To change the bit size, see the "Defining Dimension Properties" in the *RPAS Configuration Tools User Guide*.
2. Run the `reindexDomain` utility. See "Reindexing Domains Using reindexDomain" for instructions.

Or:

1. Run the `dimensionMgr` utility to update the bit size. See Setting Properties for Dimensions Using dimensionMgr for instructions.
2. Run the `reindexDomain` utility. See "Reindexing Domains Using reindexDomain" for instructions.

Reindexing Domains Using reindexDomain

Use the reindexDomain utility to compress, increase, or decrease the set of physical address space (or indexes) of the multidimensional arrays. The process of compressing and defragmenting the physical IDs makes the domain load and run faster.

Compression is achieved by removing the gaps that develop due to hierarchy operations like purge and delete. Gaps essentially block certain physical IDs within an address space and prevents them from being used to store data in the individual dimension arrays. After the gaps are removed, the dimension space of the multidimensional array is recreated.

Increasing or decreasing the multidimensional array's address space is also achieved by increasing or decreasing the bit size of the dimension arrays and then recreating the dimension space of the array with the reindexDomain utility. In addition to defragmenting and increasing and decreasing the number of indexes, the reindexDomain utility also updates any measure array affected. The update is similar to the dimension data because they are reindexed and defragmented. An affected measure is one that has a dimension that was reindexed in its base intersection.

Notes: While working with a global domain, the reindexDomain utility runs from the master domain and spawns subprocesses across subdomains if more than one process is used.

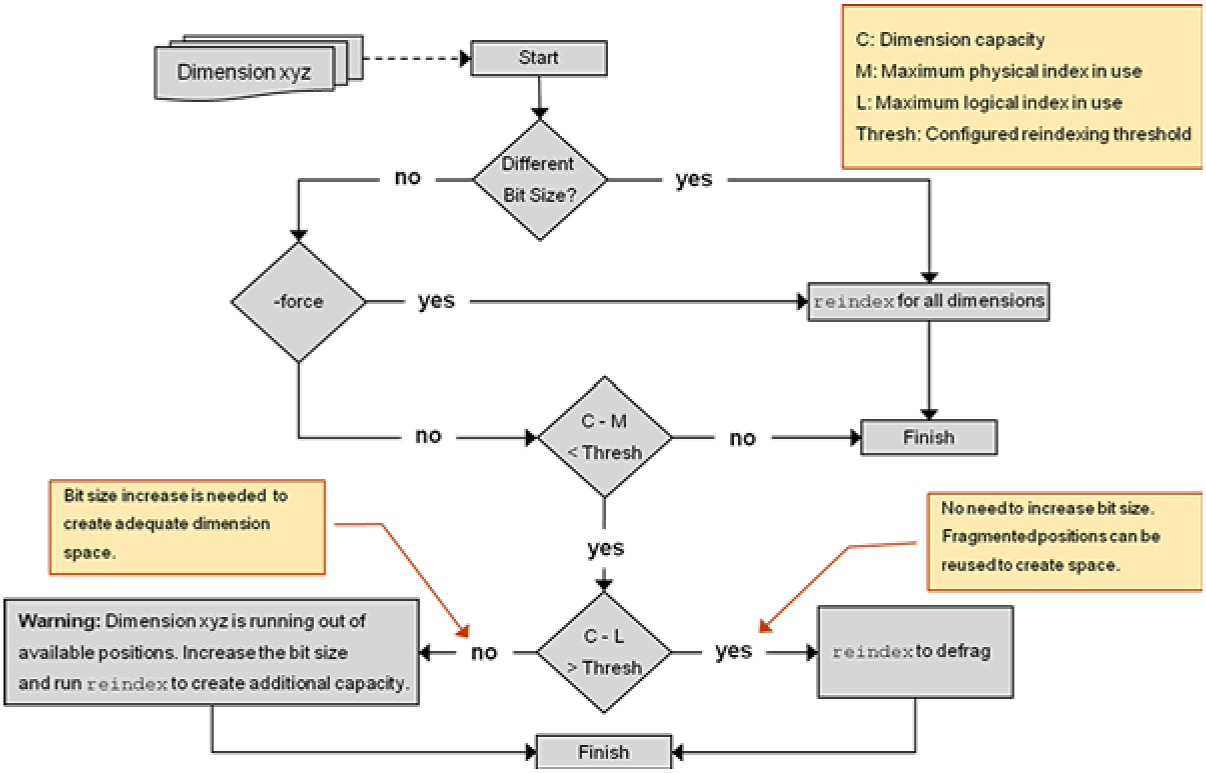
Domain reindexing should not invalidate existing workbooks.

After the reindexing process, the workbook refresh and build operations may run slower. Users who try to open a workbook that was built before the reindexing process should be prompted and informed about the potential slow performance.

A workbook is old if its dimregistry version is older than the domain dimregistry version.

Workbook commit and refresh processes will use external name dimension maps if the workbook is old.

Figure 9–1 Reindexing Domain Process Flow


C: Dimension capacity
M: Maximum physical index in use
L: Maximum logical index in use
Thresh: Configured reindexing threshold
Dimension xyz
Start
Different Bit Size?
no
yes
-force
yes
reindex for all dimensions
no
C - M < Thresh
no
Finish
yes
Bit size increase is needed to create adequate dimension space.
No need to increase bit size. Fragmented positions can be reused to create space.
Warning: Dimension xyz is running out of available positions. Increase the bit size and run reindex to create additional capacity.
no
C - L > Thresh
yes
reindex to defrag
Finish

reindexDomain Usage

```
reindexDomain -d <domainPath> {Commands} {Options}
```

The reindexing options let you reindex the entire domain, dimensions within a hierarchy, a specified list of dimensions, or prepend calendar dimension positions.

Table 9–5 reindexDomain: Reindexing Options

Example	Description
`reindexDomain -d <domainPath>`	Reindexes the entire domain. Checks the reindex condition.
`reindexDomain -d <domainPath> -hier <hierName>`	Reindexes all dimensions of the specified hierarchy.
`reindexDomain -d <domainPath> -dimSpec <Dim1,Dim2,..>`	Reindexes the comma separated list of dimensions.
`reindexDomain -d <domainPath> -hier CLND -prepend`	Creates a new DimRegistry by allocating space to prepend calendar dimensions' positions. The input to this process comes from `clndprepend.xml` file which is generated by loadHier before it throws an exception trying to prepend. After this process, loadHier must be run again to complete prepending the calendar hierarchy.

Two options can be used:

- `-force`: Forces reindexing at the domain/hierarchy/dimension level. It does not check the reindexing condition.
- `-processes max`: Specifies the number of processes to run in parallel.

ReindexDomain Option: Analysis

The analysis option of the reindexDomain utility can be used to analyze the dimensions of a hierarchy or a CSV list of dimensions.

Table 9–6 reindexDomain: Analysis

Example	Description
`reindexDomain -d <domainPath> -analyze -hier <hierName>`	Analyses the dimensions of the hierarchy and prints a report.
`reindexDomain -d <domainPath> -analyze -dimSpec <Dim1, ...>`	Analyses the CSV list of dimensions and prints a report.

The report consists of the following information:

- DimName
- DimRegistry BitSize
- DimInfo BitSize
- Capacity
- Maximum Physical ID
- Maximum Logical ID
- Threshold
- Action Required

ReindexDomain Option: Domain Properties

The reindexDomain utility can be used to review the domain properties by appending the `-prop` argument.

Table 9–7 reindexDomain: Domain Properties

Example	Description
`reindexDomain -d <domainPath> -prop`	Prints the following domain properties: ■ dimregistry_version ■ reindexing_in_progress

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

Table 9–8 reindexDomain Arguments

Argument	Description
-d	Specifies the path to the global or non-partitioned domain. This argument is required for all reindex usages.
-hier	Reindexes all the dimensions of the specified hierarchy. When used with the -analyze argument, the -hier argument specifies which hierarchies should be analyzed.

Table 9–8 (Cont.) reindexDomain Arguments

Argument	Description
-dimSpec	Reindexes the comma-separated list of dimensions. When used with the -analyze argument, the -dimSpec argument specifies which dimensions should be analyzed.
-force	Forces the reindexDomain utility to reindex the entire domain whether it needs it or not. This argument overrides the logic that checks whether reindexing is necessary.
-prepend	Use this argument only when you are prepending the calendar hierarchy. This argument creates a new DimRegistry by allocating space to prepend the calendar dimensions' positions. The input to this process comes from clndprepend.xml file. The loadHier utility generates the clndpreprend.xml file when you load the calendar hierarchy that has positions that have been prepended. After loadHier generates this file, it displays a message, stating that reindexDomain needs to be run first. The clndprepend.xml must exist in the master domain root. After you run the reindexDomain utility, you must run the loadHier utility again to complete the prepending the calendar hierarchy process. You can also generate clndprepend.xml manually without using loadHier.
-processes max	Specifies the number of processes to run in parallel.
-analyze	Checks whether reindexing is necessary for hierarchies or dimensions. This argument returns details about the dimension name, bitsize, number of available and used position IDs, the threshold ratio, and whether reindexing is necessary.
-prop	Displays the domain properties. The properties are dimregistry_version and reindexing_in_progress, where the dimension registry number and the progress indicates whether a reindex is currently running. For more information about these domain properties, see "Version and ReindexStatus".

When to Reindex

To know whether you should reindex a domain, run the `reindexDomain` utility with the `-analyze` option to generate a status report. (Detailed instructions for using the analyze option are described in "Run Reports".) RPAS calculates how much space is left in the array with the following equation:

```
(C-M) < Thresh
```

Where:

- `C` equals the maximum number of position IDs that the array has been allocated to hold. C is the dimension or bitSize cardinality, which is computed with the following equation:

  $C = 2^{BitSize}$

  BitSize is used to compute the cardinality of a dimension. The BitSize can be represented by up to a 4-byte unsigned long integer, which gives a maximum of four gigs of address space. Since four gigs is very large, the BitSize is specified at configuration time. The BitSize is determined based on the current size of the dimension and expected increase in future. If the user does not specify Bit Size for a dimension at configuration time, then the RPAS Configuration Tools provides a default BitSize of 8 for that dimension.

  For domains that were converted to 13.3.0, you must ensure that the hierarchy.xml file contains an entry for all dimensions along with the bit sizes.

- `M` equals the number of used position IDs in the dimension. This is across all position IDs currently assigned to the dimension positions. The maximum number of utilized physical IDs can differ from the maximum number of utilized logical IDs if the hierarchy is subjected to deletions and additions.
- `Thresh` is the threshold value you set for a dimension. The threshold is an integer that represents the number of unused position IDs that you want to be available in the dimension array. When the dimension array no longer has this specified amount of free space, RPAS reindexes the dimension array. The threshold is part of the configuration and is configured as a dimension attribute.

  The threshold can be specified per dimension and can be configured to any initial value. The default value is 10% of the BitSize cardinality for a particular dimension. This value helps determine what approach you need to take:

  - Increase the address space of a dimension after reindexing the dimension arrays
  - Compress the domain arrays without increasing the BitSize
  - Both
- `L` is the maximum utilized logical ID, which is the logical ID value of the last position in a dimension. The domain sequence generator displays this number, which is incremented as new positions are added to the dimension.

For example, if you have a dimension with the following characteristics:

- C: Dimension or BitSize Cardinality for the dimension: 16 (where Bitsize is 4)
- M: Maximum Utilized Position IDs: 9
- Thresh: Threshold: 10
- L: Maximum Utilized Logical ID: 4

  In this case, the calculation produces the following:

```
  (C-M)  < Thresh
(16 - 9) < 10
    7    < 10
```

  Since `7` (the number of available position IDs) is less than `10` (the threshold of required available position IDs), the report returns a a message that states the dimension needs to be defragmented.

  RPAS also checks the `C - L >` or `< Thresh` condition. If `C - L > Thresh`, then the report returns a message that states not to increase the BitSize but to defrag the dimension to recover unused address space. In the example above, `C - L` (16 - 4 =12) is less than the threshold (10). Therefore, the available address space is sufficient after the dimension is defragmented.

  If you have a dimension with the following characteristics:

  - C: Dimension or BitSize Cardinality for the dimension: 16 (where Bitsize is 4)
  - M: Maximum Utilized Position IDs: 9
  - Thresh: Threshold: 10
  - L: Maximum Utilized Logical ID: 7

    In this case, the calculation produces the following:

```
  (C-M)  < Thresh
 (16-9)  < 10
```

```
      7    < 10

    (C-L)  < Thresh
   (16-7)  < 10
      9    < 10
```

Since 9 is less than 10, the report returns a message that states that defragging alone will not help, but that a BitSize increase is required to create additional address space.

Before calculating this equation, RPAS validates the inputs. The main inputs to this utility are the domain path, hierarchy names, and dimension names. Validation logic checks the syntax of the command and verifies all inputs.

Run Reports

Three reports that you can run with the reindexDomain help you determine if you should reindex your domain. Depending on the results of these reports, RPAS may recommend that you need to reindex the dimension, that you need to increase its bit size and then reindex, or that you do not need to either of these because the cost of reindexing outweighs the benefit. To find out which one of these is the case, run the following commands:

Hierarchies and Dimensions This command analyzes one or more dimensions to check whether reindexing is necessary. You can use either the `-hier` argument or the `-dimSpec` argument.

If the `-hier` argument is used, the command analyzes all dimensions in the specified hierarchy to check whether reindexing is necessary:

```
reindexDomain -d [domainPath] -analyze -hier [hierName]
```

If the `-dimSpec` argument is used, the command analyzes all the dimensions in the CSV list to check whether reindexing is necessary:

```
reindexDomain -d [domainPath] -analyze -dimSpec [comma seperated dim names]
```

The command returns the reindexing details. The following is an example of the returned details:

```
Dim Name            :  splr
DimRegistry  BitSize:  8
DimInfo      BitSize:  8
Capacity         (C):  256
MaxPhyId         (M):  3
MaxLogId         (L):  3
Threshold:       (T):  25

Action Required     :  No action required for this dimension at this stage
```

Version and ReindexStatus To see the DimRegistry version and reindex status, run the following argument:

```
reindexDomain -d [domainPath] -prop
```

It returns details about the following two domain properties:

```
dimregistry_version       reindex_in_progress
```

These two domain properties can be set using the `domainprop` utility. But when reindexDomain is run, these two properties are set automatically. Therefore, there is no need to set them separately unless you want to see their values before using the reindexDomain utility.

- dimregistry_version: This domain property stores the current version number of the dim registry. A dim registry defines the version of the indexes. The version number is reset with a new value when reindexing is performed. For instance, if SKU123's index is 15, but reindexing causes that same SKU to be at index 2, then the dimregistry_version will be incremented. The new version is created and stored in the domain at the same level as the previous version. Each version can be identified physically as each is named after its version number. The path up to the version is the same across all versions to enable data to be processed from one version to another.
- reindex_in_progress: This domain property stores the status of the last run reindex process. It is set to TRUE when the reindexing starts and to FALSE when it successfully completes. When the process aborts in the middle, the status remains TRUE to indicate that reindexing needs to be restarted. As long as the status is TRUE, the domain cannot be used for any purpose other than reindexing.

Note: After reindexing process starts, the domain should not be used for any other purpose until the process finishes successfully. If the process aborts in the middle, then the reindexing process must be restarted and completed successfully.

Only an administrator should run this utility.

How to Reindex a Dimension

This section describes the various scenarios and steps needed for reindexing.

Reindex Entire Domain as Needed

This checks whether the reindex condition is met by at least one dimension in the domain. If so, the reindexDomain utility stops further checks and reindexes the entire domain. If the reindex condition is not met by any of the dimensions, the utility exits.

It is recommended that you not reindex the entire domain. Instead, you should reindex on a hierarchy-by-hierarchy or dimension-by-dimension basis.

```
reindexDomain -d <domainPath>
```

Reindex Entire Domain with Force Option

By using the reindexDomain with the `-force` option, you can reindex the entire domain. The `-force` option overrides the logic that checks whether reindexing is necessary and reindexes the entire domain.

```
reindexDomain -d <domainPath> -force
```

Reindex Dimensions

This reindexes one or more dimensions. You can use either the `-hier` argument or the `-dimSpec` argument.

When used with the `-dimSpec` argument, the utility reindexes only those dimensions in the CSV list that meet the reindex condition. If listing more than one dimension, use commas to separate them.

```
reindexDomain -d <domainPath> -dimSpec <comma separated dim names>
```

When used with `-hier` argument, if at least one dimension in the hierarchy meets the reindex condition, the utility reindexes all the dimensions in the specified hierarchy.

```
reindexDomain -d <domainPath> -hier <hierName>
```

Note: You can use the `-force` option with both the `-hier` and `-dimSpec` arguments.

Prepending Calendar Dimensions

This is a unique case in which you need to add calendar positions before the current start of the calendar. This scenario must be handled differently from scenarios in which you add calendar positions at the end of the calendar. This is because the DimRegistry for calendar dimensions must make space at the beginning for the new positions to be added. Here is what happens during the process of prepending positions to calendar dimensions:

1. The loadHier utility detects during its first run that clnd.dat has positions to be prepended. It generates an XML file (clndPrepend.xml) that contains the dimensions to be prepended and the number of new positions to be prepended for each dimension. The loadHier utility then displays an error message stating that the reindexDomain utility must be run first before prepending the calendar dimensions.

   Here is an example of the clndPrepend.xml file:

   ```
   <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
   - <clnd>
   - <prepend>
     <dimension>day</dimension>
     <prependsize>1457</prependsize>
     </prepend>
   - <prepend>
     <dimension>mnth</dimension>
     <prependsize>48</prependsize>
     </prepend>
   - <prepend>
     <dimension>qrtr</dimension>
     <prependsize>16</prependsize>
     </prepend>
   - <prepend>
     <dimension>ssn</dimension>
     <prependsize>8</prependsize>
     </prepend>
   - <prepend>
     <dimension>week</dimension>
     <prependsize>208</prependsize>
     </prepend>
   - <prepend>
     <dimension>year</dimension>
     <prependsize>4</prependsize>
     </prepend>
     </clnd>
   ```

2. The reindexDomain utility is run with syntax added specifically for prepending calendar dimensions:

   ```
   reindexDomain -d <domainPath> -hier CLND -prepend
   ```

3. The reindexDomain domain reads the clndPrepend.xml file and creates a new version of the DimRegistry with space allocated to prepend the calendar dimensions by shifting the physical IDs by an offset equal to the number of new positions. The reindexDomain utility then reindexes all the arrays in the domain that have these calendar dimensions.
4. After running reindexDomain, run the loadHier utility again to complete the process of prepending the calendar dimensions. During both the first and second runs of loadHier, the loadHier utility checks whether calendar dimensions are ready to be prepended or not. The DimRegistry interface is modified to verify this condition.

If you want to avoid using loadHier for generating the clndPrepend.xml file, then you can manually generate the file and place it under the domain root. Ensure that you use the same syntax as the one loadHier generates, as shown in the above example.

Reindex Domain Examples

This is not a comprehensive list of scenarios that can occur, but simply highlights some examples of the reindexDomain functionality.

Condition 1: Change of Bit Size

Table 9–9 Change of Bit Size Scenarios

Scenario	Utility Option	Result
Increase or decrease Bit Size for SKU dimension	`-dimSpec SKU`	Reindexes the SKU dimension and creates a new dimRegistry version.
Increase or decrease Bit Size for SKU and Store dimensions	`-dimSpec SKU,STR,CLSS`	Reindexes the SKU and Store dimensions only. The Class dimension does not meet any reindex condition. A new dimRegistry version is created.
Increase or decrease Bit Size for SKU dimension	`-hier PROD`	Reindexes all dimensions in the product hierarchy and creates a new dimRegistry version. For the SKU dimension, the bit size change is applied to the dimension and measures.
Increase or decrease Bit Size for SKU	`-d`	Reindexes all the dimensions in the entire domain and creates a new dimRegistry version. For the SKU dimension, the bit size change is applied to the dimension and measures.

Condition 2: Using the -force Option

Table 9–10 -force Option Scenarios

Scenario	Utility Option	Result
No bit size change or no fragmentation	`-dimSpec SKU`	Reindexes the SKU dimension and creates a new dimRegistry version.
Bit size change for SKU, fragmentation in CLSS, no fragmentation in STR	`-dimSpec SKU,CLSS, STR`	Reindexes by changing the bit size for the SKU dimension and defragmenting CLSS and STR. Creates a new version of dimRegistry.

Table 9–10 (Cont.) -force Option Scenarios

Scenario	Utility Option	Result
Bit size change for SKU and fragmentation in CLSS and STR	`-hier PROD`	Reindexes all dimensions in the product hierarchy and creates a new dimRegistry version. The SKU dimension has a bit size change applied. STR dimension and location hierarchy are not affected.
Bit size change for SKU, fragmentation in CLSS, no fragmentation in STR	`-d`	Reindexes all the dimensions in the entire domain and creates a new dimRegistry version. For the SKU dimension, the bit size change is applied to the dimension and measures.

Condition 3: Fragmentation Checks

Table 9–11 Fragmentation Check Scenarios

Scenario	Utility Option	Result
Fragmentation in the SKU dimension but does not meet reindex condition. In this case, `C-M>TR`	`-dimSpec SKU`	Nothing happens.
Fragmentation in the SKU dim but does not meet the reindex condition (here `C-M>TR`) and fragmentation in STR dimension that meets the reindex condition (`C-M<TR`)	`-dimSpec SKU,STR`	Only the STR dimension is reindexed. A new dimRegistry version is created.
Fragmentation in the SKU dimension but does not meet reindex condition. In this case, `C-M>TR`	`-hier PROD`	Nothing happens.
Fragmentation in SKU, CLSS and STR but none meet reindex condition. In this case, all three have `C-M>TR`.	`-d`	Nothing happens.
Fragmentation in the SKU dimension, where `C-M<TR` and `C-L>TR`.	`-dimSpec SKU`	Reindexes the SKU dimension and creates a new dimRegistry version. This condition means that there is fragmentation in the SKU dimension over the threshold and defragmenting can generate enough address space to resolve the initial reindex condition of `C-M<TR`.
Fragmentation in the SKU dimension with `C-M<TR` and `C-L>TR` and fragmentation in STR with `C-M>TR`.	`-dimSpec SKU,STR`	Only the SKU dimension is reindexed. A new dimRegistry version is created.
Fragmentation in the CLSS dimension with `C-M<TR`.	`-hier PROD`	All dimensions in the product hierarchy are reindexed. A new dimRegistry version is created.

Table 9–11 (Cont.) Fragmentation Check Scenarios

Scenario	Utility Option	Result
Fragmentation in both the SKU and STR dimensions. Both dimensions have `C-M<TR` and `C-L>TR`.	`-hier PROD`	All dimensions in the product hierarchy are reindexed. Nothing happens in the location hierarchy or to the STR dimension. A new dimRegistry version is created.
Fragmentation in the STR dimension with `C-M<TR` and `C-L>TR`.	`-d`	All dimensions in the entire domain are reindexed. A new dimRegistry version is created.
Fragmentation in the SKU dimension with `C-M<TR` and `C-L<TR`.	`-dimSpec SKU`	No reindexing of SKU takes place as it does not resolve the issue since a bit size increase is needed. A message is provided through the utility indicating a bit size increase is required.
Fragmentation in the SKU dimension with `C-M<TR` and `C-L<TR` and the STR dimension with no fragmentation	`-dimSpec SKU,STR`	No reindexing takes place on either SKU or STR. SKU requires a bit size increase. A message is provided through the utility indicating a bit size increase for SKU is required.
Fragmentation of all dimensions in the product hierarchy where `C-M<TR` and `C-L<TR`.	`-hier PROD`	No reindexing, but a message is provided through the utility, stating that all dimensions in the product hierarchy need an increase in bit size.
Fragmentation for the SKU and STR dimension where both have `C-M<TR` and `C-L<TR`.	`-d`	No reindexing, but a message is provided through the utility stating that the SKU and STR dimensions need an increase in bit size.

Optimizing Domains Using optimizeDomain

Because the RPAS Btree dimension arrays undergo continuous updates and changes as a result of adding and deleting existing positions, over time, measure arrays become full of stale data. This stale data is created when positions are deleted, but the associated index remains. This results in wasted space and inefficient operations. The measure arrays containing any data for these positions should be updated to reflect these deletions. Deleting positions marks the hierarchy data for the corresponding dimensions as changed, but it does not clean up the associated data from the measure arrays. The optimizeDomain utility cleans this stale data from the measure arrays.

When running this utility in a global domain environment, it is centrally administrated. This means it runs over the master domain and spawns parallel processes over the local domains.

For example, consider a measure called `meas1` that exists at intersection `sku_str_week`. When a user adds a DPM position `skudpm1` through an RPAS client, it is assigned to an integer index in the meas1 data array in addition to being assigned within the dimension's array. When this DPM position is deleted, position is flagged as inactive, but the integer index cannot be reused until the dimension is reindexed. The measure array also still contains the data associated with the inactive position. This measure array needs to be updated too in order to remove the stale data.

Note: The reindex process cleans data, but there are times when a reindex is not needed but data needs to be cleaned. That is when optimizeDomain should be run.

The same also holds true for formal positions that are purged during a normal hierarchy patch. The measures arrays need to be updated for this set of positions because they may be pointing to stale data too.

In addition, over time, existing arrays in RPAS databases may become fragmented, resulting in wasted space and a possible degradation in efficiency on array operations. This occurs because the RPAS Btree array stores data in chunks of disk memory called pages. Deleting data from the array causes empty spaces or holes to develop in these pages. These holes contain no data, yet they still increase the overall size of the array. Over time, these holes become larger and more frequent, and array operations suffer degraded performance as a result. Also, the pages begin to acquire large amounts of unused space, causing an inefficient disk usage and a larger than optimal domain size.

Notes: While reclassifying a position from one subdomain to another, data associated with the reclassified positions is moved to the new subdomain. But it is also cleaned from the source subdomain. Therefore, the reclass operation in loadHier does not cause stale data.

No matter what option is specified, when an array is picked to be processed by optimizeDomain, that array is both cleaned and defragmented after a successful run.

The `optimizeDomain` utility is used to correct these two issues. It is recommended after a full hierarchy purge that you run the `optimizeDomain` utility with the `-cleanOnly` argument to remove all stale data from the domain.

optimizeDomain Usage

```
optimizeDomain -d [domainpath] [options]
```

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

Table 9–12 optimizeDomain Usage

Argument	Description
`-d domainpath`	Defines the path to the domain that you want to optimize. If no options are specified, by default optimizeDomain defrags and cleans the entire measure set that have arrays that have a BTree density ratio lower than the threshold (80%). A BTree density ratio is the ratio of the populated positions in the array to the total available BTree pages in the array. The density threshold value (80%) is not configured.
`-defragOnly`	Selectively defrags the domain data based on database fragmentation. **Note**: `-cleanOnly` and `-defragOnly` cannot be specified together. To achieve that behavior, run the utility without any options.
`-cleanOnly`	Cleans the stale data in the domain. **Note**: `-cleanOnly` and `-defragOnly` cannot be specified together. To achieve that behavior, run the utility without any options.

Table 9–12 (Cont.) optimizeDomain Usage

Argument	Description
`-force`	Forces the utility to defrag the entire measure arrays set. This should be used in combination with the `-defragOnly` option. This option makes the defrag process run longer. This option overrides the Btree density threshold (80%) and defrag all measure arrays regardless of their BTree density.
`-excludeMeasList`	Defines the path to exclude list XML file. Optimize performance by excluding this list of measures while defragging or cleaning. This option cannot be used with `-includeMeasList`.
`-includeMeasList`:	Defines the path to include list XML file. Optimize performance by including only this list of measures while defragging or cleaning. This option cannot be used with `-excludeMeasList`.
`-returnFailure`	Returns a non-zero value if any of the databases or arrays fail to defrag or clean. The optimizeDomain process still runs to completion. The default behavior is to log a warning and continue.
`-processes max`	Defines the maximum number of processes to run in parallel.

Examples

For the following measures list and corresponding BTree density ratios, the SKU dimension has some positions that were purged from the domain and marked for cleaning.

Table 9–13 SKU Positions Purged

	Meas1	Meas2	Meas3	Meas4
Base Intersection	SKU	SKU	STR	STR
Density Ratio	60%	90%	60%	90%

Here is the list of measures that are processed based on the command specified.

Table 9–14 Processed Measures

	Meas1	Meas2	Meas3	Meas4
No specified options	X	X	X	
-includeMeas List Meas1, Meas3, Meas4	X		X	
-excludeMeasList Meas1,Meas3		X		
-defragOnly	X		X	
-cleanOnly	X	X		
-defragOnly -force or -force	X	X	X	X
-defragOnly -includeMeasList Meas1,Meas4	X			
-defragOnly -excludeMeasList Meas1			X	

Table 9–14 (Cont.) Processed Measures

	Meas1	Meas2	Meas3	Meas4
-cleanOnly -includeMeasList Meas1,Meas4	X			
-cleanOnly - excludeMeasList Meas1		X		

Adding New Dimensions to Hierarchies

Using the RPAS Configuration Tools and RPAS utilities, you can add new dimensions to hierarchies in existing domains. This process is summarized in this section and described in greater detail in the following sections.

Caution: Because of its invasive impact, adding new dimensions is different from normal domain patch process. It is a multi-step process and requires careful user intervention.

1. To add new dimensions to an existing hierarchy, use the Configuration Tools to open and modify the existing configuration so that the new configuration includes the additional dimensions.

   Notes: This process can be used only for non-partitioned hierarchies. It cannot be used to delete dimensions.

   After the configuration has been modified, use the Reports Generator in the Configuration Tools to generate a hierarchy.xml report. For more information about creating this report, see the "Report Generator" section in the *RPAS Configuration Tools User Guide*.
2. Export measure data for all measures that have one of the dimensions of the modifying hierarchies. See "Exporting Measure Data".
3. Export all hierarchy data from the hierarchy, including all DPM positions and user-defined dimension rollup information. Repeat this step for each hierarchy being modified. See "Exporting Hierarchy Data".
4. Purge all positions from the hierarchies. See "Purging Hierarchy Data".
5. Use the hierarchyMgr utility with the new hierarchy.xml to add the new dimensions. Repeat this step for each hierarchy being modified. See "Adding New Dimensions to Hierarchies".
6. Create a new data file, or update the old one, so that it contain positions for the newly added dimensions. Load it back to the domain. If the original hierarchy contains user-defined dimensions, the format of the data file must be in CSV format with a header line indicating all user-defined dimension position and label columns and using the `-includeUdd` switch. Perform one execution for each modified hierarchy. See "Reloading Formal Hierarchy Data".
7. Optional: Informalize previous DPM positions. See "Informalizing DPM Positions".
8. Reload measure data to the domain. See "Reloading Measure Data".

Exporting Measure Data

Use the exportMeasure utility to export all measures for hierarchies. It exports only measures that have storage in the domain (db property).

```
exportMeasure -d [domain path] -hier [hierarchy1, hierarchy2] -outDir
[outputDirectory]
```

To enter multiple hierarchies, separate them with commas, for example, `-hier loc,clnd`. This exports all hierarchy measures that have storage in the domain.

If the output directory that you specify does not exist, it will be created. One output file is created for each HBI (higher based intersection) measure. (In a global domain, the HBI measures are stored in the master domain because the measure's base intersection is above the partition dimension. Forced non-HBI measures are measures that should be an HBI measure and stored in the master domain, but the RPAS application has forced these measures to be stored across the local domains.) In addition, one file is created for each non-HBI or FnHBI measure per subdomain. The file names contain an internal subdomain index, for example, `sales.0.csv.rpl`, `sales.1.csv.rpl`, and so on.

For more information about exportMeasure, see "Exporting Measure Data Using exportMeasure".

Exporting Hierarchy Data

Use the exportHier utility to include user-defined dimensions. Export each hierarchy individually.

```
exportHier -d [domain path] -hier [hierarchy] -datFile [datFile] -udd [-
listInformal fileName]
```

Use the `-udd` argument to export the user-defined definitions. The `-udd` argument can only be used with the `-onlyFormal` or `-onlyInformal` options. It cannot be used with the `-fixedWidth` option. User-defined dimensions can only be exported in CSV format.

If the `-listInformal` argument is used, `then exportHier` also creates a file with the file name. This contains a list of informal positions in the domain in a format that can be used by the informalPositionMgr. Note that this option cannot be used with the `-onlyFormal` option.

For more information about exportHier, see "Exporting Hierarchy Data Using exportHier".

Purging Hierarchy Data

Use the loadHier utility with the -purgeAll argument to purge the hierarchy data. The `-purgeAll` argument purges formal, informal, and user-defined positions in the listed hierarchies. It cannot be used on a partition hierarchy or any system hierarchy.

```
loadHier -d [domain path] -purgeAll [hierarchy1,hierarchy2]
```

Use the -purgeAll argument to purge hierarchies. Specify multiple hierarchies by using CSV format (-purgeAll loc,clnd). For more information about loadHier, see "Loading Hierarchies Using loadHier".

Adding New Dimensions

Use the hierarchyMgr utility to add new dimensions. The hierarchyMgr utility parses the hierarchy.xml file and determines where to add the new dimensions. All start and width properties of all dimensions in the hierarchy are refreshed to be consistent with the new hierarchy.xml.

```
hierarchyMgr -d [domain path] -h [hierName] -addLevels hierarchy.xml
```

One or more dimensions can be added to any existing hierarchy except for partitioned or system hierarchies. Only one hierarchy can be parsed at a time. The original meta, hmaint, and language databases are backed up and can be restored if a failure occurs.

Notes:

- You cannot add or insert dimensions above or below user-defined dimensions. The hierarchy.xml file does not include user-defined dimensions.
- `Remove` and `move` are not allowed.
- This process is used for inserting formal dimensions into the hierarchy. It is not used for adding user-defined dimensions.

Reloading Formal Hierarchy Data

Use `loadHier` to load the hierarchy that was purged. The user-defined dimensions included in the input hierarchy file are also loaded.

```
loadHier -d [domain path] -load [hierarchy] [-includeUdd]
```

or

```
loadHier -d [domain path] -loadAll [-includeUdd]
```

Use the `-load` argument to load one hierarchy at a time. Use the `-loadAll` argument to load all hierarchies that have an input file in the input directory of the domain.

Use the `-includeUdd` argument to load user-defined positions back to the domain. The data file must be in CSV format with a header line. The name of the data file should follow the current standard: <hierarchy name>.csv.dat.

All user-defined dimensions must be in the data file. Any missing user-defined dimensions cause an error. All loaded positions are in formal status. See the "Informalizing DPM Positions" section for information on how to convert the status of previous DPM positions to informal.

For more information about loadHier, see "Loading Hierarchies Using loadHier".

Informalizing DPM Positions

Use `informalPositionMgr` to informalize DPM positions.

```
informalPositionMgr -d domainPath -hier hierName -operation informalize -file
inputFile
```

Use the `-file` argument to enter the input file to be processed. The inputFile is the list file exported by `exportHier` with `-listInformal`.

For more information, see "Informal Position Manager".

Reloading Measure Data

Use `loadmeasure` to load all measure files located in the specified input directory.

```
loadmeasure -d [domain path] -inDir [inputDirectory]
```

Only .rpl files can be used with this option, and only CSV format with header line is supported. The exported files have one measure per file.

For more information, see "Loading Hierarchies Using loadHier".

Adding New Hierarchy Dimensions Sample Script

Below is an example of a script that adds new hierarchy dimensions.

```
#!/bin/ksh
# -----------------------------------------------------------------------------
#
# This is a sample script demostrates different steps to add levels to an
# existing domain
#
# -----------------------------------------------------------------------------
DOMPATH=$TEST_GLOBAL_DOMAIN

# Need to be full path.
MEAS_STORAGE_DIR="C:/tak/patchHierScript/measdata"
OLD_HIER_DATA_DIR="c:/tak/patchHierScript/oldhier"
NEW_HIER_DATA_DIR="c:/tak/patchHierScript/newhier"
# can be comma separated list. If multiple is intended, exportHier, hierarchyMgr,
# and loadHier needs to be called in a loop.
# ex. HIERS="loc,clnd"
HIERS="loc"

mkdir -p $MEAS_STORAGE_DIR
mkdir -p $OLD_HIER_DATA_DIR

# Inside NEW_HIER_DATA_DIR, there should be new hierarchy xml file and new hier
data
# file.

IFS=","

# -----------------------------------------------------------------------------
# Step one, export all positions.
# -----------------------------------------------------------------------------
exportMeasure -d $DOMPATH -hier $HIERS -outDir $MEAS_STORAGE_DIR
if [ $? -ne 0 ]; then
   echo "exportMeasure for exporing $HIERS hierarchy"
   exit 1
fi

# -----------------------------------------------------------------------------
# Step two, export all hierarchy data and informal position list. One hierarchy
# at a time.
# -----------------------------------------------------------------------------
for h in "$HIERS"; do
   exportHier -d $DOMPATH -hier $h -datFile $OLD_HIER_DATA_DIR/$h.csv.dat -udd -
listInformal $OLD_HIER_DATA_DIR/$h.informal.lst
   if [ $? -ne 0 ]; then
      echo "exportHier failed for $h"
      exit 2
   fi
done

# ---------------------------------------------------------------------------
# Step 3, purge the hierarchies.
# ---------------------------------------------------------------------------
loadHier -d $DOMPATH -purgeAll $HIERS
if [ $? -ne 0 ]; then
   echo "Unable to purge $HIERS."
   exit 3
fi

# BEGIN MODIFICATION and DATA LOADING.

# ---------------------------------------------------------------------------
# Step 4, patch the hierarchy (add levels) one hierarchy at a time.
# ---------------------------------------------------------------------------
for h in $HIERS; do
   hierarchyMgr -d $DOMPATH -h $h -addLevels $NEW_HIER_DATA_DIR/hierarchy.xml
   if [ $? -ne 0 ]; then
      echo "hierarchMgr failed for $h"
      exit 4
   fi
done
# --------------------------------------------------------------------------
# Step 5, Load the hierarchy data back. The structure reflect the
# new hierarchy structure.
#
# NOTE 1:
# Data should be updated to include positions of the newly added dimensions
#
# NOTE 2:
# Even though reclass is supported by loadHier, measure data is going
# to be loaded assuming old position is still in the original subdomain.
# Position reclassification is not recommended and may cause measure data lost.
# --------------------------------------------------------------------------
for h in $HIERS; do
   # copy the data file to input directory of the domain
   cp $NEW_HIER_DATA_DIR/$h.csv.dat $DOMPATH/input/

   loadHier -d $DOMPATH -load $h -includeUdd
   if [ $? -ne 0 ]; then
      echo "loadHier failed while loading $h hierarchy"
      exit 5
   fi
done
# --------------------------------------------------------------------------
# Step 6, Set the previously DPM position back to DPM
# --------------------------------------------------------------------------
for h in $HIERS; do
   informalPositionMgr -d $DOMPATH -hier $h -file $OLD_HIER_DATA_DIR/
$h.informal.lst -operation informalize
   if [ $? -ne 0 ]; then
      echo "informalPositionMgr failed while informalizing positions in $h
hierarchy"
      exit 6
   fi
done
#
# --------------------------------------------------------------------------
# Step 7, load measure data that exported with exportMeasure back to the
# domain.
# ----------------------------------------------------------------------------
loadmeasure -d $DOMPATH -inDir $MEAS_STORAGE_DIR
if [ $? -ne 0 ]; then
   echo "Unable to purge load measure data in $MEAS_STORAGE_DIR. Please check the
logs in output directory"
   exit 7
fi

# Completed
```

Transferring Data

The transferData utility allows you to load measure data into one domain by using a second domain as the data source. While this can be achieved with the exportMeasure and loadmeasure utilities, the `transferData` utility is more efficient since it uses a single process to load a data file into the destination domain.

The transferData utility works by copying the source measure data to the destination measure data along any matching external position names between the source and destination measures. This includes any informal positions that have matching names in the destination. You can choose to exclude informal positions’ data from being copied to the destination if desired.

Note: You cannot use the transferData utility for forced non-HBI (higher based intersection) measures. Forced non-HBI measures are measures that should be HBI measures and stored in the master domain, but the RPAS application has forced these measures to be stored across the local domains. For these measures, you must transfer data using the export and import utilities described in "Adding New Dimensions to Hierarchies".

The measure names do not need to be identical in both domains. However, the source and destination measures must conform in intersection and data type.

The measures to be transferred are identified with the `-measMap` argument. The format of the argument is a colon separated list of source measure name, destination measure name, and an optional mask measure name:

```
src1:dest1[:mask1]
```

Multiple measure groups can be specified in groups if separated by a comma:

```
src1:dest1[:mask1],src2:dest2[:mask2]
```

For a complete example, see "transferData Usage".

The transferData utility can run in parallel when the destination domain is a global domain. The `-processes` argument determines how many child processes to spawn to perform the processing. This makes the transferData task faster when used in a global domain destination.

The transferData utility validates several components before transferring data. It validates that the source and destination domains exist, that the source and destination measures exist, and that the source and destination measures have conforming intersections and data types.

After it has validated everything, the transferData utility opens the source measure in read mode and opens the destination measure in write mode; this prevents data integrity issues. It iterates over positions in the source measure, pulls the values, and writes them to the destination measure.

Masks

You can use a mask measure to restrict the transfer of a range of positions from the source measure. This mask measure is a boolean measure, located in the source domain, that is at the same intersection of the source/destination measure pair or higher than that intersection.

Note: The mask measure should have an NA value of FALSE for optimal performance. The transferData utility will stop and inform the user if this is not the case.

In addition, you can use the `-clearDest` flag with or without a mask option. These combinations are described below.

No Mask with -clearDest Clears the destination arrays. Changes the destination array's NA value to the source measure's NA value at the beginning of the process. Loops over the source arrays and populates the destination.

No Mask without -clearDest Does not clear the destination arrays. Does not change the destination array's NA value. Loops over the source arrays and copies to the corresponding destination.

Mask with -clearDest Does not clear the destination arrays. Does not change the destination array's NA value. Loops over the mask and copies the source values (populated or unpopulated) to the corresponding destination cells.

Mask without -clearDest Does not clear the destination arrays. Does not change the destination array NA value. Loops over the mask and copies the populated source values to the corresponding destination cells. It does not copy unpopulated source values.

Table 9–15 illustrates the results of the mask and flag combinations.

Table 9–15 Mask and Flag Combinations and Outcomes

	No Mask, -clearDest Flag	No Mask, No Flag	Mask, -clearDest Flag	Mask, No Flag
Clears destination arrays	Yes	No	No	No
Changes the destination array's NA value to the source measure's NA value	Yes	No	No	No
Populates the destination arrays	Yes	Yes	Yes, populated and unpopulated source values	Yes, only populated source values

transferData Usage

```
transferdata -d destDomain -src srcDomain -measMap
"src1:dest1[:mask1],src2:dest2[:mask2]" {-processes n} {-clearDest} {-noInformal}
```

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

Table 9–16 transferData Utility Arguments

Argument	Description
-d pathToDestDomain	Specifies the path to the domain containing the destination measure. This is required.
-src pathToSrcDomain	Use this argument to specify the path to the domain containing the source measure. This is required.
-measMap	Enters a comma-separated list of source to destination to mask mappings (src:dest:[:mask]), where src is the name of the measure to be copied from the source domain, dest is the name of the measure to be written to in the destination domain, and mask is the name of the mask measure to be used, if any. A one-to-one relationship must exist between source measures and destination measures.
-processes max	Defines the maximum number of processes to run in parallel in a global domain setup.
-clearDest	When used with a mask option, this argument ensures that the destination measures array is not cleared and its NA value is not changed. The utility copies the populated and unpopulated data from the source measure positions that are populated in the mask. When used without a mask option, the transferData utility clears the destination arrays and changes the destination array's NA value to the source NA value at the beginning of the process. After that, the populated measure positions from the source array.
-noInformal	Ensures only formal positions that have a matching external name in the destination measure are copied over.

Scenarios

The following sections describe the four scenarios for using the transferData utility.

Simple Domain to Simple Domain When the source and destination domains are simple domains, the data is copied from the source domain to the destination domain.

Simple Domain to Global Domain When the source is a simple domain and the destination is a global domain, and if the destination measure is a non-HBI measure, the data is copied to the local domains of the destination. HBI measure data is copied to the master domain. Since the destination domain is a global domain, the data is transferred in parallel.

Global Domain to Simple Domain When the source is a global domain, for non-HBI measures, the subdomain data array is copied to the destination. HBI measure data is copied from the master domain to the destination domain.

Global Domain to Global Domain Since the destination domain is a global domain, the data is transferred in parallel. However, each subprocess visits each source subdomain when transferring a non-HBI measure. This means that you do not have to run transferData over each local domain, but only on the master domain once.

Global-to-global transferring follows this process:

1. The transferData utility maps the source's external position names to the destination's external position names.
2. The utility scans the source.
3. The utility writes the values into destination measure.

The transferData utility always runs in parallel when the destination domain is a global domain and the `-processes` option is set to a number greater than 1. When the source domain is a global domain and the source measure is a non-HBI measure, every subprocess running on each local domain of the destination domain visits every local domain on the source, even when the source local domain has no common position with that particular destination local domain. However, if there is no common position, the subprocesses visits should occur quickly.

FilterHier Utility

Sometimes, a retailer has a master file of hierarchy data that must be loaded into multiple domains. Some of these domains may be missing one or more levels from the master hierarchy, primarily because the planning levels in these domains are higher than the lowest level in the master and the domains do not need to have all the lower levels. For example, a retailer may have one domain for Merchandise Financial Planning, where the lowest level is Category, and another for Item Planning, where the lowest level is Item. The hierarchies in these two domains have their relevant hierarchy load data in one master file, but using loadhier, the retailer cannot load only what is relevant to the domain from the master. System integrators need to write custom scripts to parse out irrelevant columns from the master file to prepare load files suited for individual domains.

The filterHier utility does the filtering of columns for the system integrators, eliminating the requirement for custom scripts. The utility analyzes the target domain and trims down the master file to only have those columns that are needed by the target domain. The utility acts on CSV formatted files and requires the input file to contain a header line containing the names of the columns, for example, *SKU,SKU_label,STCO,STCO_label*. The output of the utility is file is a .csv.dat file that can be subsequently used by the loadHier utility.

filterHier Usage

```
filterHier -d domainPath -input inputPath [COMMAND] {OPTIONS}
```

Table 9–17 provides descriptions of the arguments used by the filterHier utility.

Table 9–17 Arguments Used by the filterHier Utility

Argument	Description
-d domainPath	Identifies the domain in which to load the hierarchy data.
-input inputpath	Identifies the path to the folder where the master files are located.
-filter hiername	Filters the hierarchy named in the parameter to the command.
-filterAll	Filters all hierarchies in the input directory that are relevant to the target domain.

Table 9–17 (Cont.) Arguments Used by the filterHier Utility

Argument	Description
-compress	Creates a compressed .csv.dat output file. RPAS provides a simple, proprietary compression technique to help reduce the file size and file I/O time during loads. This technique simply replaces a column's value with a '?' character to indicate that the column's value for the row matches that of the row above. The compressed file continues to print out '?' characters for a column until a change is encountered. This kind of compression is useful for hierarchy files where the lowest level positions are grouped by the higher level positions. In such cases, the output file will print out '?' characters for higher level positions until a change is encountered, thus significantly reducing the file size. Note that compressed files should not be split up or reprocessed in ways that change the order of rows.

Notes

This utility combines one or more input files into an output file that can be imported into the target domain using loadHier. The input files must contain csv data with a comma separated header line listing the name of each column. Column names must be in the format DIM,DIM_label.

Example: SKU,SKU_label,STCO,STCO_label,SIZ1,SIZ1_label

The input files must have the extension .hdr.csv.dat. Optional extensions are allowed at the end of the file.

Example: prod.hdr.csv.dat.foo1

The columns in the output file are arranged to match the hierarchy format of the target domain. Any dimensions from the input files that are not present in the output domain are filtered out.

The output file is a properly formatted .csv.dat file and is in the input directory of the target domain.

Error Conditions

The following error conditions may occur during the operation of filterHier:

- Dimension Not Found: If a dimension that exists in the domain is not found in the header, the input file will be skipped.
- No Usable Input Files: If filterHier cannot find a usable input file, it will exit with error.
- Parse Error in Data: If one or more data rows contain an error, filterHier will display an error specifying which lines contain an error, and proceed processing the rest of the file.
- Conflicting Base Positions: If a base position has multiple definitions in the input files, the first definition will be used and all others will be skipped.

Position Repartitioning

Position repartitioning is the automated process for moving positions and all corresponding measure data between local domains. This functionality is only available (and relevant) in global domain environments. Positions must 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, 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 a reclassification by RPAS customers. RPAS refers to this functionality as position repartitioning because it technically does not handle the many complex functional requirements of a true reclassification.

Loading RDF and Curve Parameters after Repartitioning

After repartitioning, default parameters for Curve and RDF are not automatically loaded in new subdomains. To load these parameters, the following scripts, which are located in the RPAS_HOME/bin directory, must 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 the default required method, default source, spreading profile, and others.

You must 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/
```

Reconfiguring the Partitions of a Global Domain Using 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 and 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:

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

Runs loadHier to apply the position addition or 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 must run the utility reconfigGlobalDomainPartitions by specifying the subdomain to which the positions 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.

---

reconfigGlobalDomainPartitions 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.

Table 9–18 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.

Table 9–18 (Cont.) Arguments Used by the reconfigGlobalDomainPartitions Utility

Argument	Description
-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 is 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.
-maxProcesses *count*	If specified, some parts of reconfig utility will run in parallel, using up to the given number of processes.
-forceInputRollups	Prevents 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 that 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.

Here 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.

Table 9–19 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 added or moved.
sample_pos	One or more positions that are being added or moved to the designated local domains.

Notes, Assumptions, and Limitations

Note the following:

- Position names are separated by commas and must be valid external position names without the prefix of a dimension (for example, 0102,0144,0152,0160).
- The utility backs up the required data and automatically restores 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, 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.

Renaming Positions Using renamePositions

The name of a position can be changed using an RPAS utility named `renamePositions`. Positions that are to be renamed should be included in a hierarchy data file. After the hierarchy data files have been updated and placed in the designated location, an administrator must run the `renamePositions` utility.

Note: The renamePositions utility cannot be used to rename any RPAS system objects such as dimensions, workbooks, hierarchies, and so on.

---

renamePositions Usage

```
renamePositions -d domainPath  -hier hierName {-input inputDirectory} {-log
logFileName} {-dryRun}
```

The `renamePositions` utility renames positions in a specified hierarchy. It looks for a file named `hierName.rn.dat` (for example, `prod.rn.dat`) in the domain's input directory when `-input` is not specified.

The input file is a CSV text file with three columns. Tab-delimited fields are not supported in the input file. This is consistent with other domain utilities in which tab-delimited fields are not supported. One line is required for each rename request. Each line consists of a dimension name, the old position name, and a new position name. For example:

```
sku,10000_old,10000_new
```

The utility ignores any line not incorrectly formatted, empty lines, and lines with a specified dimension that does not exist for the specified hierarchy.

The input file is not case sensitive, and all names are converted to lower case before the positions are renamed. Old position names that are specified but that do not actually exist are ignored. New position names cannot be an already used name. Lines with invalid position names are ignored and added to the log file. The old position name and new position name should not be prefixed with the name of the dimension.

The width property of the dimension is enforced. The width attribute in the domain must be greater than or equal to the maximum length of the new names in the input file. If the width of the new name is greater than the width attribute of the corresponding dimension, RPAS prints an error in the log file and ignores the record.

Dimensions specified in the input file must belong to the hierarchy that is specified in arguments. Otherwise, the record is ignored and RPAS prints an error in the log file.

Note: If you want to swap position names (a->b, b->a), you must perform two separate operations: (a->tempa, b->tempb) and (tempa->b, tempb->a).

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

Table 9–20 Arguments Used by the renamePositions Utility

Argument	Description
-d *domainPath*	Specifies the full path to the domain.
-input *inputDirectory*	Specifies the input directory where the input file with positions to rename is located. This is optional. The utility looks for hierarchy data files with "rn" between the hierarchy name and .dat extension (for instance, prod.rn.dat).
-hier *hierName*	Indicates the hierarchy for which positions are being renamed.

Table 9–20 (Cont.) Arguments Used by the renamePositions Utility

Argument	Description
-log *logFileName*	Generates a log file with a file name other than default. The default file name is hierRename.log. This argument can be used to name the log file other than the default (created as hierRename.log in the current directory). Optional.
-dryRun	Does not apply changes, but generates a report that identifies changes that can be applied. This argument generates a log file.

Setting Properties for Dimensions Using dimensionMgr

The dimension manager utility is used for setting a number of parameters for dimensions and positions. Multiple command arguments are allowed.

Changing the Label of a Dimension

To change the label of a dimension, use the `-label` argument with the `dimensionMgr` utility. It is not restricted to partitioned hierarchies.

```
dimensionMgr -d domainPath -dim dimName -label newLabel
```

dimensionMgr Usage

```
dimensionMgr -d pathToDomain -dim dimensionName [COMMAND] {Options}
```

Table 9–21 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.

Table 9–21 dimensionMgr Utility Arguments

Argument	Description
-d *pathToDomain*	Specifies the path to the domain.
-dim *dimensionName*	Specifies the name of the dimension to which the settings apply.
`-label` *newLabel*	Specifies a new dimension label.
`-enableDPM`	Enables Dynamic Position Maintenance (DPM) for the specified dimension (-dim dimensionName). 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*	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.

Table 9–21 (Cont.) dimensionMgr Utility Arguments

Argument	Description
`-width` *widthVal*	Sets the width of position names for the specified dimension. After you have created a position name, you can only increase its width; you cannot decrease the width.
bitSize	Changes bit size for the dimension, but does not actually update the dimension arrays. You must run reindexDomain for the dimension data and measure arrays to be updated.
reindexThreshold	Sets a new reindex threshold for the dimension.
`-specs`	Displays the properties of the specified dimension. The dimension properties indicate whether DPM and translation are enabled for the dimension and whether the dimension is image-enabled. Below is the full list of properties that the -specs argument returns: Dimension Label Associated Hierarchy of Dimension Start and Width for File Load Label Start and Width for File Load Number of Used Positions Image Enabled Bit Size Capacity Used Reindex Threshold Translation Enabled Spreads or Aggs (meaning, where the dimension fits in the rollup) If the dimension is at the base level of a hierarchy, that is indicated as well.

Exporting Hierarchy Data Using exportHier

The exportHier utility is used to export all the positions in a hierarchy, including their rollup relations. The exportHier utility cannot be run on local domains. It must be run only on the master domain or simple domains. By default, the utility assumes that the file has a CSV flat file format with fixed width format as an optional argument. The utility exports all hierarchy positions, but the file may be specified to include only formal or informal positions. The resulting file can then be used as a .DAT file with loadHier.

exportHier Usage

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

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

Table 9–22 Arguments Used by the exportHier Utility

Argument	Description
`-d` *pathToDomain*	Specifies the path to the domain.
`-hier` *hier_name*	Specifies the name of a hierarchy in the domain from which the .DAT file is generated.
`-datFile` *dat_File*	Specifies the path where the .DAT file is created. This .DAT file can then be used with loadHier to load the hierarchy into a domain.
`-fixedWidth`	Indicates that the output .DAT file is a fixed-width file instead of a comma-separated value (CSV) file format.
`-onlyFormal`	Exports only formal positions to the .DAT file. If this is option is specified, informal positions will be skipped.
`-onlyInformal`	Exports only informal positions to the .DAT file. If this is option is specified, formal positions will be skipped.
`-genheader`	Generates a header line for CSV export. The line contains fields to identify the dimension and its label column.
`-udd`	Exports the user-defined definitions. It can only be used with the `-onlyFormal` or `-onlyInformal` options. It cannot be used with `-fixedWidth` option. User-defined dimensions can only be exported in CSV format.
`-listformal`	Creates a file with the fileName, which contains a list of the informal positions in the domain in a format that can be used by the informalPositionMgr. This option cannot be used with the `-onlyFormal` option.
`-upperCase`	Converts the position names to all uppercase before writing the output data file. Without this argument, position names are in lowercase since they are stored in lowercase in the domain.

Informal Position Manager

The Informal Position Manager is a domain utility that maintains informal positions for any dimension in a domain. Informal positions are those created with the Dynamic Position Manager (DPM). Formal positions are those created with the loadHier utility. This utility can convert positions from formal to informal or from informal to formal. It can also remove informal positions, create informal positions in bulk, and copy data slices between positions in measures.

InformalPositionMgr Usage

The following main areas of use for informalPositionMgr are described in this section:

- Basic Operations: Informalize, Formalize, and Remove
- Creating Informal Positions in Bulk
- Data Slice Copying

Basic Operations: Informalize, Formalize, and Remove

Informalizing, formalizing, and removing informal positions all have similar command line options.

```
informalPositionMgr -d domainPath -hier hierName  -operation [remove | formalize |
informalize] [-dir inputDir | -file inputFile] [-retain]
```

The following table provides descriptions of the arguments used in basic operations of the informalPositionMgr utility.

Table 9–23 Arguments Used by the informalPositionMgr Utility: Basic Operations

Argument	Description
-d *domainPath*	Specifies the domain to run the utility.
-hier *hierName*	Specifies the hierarchy to operate on.
-operation *informalize | formalize | remove*	Specify one of the three basic operations: informalize, formalize, or remove. You can use only one of these operations at a time. ■ `remove`: Removes informal positions of the specified hierarchy ■ `formalize`: Formalizes positions of the specified hierarchy ■ `informalize`: Informalizes positions of the specified hierarchy

-file *inputFile*
or
`-dir` *inputDir*
	Use one and only one of these arguments to specify the input files to use during the operation specified.
	■ `-dir` *inputDirectory*: Processes all applicable files under inputDirectory
	■ `-file` *inputFile*: Processes the inputFile only
	All files that match the naming patterns below are processed:
	■ Remove files: `{hierName}.remove[.extension]`
	■ Formalize files: `{hierName}.formalize[.extension]`
	■ Informalize files: `{hierName}.informalize[.extension]`
	For example: `prod.informalize.20100220`
	The format of the returned content has one position on each line. The position has a dimension name and a position name that are comma delimited. There is no header line:
	`dimName1, positionName1`
	`dimName2, positionName2`

-retain	Use this argument if you do not want to move the input files to the processed subdirectory after a successful run.

Error Handling

The following table explains the errors associated with the InformalPositionMgr.

Table 9–24 informalPositionMgr Error Handling

Error	Behavior
Input file does not exist	A log error is generated and the operation stops.
Any dimension specified in the input file is not DPM enabled or does not exist	A log error is generated and the operation stops.
One or more positions does not exist	Log warnings are generated, the line is skipped, and the operation continues.
Selecting to convert a position to informal that is already informal or Selecting to convert a position to formal that is already formal	No action is taken on that position. A warning message is logged, and the operation continues.

Creating Informal Positions in Bulk

This feature allows user to create a number of informal positions on any DPM-enabled dimension. These positions are automatically named and labeled. However, the user must provide applicable rollup and spread information so that these new positions can be properly placed in the hierarchy.

```
informalPositionMgr -d domainPath -create -dim dimName -n posCount
-rollups dim1:pos1,dim2:pos2,... [-spreads dimA:ratioA,dimB:ratioB,...]
```

For the bulk creation of informal positions, the position names and labels are auto-generated. To ensure that the names are unique, a prefix and sequence number is used. The width and label width attributes of the dimension should be at least 10. If the width or label width is less than 10, bulk creation is not allowed, and an exception is thrown.

If the dimension specified by the option `-dim` is above the root dimension, informal positions are added to the lower levels. The numbers of spread positions for lower level dimensions are defined by the `-spreads` option. Note that the spread ratio determines the number of children for each position at the level immediately above the specified dimension. If not specified, the default is 1.

If these new descendent positions roll up to other alternate dimensions, there are a few options to handle the rollups. If those dimensions are DPM-enabled, you can either specify the rollup positions (in the `-rollups` option) or let the program create a single rollup position for you. This repeats recursively for those new positions. If any of those affected dimensions are not DPM-enabled, the rollup positions must be specified.

Note: To manage the performance impact of bulk creation, a limit is placed on how many positions can be added at one time. The default limit is 5000 per dimension, which can be overridden using the following domain property:

```
MAX_NEW_BULK_DPM_POS
```

If the limit is exceeded, the program stops and no positions are created in the domain.

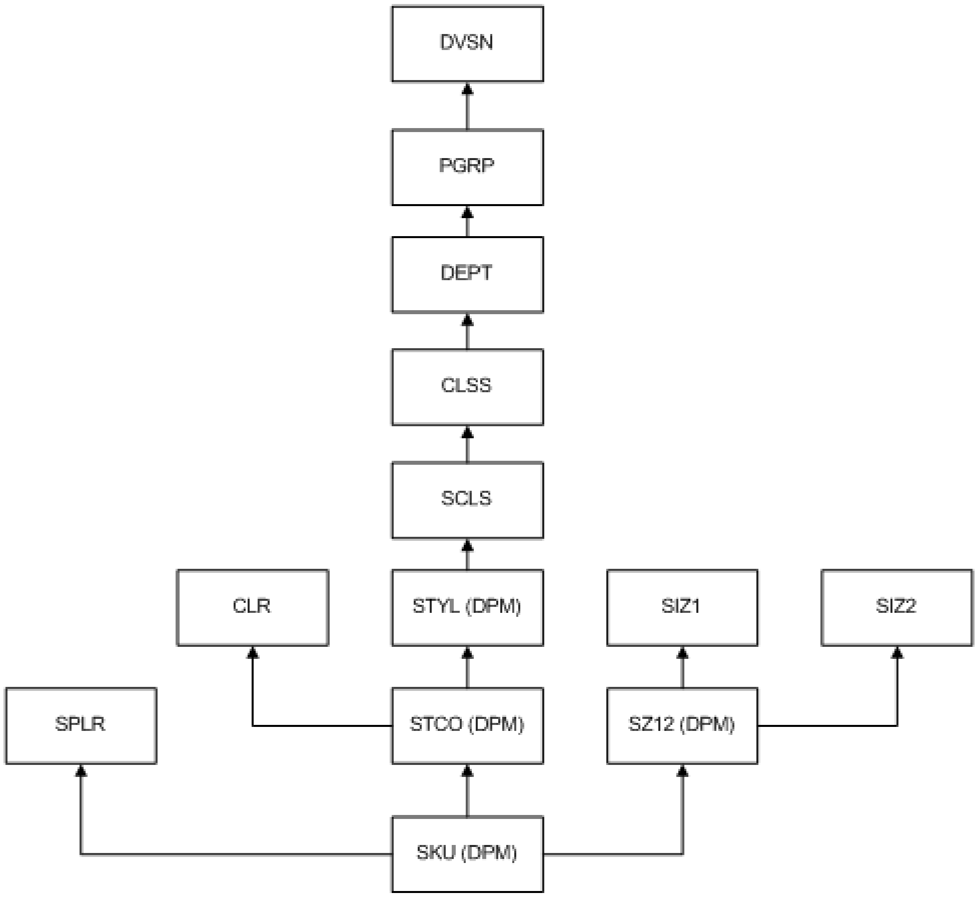
The following table provides descriptions of the arguments used by the create bulk positions functionality of the informalPositionMgr utility.

Table 9–25 Arguments Used by the informalPositionMgr Utility: Create Bulk Positions

Argument	Description
-d *domainPath*	Specifies the domain to run the utility.
-create	Batch creates new positions with auto-generated names.
`-dim` *dimName*	Specifies the dimension to operate on.
`-rollups` *dim1:pos1,dim2:pos2,...*	Specifies the rollup positions for the auto-generated positions. You should include the direct rollups for the new positions and rollups for the descendents of these new positions on alternate branches, if applicable.
`-spreads` *dimA:ratioA,dimB:ratioB,...*	Specifies the spread counts for the auto-generated positions. This is optional. It defaults to 1 if not specified.
`-n` *posCount*	Use to enter the number of positions to create on the specified dimension.

The following examples explain creating informal positions in bulk on the product hierarchy. Figure 9–2 displays a product hierarchy with alternative branches and four DPM-enabled dimensions (STYL, STCO, SKU, and SZ12).

Figure 9–2 Sample Product Hierarchy with Alternative Branches



Example 1: Creating Informal Positions on the STYL Dimension

To create informal positions on the STYL dimension, you must specify rollups for SCLS, CLR and SPLR. You have two options for SZ12:

- Option 1: Specify the rollup for SZ12.
- Option 2: Specify the rollups for SIZ1 and SIZ2. A new position is automatically created on SZ12. You can specify spread ratios for STCO and SKU. If you do not specify the spread ratio, it defaults to 1.

Option 1

```
informalPositionMgr -d domainPath -create -dim STYL -n 100 -rollups
SCLS:scls1001,CLR:clrwhite,SPLR:splr100,SZ12:sz12null -spreads STCO:2,SKU:5
```

This command creates the following:

1. Creates 100 new informal positions on the STYL dimension. These positions all roll up to SCLS:scls1001.
2. Since the spread ratio for STCO dimension is 2, then 200 new informal positions are created at the STCO level, and every two of them roll up to one STYLE position

that was created in step 1. These new informal positions all roll up to CLR:clrwhite on the alternate branch.

3. Similarly, 1000 new informal positions are created at the SKU level, and every 5 of them roll up to one STCO position that was created in step 2. They all roll up to SPLR:splr100 and SZ12:sz12null on the two alternate branches.

Option 2

```
informalPositionMgr -d domainPath -create -dim STYL -n 100 -rollups
SCLS:scls1001,CLR:clrwhite,SPLR:splr100,SIZ1:siz1null,SIZ2:siz2lt02 -spreads
STCO:2,SKU:5
```

This command creates the following:

1. Creates 100 new informal positions on the STYL dimension. These positions all roll up to SCLS:scls1001.
2. One new informal position at the SZ12 level, which rolls up to SIZ1:siz1null and SIZ2:siz2lt02.
3. Since the spread ratio for STCO dimension is 2, then 200 new informal positions are created at the STCO level, and every two of them roll up to one STYL position created in step 1. These new informal positions all roll up to CLR:clrwhite on the alternate branch.
4. Similarly, 1000 new informal positions are created at the SKU level and every 5 of them roll up to one STCO position created in step 3. They all roll up to SPLR:splr100 and the new SZ12 position created in step 2.

Example 2: Creating Informal Positions on the SZ12 Dimension

To create informal positions on SZ12 dimension, you must specify rollups for SIZ1, SIZ2 and SPLR and specify the spread ratio for SKU. For the DPM-enabled dimension STCO, you have two options:

- Option 1: Specify a rollup position.
- Option 2: Create a rollup position and specify a rollup position on CLR and rollup position on STYL (or create recursively until the non-DPM enabled dimension is reached).

Option 1

```
informalPositionMgr -d domainPath -create -dim SZ12 -n 10 -rollups
SPLR:splr100,SIZ1:siz1null,SIZ2:siz2lt02,STCO:stco2000 -spreads SKU:5
```

This command creates the following:

1. Creates 10 new informal positions on the SZ12 dimension. These positions all roll up to SIZ1:siz1null and SIZ2:siz2lt02.
2. Since the spread ratio for SKU dimension is 5, then 50 new informal positions are created at the SKU level, and every 5 of them roll up to one position at the SZ12 level, which was created in step 1. These new informal positions all roll up to SPLR:splr100 and STCO:stco2000.

Option 2

```
informalPositionMgr -d domainPath -create -dim SZ12 -n 10 -rollups
SPLR:splr100,SIZ1:siz1null,SIZ2:siz2lt02,SCLS:scls1001,CLR:clrwhite, -spreads
SKU:5
```

This command creates the following:

1. Creates10 new informal positions on the SZ12 dimension. These positions all roll up to SIZ1:siz1null and SIZ2:siz2lt02.
2. One new informal position at the STYL level, which rolls up to SCLS:scls1001.
3. One new informal position at the STCO level, which rolls up to CLR:clrwhite, and the new position at STYL level, which was created in step 2.
4. Since the spread ratio for SKU dimension is 5, then 50 new informal positions are created at the SKU level, and every 5 of them roll up to one position at the SZ12 level which was created in step 1. These new informal positions all roll up to SPLR:splr100 and the new STCO position created in step 3.

Position Naming Convention

When the bulk creation feature is used, the names and labels of the new positions are created with the following name conventions:

- **Position name**: `{dimName}{dpm}{seq#}`

  The `width` is an attribute of the dimension. For auto-generated position names, the minimum width required is 10.

  - For width = 10, {4 char dimName}d{5 char seq#}
  - For width = 11, {4 char dimName}d{6 char seq#}
  - For width = 12, {4 char dimName}d{7 char seq#}
  - For width = 13, {4 char dimName}dp{7 char seq#}
  - For width >= 14, {4 char dimName}dpm{7 char seq#}

  If the dimName is less than four characters, an underscore is appended (for example, `sku_`. Each dimension uses its own persistent sequence generator in the domain with a unique label `DPM_{dimName}`. For a global domain, the sequence generators are in the master domain.

- **Position label**: This follows the same convention as that of position name, but uses the `label width` attribute of the domain.

Data Slice Copying

This feature allows you to copy data slices of one position to those of another. It is useful when you want to merge informal positions with existing formal positions. An inclusive or exclusive measure list can be specified. A Boolean type mask measure can be used to enable selective copying of data slices.

```
informalPositionMgr -d domainPath -copy -file inputXMLFile [-retain]
```

The following table provides descriptions of the arguments used by the data slice copying functionality of the informalPositionMgr utility.

Table 9–26 Arguments Used by the informalPositionMgr Utility: Data Slice Copying

Argument	Description
-d *domainPath*	Specifies the domain to run the utility.
-copy	Copies data slices within measures.
-retain	Use if you do not want to move the input file to the processed subdirectory after a successful run.

Table 9–26 (Cont.) Arguments Used by the informalPositionMgr Utility: Data Slice

Argument	Description
-file *inputXMLFile*	Specifies the XML file that contains the configuration settings for the copying operations.

Table 9–27 shows the available settings for the XML input file. This file defines the configuration settings for the copying operation.

Table 9–27 XML Input File Settings

Section	Attribute	Description	Required	Format
copy		Settings for copying operation.		
copy	removeSource	Removes source data slices after copying.	Yes	“true” or “false”
measures		A list of measures.	No	Comma delimited measure names
measures	option	The measure list is inclusive or exclusive.	Yes, if the measures section exists	“include” or “exclude”
mask		Mask measure.	No	One Boolean type measure name
positionMap		Position mapping for the copying operation.	Yes	
positionMap	dim	The dimension of the mapped positions.	Yes	Dimension name
positions		Two positions. The first one is the position to copy from. The second one is the one to copy to.	Yes	Comma delimited

Here is a sample XML Input File Schema in XSD format file.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<xsd:schema attributeFormDefault=unqualified” elementFormDefault=”qualified”
version=”1.0” xmlns:xsd=http://www.w3.org/2001/XMLSSchema”>
   <xsd:element name=”rpas” type=”rpasType” />
   <xsd:complexType name=”rpasType”>
      <xsd:sequence>
         <xsd:element name=”copy” type=”copyType” />
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name=”copyType”>
      <xsd:sequence>
         <xsd:element name=”measure” type=”measuresType” />
         <xsd:element name=”mask” type=”xsd:string” />
         <xsd:element name=”positionMap” type=”positionMapType” />
      <xsd:sequence>
      <xsd:attribute name=”removesource” type=”xsd:boolean” />
   </xsd:complexType>
   <xsd:complexType name=”positionMapType”>
      <xsd:sequence>
          <xsd:element maxOccurs=”unbounded” name=”positions” type=”xsd:string” />
      </xsd:sequence>
      <xsd:attribute name=”dim” type=”xsd:string” />
   </xsd:complexType>
   <xsd:complexType name=”measuresType”>
      <xsd:attribute name=”option” type=”xsd:string” />
   <xsd:complexType>
</xsd:schema>
```

Here is a sample XML Input file.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<rpas>
   <copy removeSource=”true”>
        <measures option=”include”>
           r_ex_demoa, r_ex_demob, r_ex_democ
        </measures>
        <mask>maskMeasure</mask>
      <positionMap dim=”sku”>
          <positions>sku_10000010, sku_10000008</positions>
          <positions>sku_22200001, sku_22200002</positions>
      </positionMap>
   </copy>

</rpas>
```

Merging Informal Positions to Formal Positions

Here are the steps to merge informal positions to formal positions in a domain.

1. Create the XML input file by specifying the position pairs to merge. Optionally, you can specify the measures to update and a mask measure.
2. Create the mask measure if you want to use a filter to protect some data slices from being modified. For example, if you want to leave elapsed data alone, you can create a calendar-based mask measure and set all the elapsed cells to false.
3. Run informalPositionMgr with the `-copy` option to copy the data slices.
4. Create an input file for the removal of informal positions you no longer need.
5. Run informalPositionMgr with the `-operation remove` option to remove those informal positions.

Managing Position Lists as PQDs Using pqdMgr

The pqdMgr is a command-line utility used to add and delete Position Query Definitions (PQDs) using XML-formatted position lists. A position list is a multi-level listing of positions along a non-Calendar RPAS hierarchy. For example, along the product hierarchy, a position list could be a single-level listing of SKUs, or it could be a multi-level list of classes, styles, and SKUs. A PQD is used to represent a set of selected positions in a particular hierarchy. Each PQD is identified by a unique name. A user can load the PQD instead of performing manual selections on a two-tree wizard.

This utility can be used to load PQDs in master, local, and non-partitioned domains. PQDs are not shared across local domains. Loading a PQD into a global domain does not affect any local domain. Similarly, loading a PQD into a local domain does not impact the master or other local domains.

The input file must be in the following XML file format. This example shows loading and deleting lists for world and user access levels.

```
<? xml version="1.0" encoding="UTF-8" ?>
<pqdlists>
      <pqdlist name="list_name" hier="hierarchy_name">
         <access level="user">
            <comma_separated_user_names>
         </access>
         <dimension name="dimension_name">
            <pos>
               <position_external_name>
         </pos>
      </dimension>
   </pqdlist>
   <pqdlist name="list_name" hier="hierarchy_name">
      <access level="world">
      </access>
      <dimension name="dimension_name">
         <pos>
            <position_external_name>
         </pos>
      </dimension>
   </pqdlist>
   <pqdlist name="list_name" hier="hierarchy_name" operation="delete">
      <access level="world">
      </access>
   </pqdlist>
   <pqdlist name="list_name" hier="hierarchy_name" operation="delete">
      <access level="user">
         <comma_separated_list_of_users>
      </access>
   </pqdlist>
</pqdlists>
```

pqdMgr Usage

```
pqdMgr -d domainPath -load xmlFile
pqdMgr -d domainPath -delete xmlFile
pqdMgr -d domainPath -deleteAll
pqdMgr -d domainPath -validate xmlFile
pqdMgr -d domainPath -export outFile [-user userName|-world]
```

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

Table 9–28 Arguments Used by the pqdMgr Utility

Argument	Description
-d *domainpath*	Specifies the path to the domain.
-load *xmlFile*	Loads position lists from an input XML file. Position lists with an operation attribute of delete are ignored.
-delete *xmlFile*	Deletes PQDs as specified in the input XML file with an operation attribute of delete. Position lists with an operation attribute of load are ignored.
-deleteAll	Deletes all PQD lists from the domain.
-validate *xmlFile*	Validates the XML file and report any errors. No impact on the existing PQD files in the domain.
-export *outFile* [-user userName|-world]	Exports existing PQDs in the domain for a user or world access level. The file specified by outFile is overwritten. Requires one of the following options: ■ -user *userName* exports PQDs for the provided *userName* in the same XML format as used for a load. ■ -world exports PQDs with world permission in the same XML format as used for a load.

pqdMgr Notes

Note the following about pqdMgr:

- Input files are validated before loading. All dimensions, hierarchies, and user names provided in the input file must be consistent with the existing hierarchies and registered users in the domain. The utility fails (return a non-zero error code) if it finds such inconsistencies in the input file. The errors are reported to the standard output.
- Multiple list operations are allowed in the XML input file.
- The supported operations are load and delete. If no operation is specified for a list, the default is load.
- The name list_name must be unique within an access level and, if the access level is user, for the user name.
- Each list definition consists of the list name, hierarchy, and access level. One or more dimension definitions are allowed. One or more position definitions are allowed for each dimension. Only external names are allowed to describe positions.
- When an access level of user is specified, a single user name or a comma-separated list of user names is required. A PQD file is created for each user name in the list.
- For the access level of world, the PQD file that is created is saved in the following path:

  <domain_root>/wizardPQD/*<hierarchy_name>*/_world/*<list_name>*
- For the access level of user, the PQD file that is created is saved in the following path:

  <domain_root>/wizardPQD/*<hierarchy_name>*/*<user_name>*/*<list_name>*

10

Data Management

This chapter explains the processes involved in RPAS data management. It includes the following sections:

- Loading Measure Data Using loadmeasure
- Exporting Measure Data Using exportMeasure
- Exporting Measure Data Using exportData
- Mapping Data Between Domains Using mapData
- Moving Data between Arrays Using updateArray
- Scan Domain Data Using scanDomain
- Repair Domain Metadata Using fixDomain

Loading Measure Data Using loadmeasure

The loadmeasure utility is used to load measure data from text files into the domain. The administrator must specify the measure names and the path to the domain that contains the measures.

The loadmeasure utility supports the use of fixed width and CSV (comma separated variable) files for loading measure data. RPAS recommends the use of CSV files to reduce the size of the load file and to reduce disk I/O time.

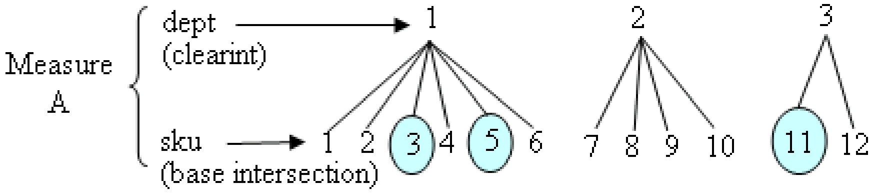
To load measure data, system administrators must copy or create one or more load files in the input folder of the domain directory. The administrator can then call loadmeasure to load data.

Example:

Measure A has a base intersection of SKU and a clearint of dept.

If there is data for only a few SKUs (3, 5, and 11) in the incoming file, and SKUs 3 and 5 roll up to dept1 while sku11 rolls up to dept3, the data in all of the SKUs that rolls into dept1 and dept3 will be cleared.

Figure 10–1 Loading Measure Data



- The base intersection is SKU.
- The clearint is dept.
- Data is present for SKUs 3, 5, and 11; which fall under dept 1 and dept3.
- Data will be cleared for dept1 (SKUs 1, 2, 3, 4, 5, 6) and dept3 (SKUs 11, and 12).
- Data for SKUs in dept2 (7, 8, 9, and 10) will be untouched.

Load File Names and Load Behavior

System integrators must pay close attention to file naming. If a file name has not specifically been configured in the domain configuration, the file must be named the same as the measure name, with the appropriate extension, depending on the type of the load.

For example, if the measure is named "rsal" and does not have a filename configured in the domain configuration, then the basic filename will also be "rsal". This name should be appended with one of the following extensions to indicate the type of load. If the load is an overlay, then the filename should be rsal.ovr; if it is an increment the file name should be rsal.inc, and so on. If a CSV file is being used, then the load type extension should be prefixed with the .csv extension; for example, rsal.csv.ovr and rsal.csv.inc.

RPAS supports the following types of loads (identified by file name extension):

- .ovr (Overlay): Existing values in the measure are overlaid with the values in the input file. Any values not included in the input file are not changed in the measure.

Note: For string type measures, an empty cell in the ovr file is treated as a valid string; as a result, the loadmeasure utility overwrites the previously loaded string with an empty string.

For other measure types, an empty cell in the ovr file is treated as invalid data. It is discarded and the previously loaded value is retained.

- .rpl (Replace): The existing measure is cleared and the values in the input file are taken as the new values for the measure. Existing values for cells that do not exist in the load file are switched to NA. In other words, all data at the base intersection for the measure is removed before cells are populated with the data from the incoming file.
- .inc Increment): Increment mode should only be used with numeric measures in which the load file contains incremental values. Therefore, if a cell had a value of 2 and the .inc file provided a value of 3 for the cell, then the new value for the cell is 5 (2 incremented with 3).

- .clr (Clear): Clear mode is a variation of replace mode. It is used when measure data is loaded in parts or staggered in time, so that data for all positions grouped by an aggregate level position is replaced if one or more positions for that group of positions are being loaded.

  In other words, data at the base intersection of a measure is partially cleared based on incoming data and the clearint attribute for the measure. The clearint attribute defines an intersection above the base intersection. All cells at the base intersection that are descended from a given position at the clearint level will be removed if data exists in the incoming file for at least one of those descending positions.

  For example, assume that there are four regions, each with several stores, and the data is loaded region by region or for a subset of regions at a time. When loading data, ensure that data for a region is completely replaced with the new load if the load file has data for one or more stores from that region; however, other regions should be left untouched. This is made possible by clear loads where the clear intersection (`clearint`) property of a measure specifies the aggregate level at which to group positions for completely replacing the data. In this example, the clear intersection is at the region level. Clear intersection does not have to be performed along one hierarchy, but can be performed at the intersection of multiple hierarchies.

  However, if you load multiple .clr files with region as the clear intersection, and data for one of those regions is contained in multiple files, then the last loaded .clr file for that region will replace any information that the previous .clr files loaded for that particular region.

The loadmeasure utility allows more than one load file to be present in the input folder at the same time for the same measure. If more than one load file is present in the input folder at the same time, each will be loaded. Since RPAS has a strict naming convention for measure file names, in order to add more than one load file at the same time, integrators must append the filenames as described above with file-distinguishing extensions.

For example, with the file names rsal.csv.ovr.1 and rsal.csv.ovr.2, RPAS does not care about the form of the multi-file extension. The extensions can be anything, number or text, and RPAS will still load them.

Note:

- Backup files should not be named as rsal.csv.ovr.bak or they will be loaded as well.
- loadmeasure does not guarantee any specific ordering of loads based on the appended extensions.

The loadmeasure utility also allows multiple types of load files to be present in the input directory at the same time. RPAS loads .rpl files first, then .clr, .ovr, and .inc files. Since .rpl files completely erase existing measure data and then load the given data, you should not have multiple .rpl files at the same time.

Loading Multiple Measures from One File

The loadmeasure utility allows multiple measures to be loaded from a single file. You can load measures from "CSV Files" or "Fixed Width Files".

Note: See the "Data Interface Tool" section of the *RPAS Configuration Tools User Guide* for more information.

CSV Files

If a CSV file is used for loading multiple measures, loadmeasure will use the start positions of measures as defined in the Data Interface Tool for the fixed file format to determine the order of columns in the CSV format. For example, if a file named multiple is used to load measures A, B, and C, where the start position (for fixed file format) for the measure values have been configured to be 40, 110, and 70 respectively, then when using the CSV file multiple.csv.ovr, `loadmeasure` will assume that after the dimension columns, the first column is measure A, then C, and then B because 40 (A) is less than 70 (C) is less than 110 (B).

It is not necessary to load all measures in a multiple measure file. Administrators can choose to load only a subset of measures from the multi-measure file. However, if Administrators wish to reuse the same file for loading different measures at different times in a batch, they must use the -noclean option to ensure that loadmeasure does not move the file to the processed folder after processing the first load request.

Note: Even though it is not required to specify all measures contained in the multi-measure CSV file in a single loadmeasure command, there is no way to skip data columns in the CSV file.

Fixed Width Files

With a fixed width file, a single measure's data can be loaded from a file containing multiple measures.

Loading Data from Below the Base Intersection of the Measures

The loadmeasure utility supports loading measure data from an intersection lower than the base intersection of the measure. The load intersection has to be pre-specified in the configuration (loadint property) and the load time aggregation (loadagg property) method must also be specified. Refer to the *RPAS Configuration Tools User Guide* for information on setting up measure properties.

When loadmeasure loads data from below the base intersection, all low-level data corresponding to a cell at the base intersection must be available in the load file for RPAS to be able to correctly aggregate the low-level data to the base level. A mistake in the values of a subset of cells that aggregate up to one cell at the base level can only be corrected by reloading the data for all low-level cells that correspond to the cell at the base level. If any low-level cells are missing, RPAS replaces their value with NA.

To perform a lower level load, RPAS first aggregates the data and then applies the appropriate load type to update the measure value, overwriting the existing value with the aggregate of the input cells if .ovr files were used, or incrementing the existing value with the aggregate of the input cells if .inc files were used.

Staging Measure Loads

RPAS supports the notion of stage-only measures. For stage-only measures, loadmeasure queues the loaded data in an intermediate staging area, but does not load it into the measure until it is called with the -applyloads parameter. For stage-only measures, loadmeasure should be called twice, once to stage the measures and then with the -applyloads parameter to subsequently load the staged data in the measure arrays. loadmeasure cannot simultaneously stage loads and apply the staged loads.

Measure staging should be performed when measure data can arrive from different sources, in different load formats, and staggered in time, when system administrators want to queue all these loads up and apply them at once while honoring the data arrival queue. Measure staging can be performed while the system is online as it does not cause measure data-related contention (it has the potential to cause metadata-related contention). When staging measure data, loadmeasure splits the data and purges the data files if data purging is enabled; it does not purge measure data until the loads are applied. This staging time preprocessing significantly reduces the load time when the loads are actually applied.

Note: The replace (.rpl) format cannot be used for staging. Furthermore, data loads from below the base intersection of the measure cannot be staged.

Running Pre-Load or Post-Load Scripts

The loadmeasure utility provides the ability to automatically run scripts before and after the utility is executed. These are referred to as preprocessing 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. Similarly, after the utility has completed running, the utility checks for the existence of scripts named post<*measurename*>.sh and executes them.

When multiple measures are loaded in a single call, only the preprocessing script for the first listed measure has any effect on the data.

Purging Old Measure Data

System administrators can purge old measure data during a load. When the base intersection of a measure involves the Calendar hierarchy, the setting for the purgeAge measure property defines how and when existing data gets purged to a NA value. If the purgeAge has not been set, the data never gets purged. If a purge age of zero or more has been set, data is purged for all dates before RPAS_TODAY - purgeAge days. That is, if the purgeAge is 5, then at data load time, all data that is older than 5 days before RPAS_TODAY will be purged.

Behavior in a Global Domain Environment

In a global domain environment, loadmeasure is centralized and can only be called in the master domain. The loadmeasure utility loads 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 then splits the input files and loads 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 only occurs 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.

loadmeasure Usage

```
loadmeasure -d pathToDomain -measure measureName{,measureName,…} {-applyloads}
{-processes max} {-noClean} {-forcePurge}{-splitOnly | -noSplit} {-defrag}
{-loglevel level} {-recordLogLevel level} {-inDir inputDirectory}
```

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

Table 10–1 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 measures 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	Applies any staged loads for the named measure. If the measure is registered to be a stage-only measure, loadmeasure will put the load in a staging area but will not update the measure until loadmeasure is called again with this argument. Upon the use of this argument, loadmeasure applies all loads that have been queued up in the staging area. It clears out the staged loads unless the measure's loadsToKeep property has been set to a non-zero number. In that case, it does not clear out the latest loadsToKeep loads. Note that only .ovr, .inc, and .clr loads can be staged. The .rpl loads cannot be staged. Additionally, staging is only allowed for base intersection loads. RPAS cannot stage loads where the load intersection is below the base intersection of the measure. This argument should not be used for measures that are not stage-only.
-processes *max*	Specifies the maximum number of child processes for parallel splitting of files and loading of measures across local domains in a global domain environment. For instance, if you specify five as the maximum number of processes, then up to five child processes can run concurrently in the split or load operations. If this argument is omitted or if only one process is specified, the application will perform 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	Prevents the input files from being moved to the processed directory. This option is used when a single file is used to load multiple measures but not all measures from the file are loaded at once. This option instructs loadmeasure to leave the load file behind for subsequent loading of unloaded measures. The user can use this option to perform intermediate processing between loads of measures available from the same file.

Table 10–1 (Cont.) Arguments Used by the loadmeasure Utility

Argument	Description
-forcePurge	Forces the purge routine to run even if no new data is loaded. This purges old measure data. This option can be applied to stage-only measures without having to apply loads. When a measure has the Calendar hierarchy in its base intersection, the setting for the purgeAge measure property defines how and when existing data gets purged to a NA value. If purgeAge has not been set, the data never gets purged. If a purge age of zero or more has been set, data is purged for all dates that are before RPAS_TODAY - purgeAge days. That is, if purgeAge is five, at data load time all data that is more than five days before RPAS_TODAY will be purged. This option does not require you to load any new data.
-noSplit	Loads the pre-split input files (created by -splitOnly) into the local domains. This option should only be used in global domain environments.
-loglevel *level*	Sets the logger verbosity level. Possible values: all, profile, audit, information, warning, error, or none.
-splitonly	Causes the input files in the global domain to be split across the local domains, but does not do any further processing of the input files. Subsequently, loadmeasure can be used with the -noSplit argument to load these pre-split input files into the local domains. File-splitting is a fairly time consuming activity and can consume up to 80% of the load time. System integrators may be able to improve batch performance by breaking away file-splitting from actual measure loading. This is useful if a multi-measure file is being used in such a way that subsets of measures are loaded at different steps in a batch process. File-splitting is a single process procedure, so the -splitOnly option is mutually exclusive with the -processes argument. This option should only be used in global domain environments.
-defrag	Defragments the domain at the end of the measure-loading process to reduce the physical size of the domain. This space-saving is achieved by replacing the existing fragmented pages with copied, fully populated BTree database pages.
-recordLogLevel *level*	Sets a logging level for record loading issues. Issues such as parsing errors, missing positions, and data conversion errors are evaluated for every record in the measure load file. By default, these are logged as errors in the log file of the loadmeasure utility. However, customers might want to downgrade the logging level for such record loading issues. They can do that using the -recordLogLevel *level* argument. The standard log levels, error, warning, information, and profile, can be used as parameters to this argument. When logging, loadmeasure compares this logging level to the utility's logging level (set using -loglevel). If the utility's logging level is less verbose than the record logging level, then record issues will not be logged. If utility's logging level is at same or higher verbosity as the record logging level, the record issues will be logged with the log indicator as set using this argument.

Table 10–1 (Cont.) Arguments Used by the loadmeasure Utility

Argument	Description
`-inDir` *InputDirectory*	Only .rpl files can be used with this option, and only CSV format with header line is supported. The header line is used to map the columns to dimensions and measures (for example: SKU,STR,DAY,Sales). Enter one measure per input file. The name of the measure is extracted from the file name; for example, sales.csv.rpl corresponds to measure sales. The input data must be at the base intersection of the measure. If the measure is normally partitioned (non-HBI), a subdomain index may be used for further performance optimization by avoiding the data-splitting step. For measure that may contain duplicate positions in different subdomains (FnHBI measures), the subdomain index is required. In either case, the name of the file is used to figure the subdomain index (for example, sales.0.csv.rpl corresponds to the first local domain; sales.1.cvs.rpl corresponds to the second local domain, and so on). **Note**: The subdomain index is designed to be used in conjunction with `exportMeasure -hier` only. Manual name-indexing of the files is not recommended. The filename property of the measure is not considered with the `-inDir` option.

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 must be stored in a measure called `r_images_<dimension name>`, where `<dimension name>` must 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 the Fusion Client* for more information on Image Display.

Example

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

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

Exporting Measure Data Using exportMeasure

The exportMeasure utility is a command-line utility that may be used to export domain or workbook measure data from RPAS in either a CSV or a 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. This utility:

- Supports export of 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.

> **Note:** The exportMeasure utility allows multiple measures to be exported into the same file when multiple measure names are provided. The same measure name cannot be specified more than once (using comma separation) in a single call; therefore, a measure can be exported only once per file.

exportMeasure Usage

```
exportMeasure -d pathToDomain -out outFile [COMMAND] [OPTIONS]
```

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

Table 10–2 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.
`-wb` *wbname*	Exports data from the specified workbook (wbname), if specified. A valid workbook name must be used.
`-intx` *intxString*	Specifies the intersection at which to export measures. If the measure's base intersection is higher than the export intersection, replication is used to spread the measure down to the export intersection. If the measure's base intersection is lower than export intersection, the measure's default method (defagg) is used for aggregation. The export intersection must be either at, above, or below the base intersection of the measure. The export intersection cannot have some dimensions above the dimension in the base intersection of the measure and some below. The RPAS dimension names in an intersection should be four characters in length. If a RPAS dimension name is less than four character long, then an underscore character ("_") should be used as a filler at the end of a dimension name.
`-mask` *measureName*	Specifies a mask measure that must be a valid Boolean measure registered. In the current measure store, its baseintx must be at same export intx.
`-range` *start:end*	Specifies a range of positions 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 the current output file. If not specified, the current output file will be erased and replaced with new data.

Table 10–2 (Cont.) Arguments Used by the exportMeasure Utility

Argument	Description
`-nomerge`	Stops exportMeasure from merging multiple output files created from each local domain to the master output files when run in a global domain environment with the exporting intersection below partition dimension, and with processes set to greater than 1. Output files created from local domain are stored in the 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 that the output file should be in the compressed CSV format.
`-hier` *hierarchy1, hierarchy2*	Exports all measures for hierarchies. It exports only measures that have storage in the domain. Multiple hierarchies can be specified in a comma-separated list.
`-outDir` *outputDirectory*	Updates the output directory. If the output directory does not exist, the utility creates one. The measure names are used to generate the output file names. A CSV file with a header line can identify the dimensions of the base intersection and the name of the measure that is generated for each file. The files always have a csv.rpl extension (for example, sales.csv.rpl). Old files are overwritten. One output directory is created for each HBI measure. In addition, one file is created for each non-HBI or FnHBI measure per subdomain. The file names contain an internal subdomain index, for example `sales.0.csv.rpl`, `sales.1.csv.rpl`, and so on.
`-upperCase`	Converts the position names to all uppercase before writing the output data file. Without this argument, position names are in lowercase since they are stored in lowercase in the domain.

Table 10–2 (Cont.) Arguments Used by the exportMeasure Utility

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.
	The modifier includes the following:
	■ .precision<double>, specifies the precision for numeric measure
	■ .format<formatString>, specifies 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")`
	For specifying date and time, the following formats are supported:
	%Y: four-digit year
	%y: two-digit year
	%m: month
	%d: day
	%B: full name of the month
	%h: three-character abbreviation for the month
	%H: hour
	%M: minute
	%S: second
	%s: milli-second
	The examples below provide valid measure specifications, given MeasNameB is a valid date/time type measure.
	Examples:
	`-meas MeasNameB`
	`-meas MeasNameB.format("%Y%m%d")`
	`-meas MeasNameB.format("%d%B%Y%H%M%S")`

Exporting Measure Data Using 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 extracts the data from either the local domains or the master domain, depending on where the data resides, which in turn 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 in which 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 goes to this array and uses the original position name to jump to the cell of the same position name. It then gets the cell value and uses 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* exports 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.

exportData Usage

```
exportData -d domainPath -out outputFile -params db array
exportData -d domainPath -out outputFile -array \"arraySpec\" {options}
exportData -d domainPath -out outputFile -meas <measspec> | -array <arrayspec> |
-params <paramspec> -wb <wbName> {options}
```

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

Table 10–3 Arguments Used by the exportData Utility

Argument	Description
`-d` *domainPath*	Specifies the domain that contains the data that to export.

Table 10–3 (Cont.) Arguments Used by the exportData Utility

Argument	Description
-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 measures 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.
`-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 .ary file where the array of parameters is stored. - *array* specifies the name of an array in the specified database that has the above parameters. - Only the string type scalar measure is allowed as a parameter array.
`-wb` *WbName*	Use after specifying any of the command arguments: -array, -meas, or -params.
`-append`	Specifies that output is appended at end of output file. The default is to overwrite output file.

Table 10–3 (Cont.) Arguments Used by the exportData Utility

Argument	Description
`-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. When using with the -wide parameter, the -dim parameter should not be used for the innermost dimension.
`-skipNA` *always* | *allna* | *anyna* | *arrayna*	- *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). Controls whether a line of data is exported based on having NAs in a cell.
`-naArray` *arrayName*	Specifies the export array that is checked to determine if data is exported when arrayna is specified using the -skipNA parameter.
`-wide`	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. The -range parameter can be used in conjunction with wide format (-wide) to specify a range along the innermost dimension. The -dim parameter should not be used for the innermost dimension when -wide is being used.
`-range` *start:end*	:Limits the export to positions in the range. The range can only be specified for the innermost dimension. May be used in conjunction with the -wide parameter.
`-time`	Specifies the YYYYMMDD format for dates.

Table 10–3 (Cont.) Arguments Used by the exportData Utility

Argument	Description
`-precision` *precisionValue*	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. In this case, 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`	Enables the use of the NA value of the array instead of the NA value specified in measSpec or arraySpec.
`-upperCase`	Converts the position names to all uppercase before writing the output data file. Without this argument, position names are in lowercase since they are stored in lowercase in the domain.
`-displayArrayNaValue`	Controls the display of measures used as a mask in -naArray option. The default is to not display the mask NA measure value. However, if this option is specified, then the NA Array measure values are also exported.
`-useLoadFormat`	Enables the format specified by the measure property. It uses the level at which the data is stored in the domain. The -dim parameter is not allowed with the -useLoadFormat.

The -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 is exported into the same column in the output file. If the full measure export specifications 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 as follows:

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

All value are exported right aligned, as in the measure loading file.

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

Users can either use the default format by specifying the measure name only, or give the full specifications. Partial measure specifications are not permitted.

If users specify that multiple measures will be exported into the same file, then each of 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 are exported by default. The external position name is 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 are left aligned in the export file.

Mapping Data Between Domains Using 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 this utility is run, 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.

---

mapData 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.

Table 10–4 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*	Applies mapdata only on the given database. Must be a valid file. If this argument is not specified, the entire domain will be included in the operation.
-array *arrayName*	Applies mapdata only on the given array. The database in which the array resides must be specified with the -db argument.

Moving Data between Arrays Using 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.

updateArray 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.

Table 10–5 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.

Table 10–5 (Cont.) Arguments Used by the updateArray Utility

Argument	Description

`-destRange` *first:last*	Optional argument.
	Default is no range.
	Defines range along innermost dimension of destination array.
	The position names of the innermost dimension are the range value. For example, if the range values is one week, the range should be specified as
	`-srcRange WEEK200811011:WEEK200811022`
	`-destRange WEEK200811011:WEEK200811022`

`-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.

Scan Domain Data Using scanDomain

The scanDomain utility is a domain utility used for detecting data loss and repairing data corruption in an RPAS database.

Data loss occurs when an RPAS process is abnormally terminated. This can happen when an external mechanism, such as a power failure, causes a sudden termination of an RPAS process. Data loss can also occur due to unexpected program breakdown.

Data corruption can occur if an external program modifies the RPAS database files or an unforeseen defect occurs in the processes using the RPAS database (an extremely rare event).

The scanDomain utility can detect both corruption and data loss, but it can only fix corruption. This utility can operate on global, non-partitioned, and local domains. It supports parallelization when repairing databases in a domain.

While the utility is attempting to perform a repair of the databases, it can use a command line option (-backup) to enable backing up the original databases. While running in detection mode (-detectDataLoss or -detectCorruption option), the utility does not change any of the RPAS databases, and therefore, it does not create such backups.

In detection mode, the utility prints a list of databases with data loss or data corruption to the screen. The output can be directed to a file.

scanDomain Usage

```
scanDomain -version
scanDomain -d domainPath [-detectDataLoss ] [-detectCorruption] [-loglevel level]
[-noheader]
scanDomain -d domainPath -repairCorruption [-backup][-processes
maximumNumberOfProcesses] [-loglevel level] [-noheader]
scanDomain -d domainPath -listUnusedData outputFile [-processes
maximumNumberOfProcesses] [-loglevel level]
```

```
scanDomain [-?|-help|-usage]
```

If the user intends to detect both corruption and data loss, it is more efficient to run the utility once with both the -detectDataLoss and -detectCorruption options. The user can run two consecutive commands for detecting corruption and data loss, although this is less efficient.

When running scanDomain to detect unused data, the user sees a list of databases that may not be needed by the domain. This information includes:

- User directories for users who are not registered in the domain
- Measure databases whose corresponding measures have been removed from the domain
- Other databases not referenced by the domain (for example, measure load databases, backup databases, and temporary databases)

The data contained in the specified databases may not be needed by the domain. If the user can confirm that the data is actually unnecessary, then the user can remove those databases before proceeding with the upgrade.

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

Table 10–6 Arguments Used by the scanDomain Utility

Argument	**Description**
-version	Prints the version of the utility.
-d *domainPath*	Specifies the path to a global, non-partitioned, or local domain. Required.
-detectDataLoss	Checks for data loss in the specified domain.
-detectCorruption	Checks for database corruption in the specified domain.
-repairCorruption	Repairs the database corruption in the specified domain. **Note:** This argument cannot be used with -detectDataLoss or -detectCorruption.
-backup	Used to back up database files before attempting to repair them. Optional. **Note:** This argument can only be used with -repairCorruption.
-listUnusedData	Lists potentially unused data in the domain. Information about this unused data is output to a file whose location is passed as the *outputFile* argument.
-processes *maximumNumberOfProcesses*	Specifies the maximum number of processes to be started to repair Btree database corruptions. Optional. **Note:** This argument can only be used with -repairCorruption.
-loglevel *level*	Sets the logger verbosity level. Possible values: all, profile, debug, audit, information, warning, error, or none. Optional.
-noheader	Disables the timestamp header. Optional.
-?|-help|-usage	Obtains the usage text. Optional.

Repair Domain Metadata Using fixDomain

Discrepancies in the metadata of a domain can cause serious problems during domain operations, and such discrepancies can be difficult to correct. The fixDomain utility analyzes a domain in order to detect problems in the metadata of a domain and can be used to attempt to fix any detected problems.

By default, fixDomain runs in report-only mode. Users should run fixDomain in this mode first in order to detect discrepancies in the metadata. Should any problems be discovered, fixDomain provides functionality to fix certain problems in the metadata of the domain.

11

Operational Utilities

This chapter details the following operational utilities of RPAS:

- Finding Alerts Using alertmgr
- Copying Domains Using copyDomain
- Moving a Domain Using moveDomain
- Setting Miscellaneous Domain Properties Using domainprop
- Calculation Engine Using mace
- Managing the Workbook Batch Queue Using wbbatch
- Workbook Manager with wbmgr
- Registering Measures Using regmeasure
- Registering Token Measures Using regTokenMeasure

Finding Alerts Using alertmgr

Batch alerts are an exception management tool for users. A batch 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.

The Fusion Client also uses Real Time alerts. Real time alerts will re-evaluate every time the measures upon which they depend are modified. Also unlike batch alerts, Real Time Alerts are not two-state (either a hit or not a hit); any number of discrete conditions can be defined for a Real Time Alert. When the alert is evaluated, whichever condition applies will be used (e.g. a Real Time Alert defined on an inventory measure could define one condition for low stock and a more severe condition for no stock).

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.

After 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 Fusion 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. Its aggregation state and base state must be read-only.
2. Create the alert (the expression) for which the alert should be evaluated using the Configuration Tools. Using alertmgr, register the alert with a category, categoryLabel, and the above expression. This flags the registered measure as an alert so that it is recognized when the alert finder is run.
3. Repeat steps 1 and 2 for any additional alerts to be registered in the domain.
4. 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.

alertmgr Usage

```
alertmgr -d domainPath [COMMAND [parameters]]
alertmgr -d pathtodomain -findAlerts {-navigationThreshold hits} {-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.

Table 11–1 provides descriptions of the arguments used by the alertmgr utility.

Table 11–1 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. It can be run from either Master or Local Domains.
`-alerts` *a1 a2…*	Evaluates specific alerts in the domain. The values *a1 a2 …* must be valid names of alerts that are defined in the domain.
`-category` *cat1 cat2 …*	Evaluates all alerts in the domain that are associated with specific categories of alerts. The values *cat1 cat2 …* must be valid names of alert categories that are defined in the domain.
`-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. It can be used only from the Master Domain. **Note**: -findAlerts must be run first to generate hit counts of alerts.
-navigationThreshold *hits*	Indicates the maximum number of alert hits for Find Next/ Previous Alert functionality to remain operational in a workbook. If over that threshold, the Find Alert functionality will only work up to that number. The default value is 5000.

Note: The alertmgr utility 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 Using 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.

copyDomain Usage

```
copyDomain -xmlConfigFile filename {OPTIONS}
copyDomain -d pathToSrc {OPTIONS}
copyDomain -version
```

Table 11–2 provides descriptions of the arguments used by the copyDomain utility.

Table 11–2 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.
`-xmlConfigFile` *pathToXmlConfigFile.* xml	Allows copyDomain to copy each subdomain into user-instructed specific locations. This argument should not be used with -d or -dest. See "copyDomain: Format of the XML Configuration File" for the file format.
The following arguments are valid for -xmlConfigFile and -d:	
`-force`	Deletes the existing domain at the specified destination path before copying the source domain.
`-clone` *dimposlist*	Copies 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.

Table 11–2 (Cont.) Arguments Used by the copyDomain Utility

Argument	Description
`-partitionPositions` *positions*	Deprecated. Use `-clone` instead.
`-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, indicating that no workbooks are copied. If this argument is not specified, all workbooks in the environment are copied.
`-skipInput`	Do not copy the input directory located in the source domain.
`-skipEmptyDir`	Do not copy the empty directory located in the source domain.
`-maxProcesses` *count*	Specifies that some parts of copyDomain run in parallel, using up to the given number of processes.
`-noSubDomains`	Do not copy any local domains in the source domain.
The following arguments are valid only with -d:	
`-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 or -relativizePaths) of the utility. If this argument is not provided, the domain is updated to have relative paths.
`-export`	Exports each database 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.
`-gzip`	Compresses the copied domain into a gzip format. This argument cannot be used when specifying an -xmlConfigFile.
`-dimDictOnly`	Copies only the source domain structure, metadata, and hierarchy data. Running copyDomain with this option result in a non-functional domain. Therefore, this argument should be used for diagnostic purposes only.

Table 11–2 (Cont.) Arguments Used by the copyDomain Utility

Argument	Description
`-relativizePaths`	Updates the existing master and subdomain path references to relative paths. If the current absolute path references are invalid paths, subdomains are searched for in the same location as the master and within the master domain directory. When this argument is used, no domain copy is made. **Note**: When using this argument, do not provide a destination using the -dest argument. For example, if you build a domain, this is what it looks like at first: (PGRP100 INFO): "C:\Oracle\Domains\1323\mfprtl\ldom3" (PGRP200 INFO): "C:\Oracle\Domains\1323\mfprtl\ldom3" (PGRP21 INFO) : "C:\Oracle\Domains\1323\mfprtl\ldom0" (PGRP22 INFO) : "C:\Oracle\Domains\1323\mfprtl\ldom1" (PGRP300 INFO): "C:\Oracle\Domains\1323\mfprtl\ldom3" (PGRP31 INFO) : "C:\Oracle\Domains\1323\mfprtl\ldom2" (PGRP32 INFO) : "C:\Oracle\Domains\1323\mfprtl\ldom2" This tells you, by partition dimension, the path to the local domain. You can see that it is a full path and that the domains are subdirectories of the master domain. If you then run copyDomain with the -relativizePaths option, the paths look like the following: (PGRP100 INFO) : "ldom3" (PGRP200 INFO) : "ldom3" (PGRP21 INFO) : "ldom0" (PGRP22 INFO) : "ldom1" (PGRP300 INFO) : "ldom3" (PGRP31 INFO) : "ldom2" (PGRP32 INFO) : "ldom2"

Notes for copyDomain

- By not providing the `-dest` or `pathToDestDomain`, the utility no longer makes the paths to the subdomains relative paths. To address this issue, you should use the argument `-relativizePaths`.
- When used with `-clone` or `-noSubDomains`, `copyDomain` does not affect workbook metadata or hierarchies.
- Workbooks that are not included in the list used with the `-copyWorkbooks` option are not included in the new domain.
- Any existing workbooks in a domain copied with the `-clone` or `-noSubDomains` options may not be able to be committed back to the new domain.
- When used with `-dimDictOnly`, the `-clone` or `-noSubDomains` options cannot be specified.
- `-dimDictOnly` switch implies `-copyWorkbooks` none.
- Use `-xmlConfigFile` to specify destination locations for individual subdomains.
- To get the usage text, use `-?`, `-help`, or `-usage`.
- To get the version of this utility, use `-version`.

- To set the logger verbosity level, use `-loglevel` with the following values: all, profile, debug, audit, information, warning, error, or none.
- To disable the timestamp header, use `-noheader`.

copyDomain: 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 subdomains. 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 stPath tag, and a subdomain tag for each subdomain. Each subdomain tag should contain one srcPath tag and one dstpath tag. Each srcPath tag should be a path to either the master or subdomain being copied. Each matching dstPath tag should be a path to where to copy that part of the domain.

The copyDomain utility validates the configuration xml file first before any files are copied. If any of the subdomain source paths do not match a subdomain path of the global domain being copied, a "can't find source subdomain 'subdomain' " error will be reported. If the global domain being copied contains any subdomain that does not have a matching srcPath tag, a “subdomain 'subdomain' does not have a subdomain xml tag" error will be reported. If the global domain srcPath tag does not 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 are 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.

Two options control the number of subdomains to be copied. These options still limit the number of subdomains that are copied; however, the configuration file must still contain entries for all domains.

Moving a Domain Using 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 is simple and designed to fit the required task. It contains fields for the locations of the source master domain and destination master domain as well as source and destination fields for each of the subdomains that need to be moved.

moveDomain Usage

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

Table 11–3 provides descriptions of the arguments used by the moveDomain utility.

Table 11–3 Arguments Used by the moveDomain Utility

Argument	Description
`-xmlConfigFile` *pathToXmlConfigFile.xml*	Allows moveDomain to move a subdomain into user-instructed specific locations-based paths specified in an xml file. This argument should not be used with the -d, -srcSubDomain, or the -dstSubDomain parameters. See "moveDomain: Format of the XML Configuration File" for the file format.
`-d` *pathTomaster*	Allows moveDomain to move each subdomain based on the user-specified paths. Enter the path to the master domain.
`-srcSubDomain` *src*	Indicates the path of the subdomain to be moved.
`-destSubDomain` *src*	Indicates the path where the subdomain is to be moved.

moveDomain: Format of the XML Configuration File

The XML configuration being used is simple and designed to fit the required task. It contains fields for the locations of the source master domain and destination master domain as well as source and destination fields for each of the subdomains 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\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. The 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 to be moved; otherwise, an error condition will be incurred.

The srcPath and dstPath tags for local domains are required only if the local domain is to be moved; otherwise, the lack of tags for a specific local domain 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 (that is, when moving global domains), all local domains that reside under the global domain folder and are not included in the XML file are moved to the destination global domain folder. Other local domains with a specified destination location are 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 subdomain paths are required only for the domains that need to be moved.
- The subdomains that need to be moved can be specified or those subdomains that will remain under the master domain can be left out. If a subdomain is not specified, it will be moved along with the master domain.
- 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) the master domain plus the size of the largest local domain.

Setting Miscellaneous Domain Properties Using 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.

domainprop Usage

```
domainprop -d pathToDomain {-property propertyname=value}
```

Table 11–4 provides descriptions of the arguments used by the domainprop utility.

Table 11–4 Arguments Used by the domainprop Utility

Argument	Description
-d *pathToDomain*	Specifies the domain path.
-property *propertyname=value*	Specifies the property to be changed. See the table of available properties below that can be set with this utility. To view the current property setting, use the property command with no value.
-reportSubDomains	Shows property values for subdomains (should all match).

Available Properties

Note: The table below lists all of the available properties of domainprop. However, the Fusion Client recognizes only `insert_measure_disabled` and `meas_fillclr_precedence`.

Table 11–5 Available Properties for the domainprop Utility

Domain Property Name	Type	Description
disable_commit_asap	Boolean	This property is not applicable to the Fusion Client because there is only one commit option.
disable_commit_later	Boolean	This property is not applicable to the Fusion Client because there is only one commit option.
domain_name	String	If this property is set, the domain name is displayed on the About page of the RPAS Client (menu Help >About).
help_path	String	This property is not applicable to the Fusion Client.
insert_measure_disabled	Boolean	If this property is set to TRUE, the "insert measure…" item under the Edit menu is disabled. By default, this property is set to FALSE.
measure_locking_disabled	Boolean	By default, this property is set to FALSE. If this property is set to FALSE, the user can lock a measure on a work sheet. To disable measure locking, set this property to TRUE.
meas_fillclr_precedence	Boolean	By default, when deciding which color to fill a particular cell with, the RPAS grid uses the following order or formatting settings: Read-only, Measure, Hierarchical, and then Read/Write. That is, if the cell is in a read-only state, it will use the read-only formatting setting. However, if that is not the case, the grid will check if there is any Measure level formatting. Failing to find it will fall through to checking for the hierarchical setting and then the read-write setting. However, some customers want RPAS to follow a different priority order for fill color formatting decision making. They want it to try Measure, then Read-only, then Hierarchical, and finally Read/Write. This change from default can be made by setting this domain property to TRUE. To reset behavior, this domain property can be reset to FALSE.

Table 11–5 (Cont.) Available Properties for the domainprop Utility

Domain Property Name	Type	Description
ovr_def_admin_privileges	Boolean	Using the Security Administration workbook, administrators can set workbook template access for every user in the system. Non-administrative users cannot access the workbook templates to which they have not explicitly been given access. However, if a user is an administrator, by default, they can see all the workbooks in the system. Some retailers want to prevent this from happening. Reasons for this include reducing clutter and having different kind of administrators manage different administrative tasks in their RPAS systems. Ability to control template access for administrators from the Security Administration workbook is made possible by setting this domain property to TRUE. By default, this property is FALSE.
r_upgrade_hash_alg	Integer	Sets the value for r_crypt_hash_alg in the upgraded domain. Because the list of algorithms can change with each release, it is not feasible to provide a list here. To get the list of values that can be used here, go to the rpasHome of the version to which you are upgrading and run: usermgr -listHashAlgs
r_upgrade_has_iter	Integer	This property sets the value for r_crypto_hash_iter in the upgrade process. It is the number of iterations of hashing to perform on the password.

Calculation Engine Using mace

The mace utility (Multi-dimensional Array Calculation Engine) allows the administrator to evaluate rule groups or expressions in order to manipulate measures. The mace utility 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

Parallelization: The mace utility can execute in parallel under the following circumstances:

1. The utility must be invoked on a master domain.
2. Parallelization is only applicable to single-expression evaluation (-run -expression argument). Parallelization does not apply to rule group evaluation.
3. The evaluated expression cannot be a SpecialExpression.

4. All of the measures appearing on the left hand side of the expression must be non-HBI; that is, the base intersection of the measures must be below the partition level.

The mace utility creates multiple child processes based on the `-processes` argument and each child mace process evaluates the expression in one local domain. This functionality enables mace to achieve higher levels of CPU utilization using parallelization on systems with multiple CPUs. It also simplifies the user script when the same expression must be evaluated in all local domains.

Centralization: When mace is run on a master domain, the following command line options apply to the master as well as all local domains. For example, running `mace -d domain -newRule` ... creates a new rule in the master and all local domains.

- -newRule: create a new rule in the domain
- -delRule: delete an existing rule from the domain
- -addRule: add a new rule to a specific rule group
- -removeRule: remove an existing rule from a specific rule group
- -newGroup: create a new rule group in the domain
- -remove Group: remove an existing rule group from the domain
- -addExpression: add an expression to a specific rule
- -purgeRules: remove all rules not contained in any rule group from the domain
- -removeAllRuleData: remove all rule and rule group data from the domain

The behavior and usage of the following commands is unchanged:

- -find: search all expressions for the specific measure and print all rules/rule groups that use it
- -check: validate the specific expression
- -resolve: order but do not evaluate expressions within the rule group
- -transit: rule calc engine by transitioning over a list of rule groups
- -print: print all specific rules and groups
- -validate: validate rule groups

mace Usage

```
mace -version
mace -d domainPath -find string
mace -d domainPath -newRule {-ruleName ruleName} {-label ruleLabel}{-processes
numProcesses}
mace -d domainPath -delRule ruleName (-processes numProcesses}
mace -d domainPath -addRule     groupName:ruleName {-processes numProcesses}
mace -d domainPath -removeRule  groupName:ruleName {-processes numProcesses}
mace -d domainPath -newGroup    groupName {-label groupLabel}{-processes
numProcesses}
mace -d domainPath -removeGroup groupName {-processes numProcesses}
mace -d domainPath -addExpression ruleName -expression exprString{-processes
numProcesses}
mace -d domainPath -check -expression exprString
mace -d domainPath -run -group groupName      {-debugRuleEngine}
mace -d domainPath -run -expression expString {-processes numProcesses}{-
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 {-processes numProcesses}
mace -d domainPath -removeAllRuleData {-processes numProcesses}
mace -d domainPath -validate calc    -ruleGroup groupName
mace -d domainPath -validate general -ruleGroup groupName
mace -d domainPath -validate refresh -ruleGroup groupName -calcRuleGroup calc
```

Table 11–6 provides descriptions of the arguments used by the mace utility.

Table 11–6 Arguments Used by the mace Utility

Argument	Description
`-d` *domainPath*	Specifies the domain in which to load the measure.
`-find` *string*	Searches all expressions for the specified string, printing all the rules and rule groups that have these expressions.
`-newRule` {-ruleName *ruleName}*	Creates a new empty rule. If desired, use the *-ruleName* argument to specify a name for the rule.
`-label` {ruleLabel | groupLabel	Specifies the label of the rule with the -newRule argument or label of the group with the -newGroup argument.
`-processes` *numProcesses*	Specifies the number of child processes to be run in parallel.
`-delRule` *ruleName*	Removes the specified rule.
`-addRule` *groupName:ruleName*	Adds the specified rule to the group specified by *groupName*.
`-removeRule` *groupName:ruleName*	Removes the specified *ruleName* from the group specified by *groupName*.
`-newGroup` *groupName*	Creates a new rule group with the specified name.
`-removeGroup` *groupName*	Removes the specified group and non-shared rules in it.
`-addExpression` *ruleName*	Adds an expression to the specified rule. -expression should be used with this argument.
`-check`	Validates the specified expression. -expression should be used with this argument.
`-run`	Evaluates the specified expression or rule group. The -expression argument should be used with this argument.
`-resolve` *groupName*	Orders (does not evaluate) expressions within rule group. Requires a comma-separated list of edited measures.
`-transit` *workbookName*	Runs 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}	Prints 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.

Table 11–6 (Cont.) Arguments Used by the mace Utility

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

mace Restrictions in Global Domains

Problems were found when running simple rule groups using the mace utility. Mace did not allow executing a rule group consisting of non-HBI measures from the master domain for the following reasons:

1. Even though the rule group contains only non-HBI measures, mace was not designed to run it from the master domain and spawn it across local domain as it required too many checks to make it work differently for special expressions as sometimes the special expressions could take more than one left-hand side measure and some of them could be HBI and some non-HBI.
2. It is not a properly configured rule group if the expressions in the rule group have both HBI and non-HBI measures which need to be evaluated. RPAS may not be able to detect this at the time of configuring the solution as the partition dimension can be changed in the globalDomainConfig.xml.
3. The rule group may contain special expressions and regular RPAS expressions (as in 1 and 2) and that adds to the complexity to the rule engine to properly evaluate these expressions when run from the master domain.
4. The above restrictions do not apply when mace is run on a single expression where the LHS can be a HBI or non-HBI measure.

There is only one situation where a rule group can be run from the master domain and that is when all the LHS measures of the expressions in the rule group are HBI measures. This type of rule group cannot be run from local domains and needs to be run from the master domain.

Proper validation is added to the mace utility to identify these conditions and flag them to the user and exit gracefully. The *Oracle Retail Predictive Application Server Administration Guide* will be updated with these additional validation points in a future release. The following validations are added:

1. If mace is executed against a master domain and the executing expressions within the rule group contain non-HBI measures on the left-hand side of expressions, excluding special expressions, mace exits with an exception.
2. If mace is executed against a local domain and the executing expressions within the rule group contain an HBI measure on the left-hand side, excluding special expressions, mace outputs a warning message but the execution continues.

The end result is that mace cannot execute a rule group that contains non-HBI measures on the LHS against the master domain and provides a warning when running a rule group with HBI measures on the LHS against a local domain.

Managing the Workbook Batch Queue Using wbbatch

The wbbatch utility is used to manage workbook batch categories and 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 clients. It is also used to add, update, and delete batch categories, to update assignments of workbook build entries to workbook batch categories, to provide workbook batch categories when adding workbooks to the refresh queue, and to update the assignments to workbooks already in the refresh queue.

RPAS can be used 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 using this utility.

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

wbbatch Usage

```
wbbatch -version
wbbatch -d pathToDomain -build queueIndex
wbbatch -d pathToDomain -refresh workbookName
wbbatch -d pathToDomain -commit workbookName
wbbatch -d pathToDomain -scheduleRefresh workbookName [-category categoryName]
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]
[-categories catName1,catName2]
wbbatch -d pathToDomain -printQueue [all|build|refresh|commit]
wbbatch -d pathToDomain -listCategories
wbbatch -d pathToDomain -addCategories catName1:Label1,catName2:Label2
wbbatch -d pathToDomain -deleteCategories catName1,catName2
wbbatch -d pathToDomain -changeCategoryLabels catName1:NewLabel1,
catName2:NewLabel2
wbbatch -d pathToDomain -queue build -updateCategories queueIndex1:newCatName1,
queueIndex2:newCatName2
wbbatch -d pathToDomain -queue refresh -updateCategories
workbookName1:newCatName1, workbookName2:newCatName2
```

Table 11–7 describes the arguments used by the wbbatch utility.

Note: Fusion Client users cannot submit workbooks into the commit later queue; therefore, the commit arguments in the wbbatch utility do not apply.

Table 11–7 Arguments Used by the wbbatch Utility

Argument	Description
-d pathToDomain	Specifies the domain containing the workbooks.
-build *queueIndex*	Runs workbook build for provided *queueIndex*.
-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.
-commit *workbookName*	This argument is not applicable to the Fusion Client because it does not allow entries into the commit later queue.
-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 refresh batch queue. If the -category option is specified, the scheduled workbook will be in that category. Otherwise, it will be in the default category.
-unscheduleRefresh *workbookName*	Removes a workbook from the workbook refresh batch queue.
-scheduleCommit *workbookName*	This argument is not applicable to the Fusion Client because it does not allow entries into the commit later queue.
-unscheduleCommit *workbookName*	This argument is not applicable to the Fusion Client because it does not allow entries into the commit later queue.
-startQueue	Runs all workbooks in provided queue. The queue options are build, refresh, and commit. If the -category option is used and one or more categories are specified, only the workbooks in those categories are built or refreshed. Categories do not apply to committing. Since the commit later queue is not applicable to the Fusion Client, the commit queue is always empty.
-printQueue	Prints the contents of the queue argument. The queue indexes for auto workbooks in the build queue are shown when printing the build queue. If “all” is specified, all three queues (build, refresh, and commit) are displayed. Since the commit later queue is not applicable to the Fusion Client, the commit queue is always empty.
-listCategories	Lists both the name and label for all categories.

Table 11–7 (Cont.) Arguments Used by the wbbatch Utility

Argument	Description
-addCategories *Name1:Label1, Name2:Label2, Name3:Label3*	Adds a new category by providing a name and label, separated by a colon. Multiple categories can be specified on the same command line if separated by a comma. If the users use a different language other than the one typed in the command line, the administrator should use the Workbook Batch Category Management wizard to create new categories.
-deleteCategories *catName1, catName2*	Deletes a category by specifying the name of that category. Multiple categories can be deleted if separated by a comma.
-changeCategoryLabels *Name1:NewLabel1, Name2:NewLabel2*	Changes the label of an existing category by specifying the category name and providing a new category label. If the users use a different language other than the one typed in the command line, the administrator should use the Workbook Batch Category Management wizard to change category labels.
-updateCategories *queueIndex1:newCatName1, queueIndex2:newCatName2* Or: -updateCategories *wbName1:newCatName1, wbName2:newCatName2*	Updates the category for an entry in the build queue or changes the workbook category of an existing entry in the refresh queue. Multiple category assignment for workbook auto build queue entries and refresh entries can be updated. If using the -build option, list the queue index. If using the -refresh option, list the workbook name. **Build example:** wbbatch -d pathToDomain -queue build -updateCategories *queueIndex1:newCatName1*, *queueIndex2:newCatName2* **Refresh example:** wbbatch -d pathToDomain -queue refresh -updateCategories *workbookName1:newCatName1*, *workbookName2:newCatName2*

Workbook Manager with 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 removing them manually.

wbmgr 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,...
```

Table 11–8 provides descriptions of the arguments used by the wbmgr utility.

Table 11–8 wbmgr Utility Arguments

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.

Table 11–8 (Cont.) wbmgr Utility Arguments

Argument	Description
-remove -all	Removes all workbooks from the domain.
-remove -user *userName*	Removes all workbooks from the domain belonging to the specified user.
-remove -user *userName* -wbList *wb1,wb2*	Removes all the workbooks in the specified list for the specified user.

Registering Measures Using regmeasure

The regmeasure utility is used for batch measure registration. The following functionality is included:

- Register a new measure in the user-specified domain with the user-specified measure properties. If the domain specified by the user is a global domain, this measure will be registered in the master domain and all its local domains. The user must provide a minimum set of measure properties, type and base intersection. Other measure properties, such as default aggregation and spreading method, are optional. If the user omits an optional measure property, the measure will be registered with default value of that property.
- Unregister an existing measure identified by its name from the user-specified domain. If the specified domain is a global domain, this measure will be removed from the master domain and all local domains. Unregistering a measure from a domain causes the measure definition and all the related measure data arrays and supporting arrays to be removed from the domain.
- Modify measure properties of an existing measure. Not all measure properties, such as type, base intersection, NV value, and database name, can be modified. These properties cannot be changed once the measure is registered. Measure properties such as default aggregation method, default spread method, base state, agg state, and so on can be modified after the measure is registered.

regmeasure Usage

```
regmeasure -version
regmeasure -d pathToDomain -add measureName -type typeName(-baseint
baseIntersection|-scalar) {-label labelString} {-db dataDbPath}{-navalue naValue}
{-defagg aggType} {-defspread spreadType}{-allowedaggs "aggType1 aggType2"}{-
refreshable (true|false)} {-insertable (true|false)}{-basestate (read|write)} {-
aggstate (read|write)}{-stageonly (true|false)} {-filename fileName}{-loadint
loadIntersectionString} {-clearint clearIntersectionString}{-loadstokeep
loadsToKeep} {-start fieldStart} {-width fieldWidth}{-loadagg loadAgg} {-range
range} {-purgeage purgeAge} {-viewtype viewType}{-syncwith syncWith} {-description
descriptionString} {-picklist}{-materialized (persistent|display)}{-lowerbound
measurename} {-upperbound measurename}{-attr attrName -attrpos attrPosName} {-
scriptname scriptName}{-specialval
action:specval:behavior,action:specval:behavior,...} {-fnhbi}{-hybridaggspec
hiername:aggop,hiername:aggop,...}{-periodstartvalue (true|false)}

regmeasure -d pathToDomain -modify measureName {-label labelString}{-defagg
aggType} {-defspread spreadType} {-allowedaggs "aggType1 aggType2..."}{-
refreshable (true|false)} {-insertable (true|false)}{-basestate (read|write)} {-
aggstate (read|write)}{-stageonly (true|false)} {-filename fileName}{-clearint
clearIntersectionString}{-loadstokeep loadsToKeep} {-start fieldStart} {-width
fieldWidth}{-loadagg loadAgg} {-range rangeString} {-purgeage purgeAge|-
clearPurgeAge}{-viewtype viewType} {-syncwith syncWith} {-description
descriptionString}{-picklist|-nopicklist} {-materialized (persistent|display)}{-
lowerbound measureName} {-upperbound measureName}{-attr attrName -attrpos
attrPosName} {-scriptname scriptName}{-specialval
action:specval:behavior,action:specval:behavior,...}{-hybridaggspec
hiername:aggOp,hiername:aggOp,...}{-periodstartvalue (true|false)}

regmeasure -d pathToDomain -remove measureName
```

Table 11–9 provides descriptions of the arguments used by the regmeasure utility.

Table 11–9 regmeasure Utility Arguments

Argument	Description
-d *pathToDomain*	Specifies the path to the domain. A valid domain path must be specified.
-add *measureName*	Adds a measure with the specified name. Set the values for the measure by using the required arguments and any of the optional arguments. Measure names can be up to 30 characters long.
-type *typeName*	Specifies the measure data type. It can be set to int, real, string, date, or boolean. Required with the -add option. Not available with the -modify option.
-baseint *baseIntersection* | -scalar	Specifies the base intersection of the measure. Non-scalar measures must use the -baseint option. Scalar measures must use the -scalar option. Required with the -add option. Not available with the -modify option.
-label *labelString*	Specifies the measure label. If not specified, it defaults to the measure name specified for the -add option.
-db *dataDbPath*	Specifies the database path for the measure's data arrays. A valid database path name must be specified. If not specified, the measure will be registered without a database. As a result, the measure will not be able to store any data in the domain. However, if the measure is not a Display only type, it will still be assigned a database in the workbook. Not available with the -modify option.
-navalue *naValue*	Specifies the NA value for the measure's base level data array. The NA value must be the same type as the measure. For date, the NA value must be formatted as 'YYYYmmddHHMMSSsss'. If not specified, it defaults to the type's default value: 0 for numeric type, false for boolean type, an empty string for string type, and 0001/01/01 for date type. Not available with the -modify option.
-defagg *aggType*	Specifies the default aggregation method for the measure. It must be an aggregation name valid for the type of measure. For a list of valid aggregation type names, see the *RPAS Configuration Tools User Guide*. If not specified, it defaults to the measure type's default aggregation method: Total for int and real, Ambig for string and date, and OR for boolean.
-defspread *spreadType*	Specifies the default spread method for the measure. It must be a spread method valid for the type of measure. For a list of valid spread methods, see the *RPAS Configuration Tools User Guide*. If not specified, it defaults to the measure type's default spread method: Ratio for int and real, and Replicate for string, date, and boolean.

Table 11–9 (Cont.) regmeasure Utility Arguments

Argument	Description
-allowedaggs "aggType1 aggType2..."	Specifies a list of aggregation methods that are allowed for this measure. The aggregation methods must be valid for the type of measure. If not specified, it defaults to the default allowed aggs for the type of measure. For numeric (int or real type) measures: total, total_pop, first, first_pop, last, last_pop, min, min_pop, max, max_pop, average, average_pop, popcount, nobcount, ambig, ambig_pop, none, period_start_total, period_end_total, period_start_average, period_end_average, median, median_pop, recalc, hybrid. For string type measures: ambig, ambig_pop, none, popcount, nobcount, first, first_pop, last, last_pop, recalc, hybrid. For date type measure: ambig, ambig_pop, pop_count, nob_count, first, first_pop, last, last_pop, min, min_pop, max, max_pop, non, recalc, hybrid. For boolean measure: boolean_and, boolean_or, pop_count, nob_count, ambig, ambig_pop, none, first, first_pop, last, last_pop, recalc, hybrid.
-refreshable (true | false)	Note: This option is no longer supported but is kept for compatibility.
-insertable (true | false)	Specifies whether the measure can be dynamically inserted into the workbook. If not specified, it defaults to true.
-basestate (read | write)	Specifies the workbook access right for the base array of the measure. If not specified, it defaults to read. The access rights of this measure are further restricted by the RPAS security features. As a result, write access specified by this option does not guarantee write access of this measure in a specific workbook.
-aggstate (read | write)	Specifies the workbook access right for the aggregated level of the measure. If not specified, it defaults to read. The access rights of this measure are further restricted by the RPAS security features. As a result, write access specified by this option does not guarantee write access of this measure in a specific workbook.
-stageonly (true | false)	Specifies whether the measure is a stage only measure. If not specified, it defaults to false. Measure data loaded by loadmeasure for stage-only measures is not automatically applied to the measure's base data array. User intervention is usually required to manually approve the loaded measure data and apply the approved loads to the measure's base data array.
-filename fileName	Specifies the file name of this measure's loading file. It should not include any extensions. If not specified, it defaults to the measure name in lower case.
-loadint loadIntersectionString	Specifies the intersection to load data for this measure. It must be a valid intersection string that is either the same or lower than the base intersection of this measure. If loadint is lower than the base intersection of the measure, the aggregation method specified by the -loadagg option will be used to aggregate the loaded data to the base array of the measure. Not available with the -modify option.
-clearint clearIntersectionString	Specifies the clear intersection for the clear load of this measure. For more information on the various loading methods including clear load, refer to the Loading Measure Data - loadmeasure section in this guide.
-loadstokeep loadsToKeep	Specifies the number of temporary measure load arrays to be kept in the staging database. If not specified, it defaults to 1.

Table 11–9 (Cont.) regmeasure Utility Arguments

Argument	Description
-start fieldStart	Specifies the starting column of this measure's data in the measure loading file. If not specified, it is calculated based on the loadint of the measure.
-width fieldWidth	Specifies the number of characters this measure's data occupies in the measure loading file. If not specified, it defaults to the default width of the measure type: 8 for integer, real, and date, 24 for string, and 1 for boolean.
-loadagg loadAgg	Specifies the aggregation method used to aggregate the temporary load array to the measure's base array if the measure's loadint is lower than its baseint. If not specified, it defaults to the measure type's default aggregation method: Total for int and real, Ambig for string and date, and OR for Boolean.
-range rangeString	Specifies the valid range for the measure. The value of the range parameter depends on the measure type. For int or real types, the format is min:max, where min is the lowest possible value of the measure and max is the highest possible value of the measure. For picklist measures, to give the allowed options, the format of the string argument is 'a(Label A),b(Label B),c,d', where a, b, c, and d are allowed measure values and Label A and Label B are optional labels for the values. In addition, the list of allowed options can be changed dynamically with the cell the user is clicking in. For this functionality, the measure's range is specified as 'measurerange=measurename' where measure name is the name of the measure that contains strings in the above format of value/label pairs. For date types, the range must be in the format `MMddyyyyhhmmss:MMddyyyyhhmmss`, where the first string is the starting date and time of the range and the second date is the ending date and time of the range. If you omit the time portion of the string (hhmmss), the default time is used. The default lower bound is 000000 (12:00:00AM), which is used for the beginning of the day. The default upper bound is 235959 (11:59:59PM), which is used for the end of the day. If the range begins with a negative number (which may confuse the command-line argument parser), enclose the entire range string in square brackets, such as -range [-10:10].
-purgeage (purgeAge)	Specifies the number of days (or whatever the base dimension of the calendar hierarchy is) of measure data that should be kept in the measure's base data array after measure load. This is used to keep the measure's data size small. If not specified, it defaults to -1, in which case the measure data will never be purged. When using the -modify option, -purgeage or -clearPurgeAge can be specified.
-clearPurgeAge	Resets the number of days (or whatever the base dimension of the calendar hierarchy is) of measure data that should be kept in the measure's base data array after measure load to -1. This means that the measure data will never be purged. clearPurgeAge is only available with the -modify option. When using the -modify option, -purgeage or -clearPurgeAge can be specified.

Table 11–9 (Cont.) regmeasure Utility Arguments

Argument	Description
-viewtype viewtype	Specifies the view type of this measure on the RPAS Client. The valid values are: 0 for none, 1 for view_only, 2 for sync_first_lag, 3 for sync_lead_last, 4 for sync_first, and 5 for sync_last. If not specified, it defaults to none. If the view type starts with "sync", the measure is called a 'Virtual Measure'. A measure of sync_first_lag type must have two sync measure names specified by the -syncwith option. The first syncwith measure name is a 'Period Start Value' type of measure, like opening stock. Measure data at the beginning period of the calendar is synchronized with this period start value kind of measure. The subsequent measure data is synchronized with the other measure data but lagged one period. A measure of sync_lead_last type must have two sync measure names specified by the -syncwith option. The first measure is a 'Period End Value' type of measure. Measure data at the last period of the calendar is synchronized with this period end value. Measure data of previous periods is synchronized with the other measure lead one period data. A measure of sync_first type must have one measure name specified by the -syncwith option. The data of the beginning period is synchronized with this syncwith measure. A measure of sync_last type must have one measure name specified by the -syncwith option. The data of the ending period is synchronized with this syncwith measure. Measures of view_only type are non-persistent. View only measures can only be used in workbooks. Their measure data is calculated during the Fetch process using a calc expression usually specified in the workbook's calc rule group.
-syncwith syncWith	Specifies the measures that the measure must be synchronized with. This option must be specified if the measure is not a virtual measure. For sync_first_lag and sync_lead_last measures, the syncwith option must have two measure names separated by a comma. The first measure is used to synchronize the data at the first or the last calendar period. The second measure is used to synchronize data at other periods. For sync_first and sync_last measures, the syncwith option must be specified with a single measure name that will be used to synchronize the first or last calendar period.
-description descriptionString	Specifies the description of the measure.
-picklist | -nopicklist	Specifies whether the measure is displayed as a picklist in the RPAS Client. The actual value of the picklist is specified by the -range option of the measure. The -nopicklist argument is only available with the -modify option. It means the measure should not be displayed as a picklist measure in the RPAS Client.

Table 11–9 (Cont.) regmeasure Utility Arguments

Argument	Description
-materialized (persistent | display)	Specifies whether the measure is either persistent or display only on the RPAS Server side. Persistent measures must have a valid database and arrays to store the measure data. Display only measures do not have permanent data arrays associated with it. The data for a display only measure must be calculated on the fly. As a result, display only measures can not be used on the RHS of any expression. Display dnly measures can still be used on the LHS of a calc expression used in a workbook, in which case a temporary array will be created in the workbook to hold the temporary data for the display measure.
-lowerbound measureName	Specifies a measure name that defines the lower bound for each cell of the measure. The difference between the -lowerbound and -range options is that the -range option specifies a single scalar as the lower bound for all cells of the measure, but the lower bound value specified by the -lowerbound option can be different from cell to cell.
-upperbound measureName	Specifies a measure name that defines the upper bound for each cell of the measure. The difference between the -upperbound and -range options is that the -range option specifies a single scalar as the upper bound for all cells of the measure, but the upper bound value specified by the -upperbound option can be different cell to cell.
-attr attrName	Specifies the measure attribute name. If not specified, it defaults to no attribute is assigned to the measure. **Note**: If this option is specified, the -attrpos option must also be specified.
-attrpos attrPosName	Specifies the measure attribute position name. Combined with the -attr option, the measure attribute provides a way to group measures together based on measure attributes. **Note**: If this option is specified, the -attr option must also be specified.
-scriptname scriptname	Specifies a shell script that must be executed as part of a specific event. Currently, the only script that is handled is to give the option of selecting a hierarchy position name as the content of a string measure. In other words, when a user clicks in a cell, the user is presented with a hierarchy dimension single-tree pop-up. The format for this is 'SingleSelect(HIER="<HIER>", DIM="<DIM>") where <HIER> and <DIM> should be replaced with the actual names of the hierarchy and dimension for which the single-tree pop-up should be created.
-specialval action:specval:behavior, action:specval:behavior,...	Specifies a list of measure special values in the form of "Action:SpecialValue:Behavior,…". The special values are stored in the domain's metadata database. For Action, the only action supported is: "DISPLAY". The only SpecialValue supported is "NAVAL". For Behavior, "NULL" means translate any NA cell to a blank cell for display. "CELLVALUE" means no translation, just display the NA value as a regular value.

Table 11–9 (Cont.) regmeasure Utility Arguments

Argument	Description
-fnhbi	Specifies that this measure is a forced non=HBI measure, which means that although the base intersection of this measure is above the partition dimension, the measure data must still be stored in each local domain. Not available with the -modify option.
-hybridaggspec hiername:aggOp,hiername:aggOp,...	Specifies the aggregation method to be used for each hierarchy in the base intersection. This option is only valid when the default aggregation method for the measure is hybrid.
-periodstatevalue (true|false)	Specifies that this measure stores a Period Start type of data, like beginning inventory. PeriodStart measures usually use Period Start Total or Period Start Average for the default aggregation method. It also has different behavior in elapsed lock. At the aggregated calendar level, if the starting period is elapse locked, then the whole aggregated period is locked.
-modify measureName	Modifies the measure with the specified name. Set the updated values for the measure by using any of the optional arguments.
-remove measureName	Removes the measure with the specified name.

Registering Token Measures Using regTokenMeasure

The regTokenMeasure utility is used to register, list, and remove RPAS Token Measures.

RPAS Token Measure provides placeholder functionality for measure names in RPAS expressions. An RPAS Token Measure is a special RPAS measure.

An RPAS Token Measure is always registered as a scalar measure of string type, with the measure property called tokenmeas set to true. Its measure data holds a valid value measure name as a single string. The data arrays for all token measures are stored in one database called token under the data directory in the RPAS domain.

Token measure can be used in RPAS expressions by prefixing `@` in front of the token measure name, either on the LHS or RHS of the expression. Before evaluation, @TokenMeasName in the expression is replaced with the value measure name that is associated with the token measure. As a result, the expression is evaluated against the value measure. A token measure name cannot be used in expression without using the prefix `@`.

In the following example, TM1 is a token measure registered with the value measure name VM1.

The following expression:

@TM1 = a + b

Is evaluated as:

VM1 = a + b

The following expression is not valid, because TM1 is used without prefixing it with `@`:

TM1 = "sth"

If mace is used for evaluation, it will throw a ParserException with the message that the token measure "TM1" is used without prefixing `@`. This functionality prevents the modification of the token measure's data, which is actually the value measure's measure name.

regTokenMeasure Usage

```
regTokenMeasure -version
regTokenMeasure -d pathToDomain -add tokenMeasure=valueMeasure {-fnhbi}
regTokenMeasure -d pathToDomain -list
regTokenMeasure -d pathToDomain -remove tokenMeasure=valueMeasure
```

Table 11–10 provides descriptions of the arguments used by the regTokenMeasure utility.

Table 11–10 regTokenMeasure Utility Arguments

Argument	Description
-d *pathToDomain*	Specifies the path to the domain. A valid domain path must be specified.
-add *tokenMeasure=valueMeasure*	Adds a token measure with the specified token measure name and value measure that the token measure points to.
-fnhbi	If specified, the token measure will be registered as an fnhbi measure in the global domain. Its data will be stored in each local domain rather than the global domain, although by definition the token measure should be always be HBI measure since it is scalar type.
-list	Prints all token measure names and the value measure names associated with the token measure, which are registered in the domain specified by the -d option.
-remove tokenMeasure=*valueMeasure*	Removes the token measure with the specified token measure name and value measure. The token measure is unregistered from the domain specified by -d option. Unregistering the token measure has no side effect to the value measure that the token measure is associated with.

Batch Plug-In Tasks: execPluginTask.sh

Many RPAS applications can be used to extend the functionality of the RPAS Configuration Tools and the rpasInstall process. Theses extensions are provided by plug-ins included as part of the solution. Most plug-in extensions execute either within the RPAS Configuration Tools or within the rpasInstall process. However, some applications may include plug-in extensions that can be executed from a command prompt.

The execPluginTask.sh utility is the method by which these command line plug-in tasks can be performed. When execPluginTask.sh is executed, it loads a configuration, locates the desired plug-in extension, and executes that extension on the loaded configuration. Then, depending on the task being executed, the configuration may or may not be saved. Extensions that do not modify the configuration, for example, ones that generate resources based on the contents of the configuration, do not require saving the configuration at the conclusion of the automation.

Individual applications document the command line extensions they support, if any, along with information on how those extensions may be used.

execPluginTask.sh usage

In addition to the standard arguments described below, it is possible to pass any number of additional arguments to execPluginTask.sh. These additional arguments are passed to the extension being executed to allow control over its operation. Details on the arguments any given command line extension accept and their function are located in the application-specific documentation of the extension.

```
execPluginTask.sh plug-inName:taskName pathToConfiguration {plug-inTaskArguments}
execPluginTask.sh -version
execPluginTask.sh -help
```

Table 11–11 Arguments Used by the execPluginTask.sh Utility

Argument	**Description**
plug-inName	Indicates the name of the plug-in containing the extension to be executed.
taskName	Indicates the name of the extension to be executed.
pathToConfiguration	The path to the configuration execPluginTask.sh should load to execute the extension. As with the rpasInstall process, paths passed as arguments should be absolute paths and not relative paths.
plug-inArguments	Additional arguments that are passed to the extension.
-version	Prints the version information for execPluginTask.sh.
-help	Prints the usage statement for execPluginTask.sh.

12

Informational Utilities

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

- Retrieving Domain Information Using domaininfo
- Checking the Validity of a Domain Using checkDomain
- Determining RPAS Server Version Using rpasversion
- List Contents of a Database Using listDb
- Printing Data from Arrays Using printArray
- Printing Data from Measures Using printMeasure

Retrieving Domain Information Using 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.

domaininfo Usage

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

Note: Domain path (-d) is required for all options except `-expectedversion`.

Table 12–1 provides descriptions of the arguments used by the `domaininfo` utility.

Table 12–1 Arguments Used by the domaininfo Utility

Argument	Description
`-d`	Indicates the path to the domain. Required for all options except `-expectedversion`.
-domainversion	Displays the RPAS version of the specified domain.
-expectedversion	Displays the expected RPAS version of the domain that the utility expects to find.

Table 12–1 (Cont.) Arguments Used by the domaininfo Utility

Argument	Description
-type	Displays 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 or master domain of a global domain environment. A Subdomain is one local domain in a global domain environment that can contain one or more partitions.
-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).
-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.
-masterdomaininfo	Lists the master domain path and partition dims for subdomains.
-all	Displays the domain version, expected domain version, domain type, associated application, history, subdomains, and master domain info (where applicable).
`-domainsize`	Displays the file size information for the domain.
`-stringstats`	Displays the number of strings of all given lengths that occur in the domain. The output returns a list by string length, for example:

```
0    758
1     21
2     8
3     69
...
```

`-stringvalues`	Displays how many occurrences of each unique string there are in the domain. The output returns a list of each unique string, for example:

```
785 ''
 1 '#copied <> #pasted'
 2 '%'
 1 '%1 <= passwd <= %2'
 2 '%1 Window'
 ...
```

`-arraydensity`	Displays array btree density.
`-arrayschemas`	Displays how many arrays in the domain are formatted for each schema. The output returns a list like the following:

```
Count of Array Schemas for Domain ‘domain’
Array schema 10: 371
Unknown schema: )
```

-version	Displays the version of this utility.
-subdomain *dim,pos*	Indicates to which local domain the specified position belongs. The position can be at or below the partition level.
`-registrypaths`	Displays all array paths to dimRegistry.

Table 12–1 (Cont.) Arguments Used by the domaininfo Utility

Argument	Description
-showrelativepaths	Indicates if paths are relative when listing subdomains. Only relevant in combination with -listsubdomains or -all.
`-terse`	Removes the header information from the output, thereby returning only the requested information.

Checking the Validity of a Domain Using checkDomain

The checkDomain 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.

checkDomain Usage

```
checkDomain -d pathToDomain -type expectedType {-q}
```

Table 12–2 provides descriptions of the arguments used by the `checkDomain` utility.

Table 12–2 Arguments Used by the checkDomain Utility

Argument	Description
`-d` *pathToDomain*	Indicates the path to the domain that needs to be validated.
`-type` *expectedType*	Indicates the expected type of domain: simple, master, or sub.
`-q`	Indicates 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 a simple domain

If checkDomain is run on a global domain, it verifies the following:

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

If it is run on a subdomain, 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 subdomain.

Determining RPAS Server Version Using rpasversion

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

rpasversion Usage

```
rpasversion -l pathToLibrary
```

List Contents of a Database Using listDb

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

listDb Usage

```
listDb pathToDb*
listDb -row -db pathToDb*
listDb -row -pageUsage -db pathToDb*
listDb -row -standardOptions -db pathToDb*
listDb -standardOptions -db pathToDb*
listDb -version
```

Table 12–3 provides descriptions of the arguments used by the listDb utility.

Table 12–3 Arguments Used by the listDb Utility

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

Printing Data from Arrays Using printArray

Use printArray to print the contents of an array.

printArray Usage

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

Table 12–4 provides descriptions of the arguments used by the printArray utility.

Table 12–4 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. The db is a full or relative path to a database. Do not specify the .ary 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 in order 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 NA value 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 Using printMeasure

Use the printMeasure utility to print measure information.

printMeasure Usage

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

Table 12–5 provides descriptions of the arguments used by the printMeasure utility.

Table 12–5 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`	Returns a list of all registered measures in the domain. This argument does not require `-d domainPath`.
`-listHBIMeasures`	Returns a list of all measures registered at or above the partition dimension in a global domain.
`-specs`	Returns the list of measure properties. Requires the `-m` parameter.
`-listDataIntersections`	Returns the base intersection of the measure.

13

Internationalization

Internationalization is the process of creating software that can be translated more easily. Changes to the code are not specific to any particular market. This section describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Oracle Retail applications have been internationalized to support 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 include the following:

- Graphical user interface (GUI)
- Error messages
- Taskflow Components

The following components are not translated:

- Documentation (online help, release notes, installation guide, user guide, operations guide)
- Batch programs and messages
- Log files
- Configuration tools
- Reports
- Demonstration data
- Training materials

The user interface has been translated into the following languages:

Note: In Table 13–1, the language identifier is used for position labels. For more information, see "Position Label Translation". The Windows Language ID is in the foundation.ini file. For more information, see "Translation Administration".

Table 13–1 Supported Languages with Language Identifiers

Language	Language Identifier	Windows Language ID
Chinese (Simplified)	CHINESE_SIMPLIFIED	2052
Chinese (Traditional)	CHINESE_TRADITIONAL	1028
Croatian	CROATIAN	26
Dutch	DUTCH	19
English	ENGLISH	9
French	FRENCH	12
German	GERMAN	7
Greek	GREEK	8
Hungarian	HUNGARIAN	14
Italian	ITALIAN	16
Japanese	JAPANESE	17
Korean	KOREAN	18
Polish	POLISH	21
Portuguese (Brazilian)	PORTUGUESE	22
Russian	RUSSIAN	25
Spanish	SPANISH	10
Swedish	SWEDISH	29
Turkish	TURKISH	31

Translation Administration

Note: For information on the translation of position labels, see "Position Label Translation".

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 processes must be followed to setup the environment to support multiple languages.

1. Build the domains with the Multi-Language setting enabled in the Configuration Tool properties.
2. Change the Web browser locale settings to reflect the relevant language settings in the application user interface using the following steps:
   a. Launch Internet Explorer.
   b. In the Tools menu, click **Internet Options**. The Internet Options window appears.
   c. In the Internet Options window, on the General tab, click **Languages**. The Language Preference window appears.
   d. In the Language Preference window, click **Add** in the Language Preference area to add the relevant language.

   e. Select the language you added, and use the **Move Up** or **Move Down** buttons to set up the order of preference.
   f. On the Language Preference window, click **OK**.
   g. On the Internet Options window, click **OK**.

3. Log on to the RPAS Fusion Client, and access the Translation Administration workbook to set up the translated text for the measure labels, workbook template names, template group names, and other domain-related elements:
   a. On the Activity Taskflow pane, click the **Expand** icon next to Administration. The Administration panel appears.
   b. In the Administration panel, click the **Expand** icon next to Administration.
4. Under Administration, select the **Create New Workbook** icon next to Translation Administration. For more information, see "Translation Administration Workbook".

Translation Administration Workbook

The Translation Administration workbook contains views 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 View

The Hierarchy Labels view 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 View

The Dimension Labels view 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 View

The Template Group Translations view 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 view affect the labels on the tabs that appear in the File - New dialog (for example (in English), Administration, Analysis, and Predict).

Workbook Template Labels View

The Template Translation view 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 View

The Measure Translations view 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 View

The Measure Descriptions view 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 View

The User Group Translations view 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 View

The Message Labels view 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 View

The RGRP Labels view 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.

Taskflow Components Configuration

The default properties file, which is located in the config/MultiSolution/resources directory and is called MultiSolutionBundle.properties, contains the strings for labels, descriptions, and instructions for Activity Groups, Activities, Tasks, and Steps. This file also contains domain information for a multi-solution installation. Additional files can be created that contain strings for translated languages. These files are in the format of MultiSolutionBundle_XX.properties, where XX is the language code. Currently, RPAS supports base translations for the languages listed in " Supported Languages with Language Identifiers".

14

Commit as Soon as Possible

Commit As Soon As Possible (Commit ASAP) allows users to schedule the commit process for workbook data so that it executes as soon as all the system resources are available. Within the RPAS Fusion Client, the user has a single commit option available that runs a Commit ASAP process. The process to commit a workbook is described in the *RPAS User Guide for the Fusion Client*.

Notes:

- If a user attempts to commit a workbook that already has a process in the queue, the original processes will be removed from the queue. That means that only one commit in the queue can be pending for a given workbook/user/template name combination at any given time.
- Data within the workbooks can be committed without saving the workbook. When users click Commit, and the workbook has not been saved by then, the RPAS Fusion Client provides an option to Save, Commit, or perform both actions on the workbook.

This chapter contains the following sections:

- Managing the Workbook Queue Using showWorkbookQueues
- Commit ASAP Settings Using configCommitAsap

Managing the Workbook Queue Using showWorkbookQueues

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

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.

showWorkbookQueues Usage

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

Table 14–1 provides descriptions of the arguments used by the showWorkbookQueues utility.

Table 14–1 showWorkbookQueues Utility Arguments

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 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. An administrator must select to purge commit processes that either succeeded or failed.

Commit ASAP Settings Using 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 by which all pending processes must be completed, after which they will be cancelled and marked as failed.

Administrators may want to use this deadline before beginning the 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.

configCommitAsap Usage

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

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

Table 14–2 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*	Indicates the time of the day when all outstanding commit ASAP operations 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*	Sets the logger verbosity level. Possible values: all, profile, information, warning, error, or none.
-noheader	Disables 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 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 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 a 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 clients.

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 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 that is currently in the commit process.
- Success directory: Contains one file per submitted Commit that has successfully completed its commit process.

- Failed directory: Contains one file per submitted Commit that either had a failure during its commit process or could not be committed prior to the deadline.
- Unknown directory: If the Commit process detects a corrupted queue file, a message gets logged and the file gets moved into the unknown directory.

15

Batch Processes and RPAS Utilities

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. 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

This chapter contains the following sections:

- CSV File Format
- RPAS Utilities Logging Options
- Using Shell Scripts to Run Batch Processes
- Common Information and Parameters for RPAS Utilities
- RPAS Intraday Enabler

CSV File Format

For those utilities that use a comma-separated value (CSV) file, the following formatting applies for any commas or double quotation marks in the data:

- If the data does not contain any commas or double quotation marks, it does not need any special formatting.
- If the data contains a comma, the string must be enclosed between opening and closing double quotation marks.
- If the data contains quotation marks, the string must be enclosed between opening and closing double quotation marks and any embedded quotation marks must be paired.

Table 15–1 shows examples of the formatting.

Table 15–1 CVS File Format

Data	Formatted Data
Item 001	Item 001
Item 001, Soda	"Item 001, Soda"
"Large Screen" TV	""Large Screen"" TV"
Item 002, "Generic Brand" Cereal	"Item 002, ""Generic Brand"" Cereal"

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 the 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. No output is provided if the utility successfully executes.

Each of the lines that contains 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 that the message is an error.
- **W** indicates that the message is a warning.
- **I** indicates that the message is informational.
- **U** indicates that the message is audit-relevant information.
- **P** indicates that the message is a performance profile.

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

Table 15–2 Utilities that Can Use the -loglevel Option

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

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 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 indicates the subdomain 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 are used to make the directory name unique. The framework attempts 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 sent to the screen and the log file.

These utilities are as follows:

- exportData
- loadmeasure
- loadHier
- exportMeasure
- reconfigGlobalDomainPartitions
- wbbatch
- reindexDomain

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.

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. It 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 b and `xx` are used to make the file name unique.

The number of these log files is 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, named as described 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 it is 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 is either `rpas Dyyyymmddhhmmbxx.log` or `rpas.log` base on the environmental variable RPAS_LOG_BACKUPS (see DomainDaemon).

loadHier

The loadHier utility uses standard logging. This utility performs part of its processing in child processes. `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 and date stamp, then the process id (pXXXX), and then b and a 2-digit number to avoid conflicts (bYY).

If any problems occur while loading specific records that belong to the partition hierarchy, they are reported in the format as shown below. Note that the record is completely reproduced in this error report in the log.

```
<E 2008Jul02 12:04:52.196> Could not find position '90000044' in line number 3:
'2001052090000044 1000 7 '. Skipping!
```

Problems with records along non-partitioned hierarchies are reported as shown in the following example:

```
<I 2008Jul02 12:04:55.482> MeasureLoader::loadDataFromFile() Loading '.ovr' file
'/vol.nas/u09/rpasqc/qc_testing/aix/1208rc2_test/RDF_12/ldom1/input/psal.ovr'
<D 2008Jul02 12:04:55.514> Error on line 1: '2001031110000044 STR1000 8 '
.Position name: STR_STR1000 not found.
<D 2008Jul02 12:04:55.514> Error on line 2: '2011041510000044 1000 9 ' .Position
name: DAY20110415 not found.
<D 2008Jul02 12:04:55.964> 2 lines had problematic data.
```

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.

reconfigGlobalDomainPartitions

Uses standard logging. The loadHier processes may be spawned as a child process. See "loadHier" 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 and de-registration. Each process creates a log file in a newly created directory in the domain output directory. The newly created directory name is formatted as `regServerYYYYMMDDHHMIbXX`, where `YYYY` is the year, `MM` is the month, `DD` is the day, `HH` is the hour, `and MI` is the minute. The character `b` and `XX` (two digits) are used to make the directory name unique. The RPAS libraries attempt to create at most eight directories for any single application.

Utilities and Database Locks

On UNIX and Linux platforms, before running any RPAS program, it is important to run "umask 0". Otherwise the RPAS databases may be permanently locked out to access by other users or groups. This applies to any program that accesses any RPAS database, including, but not limited to:

- convertDomain
- createRpasDomain
- DomainDaemon
- domaininfo
- exportMeasure
- ldrule
- listDb
- loadhier
- loadmeasure
- mace
- printArray
- printMeasure
- printMeasureInfo
- regmeasure
- rpasInstall
- RPASODBCAgent
- RPASODBCDataService
- usermgr

If database lock files (*.db.lck) have been created with limited permissions that block other users' access, a workaround is to run the following command when logged in as the lock-file-owning user:

```
$ find {domainPath} -user {owner} -name \*.lck -exec rm -f {} ';'
```

where {domainPath} is the path to the domain and {owner} is the UNIX user that owns the lock files.

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 are located 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 executes the script. In this example, it is defined to be the Korn shell. Therefore, this script is always 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.

Table 15–3 Standard Arguments for RPAS Utilities

Argument	Description
-version	Obtains the version information of the utility (for instance, RPAS 11.2.0). It does not require -d domainPath.
-d *pathtodomain*	Specifies the path to the domain against which the utility runs or from which data will be used.
-loglevel	See "RPAS Utilities Logging Options" for more information.
-n	Use 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	Disables the use of a timestamp in the header of the log file.
-help -? -usage	Any of these arguments 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 for a given utility. An administrator can set these levels for each RPAS utility. The available logger verbosity levels are as follows:

- none: Nothing is 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 line that contains the above type 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.

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 subdirectory 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.

RPAS Intraday Enabler

The RPAS Intraday Enabler (ride) functionality enables batch operations to be run over an RPAS domain while users are accessing workbooks and completing workbook operations.

This functionality enables batch operations to be executed over a domain, but does not prevent users from accessing other components that do not affect or interfere with the batch operations. The running of an exclusive batch process does not cause any pre-existing workbook operations that require domain access to fail or terminate. Users in domains that are not part of the exclusive process are not affected in any way.

In domains that have been locked by an exclusive batch process, the users are still able to perform operations that only require access to the workbook. The operations include the following:

- Workbook Edits
- Workbook Calculations
- Workbook Saves
- Workbook Opens
- Workbook Navigation
- Commit ASAP Entry

Users who enter workbooks into the commit ASAP queue while an exclusive lock is in place will have their entries processed once the exclusive process is complete.

In these same domains, users cannot perform operations that require access to data within the domain. The access can be either read or write. The operations that are prevented include the following:

- Workbook Build
- Workbook Refresh
- Workbook Custom Menu (unless configured as intraday-concurrent)
- Insert Measure
- Dynamic Position Maintenance (DPM)

When users try to access one of these operations after the exclusive lock is obtained, they see a message indicating that an exclusive process is running. A default message is provided, or it can be replaced with a message as part of the call to the ride utility.

When a user is working with a workbook in the master domain, a lock is required on the master domain and all local domains that are needed for the operation. The workbook operations in this domain are blocked when the master or local domain data is accessed by the ride process. See the Examples section for more details on domain access during different types of ride processes.

Configuration functionality is provided so that a custom menu can be flagged to run concurrently with a ride process (intraday-concurrent).

Note: See the *RPAS Configuration Tools User Guide* for details on how to configure a custom menu to be intra-day-concurrent. A custom menu that is configured to run concurrently with a ride process should only access workbook data, run a script that uses the ride utility, and run commits using the commit ASAP functionality. Custom menus that update or read directly from the domain should not be configured as intra-day-concurrent as this can conflict with the ride process.

In order for a batch job to run over a domain without interference from online activity, exclusive domain access must be granted to the job that is running. This is achieved by creating domain access control using a dual-lock control. The domain access control manages the lock request from workbooks and ride processes.

Figure 15–1 Domain Access Control

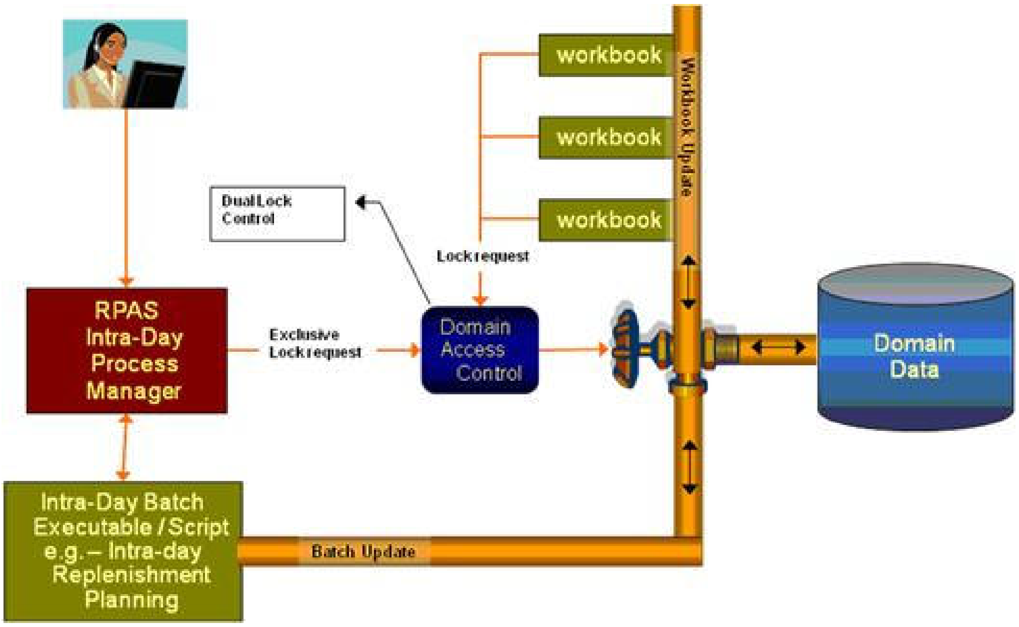

workbook
workbook
workbook
Workbook Update
Dual Lock Control
Lock request
RPAS Intra-Day Process Manager
Exclusive Lock request
Domain Access Control
Domain Data
Intra-Day Batch Executable / Script e.g. – Intra-day Replenishment Planning
Batch Update

Figure 15–1 shows the process control that is in place with the locking schema. In this case, an administrator requests exclusive access to a domain in order to run a batch job. This requires an exclusive lock on the domain so that the job can run. After the lock is received, no other workbook operations can obtain write access to the domain until the process is complete. If the exclusive lock cannot be obtained, the process should time out and the administrator notified based on the output of the ride utility. When the ride utility times out, the domainStatus utility is automatically run to provide details of the user workbook operations that are blocking the ride process. See "Domain Lock Status Using domainStatus" for details on that procedure.

ride Usage

```
ride -d domain -process pname|-script sname -args args {-message messageString} {-
timeout minutes} {-wait minutes} {-partitions pos1,pos2,…} {-masterInBatch}
```

Table 15–4 provides descriptions of the arguments used by the ride utility.

Table 15–4 Arguments Used by the ride Utility

Argument	Description
-d *domain*	Refers to a simple or master domain. When a master domain is specified, all local domains in the global domain environment, as well as the master domain, are locked.
-process *pname*	Indicates the name of the process to execute. This parameter cannot be used with the -script parameter.
-script *sname*	Indictates the name of the script to execute. This parameter cannot be used with the *-process* parameter.
-args *args*	Processes arguments that are passed to the script or the process to be executed. The -args parameter must be the last parameter or switch for this application. All parameters or switches after the -args parameter are passed on to the process or script to be started.
-message *messageString*	Indicates an optional override message that is presented to the user who is trying to perform an operation blocked by an intra-day batch process. A default message is provided to the user if this argument is not provided.
-timeout *minutes*	Indicates when a timeout will occur. The utility will time out if it cannot get access to the domains during this time. By default, there is no timeout. The timeout starts when the control utility is executed.
-wait *minutes*	Indicates the time to wait before starting the process or script. Even if domain access is granted, the process does not start until the end of wait time. The clock starts when the control utility is executed. The default is 0.
-partitions *pos1,pos2,…*	Indicates the partition positions (such as dept1, dept2, and so on) that determine the local domains that are accessed by the process or script.
-masterInBatch	When running over a global domain environment, the master domain will be accessed by the process or script in addition to any local domains selected.

Note: If neither -partitions nor -masterInBatch are provided on the command line, the entire domain will be processed when running over a global domain environment (that is, all subdomains and the master).

Scenarios

This section outlines several scenarios that are possible with the ride utility. The scenarios explain the domain access, based on how the ride utility is executed. The tables indicate whether the specific operations are blocked or allowed.

Scenario 1

This scenario describes running ride, specifying only the master domain. This locks all domains.

Usage: `ride -d master -script script.ksh`

Table 15–5 Running the ride Utility Specifying Only the Master Domain

Workbook Operation		Master Domain	Local Domain 1	Local Domain 2	Local Domain 3
Build	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Blocked	Blocked
Refresh	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Blocked	Blocked
Commit Now	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Blocked	Blocked
Custom Menu (not ride concurrent)	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Blocked	Blocked
Insert Measure	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Blocked	Blocked
DPM		NA	Blocked	Blocked	Blocked

Scenario 2

This scenario describes running ride specifying only one local domain (local domain with partition position 100).

Usage: `ride -d master -script script.ksh -partitions 100`

Table 15–6 Running the ride Utility Specifying One Local Domain

Workbook Operation		Master Domain	Local Domain 1	Local Domain 2	Local Domain 3
Build	HBI Measures	Blocked	Blocked	Allowed	Allowed
	No HBI Measures	Blocked	Blocked	Allowed	Allowed
Refresh	HBI Measures	Blocked	Blocked	Allowed	Allowed
	No HBI Measures	Blocked	Blocked	Allowed	Allowed
Commit Now	HBI Measures	Blocked	Blocked	Allowed	Allowed
	No HBI Measures	Blocked	Blocked	Allowed	Allowed
Custom Menu (not ride concurrent)	HBI Measures	Blocked	Blocked	Allowed	Allowed
	No HBI Measures	Blocked	Blocked	Allowed	Allowed
Insert Measure	HBI Measures	Blocked	Blocked	Allowed	Allowed
	No HBI Measures	Blocked	Blocked	Allowed	Allowed
DPM		NA	Blocked	Allowed	Allowed

Scenario 3

This scenario describes running ride specifying only one local domain (local domain with partition position 100) and the masterInBatch option.

Usage: `ride -d master -script script.ksh -partitions 100 -masterInBatch`

Table 15–7 Running the ride Utility Specifying One Local Domain and masterInBatch

Workbook Operation		Master Domain	Local Domain 1	Local Domain 2	Local Domain 3
Build	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Allowed	Allowed
Refresh	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Allowed	Allowed
Commit Now	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Allowed	Allowed
Custom Menu (not ride concurrent)	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Allowed	Allowed
Insert Measure	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Allowed	Allowed
DPM		NA	Blocked	Blocked	Blocked

Scenario 4

This scenario describes running ride specifying two local domains (local domain 1 with partition position 100 and local domain 2 with partition position 200).

Usage: `ride -d master -script script.ksh -partitions 100,200`

Table 15–8 Running the ride Utility Specifying Two Local Domains

Workbook Operation		Master Domain	Local Domain 1	Local Domain 2	Local Domain 3
Build	HBI Measures	Blocked	Blocked	Blocked	Allowed
	No HBI Measures	Blocked	Blocked	Blocked	Allowed
Refresh	HBI Measures	Blocked	Blocked	Blocked	Allowed
	No HBI Measures	Blocked	Blocked	Blocked	Allowed
Commit Now	HBI Measures	Blocked	Blocked	Blocked	Allowed
	No HBI Measures	Blocked	Blocked	Blocked	Allowed
Custom Menu (not ride concurrent)	HBI Measures	Blocked	Blocked	Blocked	Allowed
	No HBI Measures	Blocked	Blocked	Blocked	Allowed
Insert Measure	HBI Measures	Blocked	Blocked	Blocked	Allowed
	No HBI Measures	Blocked	Blocked	Blocked	Allowed
DPM		NA	Blocked	Blocked	Allowed

Scenario 5

This scenario describes running ride specifying two local domains (local domain 1 with partition position 100 and local domain 2 with partition position 200) and the masterInBatch option.

Usage: `ride -d master -script script.ksh -partitions 100,200 -masterInBatch`

Table 15–9 Running the ride Utility Specifying Two Local Domains and the masterInBatch Option

Workbook Operation		Master Domain	Local Domain 1	Local Domain 2	Local Domain 3
Build	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Blocked	Allowed
Refresh	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Blocked	Allowed
Commit Now	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Blocked	Allowed
Custom Menu (not ride concurrent)	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Blocked	Allowed
Insert Measure	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Blocked	Blocked	Allowed
DPM		NA	Blocked	Blocked	Blocked

Scenario 6

This scenario describes running ride to lock the master domain.

Usage: `ride -d master -script script.ksh -masterInBatch`

Table 15–10 Running the ride Utility to Lock the Master Domain

Workbook Operation		Master Domain	Local Domain 1	Local Domain 2	Local Domain 3
Build	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Allowed	Allowed	Allowed
Refresh	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Allowed	Allowed	Allowed
Commit Now	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Allowed	Allowed	Allowed
Custom Menu (not ride concurrent)	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Allowed	Allowed	Allowed

Table 15–10 (Cont.) Running the ride Utility to Lock the Master Domain

Workbook Operation		Master Domain	Local Domain 1	Local Domain 2	Local Domain 3
Insert Measure	HBI Measures	Blocked	Blocked	Blocked	Blocked
	No HBI Measures	Blocked	Allowed	Allowed	Allowed
DPM		NA	Blocked	Blocked	Blocked

Domain Lock Status Using domainStatus

The domainStatus utility provides a report on the processes that are locking the domains. The purpose is to identify the user activities that are preventing a ride process from running. This information is important since the ride process does not terminate any existing workbook operation. The utility provides output that includes the process ID, user ID, operation type, and operation start time.

With the output of this utility, the system administrator can determine why the ride process is not running. This should provide enough information to either identify the user who is causing a delay or to manually terminate the process. That will be driven by the specific client and their processes.

Note that this utility reports on domain-level locks used during the ride process. Low-level data locks are not exposed by this utility.

domainStatus Usage

```
domainStatus -d domain -autoRefresh refreshPeriod
```

Table 15–11 provides descriptions of the arguments used by the domainStatus utility.

Table 15–11 Arguments Used by the domainStatus Utility

Argument	Description
-d *domain*	Refers to a simple or master domain. When a master domain is specified, all local domains in the global domain environment, as well as the master domain, are locked.
-autoRefresh *refreshPeriod*	Refreshes the lock status information every number of seconds specified by the refreshPeriod.

16

RPAS ODBC/JDBC Driver

The RPAS ODBC/JDBC Driver provides a SQL interface to the Oracle RPAS embedded database (OREDB), which includes both domain data and workbook data.

This chapter contains the following sections:

- ODBC Configuration
- Starting the RPAS ODBC Server Process
- ODBC Client Configuration for UNIX
- Installing and Using the RPAS JDBC Driver
- Data Query
- Measure Security in the ODBC Driver
- Use Cases
- Clients
- Supported and Unsupported SQL Functions

This driver presents OREDB as a relational database to ODBC and JDBC client applications. The RPAS ODBC/JDBC Driver enables ODBC 3.51 and JDBC 3.0 compatible applications to connect to OREDB. Connectivity has been verified with the following applications:

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

The RPAS ODBC/JDBC Driver enables system users to read 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 is possible through queries in global domains.
- The ODBC/JDBC Driver does not provide support for Forced Non-HBI (FNHBI) and non-materialized measures.
- The ODBC/JDBC Driver reports only external position names in both dimension tables and fact tables. Internal position names are not reported.
- Limited support is provided for conditional queries on measure data.
- The driver is not intended to replace the exportData utility, which is used for high-speed data export to ASCII files.

Note: For information on installing the RPAS ODBC/JDBC Driver, see the *RPAS Installation Guide*.

- The ODBC driver inherits the RPAS user privilege. Note that if a user is not allowed to view any measures in RPAS, the user cannot view any measures through the ODBC driver either.

ODBC Configuration

On Windows, UNIX, and Linux platforms, configuring the system to connect the ODBC drivers and a domain environment consists of the following steps.

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. Start the RPAS ODBC Agent.
4. Configure the ODBC server components.
5. Configure the ODBC client components.
6. Create the ODBC data source name (DSN). This enables ODBC applications, such as OBIEE, to connect to the domain environments configured in the ODBC server and client configuration.
7. Start the RPAS ODBC Data Service.
8. Test the connection using Interactive SQL.

Defining the ODBC Server Configuration Settings

On UNIX and Linux platforms, upon completion of the ODBC server installation, a directory named odbcserver should be created under $RPAS_HOME. An RPAS ODBC Agent process should be started automatically. This Agent process works with the GUI ODBC Management Console installed on Windows PC to perform management and configuration tasks. The Windows version of RPAS ODBC Server must be installed on the Windows PC to make the ODBC Management Console available.

To define the ODBC Server configuration settings:

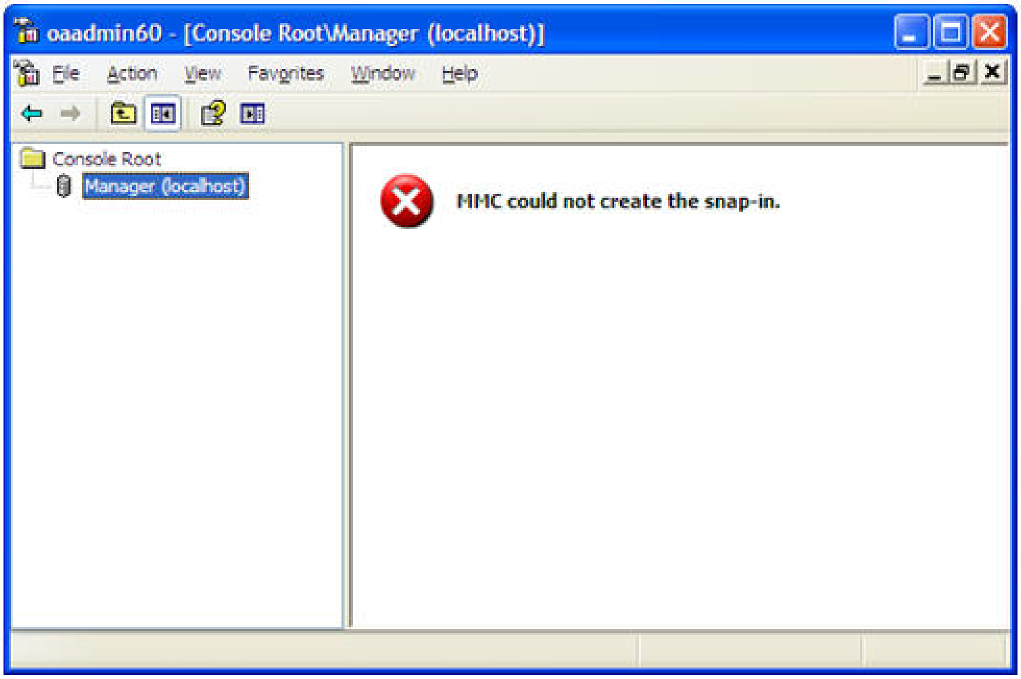
Add RPAS ODBC Manager in the Management Console

To add RPAS ODBC Manager in the Management Console, do the following:

1. From the **Start** menu, select **Oracle RPAS ODBC Server** and then **Management Console**. The Oracle RPAS ODBC Management Console appears.

Note: When you start the Management Console for the first time, an error message appears indicating the snap-in is not registered.

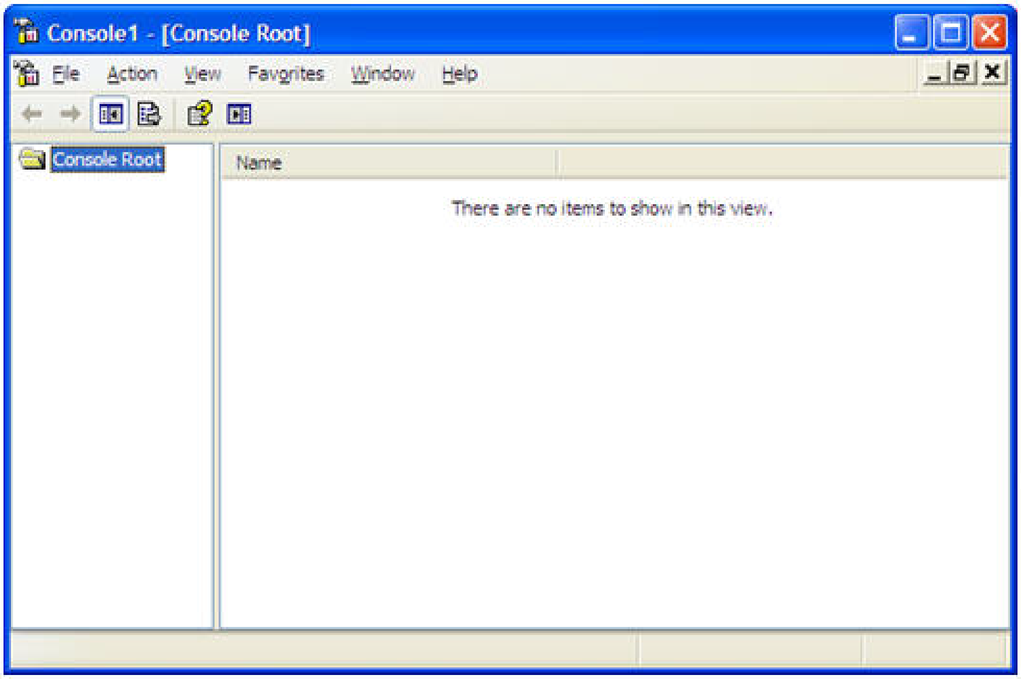
Figure 16–1 Console Manager Window Opened for First Time


oaadmin60 - [Console Root\Manager (localhost)]
File
Action
View
Favorites
Window
Help
Console Root
Manager (localhost)

MMC could not create the snap-in.

2. To create a new work space, select **File** and then **New.**

Figure 16–2 Console Manager Window Showing Console Root Directory

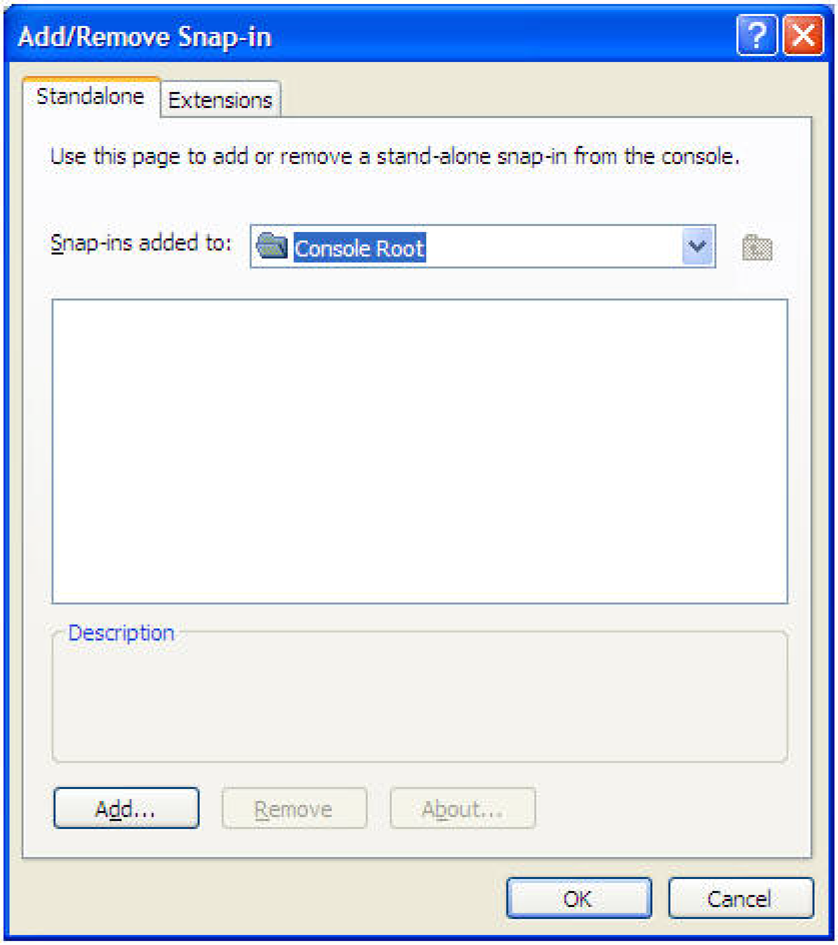

Console1 - [Console Root]
File
Action
View
Favorites
Window
Help
Console Root
Name

There are no items to show in this view.

To connect to a remote host, the Management Console is installed on a Windows PC and the Agent service is installed on a remote server.

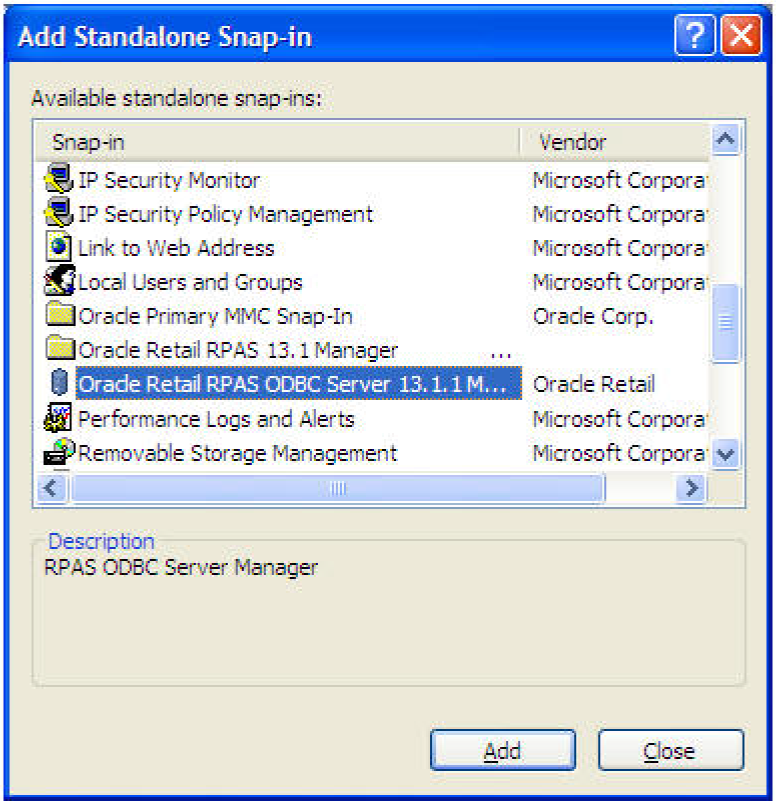
3. To add a new snap-in, select **File** and then **Add/Remove Snap-in** to start the wizard to add a new snap-in.

Figure 16–3 Add/Remove Snap-in Dialog Box


Add/Remove Snap-in
Standalone
Extensions
Use this page to add or remove a stand-alone snap-in from the console.
Snap-ins added to:
Console Root
Description
Add...
Remove
About...
OK
Cancel

4. To add a new snap-in, click **Add**.

Figure 16–4 Add Standalone Snap-in Dialog Box



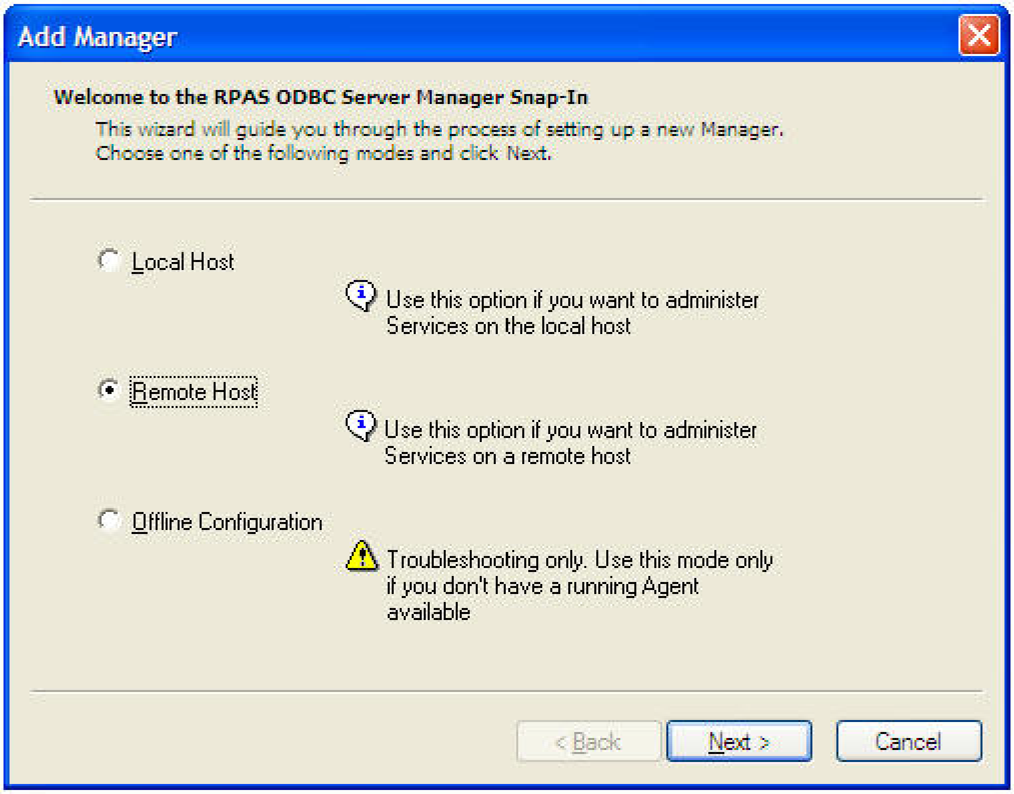
Add Standalone Snap-in
Available standalone snap-ins:
Description
Add
Close

Snap-in	Vendor
IP Security Monitor	Microsoft Corpora
IP Security Policy Management	Microsoft Corpora
Link to Web Address	Microsoft Corpora
Local Users and Groups	Microsoft Corpora
Oracle Primary MMC Snap-In	Oracle Corp.
Oracle Retail RPAS 13.1 Manager	...
Oracle Retail RPAS ODBC Server 13.1.1 M...	Oracle Retail
Performance Logs and Alerts	Microsoft Corpora
Removable Storage Management	Microsoft Corpora

Description: RPAS ODBC Server Manager

5. Select **Oracle Retail RPAS ODBC Server Manager** and click **Add**.

Figure 16–5 Add Manager Dialog Box Showing Selection of the Mode for a New Manager



Welcome to the RPAS ODBC Server Manager Snap-In

This wizard will guide you through the process of setting up a new Manager.
Choose one of the following modes and click Next.

○ Local Host

Use this option if you want to administer Services on the local host

● Remote Host

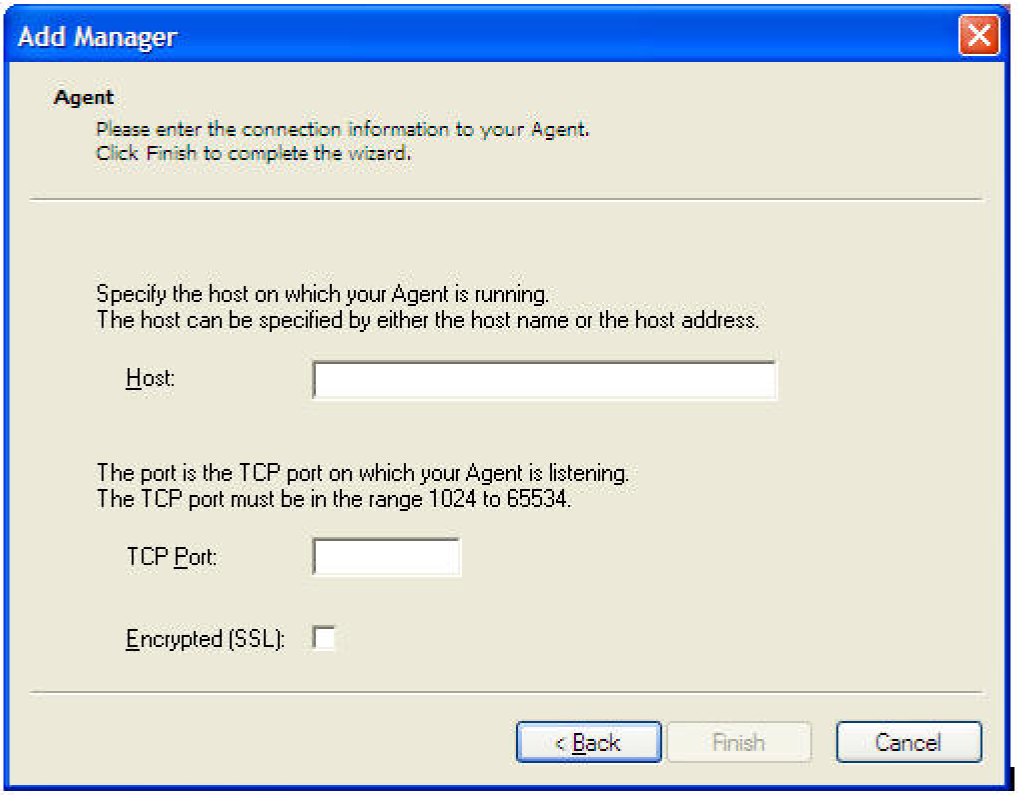
Use this option if you want to administer Services on a remote host

○ Offline Configuration

Troubleshooting only. Use this mode only if you don't have a running Agent available

6. Select Local Host or Remote Host depending on the location of the server you want to manage and then click **Next**.

Figure 16–6 Add Manager Dialog Box Showing Connection Information to be Added for the Agent



Please enter the connection information to your Agent.
Click Finish to complete the wizard.

Specify the host on which your Agent is running.
The host can be specified by either the host name or the host address.

The port is the TCP port on which your Agent is listening.
The TCP port must be in the range 1024 to 65534.

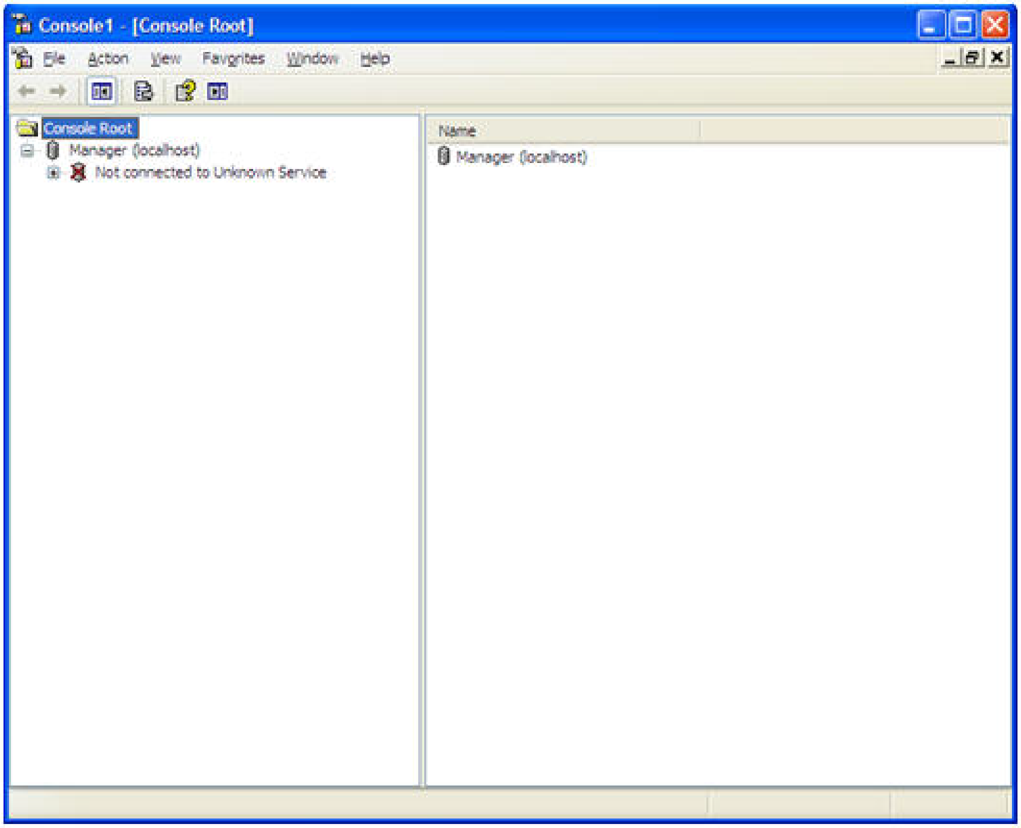
7. On the Add Manager window, enter the address of the server host machine and the TCP port number at which the RPAS ODBC Agent is listening.
8. If the Agent is not SSL enabled (which is the default), uncheck **Encrypted(SSL)**. If the Agent is SSL enabled, check **Encrypted(SSL)**.
9. Click **Finish**. You are returned to the previous window. You can click **Close** or **OK** to close the windows.

Configure Using the ODBC Manager

From the Management Console, use the ODBC Manager to perform the required configuration.

1. To configure for a remote host, click **+** to expand the ODBC Manager you just added.

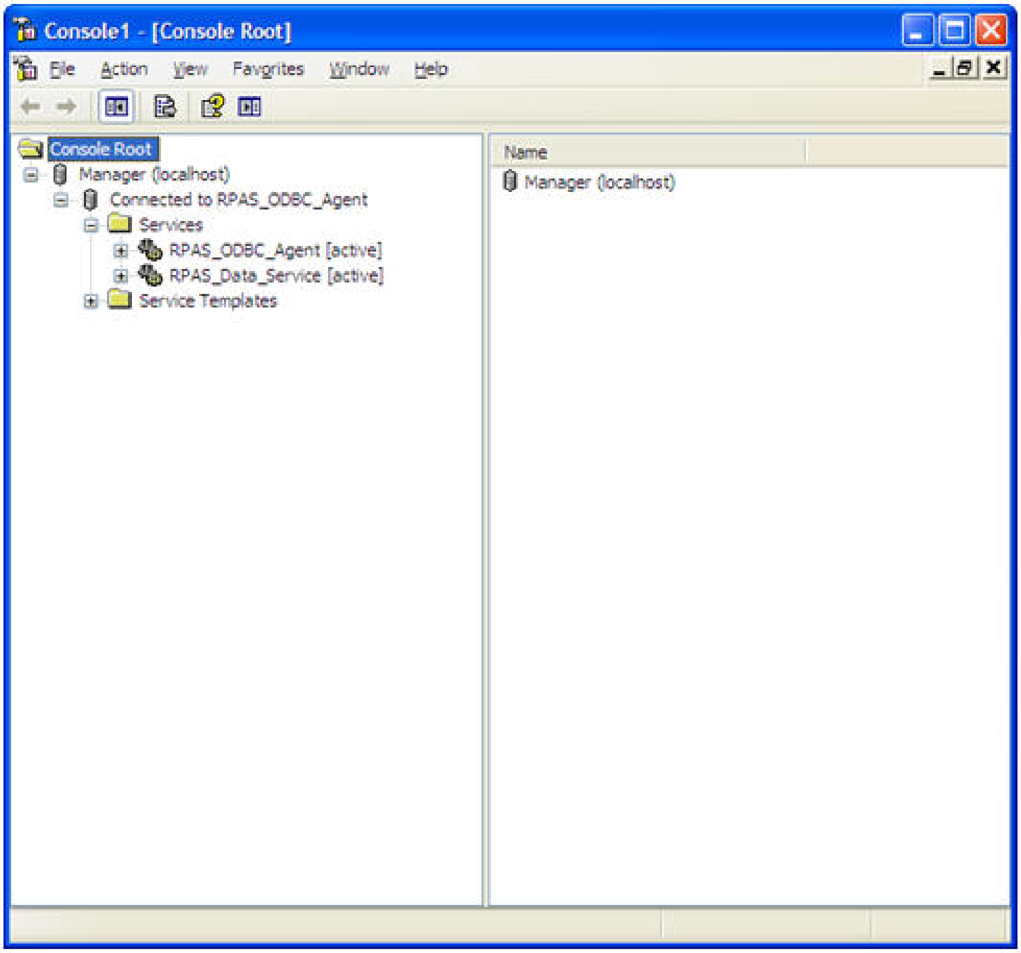
Figure 16–7 Console Manager Window Showing Expanded Console Root Directory


Console1 - [Console Root]
File
Action
View
Favorites
Window
Help
Console Root
Manager (localhost)
Not connected to Unknown Service
Name
Manager (localhost)

2. To connect to the Agent service, click **+** in front of "not connected to Unknown Service". If the Logon to Service dialog box is displayed, log on using the user name and password of a user who can administer the service on the local or remote server.

3. Expand Services by clicking **+**.

Figure 16–8 Console Manager Window Showing Expanded Connected to RPAS_ODBC_Agent Directory


Console1 - [Console Root]
File
Action
View
Favorites
Window
Help
Console Root
Manager (localhost)
Connected to RPAS_ODBC_Agent
Services
RPAS_ODBC_Agent [active]
RPAS_Data_Service [active]
Service Templates
Name
Manager (localhost)

4. Set up the environment variables.

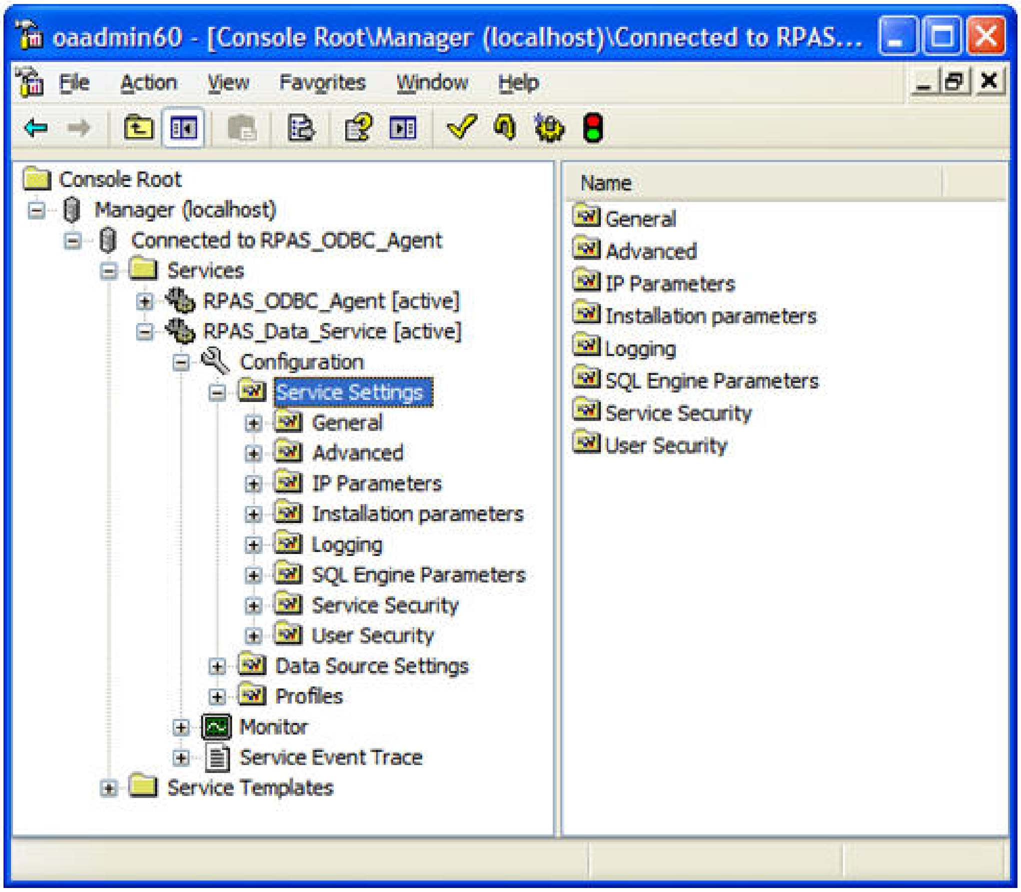
> **Note:** This step only applies to RPAS_Data_Service running on a Windows platform.

The PATH environment variable must be configured to include the following paths:

- Path to the bin subdirectory of RPAS_Home
- Path to the ODBC server binary
- Path to the ODBC server IP binary

a. Navigate to the following screen. Figure 16–9 shows the exact screen you should see.

Figure 16–9 Console Manager Window for Configuration


oaadmin60 - [Console Root\Manager (localhost)\Connected to RPAS...
File
Action
View
Favorites
Window
Help
Console Root
Manager (localhost)
Connected to RPAS_ODBC_Agent
Services
RPAS_ODBC_Agent [active]
RPAS_Data_Service [active]
Configuration
Service Settings
General
Advanced
IP Parameters
Installation parameters
Logging
SQL Engine Parameters
Service Security
User Security
Data Source Settings
Profiles
Monitor
Service Event Trace
Service Templates
Name
General
Advanced
IP Parameters
Installation parameters
Logging
SQL Engine Parameters
Service Security
User Security

b. Right-click on the blank space of the right panel.

c. Select **New/Attribute** in the menu.

d. Click **OK**.

e. Select **ServiceEnvironmentVariable** from the list for Attribute. In the Value field, enter the following:

PATH={pathToRpasHomeLib};{pathToODBCServerBin};{pathToODBCServerIPBin}

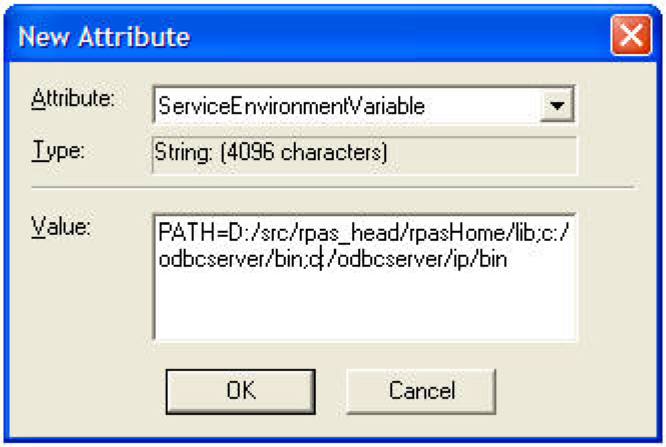
In the sample shown in Figure 16–10, the following values are set:

- {pathToRpasHomeLib} is set to D:/src/rpas_head/rpasHome/lib

- {pathToODBCServerBin} is c:/odbcserver/bin

- {pathToODBCServerIPBin} is c:/odbcserver/ip/bin

Figure 16–10 New Attribute Dialog Box


New Attribute
OK
Cancel

Attribute: ServiceEnvironmentVariable

Type: String: (4096 characters)

Value: PATH=D:/src/rpas_head/rpasHome/lib;c:/odbcserver/bin;c:/odbcserver/ip/bin

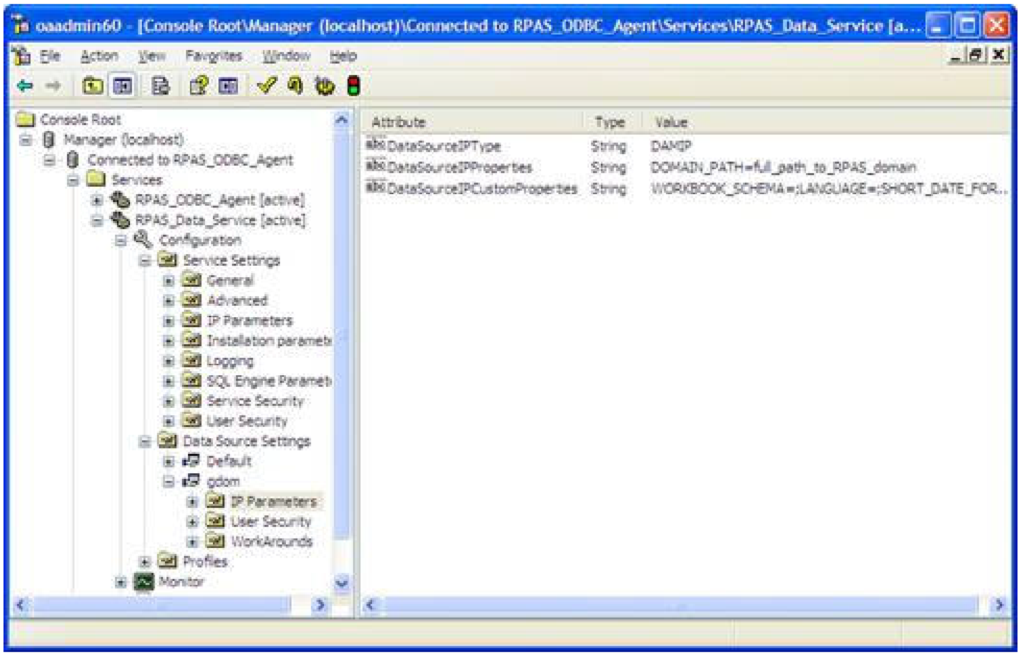
5. Expand Data Source Settings on the Console Manager window. You should find a pre-configured data source named gdom.

Note: When you create a new data source, make sure that most parameters of the new data source are identical to the parameters of the pre-configured data source gdom, except for DataSourceIPProperties and DataSourceIPCustomProperties.

The DataSourceIPCustomProperties attribute contains pre-registered property names, which are not to be modified.

6. Under gdom, select IP Parameters.

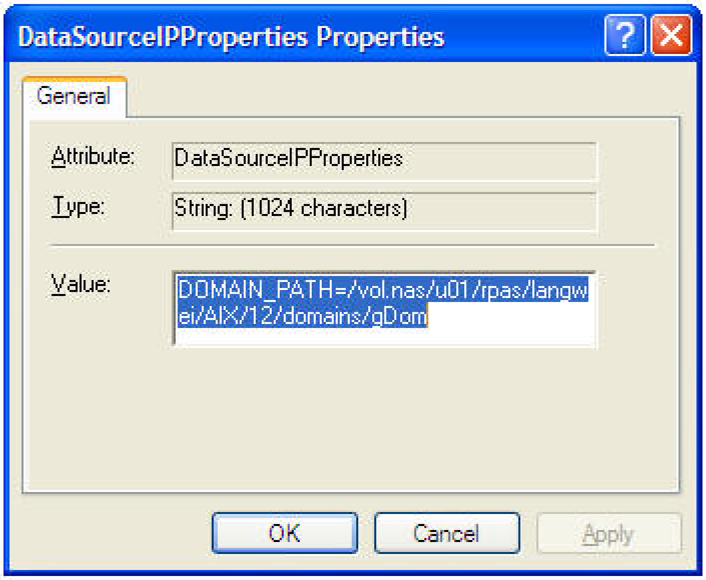
Figure 16–11 Console Manager Window with IP Parameters Selected Under gdom


oaadmin60 - [Console Root\Manager (localhost)\Connected to RPAS_ODBC_Agent\Services\RPAS_Data_Service [a...
File
Action
View
Favorites
Window
Help
Console Root
Manager (localhost)
Connected to RPAS_ODBC_Agent
Services
RPAS_ODBC_Agent [active]
RPAS_Data_Service [active]
Configuration
Service Settings
General
Advanced
IP Parameters
Installation paramete
Logging
SQL Engine Paramet
Service Security
User Security
Data Source Settings
Default
gdom
IP Parameters
User Security
WorkArounds
Profiles
Monitor

Attribute	Type	Value
DataSourceIPType	String	DAMIP
DataSourceIPProperties	String	DOMAIN_PATH=full_path_to_RPAS_domain
DataSourceIPCustomProperties	String	WORKBOOK_SCHEMA=;LANGUAGE=;SHORT_DATE_FOR...

7. On the right panel of the window, double click DataSourceIPProperties attribute. The following window appears.

Figure 16–12 DataSourceIPProperties Properties Dialog Box


DataSourceIPProperties Properties
General
OK
Cancel
Apply

Attribute: DataSourceIPProperties

Type: String: (1024 characters)

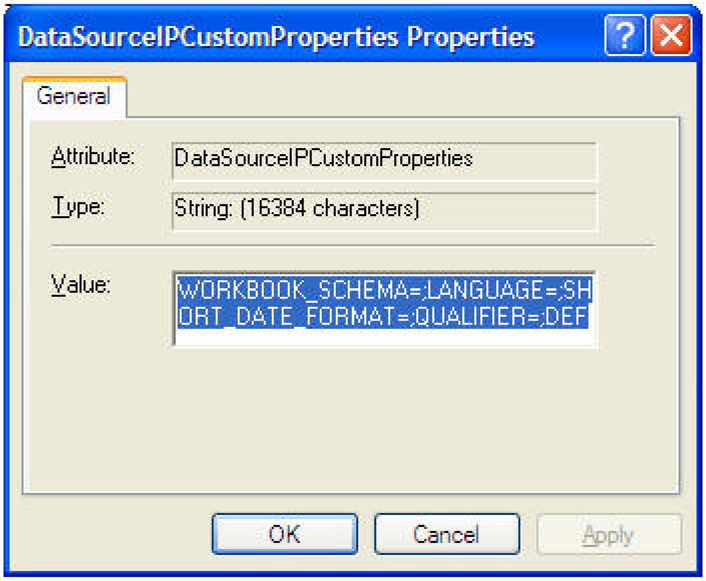
Value: DOMAIN_PATH=/vol.nas/u01/rpas/langwei/AIX/12/domains/gDom

Make sure the Value field has the keyword **DOMAIN_PATH=** and that the string after the "=" sign is the absolute path to the RPAS domain you want connect to on the server side.

The following example shows the path to connect to a domain on the local host: `DOMAIN_PATH=C:\RPAS\11\Test\ODBC\nt_testGlobalDomain`

8. On the right panel of the window, double click the DataSourceIPCustomProperties attribute. The following window appears.

Figure 16–13 DataSourceIPCustomProperties Properties Dialog Box


DataSourceIPCustomProperties Properties
General
OK
Cancel
Apply

Attribute: DataSourceIPCustomProperties

Type: String: (16384 characters)

Value: WORKBOOK_SCHEMA=;LANGUAGE=;SHORT_DATE_FORMAT=;QUALIFIER=;DEF

Copy the value from the DataSourceIPCustomProperties of the sample gdom data source to make sure all pre-registered properties are included.

Import the configuration changes

1. To save the configuration, right-click **Services** in the Console Manager window. In the menu, select **All Tasks** and then **Save Configuration**.
2. To save the snap-in configuration, select **File** and then **Save as**. Select <*installdir*>\admin\oaadmin60.msc for the file name. This overwrites the original file (which was basically empty).
3. Stop and start the RPAS Data Service.

   Right-click **RPAS_Data_Service**. In the menu, click **Stop RPAS_Data_Service** or **Start_RPAS_Data_Service**. If you have made changes to any of the service attributes, you need to restart the Data Service.

The RPAS ODBC Data Service is now ready to accept connections.

Defining the ODBC Client Configuration for Windows

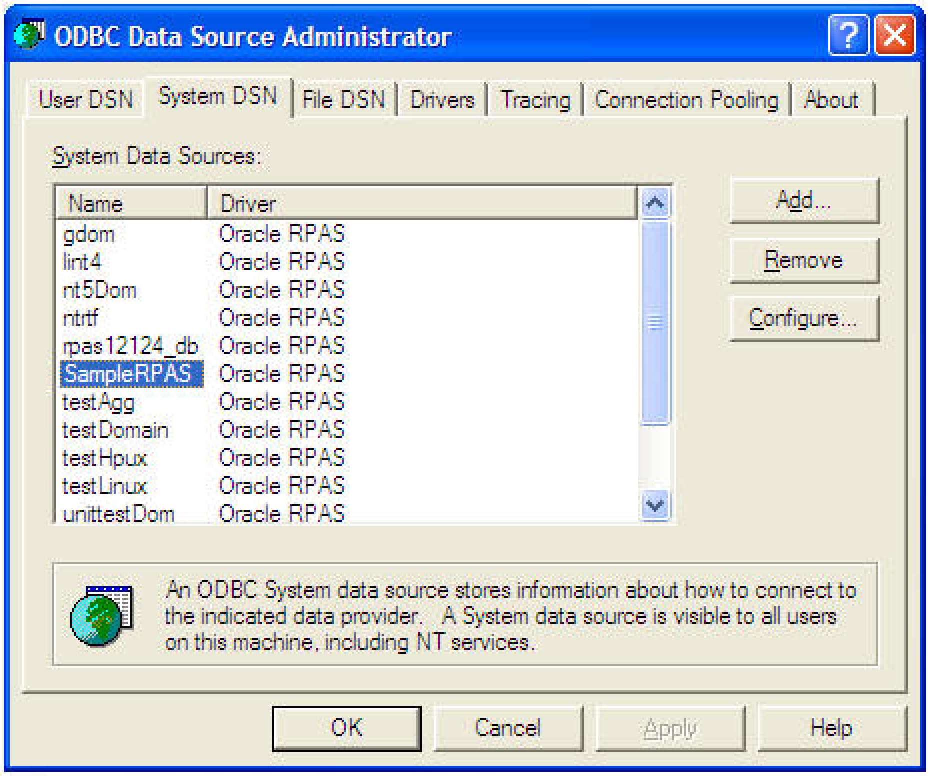
To define the ODBC Client configuration settings, do the following:

1. From the Start menu, select **ODBC Administrator** under the Oracle RPAS odbc driver menu item.

> **Note:** Upon successful installation of RPAS ODBC Client, a sample DSN named "SampleRPAS" is automatically created and configured to connect to the data source gdom on the server side. To create a new DSN, see the following steps.

2. In the ODBC Data Source Administrator window, select **Add** to add a data source.

Figure 16–14 ODBC Data Source Administrator Dialog Box

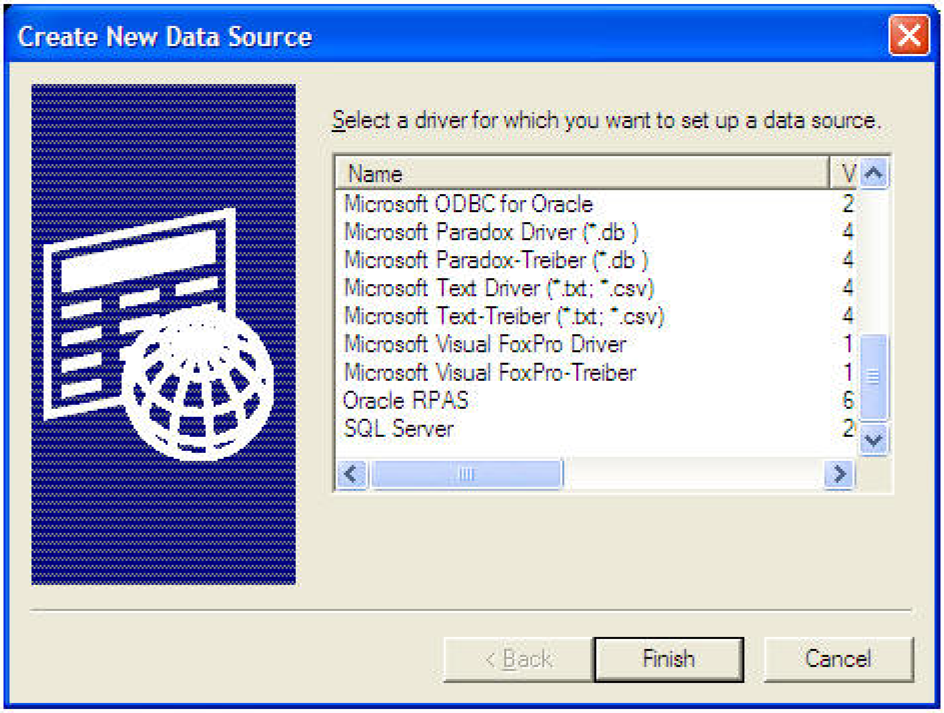


Name	Driver
gdom	Oracle RPAS
lint4	Oracle RPAS
nt5Dom	Oracle RPAS
ntrtf	Oracle RPAS
rpas12124_db	Oracle RPAS
SampleRPAS	Oracle RPAS
testAgg	Oracle RPAS
testDomain	Oracle RPAS
testHpux	Oracle RPAS
testLinux	Oracle RPAS
unittestDom	Oracle RPAS

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

3. In the Create New Data source window, select **Oracle RPAS**.

Figure 16–15 Create New Data Source Dialog Box



Select a driver for which you want to set up a data source.

Name	V
Microsoft ODBC for Oracle	2
Microsoft Paradox Driver (*.db)	4
Microsoft Paradox-Treiber (*.db)	4
Microsoft Text Driver (*.txt; *.csv)	4
Microsoft Text-Treiber (*.txt; *.csv)	4
Microsoft Visual FoxPro Driver	1
Microsoft Visual FoxPro-Treiber	1
Oracle RPAS	6
SQL Server	2

4. In the Oracle Retail RPAS ODBC Driver Setup window, enter the following information:

   - Name and description of the ODBC data source.
   - In the Service Host field, enter the name of the server. If connecting to a service running on a local host, enter **localhost** or the name of the local host server. If the Agent service is running on a remote server, enter the name of the remote host server.
   - In the Service Port field, enter the port number that the data service is listening on.

   Note: This is the port number of the data service and not the Agent service port number.

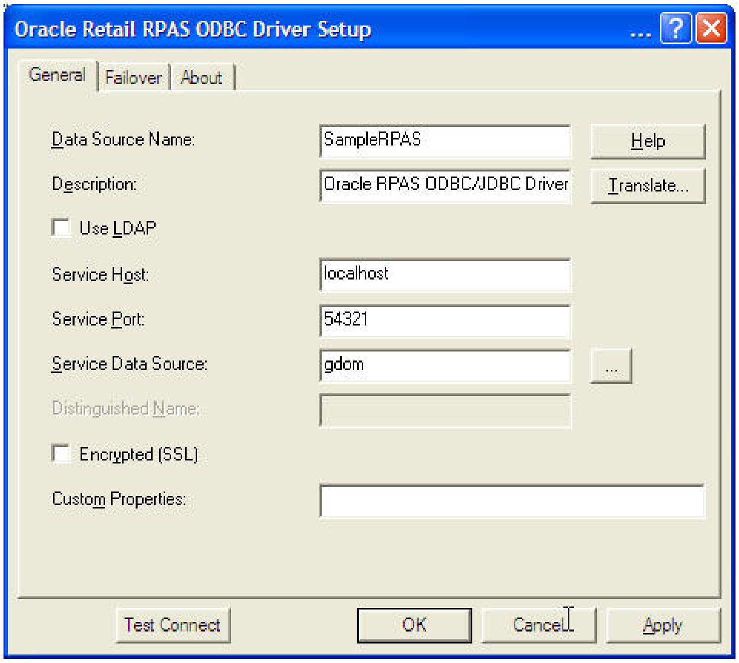
   - In the Service Data Source field, enter the name of the service data source that has been confided for the data service. The default for this field is gdom.
   - If the data service is not SSL enabled, uncheck **Encrypted(SSL)**. If the data service is SSL enabled, check **Encrypted(SSL)**.
   - Enter custom properties, if needed. Custom properties are entered in the format of [name]=[value]. Multiple properties should be separated by a semicolon. For example: LANGUAGE=Japanese;WORKBOOK_SCHEMA=DOMAIN_T0

   The following table lists the available custom properties (all optional):

Table 16–1 Available Custom Properties

Property Name	Description
LANGUAGE	Name of the language you use. The RPAS ODBC/JDBC driver is multi-language enabled. If data is in any language other than English, the LANGUAGE property should be set to the name of that language. If multiple languages are used in the domain, set this property to the name of the language other than English. For example, if some position names are in English and some are in Japanese, then LANGUAGE should be set to Japanese. The default is English.
WORKBOOK_SCHEMA	Name of the workbook you wish to connect to. If not set, the driver connects to the domain.
SHORT_DATE_FORMAT	Valid short date format used in RPAS.
DEFAULT_SCHEMA	Default schema name if the table name in the query is not qualified. This property is set to DOMAIN by the default configuration.
AGG_TABLE_NAMES	This property can be set to a list of valid aggregate table names separated by commas. When this is set, the driver presents the tables specified in the system tables. When this property is not set, the valid aggregate tables can still be queried even though they do not exist in the system tables.
NORMALIZE_DIM_TABLES	Valid values are Yes and No. The default value is No. If set to Yes, the dimension tables will only contain columns for this dimension and its immediate parent dimension. If set to No, the dimension tables will contain columns for this dimension and all parent dimensions within the hierarchy.

Figure 16–16 Oracle Retail RPAS ODBC Driver Setup Dialog Box



Data Source Name: SampleRPAS

Description: Oracle RPAS ODBC/JDBC Driver

☐ Use LDAP

Service Host: localhost

Service Port: 54321

Service Data Source: gdom

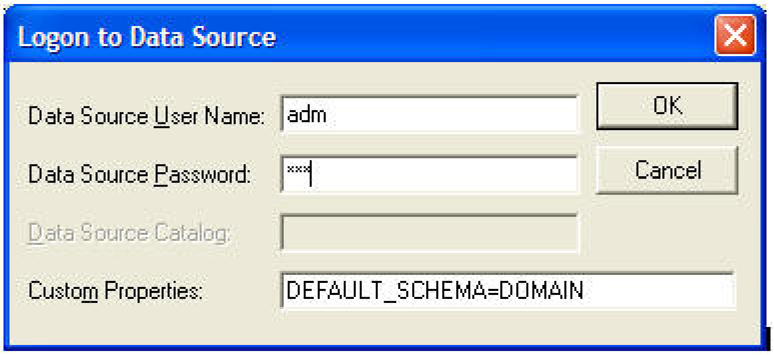
Distinguished Name:

☐ Encrypted (SSL)

Custom Properties:

5. Click **Test Connect**. If the user security of the gdom Data source setting has been set to DBMSLogon, the Logon to Data Source dialog is displayed. Enter the data source user name and password configured for the data source.

Figure 16–17 Logon to Data Source Dialog Box



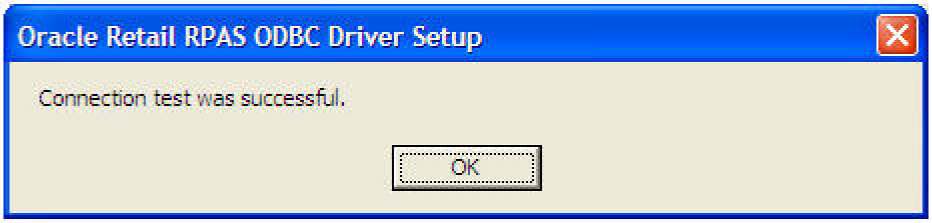
Data Source User Name: adm

Data Source Password: ***

Custom Properties: DEFAULT_SCHEMA=DOMAIN

If the connection is successful, the following dialog box is displayed.

Figure 16–18 Oracle Retail RPAS ODBC Driver Setup Dialog Box for Successful Connection



Connection test was successful.

6. Click **OK**.

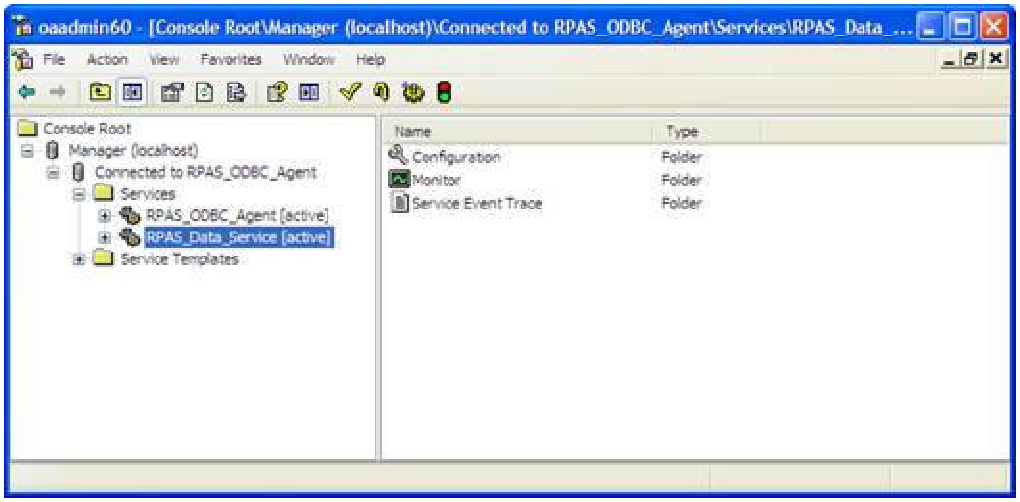
Starting the RPAS ODBC Server Process

The RPAS ODBC Agent and Data Services should have automatically started after successful completion of the server installation.

The RPAS ODBC Data Service should be stopped and restarted using the Management Console.

1. To stop a data service, right-click the service name and then click **Stop** in the menu.
2. Right-click RPAS_Data_Service and then click **Start RPAS_Data_Service** in the menu.

Figure 16–19 Console Manager Window with Data Service Selected


oaadmin60 - [Console Root\Manager (localhost)\Connected to RPAS_ODBC_Agent\Services\RPAS_Data_...
File
Action
View
Favorites
Window
Help
Console Root
Manager (localhost)
Connected to RPAS_ODBC_Agent
Services
RPAS_ODBC_Agent [active]
RPAS_Data_Service [active]
Service Templates
Name
Type
Configuration
Folder
Monitor
Folder
Service Event Trace
Folder

Testing the Connection Using Interactive SQL

After 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 driver**, and then **Interactive SQL (ODBC)**. The Interactive SQL command window appears.
2. Enter 'connect <*user name*>*<*password*>@<*dsn_name*>' where <*dsn_name*> is the name of the connection defined in the ODBC Server and ODBC Client configuration. The following is an example.

   'connect adm*adm@SampleRPAS'

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

ODBC Client Configuration for UNIX

Configuring the UNIX system to connect the ODBC drivers and a domain environment consists of the following steps:

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 Data Service if it is not already started.
6. Test the connection using Interactive SQL.

Client Configuration

Both 32-bit and 64-bit ODBC Clients are available. They are delivered in directories named odbcclient32 and odbcclient64 respectively. The configuration steps are identical for 32-bit and 64-bit ODBC Client.

Note: For the remainder of this chapter, the 32-bit or 64-bit ODBC Client are referred to as ODBC Client, and odbcclient32 or odbcclient64 are referred to as odbcclient.

If it comes with RPAS, then odbcclient directory is under your $RPAS_HOME. If it comes separately, the installer determines its location.

1. Set up the environment for the ODBC Client.

   If the ODBC client does not come with RPAS (meaning the odbcclient directory is not under $RPAS_HOME), edit the oaodbc.sh file (oaodbc64.sh for 64-bit Client) in odbcclient:

   a. Make sure the following environment variables are set correctly:

      - LIBPATH and OASDK_ODBC_HOME are set to the full path of the lib directory inside the odbcclient directory

      - ODBCINI is set to the full path of the odbc.ini file (odbc64.ini for 64-bit Client), including the file name, inside the odbcclient directory

   b. Source oaodbc.sh by running the following command in the odbcclient directory:

      . ./oaodbc.sh

2. Create and configure the data sources in odbc.ini.

   The odbc.ini file in the odbcclient directory has three sections: [ODBC], [ODBC Data Sources], and [SampleRpas] which is a section for the sample RPAS data source.

   a. Edit the [ODBC] section: Set TraceDll to the full path to lib/odbctrac.so and InstallDir to the full path of the odbcclient directory.

   b. Edit the [ODBC Data Sources] section: Add an entry for the new data source you are creating. The entry has the following format:

      ```
      MyRPASDataSource= Oracle RPAS ODBC Driver
      ```

   c. Create a new [MyRPASDataSource] section: The [SampleRPAS] section can be copied and modified. In the new [MyRPASDataSource] section:

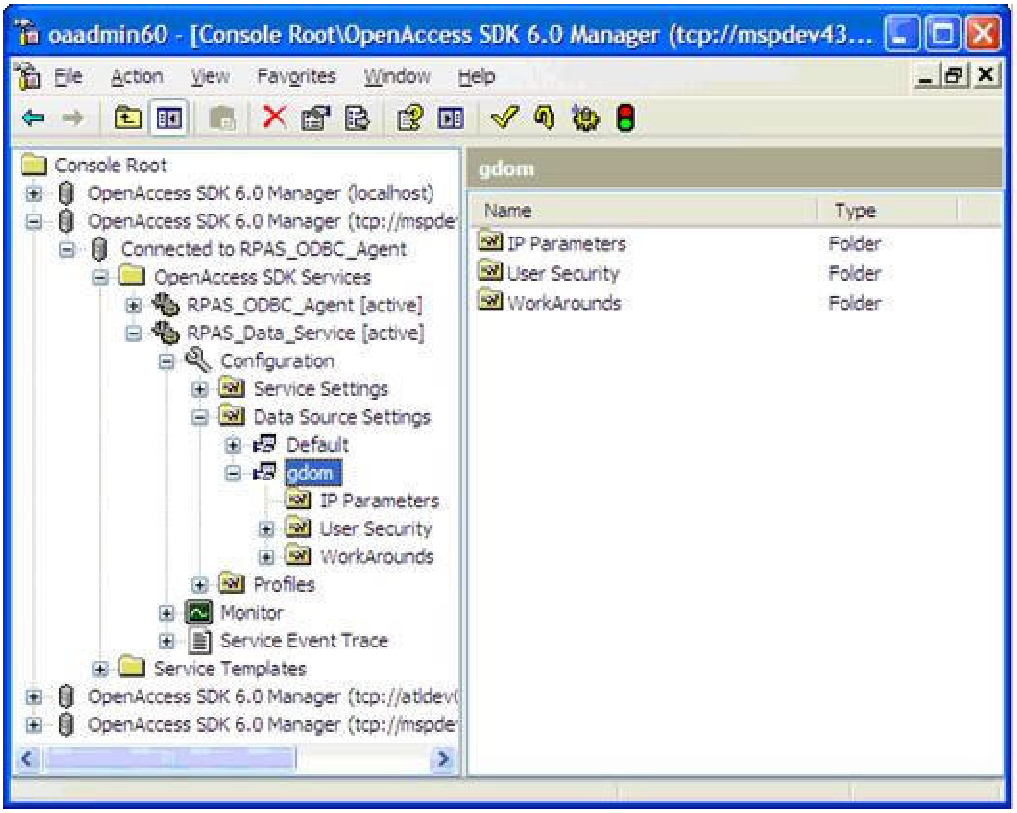
      - Set Driver to the full path of odbcclient/lib/ivoa22.so.

      - Set Host to the name or IP address of the server.

      - Set Port to the port number the RPAS_Data_Service listens at. This is not the port number used by the RPAS ODBC Agent.

      - ServerDataSource should be set to the name of the data source you created in the server configuration. For [SampleRPAS], this entry is set to gdom, since that is the data source created on the server as an example. This is shown in the following figure.

Figure 16–20 Console Manager Window with gdom Selected


oaadmin60 - [Console Root\OpenAccess SDK 6.0 Manager (tcp://mspdev43...
File Action View Favorites Window Help
Console Root
OpenAccess SDK 6.0 Manager (localhost)
OpenAccess SDK 6.0 Manager (tcp://mspde
Connected to RPAS_ODBC_Agent
OpenAccess SDK Services
RPAS_ODBC_Agent [active]
RPAS_Data_Service [active]
Configuration
Service Settings
Data Source Settings
Default
gdom
IP Parameters
User Security
WorkArounds
Profiles
Monitor
Service Event Trace
Service Templates
OpenAccess SDK 6.0 Manager (tcp://atldev(
OpenAccess SDK 6.0 Manager (tcp://mspde
gdom
Name
Type
IP Parameters
Folder
User Security
Folder
WorkArounds
Folder

Testing the Connection

After the ODBC Server and ODBC client have been configured, you can test the connection using Interactive SQL. The RPAS ODBC Data Service must be started.

1. In the odbcclient directory, source oaodbc.sh if you have not already done so.
2. Change to the tools directory and run the executable odbcisql. Then, at the ISQL prompt, enter 'connect <*user Name*>*<*password*>@<*DataSourceName*>' where <*DataSourceName*> is the name of the data source you defined in odbc.ini, as described in "Client Configuration". The following is an example.

   'connect adm*adm@MyRPASDataSource'

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

Installing and Using the RPAS JDBC Driver

This section describes how to install, set up, and use the RPAS JDBC driver on all UNIX and Linux and Windows platforms.

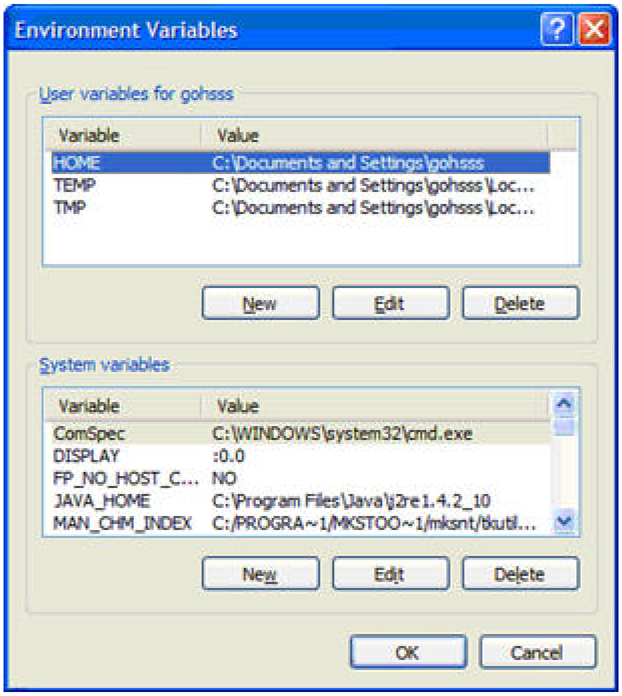
The RPAS JDBC driver is delivered in a single (zipped) jdbcclient directory. The user has full control on the location of the jdbcclient directory. For installation guidelines, refer to the *RPAS Installation Guide*.

After the JDBC driver is installed on your system, you need to update the CLASSPATH environment variable. This variable ensures that the JDBC client can access the appropriate Java classes needed to connect to the database.

Updating Environment Variables for the JDBC Driver on Windows

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.

Figure 16–21 Environment Variables Dialog Box



User variables for gohsss

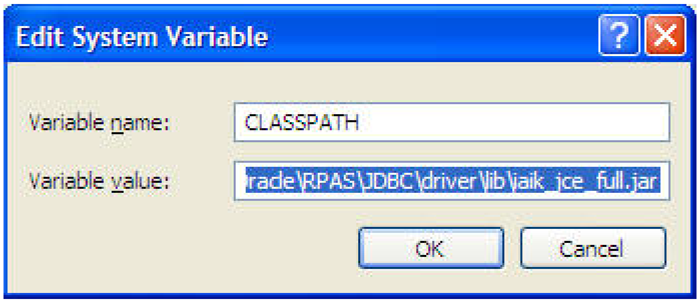
Variable	Value
HOME	C:\Documents and Settings\gohsss
TEMP	C:\Documents and Settings\gohsss\Loc...
TMP	C:\Documents and Settings\gohsss\Loc...

System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DISPLAY	:0.0
FP_NO_HOST_C...	NO
JAVA_HOME	C:\Program Files\Java\j2re1.4.2_10
MAN_CHM_INDEX	C:/PROGRA~1/MKSTOO~1/mksnt/tkutil...

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

Figure 16–22 Edit System Variable Dialog Box



Variable name: CLASSPATH

Variable value: racle\RPAS\JDBC\driver\lib\iaik_jce_full.jar

4. Add the current working directory ".",driver_home/driver/lib/ORjc.jar, driver_home/driver/ORssl14.jar, and driver_home/driver/ iaik_jce_full.jar to the CLASSPATH environment variable and click **OK**.

   The driver_home is the location where the jdbcclient was installed. If your jdbcclient was installed in C:/jdbcclient, you see the following in your CLASSPATH:

```
.;C:/jdbcclient/driver/lib/ORjc.jar; C:/jdbcclient/driver/lib/ORssl14.jar; C:/
jdbcclient/driver/lib/iaik_jce_full.jar
```

Note: Separate paths with semi-colons (;).

---

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

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

Updating Environment Variables for JDBC Driver on UNIX/Linux

On UNIX/Linux systems, use "export" (or "set", depending what shell you use) to add the following to your CLASSPATH:

```
export CLASSPATH=.:jdbc_home/driver/lib/ORjc.jar: jdbc_home/driver/lib/
ORssl14.jar: jdbc_home/driver/lib/iaik_jce_full.jar:$CLASSPATH
```

where jdbc_home is the full path of the directory where jdbcclient is installed. If you installed jdbcclient at /usr/products/oracle, then you should replace jdbc_home with /usr/products/oracle/jdbcclient.

The above export command can be added to your .profile.

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: com.oracle.ard.jdbc.openaccess.OpenAccessDriver
- URL: "jdbc:RPAS://<host>:<port>;ServerDataSource=<DataSourceName>"

<host> is the name or IP address of the server, <port> is the port number the RPAS Data Service listens at, and <DataSourceName> is the name of data source you created for the RPAS Data Server (it is gdom in the default configuration).

Enabling Spy for RPAS JDBC Driver

Spy is a logging facility for the JDBC driver. To enable spy for the RPAS JDBC connection:

1. Add jdbc_home/spy/lib/ORy.jar to your CLASSPATH, where jdbc_home is the installation directory of jdbcclient.
2. Set your driver class to com.oracle.ard.jdbcspy.SpyDriver.
3. Use the following URL:

   "jdbc:spy:{jdbc:RPAS://<host>:<port>;ServerDataSource=<DataSourceName>};load=com.oracle.ard.jdbc.openaccess.OpenAccessDriver;[key=value];…"

   <host> is the name or IP address of the server, <port> is the port number the RPAS Data Service listens at, and <DataSourceName> is the name of data source you created for the RPAS Data Server. (It is gdom in the default configuration.) The key and value pairs are the attributes of the Spy class.

The following table lists the available attributes:

Table 16–2 Attributes Available

Key and Value	Description
log=System.out	Redirects logging to the Java output standard, System.out.
log=(file)*filename*	Redirects logging to the file specified by *filename*. For example, C:\temp\spy.log.
linelimit=*numberofchars*	Indicates the maximum number of characters, specified by *numberofchars*, that Spy logs on one line. When set to no (the default), there is no maximum limit on the number of characters.
logLobs={yes | no}	Specifies whether Spy logs activity on Blob/Clob. The initial default is no.
logIS={yes | no | nosingleread}	Specifies whether Spy logs activity on InputStreams. When logIS=nosingleread, logging on InputStream and Reader objects is active; however, logging of the single-byte read InputStream.read or single-character Reader.read is suppressed to prevent generating large log files that contain single-byte or single character read messages. When set to no (the default), Spy does not log activity on InputStreams.
logTName={yes | no}	Specifies whether Spy logs the name of the current thread. When set to no (the default), Spy does not log the name of the current thread.
timestamp={yes | no}	Specifies whether a timestamp should be included on each line of the Spy log. When set to no (the default), Spy does not include a timestamp on each line.

Using the jdbcisql Utility Provided with RPAS JDBC Driver

Oracle Retail suggests that you use the jdbcisql.bat (for Windows) or jdbcisql.sh (for UNIX/Linux) located in jdbcclient/isql to start jdbcisql. Edit jdbcisql.bat or jdbcisql.sh to make sure it uses the appropriate URL (the argument of the -u option):

```
java jdbcisql -d com.oracle.ard.jdbc.openaccess.OpenAccessDriver -u "jdbc:RPAS://
<host>:<port>;ServerDataSource=gdom"
```

Note: The value of the ServerDataSource setting in the URL is case-sensitive on Windows and UNIX systems and has to match the Data source setting defined for the data service.

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

```
Connect adm*adm@
```

To enable Spy for jdbcisql, use a command line similar to the following:

```
java jdbcisql -d com.oracle.ard.jdbcspy.SpyDriver -u "jdbc:spy:{jdbc:RPAS:// ://
<host>:<port>;;ServerDataSource=gdom};load=com.oracle.ard.jdbc.openaccess.OpenAcce
ssDriver;log=(file)C:\temp\spy.log;logIS=yes;logTName=yes;timestamp=yes"
```

Using Oracle SQL Developer

Create an XML file with the following content:

```
<?xml version = '1.0'?>
<!DOCTYPE connections>
<connections>
   <connection>
      <URL>"jdbc:RPAS://<host>:<port>;ServerDataSource=<DataSourceName>"</URL>
      <ConnectionName>MyConnection</ConnectionName>
      <user>adm</user>
      <ConnectionType>OTHER_JDBC</ConnectionType>
      <JdbcDriver>com.oracle.ard.jdbc.openaccess.OpenAccessDriver</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 you like. This field can be changed later using the client application.
- Enter a user name for the connection. This field can also be changed later using the application.
- Leave the remaining information as shown in the code sample above.

To set up the connection:

1. Save this XML file with any name you like.
2. In SQL Developer, using the Tools/Preferences/Database/Third Party Drivers, add the ORjc.jar, Orssl14.jar, and iaik_jce_full.jar files to the list of third party drivers used by SQL Developer. (SQL Developer does not look in the classpath for drivers.)
3. Go to the Connection Navigator and right-click on Connections. Select **Import Connections**.
4. Browse to the XML file. The dialog displays the list of connections you specified in the file. Choose your connection in the sample code, MyConnection.

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 the Create New Database Connection wizard, enter a connection name and choose **Third Party JDBC driver** for **Connection Type**.
5. On the second screen, enter the 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 ORjc.jar, Orssl14.jar, and iaik_jce_full.jar files and their path. These jar files are available from the installation of the RPAS JDBC Client.

   c. Enter the RPAS JDBC Driver connection URL, as specified at the beginning of this section.
   d. In the **Driver Class** field, enter com.oracle.ard.jdbc.openaccess.OpenAccessDriver.
7. Follow the instructions to finish creating the connection.

Using a Java Program

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

- new oadriver();
- Class.forName ("com.oracle.ard.jdbc.openaccess.OpenAccessDriver").newInstance();

Make sure *driver_home*/driver/lib/ORjc.jar, *driver_home*/driver/lib/Orssl14.jar, and *driver_home*/driver/lib/oaik_jce_full.jar are included in the CLASSPATH.

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

Java Code Sample

```
import java.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;
   Driver d =
(Driver)Class.forName("com.oracle.ard.jdbc.openaccess.OpenAccessDriver").newInstan
ce();
   String url = "jdbc:RPAS://";
   String database = args[0];
   String uid = args[1];
   String pwd = args[2];
   url += database;
   System.out.println("Trying to connect to url: " + url );
   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_YEAR";
   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 ();
  }
 }
}
```

Running the Program

After compilation, run the program as:

```
java RPASDriverTest.class "<host>:<port>;ServerDataSource=<DSN>" <uid> <pwd>
```

In the example above, <host> is the IP of the server box where RPAS ODBC Server is running and where <port> is port number of the ODBC Server. <DSN> is the data source name that is created on the server. Note that the double quotes must be included because of the semicolon.

Data Query

This section provides the details of data query, including the limitations, metadata, and dimension tables.

Limitations

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

Contention

Workbook commit requests issued by RPAS users compete with real-time intra-day reports (ODBC queries) when accessing the domain data. Commit requests imply 'write' locks on certain measures in the domain. While such 'write' locks are in place, reporting tools cannot access the same data. In such situations both reporting and workbook users can experience latencies or, in more sever scenarios, they can encounter system feedback informing them that their last operation did not succeed and they need to re-execute their last request. The likelihood of such concurrency issues depends on a number of factors such as hardware capabilities (CPU and IO capacity), reporting volume, reporting granularity, number of reporting users, number of workbook users, and the commit data volume. To minimize the chance of such concurrency issues, all workbook commits should leverage the commit ASAP framework. Additionally, RPAS allows reporting from workbooks that is less prone to concurrency issues that may be experienced while reporting from domains.

Workbook Queries

All workbook queries must be performed against saved workbooks. The workbook can be open or closed. The user has to save the workbook before reporting.

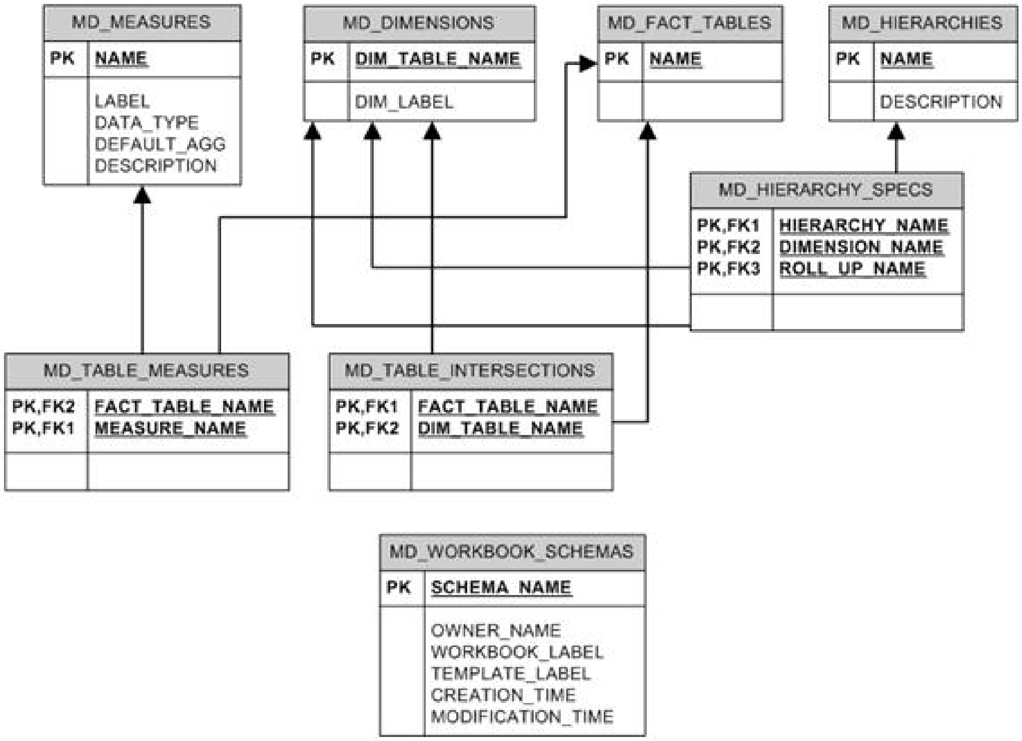
Metadata

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

- 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 16–23 Database Diagram for All Metadata Tables in a Domain or in Each Workbook


MD_MEASURES
PK NAME
LABEL
DATA_TYPE
DEFAULT_AGG
DESCRIPTION
MD_DIMENSIONS
PK DIM_TABLE_NAME
DIM_LABEL
MD_FACT_TABLES
PK NAME
MD_HIERARCHIES
PK NAME
DESCRIPTION
MD_HIERARCHY_SPECS
PK,FK1 HIERARCHY_NAME
PK,FK2 DIMENSION_NAME
PK,FK3 ROLL_UP_NAME
MD_TABLE_MEASURES
PK,FK2 FACT_TABLE_NAME
PK,FK1 MEASURE_NAME
MD_TABLE_INTERSECTIONS
PK,FK1 FACT_TABLE_NAME
PK,FK2 DIM_TABLE_NAME
MD_WORKBOOK_SCHEMAS
PK SCHEMA_NAME
OWNER_NAME
WORKBOOK_LABEL
TEMPLATE_LABEL
CREATION_TIME
MODIFICATION_TIME

Note: The MD_WORKBOOK_SCHEMAS table is not included in workbooks.

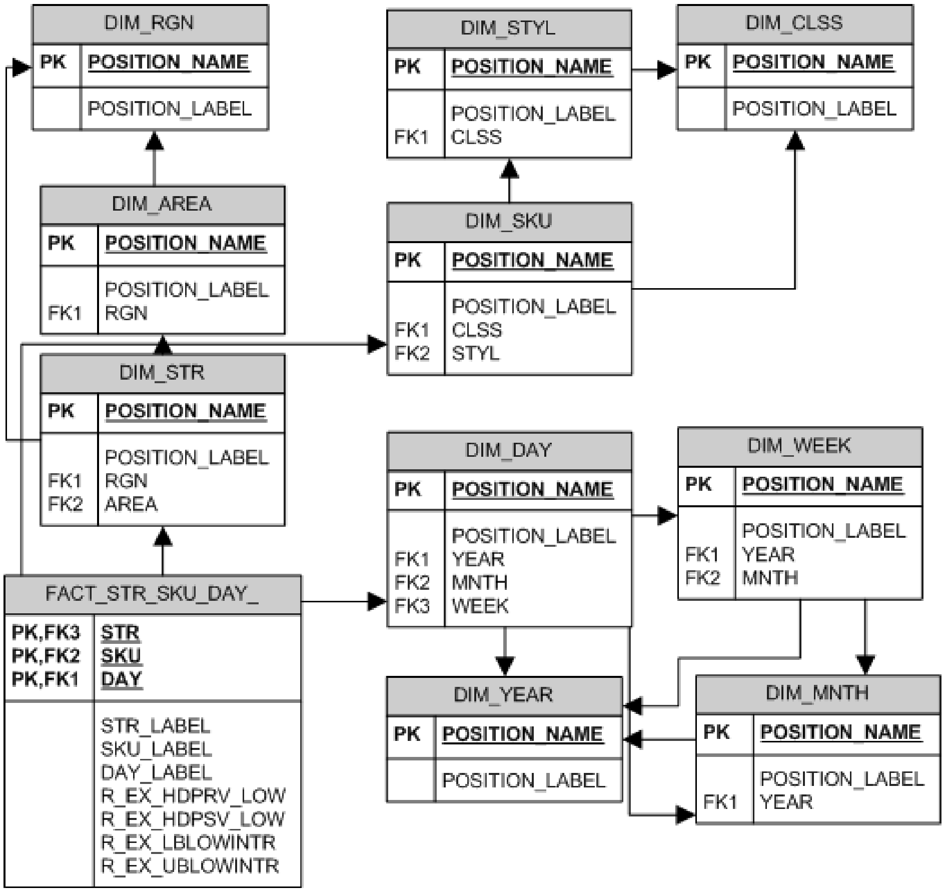
Fact and Dimension Tables

The following figure shows an example of the structure of fact and dimension tables and the relationships between them. A fact table represents an intersection where one or more measures' data is stored. Each measure is represented by a column in the table. Additionally, each dimension on the intersection is represented by a column. A record in the fact table is uniquely identified by a unique combination of position names for the intersecting dimensions.

A dimension table represents a dimension. It includes a column to list all position names, their labels, and their rollup mapping to each dimension at higher levels in the hierarchy.

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

Figure 16–24   Example of Star - De-normalized Schema to Represent Facts and Dimensions in RPAS


DIM_RGN
PK POSITION_NAME
POSITION_LABEL
DIM_STYL
PK POSITION_NAME
POSITION_LABEL
FK1 CLSS
DIM_CLSS
PK POSITION_NAME
POSITION_LABEL
DIM_AREA
PK POSITION_NAME
POSITION_LABEL
FK1 RGN
DIM_SKU
PK POSITION_NAME
POSITION_LABEL
FK1 CLSS
FK2 STYL
DIM_STR
PK POSITION_NAME
POSITION_LABEL
FK1 RGN
FK2 AREA
DIM_DAY
PK POSITION_NAME
POSITION_LABEL
FK1 YEAR
FK2 MNTH
FK3 WEEK
DIM_WEEK
PK POSITION_NAME
POSITION_LABEL
FK1 YEAR
FK2 MNTH
FACT_STR_SKU_DAY_
PK,FK3 STR
PK,FK2 SKU
PK,FK1 DAY
STR_LABEL
SKU_LABEL
DAY_LABEL
R_EX_HDPRV_LOW
R_EX_HDPSV_LOW
R_EX_LBLOWINTR
R_EX_UBLOWINTR
DIM_YEAR
PK POSITION_NAME
POSITION_LABEL
DIM_MNTH
PK POSITION_NAME
POSITION_LABEL
FK1 YEAR

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

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

For information on limitations when performing queries, see "Limitations".

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 to which they have access.

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

When users 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 exist in the workbook that the users can access, those measures (their workbook copies) are accessible to the users.

Use Cases

This section describes some use cases of RPAS.

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

1. Fetch the list of workbook schema names. In this example, the workbook is owned by 'USER01', built using the template 'TestTemplate', and 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 the 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 explain the structure of a particular hierarchy (for example: CLND) the following table lists 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 ODBC/JDBC Driver 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. Select **Start**, **Settings**, **Control Panel**, **Administrative Tools**, and then **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 to which you want to connect. 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**, and then **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, to see a fact table for the intersections

'DEPT' and 'DEPT_YEAR', use the following values for this custom property: 'AGG_TABLE_NAMES=FACT_DEPT, FACT_DEPT_YEAR'.

4. Click **OK**.

   If you are entering more than one connection property (that is, both the WORKBOOK_SCHEMA and AGG_TABLE_NAMES properties), separate the property key value pairs by a semicolon. Using examples above, the content of the custom properties input box should be: 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 datastore through the RPAS ODBC/JDBC Driver. The examples in this section do not include all client applications that can connect to the RPAS ODBC/JDBC Driver.

Note: In a client/server configuration, the server (executable) must be started before a client can connect to it.

Oracle Business Intelligence Enterprise Edition (OBIEE)

This section outlines 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, refer to OBIEE documentation. The user must install and configure the ODBC client first on the OBIEE server host (refer to section **ODBC Client Configuration for UNIX**) and test the connection. The ODBC client and the OBIEE server must both be 32-bit or 64-bit. The administrator must source the oaodbc.sh or oaodbc64.sh script under the ODBC client home directory before starting or restarting the OBIEE server.

Configuring the ODBC Client for OBIEE

The following example provides a sample configuration of the ODBC client for OBIEE. This example was developed for OBIEE on AIX, but the process is the same for other environments.

1. Open the $BIEE_HOME/setup/odbc.ini file, where $BIEE_HOME is the directory where OBIEE is installed.
2. Set the TraceDll to the odbctrac.so that comes with RPAS odbcclient. Set InstallDir to the RPAS odbcclient installation directory.
3. 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.

4. Create a section in the file for the rpas_domain. The following example is subject to changes. Refer to the [SampleRPAS] section in odbc.ini or odbc64.ini under ODBC client home directory for all up-to-date settings.

   Example:

```
[rpas_domain]
Driver= absolute_path_to_odbc_client/lib[64]/ivoa22.so
Description=Oracle Retail RPAS ODBC Driver
Host=<RPAS ODBC Server host>
Port=<odbc_data_service_port>
ServerDataSource=<data_source_name>
UseLDAP=0
DistinguishedName=
Encrypted=0
LoadBalancing=0
AlternateServers=
ConnectionRetryCount=0
ConnectionRetryDelay=3
CustomProperties=
```

Save your changes to the file.

Connecting OBIEE to an RPAS Domain

To connect OBIEE to a predefined DSN for an RPAS Domain:

1. Make sure the following Windows services are running:
   - Oracle BI Java Host
   - Oracle BI Server
2. Start the OBIEE Administration Tool. Select **Start**, **All Programs**, **Oracle Business Intelligence**, and then **Administration**.
3. From the File menu, select **open - online**. A window appears to enter login credentials.
4. Enter the 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, select **Import-From database**. A window appears to select the connection type and RPAS user information.
6. In order to optimize the OBIEE queries issued to the RPAS ODBC driver, you must select the appropriate connection type when configuring OBIEE. Selecting the RPAS-specific connection type will cause OBIEE to generate physical queries that delegate data aggregations to RPAS. For more, see *Oracle Fusion Middleware Integrator's Guide for Oracle Business Intelligence Enterprise Edition* /11g Release 1 (11.1.1).

   Once you select the connection type, the RPAS schemas and tables appear in a new window.
7. Select the objects you want to import, and then click **import**. After 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**. Click **OK** to exit.
9. In the Admin Tool window, click **Save** to save your physical model.

Now that 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, presentation layer, and OBIEE Web interface, refer to OBIEE documentation.

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 the File menu, select **Get External Data- Link Tables** (or **Import** if you want to import the data from RPAS datastore to Access). A dialog box appears.
4. In the Files of type box, select **ODBC Databases()**.
5. Click the **Machine Data Source** tab and then double-click the pre-configured ODBC data source from which you want to link.
6. At the logon prompt, enter 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 and tables that you can import or link.
7. Click each table that you want to import or link and then click **OK**. If you are linking a table that does not have an index that uniquely identifies each record, then you see displayed a list of the fields in the linked table. Select a field, or a combination of fields, that uniquely identify each record and then click **OK**.

JDeveloper

JDeveloper works best with a native JDBC driver, which is included in the RPAS ODBC/JDBC Driver package.

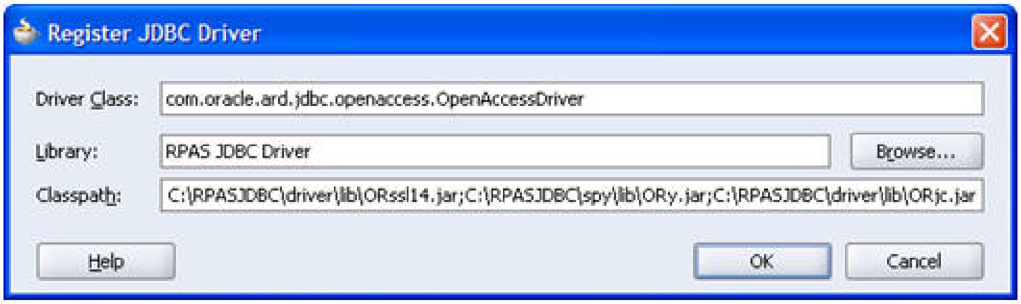
To connect using JDeveloper:

1. Start JDeveloper.
2. On the JDeveloper left panel, select the **Connections** tab.
3. Right-click on **Databases**, and select **New Database Connection**.
4. On the first screen of the Create New Database Connection wizard, enter a connection name and select **Third Party JDBC driver** for Connection Type.
5. On the second screen, enter the user name and password and then click **Next**.
6. On the third screen, click **New** to add the driver. It opens up the following dialog. You need to find the jar files ORjc.jar, ORssl14.jar, iaik_jce_full.jar, and ORy.jar and their paths. (They are made available from the installation of the JDBC Driver.) Then, create a library group "RPAS JDBC Driver" with those four jar files.

   In the Driver Class field, enter**com.oracle.ard.jdbc.openaccess.OpenAccessDriver**.

   Then enter in the URL field: **jdbc:RPAS://{host_name}:{port_number};ServerDataSource={data_source_name}**, where **host_name** is the host name or IP address of the ODBC server, **port_number** is the RPAS_Data_Service port number and **data_source_name** is the Data Source Name that is configured on the ODBC server.

Figure 16–25 Register JDBC Driver Dialog Box



Driver Class: com.oracle.ard.jdbc.openaccess.OpenAccessDriver

Library: RPAS JDBC Driver

Classpath: C:\RPASJDBC\driver\lib\ORssl14.jar;C:\RPASJDBC\spy\lib\ORy.jar;C:\RPASJDBC\driver\lib\ORjc.jar

7. Follow the instructions to finish creating the connection.

   After the connection is established, you can expand the Connection (and the nodes under the Connection) to browse the objects in the RPAS datastore. You can also open a SQL worksheet (by selecting **SQL Worksheet** from the Tools menu) to write or execute SQL statements.

XML Publisher

This section describes how to make the connection from XML Publisher to RPAS using the JDBC driver (XML only supports JDBC).

1. Install and configure the JDBC client driver and ODBC/JDBC server. Start the server.
2. Copy the JDBC client jar files ORjc.jar, ORssl14.jar, iaik_jce_full.jar, and ORy.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 the XML Publisher server. (Select **Start, All Programs**, **Oracle XML Publisher Server**, **OUIHome1**, and then **Oracle XML Publisher Enterprise Start**.)
4. Start a Web browser and 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. Log in as admin/admin.
6. Select the **Admin** tab and 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:RPAS://{host_name}:{port_number};ServerDataSource={data_source_name}** for the URL, where **host_name** is the host name or IP address of the ODBC server, **port_number** is the RPAS_Data_Service port number, and **data_source_name** is the Data Source Name that is configured on the ODBC server.
   - Enter **adm** for user name and password.
   - Enter "**com.oracle.ard.jdbc.openaccess.OpenAccessDriver**" 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 ODBC/JDBC SDK.

To connect to the remote ODBC/JDBC server, use odbcisql.exe (with ODBC client installed) or jdbcisql.class (with JDBC client installed).

> **Note:** Users are expected to know basic SQL in order to use ISQL.

To connect to the ODBC/JDBC server using odbcisql, start odbcisql and 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
```

After you are connected, you can issue various SQL DML and DDL statements to inspect and modify the data in the RPAS datastore.

Supported and Unsupported SQL Functions

This section contains the following information:

- Detailed descriptions of various functions supported by the RPAS ODBC Driver.
- Descriptions of the SQL92 and SQL99 functions that are not supported.

Supported SQL Functions

Use extreme caution when applying functions to any dimension name or label columns because the driver is not able to use the corresponding internal indexes to optimize row selection when functions are applied to those columns (which could be a significant performance hit).

It is suggested that users avoid applying functions to dimension name or label columns whenever possible.

Consider the following query:

```
Select * from fact_str_sku_day
where convert(day, SQL_DATE) = curdate();
```

Even though this query selects the data for only one day, the driver has to scan the entire fact table and then apply the convert function to every row of the table.

Working with OBIEE, the same can be achieved using a variable, which holds the converted string value of the current date (in the same format as the "day" column). The query then becomes:

```
Select * from fact_str_sku_day
```

```
Where day = @curDateString;
```

The driver only reads the rows that meet the condition.

Numeric Functions

Table 16–3 Numeric Functions

Description	Function
ABS(*numeric_exp*)	Returns the absolute value of numeric_exp. For example: SELECT ABS(-1.0), ABS(0.0), ABS(1.0) FROM emp WHERE empno = 1; This returns three result columns with values 1, 0, and 1.
ACOS(*float_exp*)	Returns the arccosine of float_exp as an angle, expressed in radians. For example: SELECT ACOS(-1) FROM emp WHERE empno = 1; This returns 3.14159.
ASIN(*float_exp*)	Returns the arcsine of float_exp as an angle, expressed in radians. For example: SELECT ASIN(-1.0) FROM emp WHERE empno = 1; This returns -1.57079.
ATAN(*float_exp*)	Returns the arctangent of float_exp as an angle, expressed in radians. For example: SELECT ATAN(45.0) FROM emp WHERE empno = 1; This returns 1.54857.
ATAN2(*float_exp1, float_exp2*)	Returns the arctangent of the x and y coordinates, specified by float_exp1 and float_exp2, respectively, as an angle, expressed in radians. For example: SELECT ATAN2(35.175, 129.44) FROM emp WHERE empno = 1; This returns 0.2653399.
CEILING(*numeric_exp*)	Returns the smallest integer greater than or equal to numeric_exp. The return value is of the same data type as the input parameter. For example: SELECT CEILING(123.45), CEILING(-123.45), CEILING(0.0) FROM emp WHERE empno = 1; This returns 124, -123 and 0.
COS(*float_exp*)	Returns the cosine of *float_exp*, where *float_exp* is an angle expressed in radians. For example: SELECT COS(14.78) FROM emp WHERE empno = 1; This returns -0.59946542.
COT(*float_exp*)	Returns the cotangent of *float_exp*, where *float_exp* is an angle expressed in radians. For example: SELECT COT(124.78) FROM emp WHERE empno = 1; This returns -0.82045588.
DEGREES(*numeric_exp*)	Returns the number of degrees converted from *numeric_exp* radians. For example: SELECT DEGREES(3.143) FROM emp WHERE empno = 1; This returns 180.0806.

Table 16–3 (Cont.) Numeric Functions

Description	Function
EXP(*float_exp*)	Returns the exponential value of *float_exp*. For example: SELECT EXP(378.615) FROM emp WHERE empno = 1; This returns 2.69404760606322E+164.
FLOOR(*numeric_exp*)	Returns the largest integer less than or equal to *numeric_exp*. The return value is of the same data type as the input parameter. For example: SELECT FLOOR(123.45), FLOOR(-123.45) FROM emp WHERE empno = 1; This returns 123 and -124.
LOG(*float_exp*)	Returns the natural logarithm of *float_exp*. For example: SELECT LOG(5.175643) FROM emp WHERE empno = 1; This returns 1.64396358.
LOG10(*float_exp*)	Returns the base 10 logarithm of *float_exp*. For example: SELECT LOG10(145.175643) FROM emp WHERE empno = 1; This returns 2.161893758. SELECT LOG10(0), LOG10(-1), LOG10(1) FROM emp WHERE empno = 1; This returns -1.#INF, -1.#IND and 0.
MOD(*integer_exp1, integer_exp2*)	Returns the remainder (modulus) of *integer_exp1* divided by *integer_exp2*. For example: SELECT mod(empno, 2) FROM emp WHERE empno = 11; This returns 1.
PI( )	Returns the constant value of pi as a floating-point value. For example: SELECT PI() FROM emp WHERE empno = 1; This returns 3.14159265358979.
POWER(*numeric_exp, integer_exp*)	Returns the value of *numeric_exp* to the power of *integer_exp*. For example: SELECT POWER(2, -5), POWER(2, 5) FROM emp WHERE empno = 1; This returns 0, 32.
RADIANS(*numeric_exp*)	Returns the number of radians converted from *numeric_exp* degrees. For example: SELECT RADIANS(45.0) FROM emp WHERE empno = 1; This returns 0.785398.
RAND([*integer_exp*])	Returns a random floating-point value using *integer_exp* as the optional seed value. For example: SELECT RAND(0) FROM emp WHERE empno = 1; This returns 38.

Table 16–3 (Cont.) Numeric Functions

Description	Function
ROUND(*numeric_exp*, *integer_exp*)	Returns *numeric_exp* rounded to *integer_exp* places right of the decimal point. If *integer_exp* is negative, *numeric_exp* is rounded to |*integer_exp*| places to the left of the decimal point. For example: SELECT ROUND(123.344, 2), ROUND(123.345, 2) FROM emp WHERE empno = 1; This returns 123.34 and 123.35. SELECT ROUND(748.58, -1), ROUND(748.58, -2), ROUND(748.58, 3), FROM emp WHERE empno = 1; This returns 750, 700 and 1000.
SIGN(*numeric_exp*)	Returns the positive (+1), zero (0), or negative (-1) sign of the given expression. For example: SELECT SIGN(empno) FROM emp WHERE empno = 11; This returns 1. SELECT SIGN(-1 * empno), SIGN(0) FROM emp WHERE empno = 1; This returns two result columns with values -1 and 0.
SIN(*float_exp*)	Returns the sine of *float_exp*, where *float_exp* is an angle expressed in radians. For example: SELECT SIN(1.570796) FROM emp WHERE empno = 11; This returns 0.999999.
SQRT(*float_exp*)	Returns the square root of *float_exp*. For example: SELECT SQRT(45.35) FROM emp WHERE empno = 11; This returns 6.7342.
TAN(*float_exp*)	Returns the tangent of float_exp, where *float_exp* is an angle expressed in radians. For example: SELECT TAN(0.785398) FROM emp WHERE empno = 11; This returns 0.999999.
TRUNCATE(*numeric_exp*, *integer_exp*)	Returns *numeric_exp* truncated to *integer_exp* places right of the decimal point. If *integer_exp* is negative, *numeric_exp* is truncated to |*integer_exp*| places to the left of the decimal point.
NCHAR(code)	Returns the Unicode character that has the specified code as a SQL_WCHAR value. The valued of code should be between 0 and 65535. Example: "SELECT NCHAR(945)" returns the character α.

String Functions

Table 16–4 String Functions

Function	Description
ASCII(*string_exp*)	Returns the ASCII code value of the leftmost character of string_exp as an integer. For example: SELECT ASCII(ename) FROM emp WHERE ename = 'Adam'; This returns 65 which is the ASCII code of A.

Table 16–4 (Cont.) String Functions

Function	Description
BIT_LENGTH(*string_exp*)	Returns the length in bits of the string expression. For example: SELECT BIT_LENGTH(ename) FROM emp WHERE ename = 'John'; This returns 32, which is the number of bits.
CHAR(*code*)	Returns the character that has the ASCII code value specified by code. The value of code should be between 0 and 255; otherwise, the return value is data source-dependent. For example: SELECT CHAR(65) FROM emp; This returns A which is the character for ASCII code A.
CHAR_LENGTH(*string_exp*) CHARACTER_LENGTH(*string_exp*)	Returns the length in characters of the string expression, if the string expression is of a character data type; otherwise, returns the length in bytes of the string expression (the smallest integer not less than the number of bits divided by 8). (This function is the same as the CHARACTER_LENGTH function.) For example: SELECT CHAR_LENGTH(ename) FROM emp where ename = 'John'; This returns 4.
CONCAT(*string_exp1, string_exp2*)	Returns a character string that is the result of concatenating string_exp2 to string_exp1. If either of string_exp1 or string_exp2 is NULL value, it returns NULL string. If either of string_exp1 or string_exp2 is wide character string, the return value is a wide character string. For example: SELECT CONCAT('Name is: ', ename) FROM emp WHERE ename = 'John'; This returns 'Name is: John'. SELECT CONCAT(N'Name is: ', ename) FROM emp WHERE ename = N'John'; This returns wide character string N'Name is: John'.
INSERT(*string_exp1, start, length, string_exp2*)	Returns a character string where length characters have been deleted from string_exp1, beginning at start, and where string_exp2 has been inserted into string_exp, beginning at start. If string_exp1 is wide character string, the return value is a wide character string. Offsets (start and length) must be specified in number of characters. For example: SELECT INSERT(ename, 1, 0, 'Name is: ') FROM emp WHERE ename = 'John'; This returns 'Name is: John'. SELECT INSERT(ename, 1, 0, N'Name is: ') FROM emp WHERE ename = N'John'; If ename is a column of wide character data type, this returns wide character string: N'Name is: John'.
LCASE(string_exp) LOWER(string_exp)	Returns a string equal to that in string_exp, with all uppercase characters converted to lowercase. For example: SELECT LCASE(ename) FROM emp WHERE ename is 'John'; This returns 'john'.

Table 16–4 (Cont.) String Functions

Function	Description
LEFT(*string_exp*, *count*)	Returns the leftmost count characters of string_exp. If string_exp is wide character string, the return value is a wide character string. Offset (count) must be specified in number of characters. For example: SELECT LEFT(ename, 2) FROM emp WHERE ename = 'John'; This returns 'jo'. SELECT LEFT(ename, 2) FROM emp WHERE ename = N'John'; If ename is a column of wide character data type, this returns wide character string N'jo'.
LENGTH(*string_exp*)	Returns the number of characters in string_exp, excluding trailing blanks. If string_exp is wide character string, the return value is a number of wide characters in string_exp. Trailing blanks are not checked in wide character implementation. For example: SELECT LENGTH('John '), LENGTH('John') FROM emp; This returns 4 for both result columns as trailing blanks are excluded. SELECT LENGTH(N'John '), LENGTH(N'John') FROM emp; This returns 7 for the first result column and 4 for the second result column. Trailing blanks are not checked in wide character implementation.
LOCATE(*string_exp1*, *string_exp2[*, *start*])	Returns the starting position of the first occurrence of string_exp1 within string_exp2. The search for the first occurrence of string_exp1 begins with the first character position in string_exp2 unless the optional argument, start, is specified. If start is specified, the search begins with the character position indicated by the value of start. The first character position in string_exp2 is indicated by the value 1. If string_exp1 is not found within string_exp2, the value 0 is returned. If string_exp2 is a wide character string, returns the starting position of the first occurrence of string_exp1 within the wide character string string_exp2. Offset (start) must be specified in number of characters. If string _exp2 is a wide character exp, the result is computed by treating both arguments as wide character string. For example: SELECT LOCATE('h', 'John', 1) FROM emp; This returns 3 as 'h' is the found at the third position. SELECT LOCATE(N'h', N'John', 1) FROM emp; This returns 3 as N'h' is the found at the third position.
LTRIM(*string_exp*)	Returns the characters of string_exp, with leading blanks removed. For example: SELECT LTRIM(' ABC') FROM emp; This returns 'ABC'.
OCTET_LENGTH(*string_exp*)	Returns the length in bytes of the string expression. The result is the smallest integer not less than the number of bits divided by 8. For example: SELECT OCTET_LENGTH(ename) FROM emp WHERE ename = 'John'; This returns 4.

Table 16–4 (Cont.) String Functions

Function	Description
POSITION(*character_exp1 character_exp2*)	Returns the position of the first character expression in the second character expression. The result is an exact numeric with an implementation-defined precision and a scale of 0. If character_exp1 and character_exp2 are wide character strings, returns the position of the first wide character expression in the second wide character expression. If character_exp2 is a wide character string, the result is computed by treating both arguments as wide character strings. For example: SELECT POSITION('abc', '1234abc def') FROM emp; This returns 5. SELECT POSITION(N'abc', N'1234abc def') FROM emp; This returns 5.
REPEAT(*string_exp, count*)	Returns a character string composed of string_exp repeated count times. If string_exp is wide character string, the return value is a wide character string. For example: SELECT REPEAT(ename, 2) FROM emp WHERE ename = 'John'; This returns 'JohnJohn'. SELECT REPEAT(ename, 2) FROM emp WHERE ename = N'John'; If ename is a column of wide character data type, this returns N'JohnJohn'.
REPLACE(*string_exp1, string_exp2, string_exp3*)	Search string_exp1 for occurrences of string_exp2, and replace with string_exp3. If string_exp1 is wide character string, the return value is a wide character string. For example: SELECT REPLACE(address, 'San Francisco', 'SFO') FROM emp where address = '100 Vanness, San Francisco'; This returns '100 Vanness, SFO'. SELECT REPLACE(address, N'San Francisco', N'SFO') FROM emp WHERE address = N'100 Vanness, San Francisco'; If address is a column of wide character data type, this returns N'100 Vanness, SFO'.
RIGHT(*string_exp, count*)	Returns the right-most count characters of string_exp. If string_exp is wide character string, the return value is a wide character string. Offset (count) must be specified in number of characters. For example: SELECT RIGHT(ename, 2) FROM emp WHERE ename = 'John'; This returns 'hn'. SELECT RIGHT(ename, 2) FROM emp WHERE ename = N'John'; If ename is a column of wide character data type, this returns N'hn'.
RTRIM(*string_exp*)	Returns the characters of string_exp with trailing blanks removed. For example: SELECT RTRIM('abc ') FROM emp; This returns 'abc'.

Table 16–4 (Cont.) String Functions

Function	Description
SPACE(*count*)	Returns a character string consisting of count spaces. For example: SELECT ename+space(5)+ename FROM emp WHERE ename = 'John'; This returns 'John John'.
SUBSTRING(*string_exp*, *start*, length) SUBSTR(*string_exp*, length)	Returns a character string that is derived from string_exp, beginning at the character position specified by start for length characters. If string_exp is wide character string, the return value is a wide character string. Offset (start and length) must be specified in number of characters. For example: SELECT SUBSTR(ename, 1, 3) FROM emp WHERE ename ='John'; This returns 'Joh'. SELECT SUBSTR(ename, 1, 3) FROM emp WHERE ename = N'John'; If ename is a column of wide character data type, this returns N'Joh'.
UCASE(*string_exp*) UPPER(*string_exp*)	Returns a string equal to that in string_exp, with all lowercase characters converted to uppercase. For example: SELECT UCASE(ename) FROM emp WHERE ename = 'John'; This returns 'JOHN'.
UNICODE(string_exp)	Returns the Unicode code of the first character of the string_exp as a SQL_INTEGER value. Example: "SELECT UNICODE('αβγ')" returns an integer value of 945.

Time and Date Functions

Table 16–5 Time and Date Functions

Function	Description
CURDATE()	Returns the current date. For example: SELECT CURDATE() FROM emp; Returns the current date as: 2008-10-25
CURTIME()	Returns the current local time. For example: SELECT CURTIME() FROM emp; Returns the current time as: 10:20:05
CURTIMESTAMP()	Returns the current local date and local time as a timestamp value. For example: SELECT CURTIMESTAMP() FROM emp; Returns current date and time as: 2003-03-31 14:08:57

Table 16–5 (Cont.) Time and Date Functions

Function	Description
DATEADD(*datepart* , *number* ,*date*) TIMESTAMPADD(*datepart* , *number* , *date*)	Returns a new date time value based on adding an interval to the specified date. The return date-time data type is same as the input *date* value. *datepart*: the parameter that specifies on which part of the date to return a new value. Both ODBC notation and SQL Server notation for datepart are supported. Datepart Abbreviations Year SQL_TSI_YEAR , year, yy, yyyy, quarter SQL_TSI_QUARTER, quarter, qq, q Month SQL_TSI_MONTH, month, mm, m dayofyear DAYOFYEAR, dy, y Day SQL_TSI_DAY, day, dd, d Week SQL_TSI_WEEK, week, wk, ww Hour SQL_TSI_HOUR, hour, hh minute SQL_TSI_MINUTE , minute, mi, n second SQL_TSI_SECOND, second, ss, s The current implementation does not support millisecond and fractional second specifications. *number*: the value used to increment the datepart. If value is not an integer, the fractional part of the value is discarded. For example, if you specify **day** for *datepart* and **1.75** for *number*, *date* is incremented by 1. *date:* an expression that returns a date or timestamp value or a character string in a date-time format.

Table 16–5 (Cont.) Time and Date Functions

Function	Description
DATEDIFF(*datepart* , *startdate* , *enddate*) TIMESTAMPDIFF(*datepart* , *startdate* , *enddate*)	Returns the number of date and time boundaries crossed between two specified dates. *startdate* is subtracted from *enddate*. If *startdate* is later than *enddate*, a negative value is returned. *datepart*: the parameter that specifies on which part of the date to calculate the difference. Both ODBC notation and SQLServer notation for datepart are supported. Datepart Abbreviations Year SQL_TSI_YEAR, year, *yy*, *yyyy*, quarter SQL_TSI_QUARTER, quarter, *qq*, *q* Month SQL_TSI_MONTH, month, *mm*, *m* dayofyear DAYOFYEAR, *dy*, *y* Day SQL_TSI_DAY, day, *dd*, *d* Week SQL_TSI_WEEK, week, *wk*, *ww* Hour SQL_TSI_HOUR, hour, *hh* minute SQL_TSI_MINUTE, minute, *mi*, *n* second SQL_TSI_SECOND, second, *ss*, *s* The current implementation does not support dayofyear, millisecond and factional second specifications. For example: SELECT DATEDIFF(year, hiredate, curdate()) FROM emp WHERE hiredate = '2000-10-01'; SELECT DATEDIFF(SQL_TSI_YEAR, hiredate, curdate()) FROM emp WHERE hiredate = '2000-10-01'; This returns 2 (assuming that the curdate() returns year 2002).
DAYNAME(*date_exp*)	Returns a character string containing the data source-specific name of the day (for example, Sunday through Saturday or Sun. through Sat. for a data source that uses English, or Sonntag through Samstag for a data source that uses German) for the day portion of *date_exp*. For example: SELECT DAYNAME('2002-01-01'), DAYNAME('2002-01-02') FROM emp; This returns 'Tuesday' and 'Wednesday'.
DAYOFMONTH(*date_exp*)	Returns the day of the month based on the month field in *date_exp* as an integer value in the range of 1-31. For example: SELECT DAYOFMONTH('2002-01-05') FROM emp; This returns 5.
DAYOFWEEK(*date_exp*)	Returns the day of the week based on the week field in *date_exp* as an integer value in the range of 1-7, where 1 represents Sunday. For example: SELECT DAYOFWEEK('2002-01-05') FROM emp; This returns 7.
DAYOFYEAR(*date_exp*)	Returns the day of the year based on the year field in *date_exp* as an integer value in the range of 1-366. For example: SELECT DAYOFYEAR('2002-01-05') FROM emp; This returns 5.

Table 16–5 (Cont.) Time and Date Functions

Function	Description
HOUR(*time_exp*)	Returns the hour based on the hour field in *time_exp* as an integer value in the range of 0-23. For example: SELECT HOUR('22:20:20') FROM emp; This returns 22.
MINUTE(*time_exp*)	Returns the minute based on the minute field in *time_exp* as an integer value in the range of 0-59. For example: SELECT MINUTE('22:21:20') FROM emp; This returns 21.
MONTH(*date_exp*)	Returns the month based on the month field in *date_exp* as an integer value in the range of 1-12. For example: SELECT MONTH('2002-01-05') FROM emp; This returns 1.
MONTHNAME(*date_exp*)	Returns a character string containing the data source-specific name of the month (for example, January through December or Jan. through Dec. for a data source that uses English, or Januar through Dezember for a data source that uses German) for the month portion of *date_exp*. For example: SELECT MONTHNAME('2002-01-05') FROM emp; This returns January.
NOW( )	Returns current date and time as a timestamp value. For example: SELECT NOW() FROM emp; This returns the current date and time: 2002-10-25 10:20:05.
QUARTER(*date_exp*)	Returns the quarter in *date_exp* as an integer value in the range of 1-4, where 1 represents January 1 through March 31. For example: SELECT QUARTER('2002-01-05') FROM emp; This returns 1.
SECOND(*time_exp*)	Returns the second based on the second field in *time_exp* as an integer value in the range of 0-59. For example: SELECT SECOND('22:21:20') FROM emp; This returns 20.
WEEK(*date_exp*)	Returns the week of the year based on the week field in *date_exp* as an integer value in the range of 1-53. For example: SELECT WEEK('2002-01-05') FROM emp; This returns 1.
YEAR(*date_exp*)	Returns the year based on the year field in *date_exp* as an integer value. For example: SELECT YEAR('2002-01-01') FROM emp; This returns 2002.

System Functions

Table 16–6 System Functions

Function	**Description**
DATABASE( )	Returns the name of the database corresponding to the connection handle. The name of the database is also available by calling SQLGetConnectOption with the SQL_CURRENT_QUALIFIER connection option.
USER( )	Returns the user name in the DBMS. The user name is also available using SQLGetInfo by specifying the information type: SQL_USER_NAME. This can be different than the login name.

Aggregate Functions

Aggregate functions return a single row based on groups of rows, rather than on single rows.

Table 16–7 Aggregate Functions

Function	**Description**
AVG([ALL | DISTINCT] *expression*)	Returns the average of the values in a group. Null values are ignored.
SUM([ALL | DISTINCT] *expression*)	Returns the sum of all the values, or only the DISTINCT values, in the expression. SUM can be used with numeric columns only. Null values are ignored.
COUNT({[ALL | DISTINCT] *expression*] | *})	Returns the number of items in a group. COUNT(*) returns the number of items in a group, including NULL values and duplicates. COUNT(ALL *expression*) evaluates *expression* for each row in a group and returns the number of non-null values. COUNT(DISTINCT *expression*) evaluates *expression* for each row in a group and returns the number of unique, non-null values.
MAX([ALL | DISTINCT] *expression*)	Returns the maximum value in the expression.
MIN([ALL | DISTINCT] *expression*)	Returns the minimum value in the expression.

Aggregate functions can appear in SELECT lists and HAVING clauses. If you use the GROUP BY clause in a SELECT statement, OpenAccess SDK divides the rows of a queried table or view into groups. In a query containing a GROUP BY clause, all elements of the SELECT list must be expressions from the GROUP BY clause, expressions containing aggregate functions, or constants. OpenAccess SDK applies the aggregate functions in the SELECT list to each group of rows and returns a single result row for each group.

If you omit the GROUP BY clause, OpenAccess SDK applies aggregate functions in the SELECT list to all the rows in the queried table or view.

Many aggregate functions accept these options:

- DISTINCT causes an aggregate function to consider only distinct values of the argument expression.
- ALL causes an aggregate function to consider all values, including all duplicates.

For example:

```
SELECT max(sal), MIN(sal), AVG(sal) FROM emp;
SELECT deptno, MAX(sal), SUM(sal) FROM emp GROUP BY deptno;
SELECT deptno, COUNT(empno) FROM emp GROUP BY deptno;
```

Other Functions

DECODE

Syntax

```
DECODE (expr, [search, result]..., default)
```

Example

```
SELECT DECODE (deptno,10, 'ACCOUNTING',
               20, 'RESEARCH',
               30, 'SALES',
               40, 'OPERATION',
                    'NONE')
      FROM dept
```

To evaluate this expression, the OpenAccess SDK SQL engine compares expr to each search value one by one. If expr is equal to a search, the OpenAccess SDK SQL engine returns the corresponding result. If no match is found, the OpenAccess SDK SQL engine returns default, or if default is omitted, returns null. The return value is the same data type as the first result expression. The search, result, and default values can be derived from expressions.

The OpenAccess SDK SQL engine evaluates each search value only before comparing it to expr, rather than evaluating all search values before comparing any of them with expr. Consequently, OpenAccess SDK SQL engine never evaluates a search if a previous search is equal to expr.

The OpenAccess SDK SQL engine automatically converts expr and each search value to the datatype of the first search value before comparing. The OpenAccess SDK SQL engine automatically converts the return value to the same datatype as the first result. If the first result has the datatype CHAR or if the first result is null, then the OpenAccess SDK SQL engine converts the return value to the datatype of CHAR.

In a DECODE expression, the OpenAccess SDK SQL engine considers two nulls to be equivalent.

If expr is null, the OpenAccess SDK SQL engine returns the result of the first search that is also null. The maximum number of components in the DECODE expression, including expr, searches, results, and default is 255.

Examples:

```
SELECT DECODE(empno, 1, 'E1', 2, 'E2', 'DEFAULT') FROM emp;

# First Result expression is NULL. Result should be type
XO_TYPE_CHAR
SELECT DECODE(empno, 1, NULL, 2, 'E2', 'DEFAULT') FROM emp;

# Input expression is NULL, Result should match the result
of NULL search expr
SELECT DECODE(ename, 'Bob', 'My Bob', 'Mary', 'My Mary',
NULL, 'New Name', 'Default Name') FROM emp;

# no default value, so return NULL for non-match values
SELECT DECODE(empno, 1, 'E1', 2, 'E2') FROM emp;
```

```
# ERROR CHECKING
# Invalid number of arguments
# Invalid syntax used with scalar function:DECODE. Function
expects 3 arguments.
SELECT DECODE() FROM emp;
SELECT DECODE(empno, 1) FROM emp;

# Conversion errors
# decode() Error converting value of result expression to
XoType:<4>

SELECT DECODE(empno, 1, 10, 2, 20, 'abc') FROM emp;
```

IFNULL, ISNULL, NVL

These functions allow the NULL value to be replaced by a default value. OpenAccess SDK supports IFNULL as defined by ODBC, ISNULL as defined by SQL Server, and NVL as defined by Oracle.

Syntax

```
IFNULL (expr, default_val)
ISNULL (expr, default_val)
NVL (expr, default_val)
```

The OpenAccess SDK SQL engine evaluates the input expression and returns the expression value if it is non-NULL. If the expression value is NULL, default_val is returned. The return value is of the same data type as the input expression.

Example

```
SELECT ename, IFNULL (sal, 1000) FROM emp;
SELECT ename, ISNULL (sal, 1000) FROM emp;
SELECT ename, NVL (sal, 1000) FROM emp;
SELECT ename, IFNULL (hiredate, '2001-01-01') FROM emp;
```

CAST

Syntax

`CAST (`*`value_exp`*` AS `*`data_type`*`)`

Example

```
SELECT empno, CAST(empno AS VARCHAR) FROM emp
SELECT empno, CAST(empno AS SMALLINT) FROM emp
```

The function returns the value specified by value_exp converted to the specified *data_type,* where *data_type* is one of the following:

- CHAR
- NUMERIC
- DECIMAL
- INTEGER
- SMALLINT
- FLOAT
- REAL

- DOUBLE
- DATE
- TIME
- TIMESTAMP
- VARCHAR
- LONGVARCHAR
- BINARY
- VARBINARY
- LONGVARBINARY
- TINYINT
- BIT
- WCHAR,
- WVARCHAR
- WLONGVARCHAR

The following table defines the precision, length, and scale keywords of the CAST function.

Table 16–8 CAST Function

Keyword	Value
CHAR	255
BINARY	255
BIT	1
DATE	6
DOUBLE	8
FLOAT	8
INTEGER	4
LONGVARBINARY	1000000
LONGVARCHAR	1000000
NUMERIC	34
SMALLINT	2
REAL	4
TIME	6
TIMESTAMP	16
TINYINT	1
VARBINARY	1024
VARCHAR	1024
WLONGVARCHAR	2000000
WVARCHAR	512
WVARCHAR	2048

Table 16–8 (Cont.) CAST Function

Keyword	Value
CHAR, BINARY	255
DATE	10
DOUBLE	15
FLOAT	15
INTEGER	10
LONGVARBINARY	1000000
LONGVARCHAR	1000000
NUMERIC	32
REAL	7
SMALLINT	5
TIME	6
TINYINT	3
VARBINARY	1024
VARCHAR	1024
WCHAR	255
WVARCHAR	1024
WLONGVARCHAR	1000000
NUMERIC	5
All other types	0

CONVERT

Syntax

```
CONVERT (value_exp, data_type)
```

Example

```
SELECT empno, CONVERT(empno, SQL_VARCHAR) FROM emp
SELECT empno, CONVERT(empno, SQL_SMALLINT) FROM emp
```

The function returns the value specified by value_exp converted to the specified data_type, where data_type is one of the following:

- SQL_CHAR
- SQL_NUMERIC
- SQL_DECIMAL
- SQL_INTEGER
- SQL_SMALLINT
- SQL_FLOAT
- SQL_REAL
- SQL_DOUBLE
- SQL_DATE

- SQL_TIME
- SQL_TIMESTAMP
- SQL_VARCHAR
- SQL_LONGVARCHAR
- SQL_BINARY
- SQL_VARBINARY
- SQL_LONGVARBINARY
- SQL_TINYINT
- SQL_BIT
- SQL_WCHAR
- SQL_WVARCHAR
- SQL_WLONGVARCHAR

The following tables define the length, precision, and scale keywords of the CONVERT function.

Table 16–9 CONVERT Function-Length

Keyword	Length
SQL_CHAR	256
SQL_BINARY	256
SQL_BIT	1
SQL_DATE	6
SQL_DOUBLE	8
SQL_FLOAT	8
SQL_INTEGER	4
SQL_LONGVARBINARY	1000000
SQL_LONGVARCHAR	1000000
SQL_NUMERIC	34
SQL_SMALLINT	2
SQL_REAL	4
SQL_TIME	6
TIMESTAMP	16
TINYINT	1
VARBINARY	1024
VARCHAR	1024
WLONGVARCHAR	2000000
WVARCHAR	512
WVARCHAR	2048

Table 16–10 CONVERT Function-Precision

Keyword	Precision
SQL_BINARY	255
SQL_BIT	1
SQL_CHAR	255
SQL_DATE	10
SQL_DOUBLE	15
SQL_FLOAT	15
SQL_INTEGER	10
SQL_LONGVARBINARY	1000000
SQL_LONGVARCHAR	1000000
SQL_NUMERIC	32
SQL_REAL	7
SQL_SMALLINT	5
SQL_TIME	8
SQL_TINYINT	3
SQL_VARBINARY	1024
SQL_VARCHAR	1024
SQL_WCHAR	255
SQL_WVARCHAR	2048
SQL_WLONGVARCHAR	1000000

Table 16–11 CONVERT Function-Scale

Keyword	Scale
SQL_NUMERIC	5
All other types	0

Unsupported SQL Functions

The SQL engine of the RPAS ODBC Server implements a large portion of the entry level SQL as defined in the X3.135-1992, "Database Language SQL" specification and commercial databases like SQL Server and Oracle. It is compliant with the ODBC minimal grammar specification.

This section describes the unsupported features.

Handling of NULLS

NOT IN should return FALSE if any member of the set is NULL. When evaluating the IN condition, the OpenAccess SDK SQL engine treats the comparison of any value with NULL as FALSE, so NOT IN becomes TRUE.

SELECT * FROM emp WHERE job NOT IN

(SELECT job FROM emp WHERE job IS NULL)

This example should return no results if there is an emp record with a NULL value for Job.

Schema Information

The SQL engine of the RPAS ODBC Driver does not support the following:

- Collate sequence and character set
- DEFAULT clause for column values

Data Definition Language (DDL)

The only DDL that is supported is "create view".

Insert

- Insert statements are not supported.
- Update measure data on fact table is supported.

SELECT Syntax

Subqueries are not supported in a SELECT list.

Example:

SELECT

(SELECT a.empno FROM emp a WHERE a.deptno = b.deptno)

FROM

dept b

Value Expressions

Special Values- The SQL engine does not support the use of special values (CURRENT_USER, SESSION_USER, CURRENT_TIMESTAMP) in value specification.

Value Functions

The SDK SQL engine does not support the following functions:

- TRANSLATE
- TRIM

> **Note:** The OpenAccess SDK SQL engine supports LTRIM and RTRIM.

- DIFFERENCE
- SOUNDEX

Date and Time Functions

The OpenAccess SDK SQL engine does not support the following Date and Time functions:

- CURRENT_TIME[(*time-precision*)] - The SQL engine does not support time-precision argument.
- CURRENT_TIMESTAMP[(*time-precision*)]

- EXTRACT(*extract-field FROMextract-source*)

Advanced Value Expressions

NULLIF

NULLIF is shorthand for a frequently used variation of CASE.

Syntax

```
NULLIF(value1, target_value)
is equivalent to
CASE
    WHEN value1 = = target_value THEN NULL
ELSE value1
END
Example:
.. WHERE sales_revenue / NULLIF(our_cost, -1) > 50
```

COALESCE

Coalesce is shorthand for a frequently used variation of CASE.

Syntax:

COALESCE (value1, value2, value3)

is equivalent to:

CASE

WHEN value1 IS NOT NULL THEN value1

WHEN value2 IS NOT NULL THEN value2

ELSE value3

END

Example:

SELECT name, job_title, COALESCE (salary, commission,

subsistence)

FROM job_assignments

Row Value Constructor

A row value constructor is a parenthesized list of values.

Example:

The following expression:

WHERE c1=CA AND c2=CB AND c3=CC

can be written using row value constructor as:

WHERE (c1, c2, c3) = (CA, CB, CC)

Predicates

The SQL engine does not support the following predicates.

- OVERLAPS predicate: Determines whether two intervals of time overlap with one another.event-information OVERLAPS event-information
- MATCH predicate

Join Operators

This section explains which Join operations are supported by the RPAS SQL engine, and which are not.

Supported Join Operators

The OpenAccess SDK SQL engine supports the following join operations:

- Implicit JOIN. The WHERE clause explicitly specifies the join condition.
- INNER JOIN. All joins that are not OUTER JOINS are considered in SQL terminology as INNER joins. The use of keyword INNER has no additional effects, but helps the statement to be completely self-documenting.
  - SELECT * FROM t1 INNER JOIN t2 ON t1.c1 = t2.c3 WHERE search-LEFT OUTER JOIN - This join preserves unmatched rows from the left table.
  - SELECT * FROM t1 LEFT OUTER JOIN t2 ON t1.c1 = t2.c3 WHERE search-condition
- UNION and UNION ALL operators. UNION is used for combining two result tables that are union compatible.

  SELECT c1, c2 FROM t1 UNION SELECT c3, c4 FROM t2

Unsupported Join Operators

The following join operations (syntax) are not supported in this release:

- CROSS JOIN: Functionally similar to the implicit joins.

  SELECT * FROM t1 CROSS JOIN t2
- NATURAL JOIN: Also referred to as natural, equi-join selects rows from the tables that have same value for columns with the same name.

  SELECT * FROM t1 NATURAL JOIN t2
- Condition JOIN: Uses the keyword ON to specify the JOIN condition between tables. The scope of fields referred in the ON condition is restricted.

  SELECT * FROM t1 JOIN t2 ON t1.c1 = t2.c3 WHERE search-condition
- Column Name JOIN: Specifies a more restricted form of NATURAL join. NATURAL joins use all columns with the same names to manage the matching process. The column name JOIN specifies which column values should be matched.

  SELECT * FROM t1 JOIN t2 USING (c1, c2)
- RIGHT OUTER JOIN: Preserves unmatched rows from the right table.

  SELECT * FROM t1 RIGHT OUTER JOIN t2 ON t1.c1 = t2.c3 WHERE search-condition
- FULL OUTER JOIN: Preserves unmatched rows from both the left and right tables.

  SELECT * FROM t1 FULL OUTER JOIN t2 ON t1.c1 = t2.c3 WHERE search-condition
- UNION JOIN: Creates a new virtual table with the union of all columns from the source tables. The UNION join has no provision for column matching.

17

Publishing Measure Change Events

Event driven planning requires the ability to identify events when they arise. This includes events that result from changes in plans and those that arise because of advancement in the planning activity, for example, approval of a plan or creation of new items. The ability to get notification of the event when it occurs is therefore essential. In the context of RPAS applications, many measure changes result from business activities and therefore fall into the category of notification-worthy events.

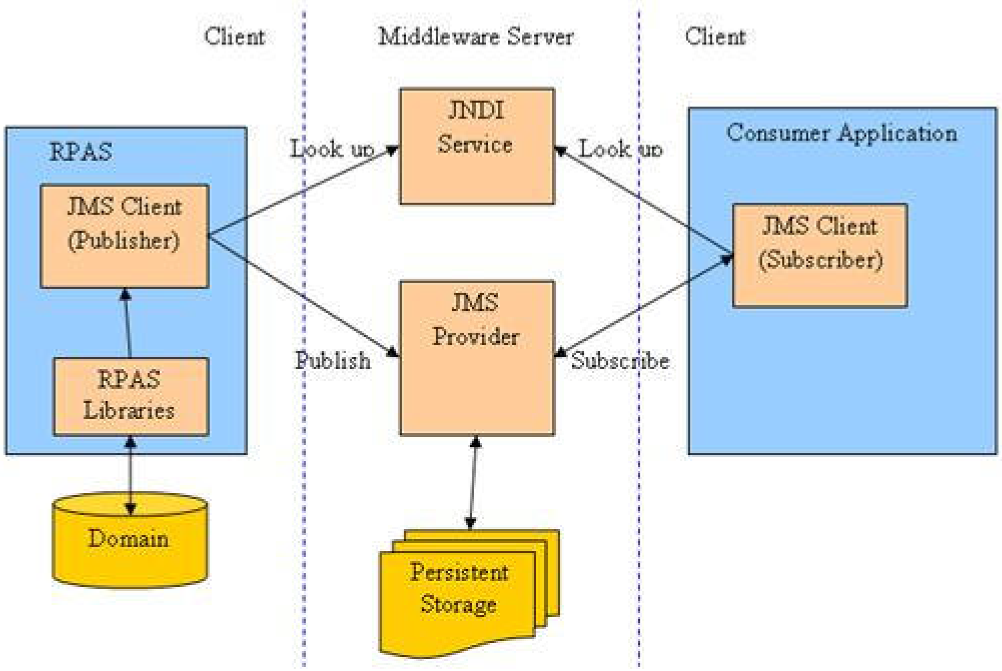
RPAS Publishing Measure Changes (PMC) provides a mechanism so that users can monitor measure changes and receive notification messages through a standard JMS messaging service. A measure change event is defined as the measure data being written by any means. The following is the list of the sources for PMC events:

- MACE: all left-hand side measures in the expression or expressions of the rule group specified in the command line.
- Workbook Commit: all left-hand side measures in the commit rule group.
- loadMeasure: all successfully loaded measures.
- loadHier: measures whose base intersection contains at least one affected dimension. Affected means adding and deleting one or more logical positions.
- Dynamic Position Management (DPM): measures whose base intersection contains a dimension affected by DPM. Adding or removing positions to a dimension or its lower level children constitutes a change in the measure.

Figure 17–1 demonstrates a JMS system in the context of RPAS. The JMS system consists of four basic components:

- JMS Provider: The central daemon that accepts connections from clients and routes and queues messages. It is sometimes called JMS Broker.
- Java Naming and Directory Interface (JNDI) Service: A service that provides mapping between names and objects. It adds abstraction to the more complex structure of underlying objects.
- JMS Client (Publisher): A standalone daemon process running with the other RPAS server side processes such as DomainDaemon and RpasDbServer.
- JMS Client (Subscriber): Any applications interested in getting notifications of measure changes.

Figure 17–1 JMS System in RPAS


Client
Middleware Server
Client
RPAS
JMS Client (Publisher)
RPAS Libraries
Domain
Look up
Publish
JNDI Service
JMS Provider
Persistent Storage
Look up
Subscribe
Consumer Application
JMS Client (Subscriber)

Configuring Subjects and Measures for Monitoring

A subject is defined as a logical grouping of measures. A subject is mapped to one or more measures in one domain, while a measure can belong to one or more subjects. For example, a subject named PlanningMeasures can include all measures associated with planning.

The configuration file of measure change monitoring serves two purposes:

- Defines the mappings between subjects and measures.
- Defines the inclusion filters for monitored measures.

By default, no measures are monitored unless they are specifically included in the configuration.

For each domain, there is one measure change monitor configuration file. It is named MeasureChangeMonitor.properties and must be located under {domainPath}/config/.

Note: There is only one property file for a global domain. It is placed under the config directory of the master domain.

This configuration file is in standard Java properties file format. Each line defines the relationship between one subject and one measure:

```
Subject.MeasureName=true
```

where

- true means inclusion in monitoring.
- false is used to exclude a subject or measure from monitoring. Use of false is not recommended since, by default, a measure is not monitored and such a line can be

deleted or commented out from the file. Comments are any contents on a line from "#" to the end of the line.

Example:

```
TestMeasures.R_EX_DEMOA=true
TestMeasures.R_EX_DEMOB=true
LanguageMeasures.R_MsgLabel=true
```

The following are defined in the above example:

- Two subjects: `TestMeasures` and `LanguageMeasures`.
- The subject `TestMeasures` represents two measures: `R_EX_DEMOA` and `R_EX_DEMOB`.
- The subject `LanguageMeasures` represents one measure: `R_MsgLabel`.

A subject name must be a valid hierarchical variable name, that is, consisting of only alphanumeric characters, underscores, and periods. RPAS does not enforce any naming convention for a subject. It is up to the retailer to define its own naming convention.

After the configuration file is modified, RPAS processes can detect that the file has changed and automatically reload it. There is no need to restart any RPAS processes.

Configuring the RPAS JMS Publisher

JMS Publisher for measure change events is implemented in Java and runs as a standalone process. It is decoupled from any other RPAS server side processes in terms of interprocess communications and domain data file locking.

Each JMS Publisher process is tied to a domain and a JMS topic. When the publisher detects an event for its domain, it generates a JMS message and sends the message to a JMS provider. The format of the JMS message is defined by a template file that can contain any of the macros listed in the following table. The macros are replaced by actual values at run time.

Table 17–1 Configuration Macros

Macro Name	Format	Notes
__EventDateTime__	YYYY-MM-DDThh:mm:ss	Local time of the server.
__SourceURI__	RPAS/JMS/{hostname}	{hostname} is the name of the server where the publisher is running.
__SUBJECT__	String	Subject of the event as defined in MeasureChangeMonitor.properties.
__TYPE__	"MeasureChange"	Type is a constant string.
__DOMAIN__	String	Path to the domain.
__MEASURE__	String	RPAS internal measure name.
__ORIGINUSER__	String	RPAS User ID, only available for workbook commit or DPM. Use "-" if not available.

Here are two examples of a JMS message template.

Example 1: Simple name/value pairs:

```
type=__TYPE__
domain=__DOMAIN__
measure=__MEASURE__
time=__EventDateTime__
user=__ORIGINUSER__
```

Example 2: XML-based Notification Event Architecture for Retail (NEAR) format:

```
<?xml version="1.0" encoding="UTF-8" ?>
<AlertEvent MajorVersion="1" MinorVersion="0" TypeCode="RPASEvent"
            Priority="0" Severity="Information" Mode="Test" FixVersion="0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.retail.oracle.com/workspace/alerts/AlertEventV1.0.0
.xsd"
            xmlns="http://www.retail.oracle.com/workspace/alerts/">
      <SequenceNumber>0</SequenceNumber>
      <EventDateTime>__EventDateTime__</EventDateTime>
      <EventDescription>RPAS Measure Change Event</EventDescription>
      <SourceName>RPAS</SourceName>
      <SourceURI>__SourceURI__</SourceURI>
      <Instance>1</Instance>
      <RoutingInfo TypeCode="SubjectInfo">__SUBJECT__</RoutingInfo>
      <AlertData><![CDATA[<type>__TYPE__</type>
<domain>__DOMAIN__</domain>
<measure>__MEASURE__</measure>
<originUser>__ORIGINUSER__</originUser>]]></AlertData>
</AlertEvent>
```

Command Line

The following command line is used for JMS Publisher:

```
java -cp {classpath} oracle.rpas.pmc.MCPublisher -d {domainPath} [-c
{configFileName}] [name1=value1 [name2=value2 [...]]]
```

The following table provides descriptions of the arguments used in the command line.

Table 17–2 Arguments Used in the Command Line

Argument	Description
classpath	Should include rpaspmc.jar and jms.jar under $RPAS_HOME/lib and any vendor-specific JMS implementation jar files.
-d *domainPath*	Path to the monitored domain. It must be a simple or master domain.
-c *configFileName*	Configuration file in Java properties file format. This argument is optional.
name1=value1 name2=value2 …	If the configuration file is not specified, the name/value pairs that are provided are used. If the configuration file is specified, the name/value pairs are added to the configuration and overwrite any value with the same name already present in the file. This argument is optional.

Configuration Settings

The configuration settings for the JMS Publisher can be categorized into general and vendor-specific settings.

General Settings

The following table lists all vendor-neutral configuration settings for the JMS Publisher:

Table 17–3 Vendor-Neutral Configuration Settings for the JMS Publisher

Setting Name	Description	Required	Value
topic	JMS topic lookup name.	Yes	A string with only alphanumeric characters, underscores, and periods. Maximum length is 100 bytes.
topicConnectionFactory	JMS topic connection factory lookup name.	Yes	A string with only alphanumeric characters, underscores, and periods. Maximum length is 100 bytes.
messageTemplate	Template file name for the JMS message. This file must be placed under the config directory of the domain.	Yes	Standard template file PMCMessageTemplate.xml is available under $RPAS_ HOME/domain/config. This file must be copied to the config directory of the domain.
logLevel	Logging level. A log level is a cut-off level. This means logs with a lower level are filtered out.	No	Specify one of the following values, listed from low to high. VERBOSE: All logs. DEBUG: Debug logs. INFO: Informational logs. WARN: Warning logs. ERROR: Error logs. SUPPRESS: No logs. If not specified, the default is INFO.
logFile	The path to the log file. Can be a relative or absolute path.	No	If not specified, output to the console.
restartAfterException	Flag to restart after encountering any JMS-related exceptions.	No	If true, the publisher will try to restart after catching any JMS related exceptions. Interval between retries is 180 seconds. To stop the publisher from trying to restart, the process must be ended manually. Default is true.
message.deliveryMode	Delivery mode.	No	NON_PERSISTENT or PERSISTENT. Default is PERSISTENT.
message.priority	A priority number for the JMS message.	No	0 to 9. Default is 4.
message.timeToLive	Time to live in milliseconds.	No	0 or any positive integer. Default is 0 (unlimited).

Vendor-Specific Settings

Sun Open Message Queue is used as an example for these settings. For further details, consult the vendor documentation for your JMS implementation.

File-Based JNDI Object Store

File-based JNDI object store is used primarily for development and testing. It is very easy to set up, but has weak built-in security.

Table 17–4 File-Based JNDI Object Store

Property Name	Description	Required	Value
java.naming.factory.initial	Initial context for JNDI lookup.	Yes	For Open Message Queue, it must be: com.sun.jndi.fscontext.RefFSContextFactory
java.naming.provider.url	Directory path to the object store.	Yes	Example: file:///C:/myapp/mqobjs

LDAP-Based JNDI Object Store

An LDAP server is the recommended object store for production JMS messaging systems. LDAP servers are designed for use in distributed systems and provide security features that are required in production environments.

Table 17–5 LDAP-Based JNDI Object Store

Property Name	Description	Required	Value
java.naming.factory.initial	Initial context for JNDI lookup.	Yes	For Open Message Queue, it must be: com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url	Server URL and directory path to the object store.	Yes	Example: ldap://myD.com:389/ou=mq1,o=App where administered objects are stored in the directory /App/mq1
java.naming.security.principal	Identity of the principal for authenticating callers.	No	The format of this attribute depends on the authentication scheme. For example: uid=homerSimpson,ou=People,o=mq If this attribute is not specified, the behavior is determined by the LDAP service provider.
java.naming.security.credentials	Credentials of the authentication principal.	No	The value of this attribute depends on the authentication scheme. For example, it might be a hashed password, clear-text password, key, or certificate. If this property is not specified, the behavior is determined by the LDAP service provider.

Table 17–5 (Cont.) LDAP-Based JNDI Object Store

Property Name	Description	Required	Value
java.naming.security.authentication	Security level for authentication.	No	Specify one of these options: none: No security simple: Simple security strong: Strong security For example, if you specify simple, you are prompted for any missing principal or credential values. This enables a more secure way of providing identifying information. If this property is not specified, the behavior is determined by the LDAP service provider.

The following is a sample configuration file for RPAS JMS Publisher:

```
java.naming.factory.initial=com.sun.jndi.fscontext.RefFSContextFactory
java.naming.provider.url=file:///C:/Temp
topic=RPASEventTopic
topicConnectionFactory=RPASEventTopicConnectionFactory
messageTemplate=PMCMessageTemplate.xml
```

Configuring the RPAS JMS Subscriber

A sample JMS Subscriber is implemented to use for testing or trouble-shooting. This subscriber is packaged and deployed along with the publisher. It simply writes the messages it receives to logging output, which can be console or a log file.

Command Line

Here is the command line for the sample JMS Subscriber:

```
java -cp {classpath} oracle.rpas.pmc.MCSubscriber [-c {configFileName}]
[name1=value1 [name2=value2 [...]]]
```

where

- {classpath} should include rpaspmc.jar and jms.jar under $RPAS_HOME/lib and any vendor-specific JMS implementation jar files.
- {configFileName} is a configuration file in Java properties file format. It is optional.
- name/value pairs. If the configuration file is not specified, the name/value pairs are used. Otherwise, the name/value pairs are added to the configuration file and any value with the same name already present in the file is overwritten.

Configuration Settings

Vendor-specific settings are the same as the JMS publisher. Refer to "Configuring the RPAS JMS Publisher" for details.

The following table lists all the general settings for the JMS Subscriber:

Table 17–6 General Settings for the JMS Subscriber

Property Name	Description	Required	Value
topic	JMS topic lookup name.	Yes	A string with only alphanumeric characters, underscores, and periods. Maximum length is 100 bytes.
topicConnectionFactory	JMS topic connection factory lookup name.	Yes	A string with only alphanumeric characters, underscores, and periods. Maximum length is 100 bytes.
logLevel	Logging level. A log level is a cut-off level which means logs with a lower level are filtered out.	No	Specify one of the following values, in low to high order. ■ VERBOSE: All logs. ■ DEBUG: Debug logs. ■ INFO: Informational logs. ■ WARN: Warning logs. ■ ERROR: Error logs. ■ SUPPRESS: No logs. If not specified, default is INFO.
logFile	The path to the log file. Can be a relative or absolute path.	No	If not specified, output to the console.
clientID	JMS Client ID. Only required for a durable subscription. Not recommended for other use.	No	This must be a unique string for all JMS clients using the same connection factory.
durableSubscriptionName	A unique name for a durable subscription.	No	If a name is not provided, the subscription will be a transient subscription. This means messages are not queued up if the connection for the subscriber is lost or the subscriber is not running.

The following is a sample configuration file for RPAS JMS Subscriber:

```
java.naming.factory.initial=com.sun.jndi.fscontext.RefFSContextFactory
java.naming.provider.url=file:///C:/Temp
topic=RPASEventTopic
topicConnectionFactory=RPASEventTopicConnectionFactory
```

18

Guided Launch

Guided Launch is an RPAS feature that provides retailers with a planning-centered, integrated UI solution that can be used to respond to events by adjusting the affected plans within the context of the changed circumstances and interacting plans. A planner's workspace can be developed in which relevant events are announced in role-specific dashboards. The planner can drill down into the event report to identify the context of the event. Appropriate planning applications can be launched to react to the event by either replanning or deriving further insight from the plans, in order to justify the event.

If the RPAS Fusion Client is deployed in an SSO environment, then the launching pad application (that is, the planner's workspace) must also be deployed to the same SSO environment. If this is not done, the launch context may not always be passed through to the Fusion Client.

RPAS launches the Fusion Client given a context, that is, the template or workbook to launch and the wizard parameters that define the scope if a new workbook is to be built. The user can open the launched Fusion Client within an area inside the launching portal page or in a separate browser window. Launching the Fusion Client involves the following steps:

1. The Fusion Client application is launched by a user.
2. The context is created, asking to open a workbook or create one. In the case of the latter, the positions to include in the workbook are also specified. The context is passed to the Fusion Client when a user initiates the guided launch.
3. If a new workbook is being built, it is created
4. A new or existing workbook is opened, if that is what the context specified.

The following topics are covered in this chapter:

- Opening a pre-built workbook.
- Building a new workbook given a specification of the wizard parameters.
- Issuing the open or build requests from a web application running in a web portal.

Launching the RPAS Fusion Client

A system integrator is responsible for creating a web-based application that can launch RPAS. Creating an application involves the following steps:

1. Configure the navigational pane to appropriately launch RPAS.
2. Create static or dynamic web reports and dashboards, including OBIEE and other Oracle Fusion technologies, to present integrated data to the end-user with links or

buttons that create Guided Launch links in the launching application page that either target a frame within the launching page or a separate browser window.

3. Create alerts or alert reports, using BPM, OBIEE, or other Oracle Fusion technologies that have links or other UI mechanisms that can launch RPAS.

RPAS provides the framework that system integrators can use to launch any RPAS-based workbook, independent of whether the workbook uses a standard wizard or a custom wizard.

RPAS supports launching workbooks that are built using the dynamic-template API. This includes all templates built using RPAS Configuration Tools but does not include templates that have been coded in C++ using an API lower that the Dynamic Template.

Note: Successfully launching RPAS administrative templates that are not built using RPAS Configuration Tools is not guaranteed.

Issuing a Launch Request from a Web Page

RPAS Guided Launch is implemented as an extension of the RPAS Web Launch feature. As a prerequisite, the RPAS Web Launch Servlet and RPAS Client web installation package must be deployed to a web server and made available to system integrators. For more information, see the *Oracle Retail Predictive Application Server Installation Guide*.

The RPAS Fusion Client is expected to be deployed and running. The home page URL for the Fusion Client would be something like http://<ipaddr>:<port>/<fusionclientapplicationname>.

System integrators must design the report page to have a link or other web GUI widget that invokes client-side Java Script code to initiate an RPAS Guided Launch.

An HTTP POST request is required to be constructed and submitted in the following format:

- target host: URL of Fusion Client home page. example: http://example.com:8001/rpas
- POST body (name/value pair):
- name: XMLString
- value: An XML string that represents the context

Here is an example of a guided launch page that displays the Fusion Client pages in an i-frame:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<html>
<head>
    <title>In context launch test page</title>
    <script type="text/javascript">
    function prepareForLaunch() {
    var iframe = document.getElementsByTagName("iframe")[0];
    iframe.src="about:blank";
    return true;
    }
    </script>
</head>
```

```
<body>
    <form id="launch" method="POST" action="http://a.example.com:8001/rpas"
target="ifname" onsubmit="prepareForLaunch()">
            <input type="hidden" name="XMLString" value="<?xml version='1.0'
encoding='UTF-8' ?>
<RPASLaunchContext>
    <Workbook>
    <open>
    <solutionId>pcgd</solutionId>
    <name>t1</name>
    </open>
    </Workbook>
</RPASLaunchContext>" >
      <p>
        <input type="submit" name="submitbutton" value="RPAS InContextLaunch
Workbook by name" />
      </p>
    </form>
    <iframe id="ifname" name="ifname" height="500" width="100%" scrolling="yes"/>
</body>
</html>
```

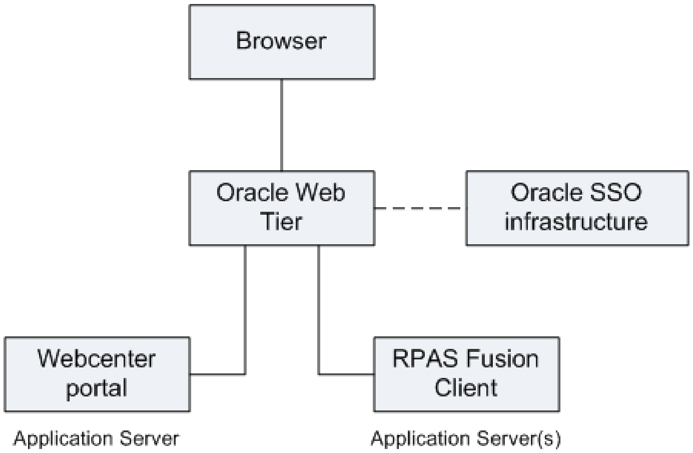
User Authentication

This section describes the authentication of users.

OSSO-Based Authentication

If the HTML active content is protected by OSSO, it can be assumed that the user has already been authenticated.

Figure 18–1 OSSO-Based Authentication


Browser
Oracle Web Tier
Oracle SSO infrastructure
Webcenter portal
RPAS Fusion Client
Application Server
Application Server(s)

Deploying Guided Launch

Consider the following details in deploying an RPAS Guided Launch enabled application. The RPAS Web Launch Servlet has to be deployed on a web server. This web server does not have to be the same as the web server that is used for any reporting, dashboard, or BAM applications. In this release, RPAS Guided Launch is able to interact with the dynamic content of the web page using mechanisms described in the previous sections. As long as system integrators can ensure that the RPAS Web Launch Servlet is available, Guided Launch can be achieved from any web-based application.

In an all-Oracle solution, the portal would be a Webcenter application. The portal page would have links to launch the RPAS Fusion Client either in a separate window or within a pane on the portal page. The two applications (portal and RPAS Fusion Client) can run on different application servers. But they must both use the same OSSO infrastructure for authentication and authorization.

When using Guided Launch, make sure that the launching portal application and the Fusion Client are both deployed in the same SSO environment. In order to access a portal page, you must be logged into the SSO environment. Because of this, you are not prompted to log in when you attempt to access a portal page via a guided launch of the Fusion Client.

If you log out of SSO and then attempt a guided launch using a portal page link, you will be prompted to log into SSO again. However, the context will be lost and you will be taken to the Fusion Client home page. If this happens, click the link again. You will not be prompted to log in and the context will be properly executed.

Use Cases

This section lists some RPAS use cases.

Launching Without Context

A system integrator may want to launch RPAS without any context. This means starting up a RPAS Fusion Client window and logging on automatically, but without opening or building a workbook. This type of launch is required to support a generic link in the navigation pane where an end-user can simply launch RPAS to do planning using RPAS Today. The navigation pane only serves as a launchpad for RPAS where RPAS can be started and logged into by clicking a static link or button.

To code this type of launch, the user must either remove the <Workbook> tag from the context or ensure that the <Workbook> tag does not include an <open> or <build> request.

Launching With Context

An RPAS launch that includes more than a simple login into RPAS is considered a guided launch. This includes launching a RPAS Fusion Client instance to open an existing workbook or build a workbook given the template and wizard parameters.

Opening an Existing Workbook

A system integrator may want to design an application that opens a pre-built workbook. This workbook may have been built by a system integrator, the user, workbook build batch, or previous guided launch request.

It is assumed that system integrators are aware of either the internal, unique workbook name that RPAS uses or the workbook label, and optionally, the template name and creation date and time of the workbook. These parameters can be specified in the XML context specification to have RPAS open the workbook. For information, see "Open Workbook".

To enable opening the latest workbook from a nightly auto-workbook-build batch, system integrators can provide the label and template name. Guided Launch automatically defaults to the latest workbook of the same label.

Note: The last workbook that was created is opened, not the last workbook that was modified.

Building a New Workbook

A system integrator may want to ensure that a new workbook is built for the end-user that depends on what the end-user is or has been doing in the web application. The following examples illustrate this concept.

- If a link for Merchandise Financial Planning is clicked in the navigation pane, RPAS is launched to build a workbook for the Merchandise Financial Planning template. The end-user is required to make all the wizard selections.
- If the user is navigating through a report and has identified some outliers at the SKU/Store level, RPAS is launched to automatically build a workbook for those SKU/Stores without requiring the end-user to reselect them in the wizard.

This can be achieved using the build context detailed in "Building a Workbook".

RPAS Guided Launch Format

The guided launch is described in XML format. It consists of one component, the workbook. The workbook component contains information for opening or building a workbook.

Workbook Settings

The Workbook section has two options, opening a workbook or building a workbook. If the options or the whole Workbook section is not specified, the RPAS client logs in to the domain without any workbook operation. For information on opening a workbook, see "Open Workbook". For information on building a workbook, see "Building a Workbook".

XML Schema

The following is the full schema for the RPAS Guided Launch.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="RPASLaunchContext">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Workbook"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Workbook">
    <xs:complexType>
      <xs:sequence>
        <xs:choice>
          <xs:element ref="solutionId"/>
          <xs:element ref="taskName"/>
        </xs:choice>
        <xs:choice minOccurs="0">
          <xs:element ref="selectDomain"/>
          <xs:element ref="domainName"/>
        </xs:choice>
        <xs:choice>
          <xs:element ref="build"/>
          <xs:element ref="open"/>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="open">
  <xs:complexType>
    <xs:sequence>
      <xs:choice>
        <xs:element name="name" type="xs:string"/>
        <xs:sequence>
          <xs:element name="label" type="xs:string"/>
          <xs:element name="templateName" type="xs:string" minOccurs="0"/>
          <xs:element name="createdTime" type="xs:string" minOccurs="0"/>
        </xs:sequence>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="selectDomain">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="selection"/>
    </xs:sequence>
    <xs:attribute name="level" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="solutionId" type="xs:string"/>
<xs:element name="taskName" type="xs:string"/>
<xs:element name="domainName" type="xs:string"/>
<xs:element name="build">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="templateName"/>
      <xs:element name="allowManualOverride" type="xs:boolean" minOccurs="0"/>
      <xs:element name="customErrorMessage" type="xs:string" minOccurs="0"/>
      <xs:element name="customWaitMessage" type="xs:string" minOccurs="0"/>
      <xs:element ref="WizardPages"/>
      <xs:element ref="save" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="templateName" type="xs:string"/>
<xs:element name="save">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="label" type="xs:string"/>
      <xs:element name="access" type="xs:string" minOccurs="0"/>
      <xs:element name="group" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="WizardPage">
  <xs:complexType>
    <xs:sequence>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="Checkbox"/>
        <xs:element ref="DatePicker"/>
        <xs:element ref="DropDownList"/>
        <xs:element ref="Edit"/>
        <xs:element ref="Listbox"/>
        <xs:element ref="RadioButton"/>
        <xs:element ref="SingleHierSelect"/>
        <xs:element ref="SuperTree"/>
        <xs:element ref="Tree"/>
      </xs:choice>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
      <xs:attribute name="option" type="xs:string" use="optional"/>
      <xs:attribute name="fallback" type="xs:string" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="WizardPages">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="WizardPage" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="BooleanDataType">
    <xs:sequence>
      <xs:element name="value" type="xs:boolean"/>
      <xs:element name="default" type="xs:boolean" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="fallback" type="xs:string" use="optional"/>
  </xs:complexType>
  <xs:complexType name="StringDataType">
    <xs:sequence>
      <xs:element name="value" type="xs:string"/>
      <xs:element name="default" type="xs:string" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="fallback" type="xs:string" use="optional"/>
  </xs:complexType>
  <xs:complexType name="SelectionType">
    <xs:sequence>
      <xs:element name="selection" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StringSetDataType">
    <xs:sequence>
      <xs:element name="value" type="SelectionType"/>
      <xs:element name="default" type="SelectionType" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="fallback" type="xs:string" use="optional"/>
  </xs:complexType>
  <xs:complexType name="StringVectorDataType">
    <xs:sequence>
      <xs:element name="value" type="SelectionType"/>
      <xs:element name="default" type="SelectionType" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="fallback" type="xs:string" use="optional"/>
  </xs:complexType>
  <xs:complexType name="DimDataType">
    <xs:sequence>
      <xs:element name="dim" maxOccurs="unbounded">
        <xs:complexType>
          <xs:complexContent>
            <xs:extension base="SelectionType">
              <xs:attribute name="name" type="xs:string" use="required"/>
            </xs:extension>
          </xs:complexContent>
        </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="SelectDomainDataType">
  <xs:sequence>
    <xs:element name="selectDomain" maxOccurs="unbounded">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="SelectionType">
            <xs:attribute name="level" type="xs:string" use="required"/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="PQDDataType">
  <xs:sequence>
    <xs:element name="pqd">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="type" type="xs:string" use="required"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:element name="Checkbox" type="BooleanDataType"/>
<xs:element name="RadioButton" type="BooleanDataType"/>
<xs:element name="DatePicker" type="StringDataType"/>
<xs:element name="DropDownList" type="StringDataType"/>
<xs:element name="Edit" type="StringDataType"/>
<xs:element name="Listbox" type="StringSetDataType"/>
<xs:element name="SingleHierSelect" type="StringVectorDataType"/>
<xs:element name="Tree" type="StringSetDataType"/>
<xs:element name="SuperTree">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="value"/>
      <xs:element ref="default"/>
    </xs:sequence>
    <xs:attribute name="fallback" type="xs:string"/>
    <xs:attribute name="name" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="value">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="pqd"/>
      <xs:element maxOccurs="unbounded" ref="dim"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:element name="default">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="pqd"/>
      <xs:element maxOccurs="unbounded" ref="dim"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="dim">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="selection"/>
      </xs:sequence>
      <xs:attribute name="name" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="selection" type="xs:string"/>
  <xs:element name="pqd">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="type" use="required" type="xs:string"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Open Workbook

In any retail-planning scenario, a particular workbook or an auto-build workbook may be used to plan for a specific grouping of products and locations. When reacting to an event or alert related to these products and locations, it may be necessary to always open a particular workbook or auto-build workbook. System integrators configuring an application can configure and code a guided launch opening of the specific workbook.

System integrators can programmatically specify the parameters that RPAS needs to open a pre-built workbook. This does not involve the use of an API, but involves the XML specification that Guided Launch can use to identify and open the desired workbook.

System integrators can identify the target workbook in either of two ways:

- Using the RPAS name of the workbook.
- Using the auto-build parameters, that is, user, template name, label, and creation date. Template name and creation date are optional parameters. If template name or creation date are not specified and multiple workbooks with the same label are found, RPAS chooses the workbook that was created last and then further breaks the tie based on the alphabetical ordering of the template name.

Specifications

Option 1

Table 18–1 Option 1- Workbook Specification

Setting	Description	Type	Required
name	The RPAS name of the workbook (also known as the "gem name", such as "t1" and "t18" which is the root directory name of the workbook)	String	Yes

Option 2

Table 18–2 Option 2- Workbook Specification

Setting	Description	Type	Required
label	The workbook label	String	Yes
templateName	The template name of the workbook	String	No
createdTime	The created time of the workbook in the format YYYY-MM-DD hh:mm:ss	String	No

Examples

Here is sample guided launch description for opening a workbook using the RPAS name. Note that if the solution is a global domain environment, you should specify the domain using either <domainName> of <selectDomain>. If you do not specify a domain, the default is tried instead.

```
<?xml version="1.0" encoding="UTF-8" ?>
<RPASLaunchContext>
<solutionId>MFP</solutionId>
<Workbook>
   <domainName>
     <open>
        <name>t1</name>
     </open>
   </domainName>
</Workbook>
</RPASLaunchContext>
```

Here is sample guided launch description for opening a workbook by the label and template name:

```
<RPASLaunchContext>
… … (Same as above, omitted here.)
<Workbook>
    <open>
        <label>wb2</label>
        <templateName>MEASUREANALYSIS</templateName>
    </open>
</Workbook>
</RPASLaunchContext>
```

Building a Workbook

RPAS users generally build workbooks through a manual process using RPAS Wizard Pages from workbook templates. This process is automated in Guided Launch by specifying the content and selection of the wizard pages in the guided launch description.

The automated process enables system integrators to use the XML schema specify the parameters that RPAS needs to build a new workbook. The integrator can program RPAS guided launch to build virtually any workbook from any workbook templates, simple or complex, by constructing an XML string according to the guided launch schema and the configuration of the wizard pages.

Users can think of the XML specifications as macro definitions of what they do when they build the same workbook manually. The specifications for building a workbook start from a workbook template, which contains a list of wizard pages.

Right after the Workbook tag, the user specifies either solutionId or taskName, this is mandatory for the Fusion Client. After that, the user specifies domain information, which is needed for building a workbook in a global domain environment. The domain information can be either the name itself (domainName element) or a selectDomain element, as shown in the following example:

```
<selectDomain level='pgrp'>
</selectDomain>
<selection>1000</selection>
```

Each wizard page has a set of pre-defined widgets:

- RadioButton
- Checkbox
- Edit
- DropDownList
- SingleHierSelect
- DatePicker
- Listbox
- Tree
- SuperTree

Each widget has its own specifications, depending on its type.

The fallback option can be specified at the widget and wizard page level. The widget level fallback option is considered first when the specifications for that widget are incomplete or incorrect. If a specification cannot be resolved at the widget level, the wizard page fallback option is used. If the fallback option on the wizard page is not specified or the option has errors, an error message is displayed to the user.

If any unrecoverable errors occur on a wizard page, the workbook build process falls back to manual mode if the allowManualOverride option is set to true. The wizard starts from the first page. All selections prior to the problematic page are preserved, and the user can click **Next** to skip them and go to the page with errors, which is blank, along with all subsequent pages.

Specifications

The specifications to build a workbook consist of basic settings and a list of wizard pages. Each wizard page contains a list of widget specifications.

Table 18–3 Basic Settings

Setting	Description	Type	Required
templateName	The template name for the workbook to be built.	String	Yes
allowManualOverride	Allow manual override in case of fatal errors in the XML data. The default is true. The wizard starts from the first page.	Boolean	No
customErrorMessage	Custom error message to display in case of fatal errors in the launch context data.	String	No
customWaitMessage	Custom wait message to display while the workbook is being built.	String	No

Table 18–4 Wizard Page Specifications

Setting	Description	Type	Required
option (attribute)	Wizard page handling option. Can be one of the following values: manual: forces manual selection for all widgets. This option forces manual input for this and all subsequent pages. However, the wizard still starts from the first page. skippable: this page can be skipped. Normally used for a conditional wizard page. This page only exists if some particular value is selected for some widget on the previous page.	String	No
fallback (attribute)	Wizard page level fallback option. Can be one of the following values: default: use values from the default settings for all widgets. cache: use selections from the cache selection file. Cache files are from last successful workbook build of the same template.	String	No

Table 18–5 Wizard Widget Specifications: RadioButton and Checkbox

Setting	Description	Type	Required
name (attribute)	Widget name.	String	Yes
fallback (attribute)	Widget level fallback option. Can be: default: use value from the default setting.	String	No
value	Boolean type: selected value for the widget. Value must be either false or true.	XML section	Yes
default	Default value for the widget. Must be a Boolean type.	XML section	No

Table 18–6 Wizard Widget Specifications: Edit, DropDownList, SingleHierSelect, and DatePicker

Setting	Description	Type	Required
name (attribute)	Widget name.	String	Yes
fallback (attribute)	Widget level fallback option. Can be: default: use value from the default setting.	String	No
value	String type: selected value for the widget.	XML section	Yes
default	Default value for the widget. Must be a String type.	XML section	No

Table 18–7 Wizard Widget Specifications: Listbox and Tree

Setting	Description	Type	Required
name (attribute)	Widget name.	String	Yes

Table 18–7 (Cont.) Wizard Widget Specifications: Listbox and Tree

Setting	Description	Type	Required
fallback (attribute)	Widget level fall back option. Can be one of the following values: ■ default: use value from the default setting. ■ all: select all available items or positions. ■ intersection: use intersection of the specified items/positions and all available items/positions if any of the specified items/positions do not exist.	String	No
value	String Set type: selected value for the widget. Value is a set of selections.	XML section	Yes
default	Default value for the widget. Must be a String type.	XML section	No

Table 18–8 Wizard Widget Specifications:SuperTree

Setting	Description	Type	Required
name (attribute)	Widget name.	String	Yes
fallback (attribute)	Widget level fall back option. Can be one of the following values: ■ default: use value from the default setting. ■ all: select all available items or positions. ■ use intersection of specified positions and all available positions if any of the specified positions do not exist.	String	No
value	String to String Set Map Type/PQD File Type: selected value for the widget. Can be one of the following values: ■ Value is one or more dimensions, each having a set of selections. ■ Value is a Position Query Definition (PQD) file name in the format: <pqd type="{pqd_type}"> pqd_file_name </pqd> where {pqd_type} must be either PointAndClickPQD or RollingCalendarPQD.	XML section	Yes
default	Default value for the widget. Must be a String to String Set Map type or PQD file type.	XML section	No

Workbook Save Specifications

These save specifications are optional. If specified, the workbook is immediately saved after it is built.

Table 18–9 Workbook Save Specifications

Setting	Description	Type	Required
label	Workbook label.	String	Yes
access	Workbook access. Must be user, group or world. If not specified, the default is user.	String	No

Table 18–9 (Cont.) Workbook Save Specifications

Setting	Description	Type	Required
group	Workbook group. Required when workbook access is group. There is no default.	String	Conditional

Example

The following example shows a sample launch context for building a workbook. It specifies a new Measure Analysis workbook for a simple domain (TestDomain). The domain is created by using the RPAS utility createRpasDomain with the command line option "-x".

```
<?xml version="1.0" encoding="UTF-8" ?>
<RPASLaunchContext>
<solutionId>MFP</solutionId>
<domainName>global</domainName>
<Workbook>
  <build>
    <templateName>MEASUREANALYSIS</templateName>
    <WizardPages>
      <WizardPage id="ExtraMeasuresPage" fallback="none">
        <Listbox name="lb1">
          <value>
            <selection>R_EX_DEMOA</selection>
            <selection>R_EX_DEMOB</selection>
            <selection>R_EX_DEMOC</selection>
          </value>
        </Listbox>
      </WizardPage>
      <WizardPage id="DAY" fallback="none">
        <SuperTree name="tree1" fallback="all">
          <value>
            <dim name="WEEK">
              <selection>W01_1997</selection>
              <selection>W01_1998</selection>
              <selection>W01_1999</selection>
              <selection>W01_2000</selection>
            </dim>
            <dim name="DAY">
              <selection>19970108</selection>
            </dim>
          </value>
          <default>
            <dim name="DAY">
              <selection>19970108</selection>
            </dim>
          </default>
        </SuperTree>
      </WizardPage>
      <WizardPage id="STR" fallback="none">
        <SuperTree name="tree1">
          <value>
            <dim name="STR">
              <selection>0102</selection>
              <selection>0144</selection>
              <selection>0152</selection>
              <selection>0557</selection>
              <selection>0594</selection>
              <selection>0959</selection>
          </dim>
        </value>
      </SuperTree>
    </WizardPage>
    <WizardPage id="SKU" fallback="none">
      <SuperTree name="tree1">
        <value>
          <dim name="CLSS">
            <selection>021</selection>
          </dim>
        </value>
      </SuperTree>
    </WizardPage>
  </WizardPages>
  <save>
    <label>Testwb</label>
    <access>Group</access>
    <group>ADMIN</group>
  </save>
 </build>
</Workbook>
</RPASLaunchContext>
```

buildWorkbookContext Utility

The buildWorkbookContext utility is a server side command line utility. It runs under both global domains and local domains.

Before running this utility, you should first manually create a workbook using the workbook template you intend to use in the context. This step generates selection files. In order to capture as much information as possible in the selection files, you can choose to select all for the widgets on wizard pages. This is especially useful for listbox widgets.

After the XML file is generated, you can use the data in the file directly if you want to build an identical workbook. Or, you can use it as a starting point and tailor it to meet some specific requirements.

Command Line Usage

```
buildWorkbookContext -d [domain_path] -user [user_name] -wbt [wbt_name] -out
[output_file]
```

- [domain_path]: The path to the global or local domain.
- [user_name]: The username whose selection files are used. This is the user who can manually run the workbook template wizard.
- [wbt_name]: The name of the workbook template for which the XML context is built. This is the internal name for the template. It can be identified by looking at the selection files under DOMAIN_PATH/users/[user]/selections. The names of the selection files are in this format: [templateName]_[wizpageID].sel.
- [output_file]: Output file name for the XML context.

A

Appendix: Curve Administration Guide

This appendix provides the details of the utilities used in Curve administration. It contains the following sections:

- curvevalidate
- curvebatch

curvevalidate

The curvevalidate utility automatically executes during the domain install, and it can also be run at any time against a master domain 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 that subdomain (not all other subdomains). This process verifies that:

- Profile and Source intersections and source data are properly defined
- Profile intersections respect the partition dimension

curvevalidate Usage

```
curvevalidate -d domainpath [-s]
```

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

Table A–1 Arguments Used by the curvevalidate Utility

Argument	Description
-d *domainpath*	Indicates the path to the domain.
-s	Sets defaults.
-debug	Causes temporary measures to be retained for debugging purposes.
-h	Displays information about curvebatch in the terminal screen.
-version	Displays version information.

Table A–1 (Cont.) Arguments Used by the curvevalidate Utility

Argument	Description
-loglevel	Sets the logger verbosity level. Available verbosity levels are as follows: ■ all ■ profile ■ information ■ warning ■ error ■ none
-noheader	Standard argument to disable the use of the 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:

    - The Profile Type must not be diff(8).

    - The Aggregation Intersection must be above the Profile Intersection.

    - The Aggregation Intersection must be above the Stored Intersection.

    h. If Aggregation Intersection does not conform to Profile Intersection:

    - The Profile Type must be diff (8).

    - There must be at least one common hierarchy between the Aggregation Intersection and X in {Profile Intersection, Stored Intersection}.

    - 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:

- The Profile Type must not be diff(8).

- The Aggregation Intersection must be above the Profile Intersection.

- The Aggregation Intersection must be above the Stored Intersection.

f. If Aggregation Intersection does not conform to Profile Intersection:

- The Parent Profile Type must be diff (8).

- There must be at least one common hierarchy between the Aggregation Intersection and X in {Profile Intersection and Stored Intersection}.

- 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

This section provides the details of the curvebatch utility.

curvebatch Usage

```
curvebatch -d domainpath [-level # ] [-debug] | -h | -version
```

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

Table A–2 Arguments Used by the curvebatch Utility

Argument	Description
-d *domainpath*	Indicates the path to the domain.
-level #	The # signifies the profile ID. When using level, a valid profile ID must be provided.
-debug	Causes temporary measures to be retained for debugging purposes.
-h	Display information about curvebatch in the terminal screen.
-version	Displays version information.
-loglevel	Sets the logger verbosity level. Available verbosity levels are as follows: ■ all ■ profile ■ information ■ warning ■ error ■ none

B

Appendix: RPAS Test Automation

This appendix describes an RPAS utility called rpac (RPAS Pluggable Automation Component). It can be used to perform various tests on an existing domain, based on a set of test cases contained in XML files.

This appendix contains the following sections:

- Introduction
- Usage
- Writing Test Cases
- Example
- Output
- Schema

Introduction

The rpac utility enables the creation of highly customizable automated test suites for the RPAS server. The XML-based framework allows rapid development of tests and ensures that test cases can be quickly copied across locations. The flexible XML schema allows the user to test a large number of RPAS features such as writing data to a domain, building a workbook, running custom menu options, and verifying the results.

The XML framework also introduces consistency, accuracy, and reproducibility in the test results. To reproduce a test case, all that is needed is the XML file and a copy of the domain.

Note that rpac operates directly against the domain itself and therefore does not test any Client/Server interactions.

Usage

To run an rpac test, run the following from the command line:

```
rpac -d <path to domain> -title <title> -input <path to xml input>  [commands]
[options]
```

The output of this command is written to unittest.xml in the local directory. The results of this file contain only the failing tests. To display both passing and failing tests, use the -coverage option. This generates a coverage.xml in the local directory that contains both passing and failing results.

Required Parameters

Table B–1 Required Parameters

Option	Parameter	Description
-d	Path to domain	Specifies the path to the domain that the test runs against.
-title	Title	Specifies the name that this test run is given in the output files. This is used when combining the results of multiple test runs.

Commands

Table B–2 Commands

Command	Description
<none>	If you do not specify a command, then rpac will run the test script provided.
-listTests	When this is specified, rpac writes the tests to be executed to the console. The tests are not run.
-traceTests	When this is specified, rpac writes the commands to be executed to the console. The tests are not run.
-printTagDoc	Prints the rpac schema to the console. This option does not require -input.
-writeTagDoc	Writes the rpac schema to a file. This option does not require -input.

Options

Table B–3 Options

Option	Parameter	Description
-input	path to XML script	Specifies the path to the XML script containing the test suite.
-suite	pattern	A regular expression pattern to specify which test suites should be run. Any test suite whose name does not match the pattern is skipped. Example: If you have the following test suites (test1, test2, test3, test4, test5) and you just want to run suites 1, 2, and 4, specify: -suite test[124]
-coverage	<none>	When specified, rpac writes all passing and failing results to coverage.xml.

Writing Test Cases

The structure of an rpac test script is defined by the following tags. For more information, see "Schema" on page B-7.

Table B–4 Test Case Tags

Tag	Description
<testscript>	The outermost tag for the test script. This tag contains the one or more <testsuite> tags.
<testsuite>	This tag defines a suite of tests that are designed to run together. Each suite contains its own <setup>, <teardown>, <testcase>, and <workbook-operations> tags. This tag has a name attribute that can be used with the -suite command line attribute to specify which suites are executed in this run.
<setup>	This tag sets the preconditions that are common to all test cases. Operations inside this tag are run before every test case is executed. Example operations: ■ Set RpasToday so the test date is consistent for every test run. ■ Pre-set measure values so tests operate in a known state. ■ Check any preconditions to make sure this test run is valid.
<teardown>	This tag contains cleanup code that is run after all test cases. This is used to restore a domain or workbook to its default state to remove dependencies between independent test cases. Example operations: ■ Set domain measure values back to their initial state. ■ Run batch scripts to remove any test-specific files that were generated.
<workbook-operations>	This tag allows for the test script to build, refresh, commit, and close workbooks. Workbook operations may exist outside of test cases, but you must encapsulate them wherever possible inside a testsuite tag for logging purposes.
<testcase>	This tag is where the code to perform each test goes. It should perform operations, then use the assert tags to verify the results. The name of this tag is used in the output to log timing results and success/failure. **Note:** Code in the <setup> tag is executed prior to each test case, and code in the <teardown> tag is executed after each test case.

It is highly recommended that the teardown section restore the domain to its previous state. This removes dependencies between test cases and ensures consistency from one run to the next.

Example

Below is a sample XML file that contains some rpac tests that run against an MFP domain. This test illustrates how the setup and teardown methods work, as well as how to do the various workbook operations.

The first test case builds the workbook. The first step in this operation is to shell out to the wbmgr utility to clear out all existing workbooks. This step prevents the domain from growing after repeated operations, but you should only use this step in cases where you know that existing workbooks are not needed.

After that, the test verifies the preconditions of the test case. In this example, the test verifies the pre-conditions performed in the setup code by checking that the domain measure value is "true". After that, the test makes the wizard selections and builds the workbook. Once the workbook is built, an assertion verifies that the load rules correctly set the workbook value.

Once this test case is complete, the teardown code is executed, along with the setup code for the next test case. The test verifies that the teardown code correctly set the workbook's measure value to "false" and then issues a commit. Once the workbook is committed, the domain value is checked to see if it was properly updated. Once again, the teardown code is executed and the setup code is run at the beginning of the next test case.

For the third test case, the setup code should have initialized the domain value back to "true", and the workbook value back to "false". The test case verifies these operations, then performs a workbook refresh and verifies that the workbook value was correctly set to "true".

After that, the workbook is closed and the test suite is complete.

Note: Because the teardown code modifies workbook data, the workbook close step cannot be encapsulated inside of a test case. If it was, the teardown code would try to modify a workbook that was no longer there and produce a failing result.

```
<?xml version="1.0" encoding="UTF-8"?>
<!---
  Copyright (c) 2004, 2012, Oracle and/or its affiliates. All rights reserved.
  File: wbtest.xml
  TestCase Description:
  - Setup:
  Set the RPAS Today value
  Set the domain's measure value to "true"
  - Teardown:
  Set the workbook's measure value to "false"
  - Build: Build a workbook and validate the load rule
  - Commit: Validate the commit rule
  - Refresh: Validate the refresh rule
  - Close the workbook
-->
<testscript>
  <testsuite name="MFP RPAC Test 11">
    <!--Setup executed at the beginning of every testcase-->
    <setup>
      <rpas-today>20070101</rpas-today>
      <edit-domain-measure-value name="ipopappenbb" namedkeys="h1_2007:100:1"
value="true"/>
    </setup>

    <testcase name="Enable OP Approval Workbook: Build">
      <!--Clear all existing workbooks-->
      <shell>wbmgr -d [DOMAIN] -remove -all</shell>

      <!--Verify pre-conditions are true-->
      <assert-domain-measure-value-eq name="ipopappenbb" namedkeys="h1_2007:100:1"
value="true"/>

      <workbook-operations>
        <build template-name="EnableOp" label="Enable OP Approval" user="adm">
          <!--Make positions selection for the wizard page-->
          <wizard-page-settings id="wiz_EnableOpssn">
            <set-tree-selections control-name="tree1" dim-name="ssn"
rollup-name="ssn">
             h1_2007
            </set-tree-selections>
          </wizard-page-settings>
          <wizard-page-settings id="wiz_EnableOpdept">
            <set-tree-selections control-name="tree1" dim-name="dept"
rollup-name="dept">
```

```
              100
            </set-tree-selections>
          </wizard-page-settings>
          <wizard-page-settings id="wiz_EnableOpchnl">
            <set-tree-selections control-name="tree1" dim-name="chnl"
rollup-name="chnl" >
              1
            </set-tree-selections>
          </wizard-page-settings>
        </build>
      </workbook-operations>

      <!--Validate the load rule worked correctly->
      <assert-workbook-measure-value-eq name="ipopappenbb" keys="0:0:0"
value="true"/>
    </testcase>

    <!--Commit the workbook and assert the measure values in domain-->
    <testcase name="Enable OP Approval Workbook: Commit">
      <assert-domain-measure-value-eq name="ipopappenbb" namedkeys="h1_2007:100:1"
value="true"/>
      <assert-workbook-measure-value-eq name="ipopappenbb" namedkeys="h1_
2007:100:1 value="false"/>
      <workbook-operations>
        <commit type ="now" template-name="EnableOp" label="Enable OP Approval"
user="adm"/>
      </workbook-operations>

      <!--Validate the commit rule worked correctly->
      <assert-domain-measure-value-eq name="ipopappenbb" namedkeys="h1_2007:100:1"
value="false"/>
    </testcase>

    <!--Refresh the workbook and assert measure values in workbook and domain-->
    <testcase name="Enable OP Approval Workbook: Refresh">
      <assert-domain-measure-value-eq name="ipopappenbb" namedkeys="h1_2007:100:1"
value="true"/>
      <assert-workbook-measure-value-eq name="ipopappenbb" namedkeys="h1_
2007:100:1" value="false"/>
      <workbook-operations>
        <refresh template-name="EnableOp" label="Enable OP Approval" user="adm"/>
      </workbook-operations>

      <!--Validate the refresh rule worked correctly->
      <assert-workbook-measure-value-eq name="ipopappenbb" namedkeys="h1_
2007:100:1" value="true"/>
    </testcase>

    <!--Close the workbook-->
    <workbook-operations>
      <close template-name="EnableOp" label="Enable OP Approval" user="adm"/>
    </workbook-operations>

    <!--Teardown executed at the end of every testcase-->
    <teardown>
      <edit-workbook-measure name="ipopappenbb" keys="0:0:0" value="false"/>
    </teardown>
  </testsuite>
</testscript>
```

Output

When rpac executes, it prints the status of current execution to the console. On completion of all the test cases, the summary of test cases is stored in unitest.xml under the current directory. These results contain the failing test case, as well as the error message that was captured. If -coverage was specified, then a coverage.xml file is generated containing the passing and failing results.

Example:

```
[rpasHome] rpac -d testDomain -input wbTest.xml -title "'MFP OpApproval RPAC Test"

Running rpac tests for RPAC : MFP Tests...

Now executing wbmgr in domain.
Deleting all workbooks in the domain...
Destroying workbook 't0000000000' in domain.
wbmgr completed successfully
5.292541
0.308167
0.244300

Ran 3   tests: 0   errors, 0   failures in 5.85 seconds
```

If something had gone wrong, the output would look like this:

Example:

```
 [rpasHome] rpac -d testDomain -input wbTest.xml -title "'MFP OpApproval RPAC
Test"

Now executing wbmgr in domain
Deleting all workbooks in the domain...
Destroying workbook 't0000000000' in domain
wbmgr completed successfully

Failure
0.192497
Failure

Ran 3   tests: 0   errors, 2   failures in 6.42 seconds

Contents of unittest.xml
After execution, rpac creates a unittest.xml file containing the test results.
Here is an example from the successful execution above:

<unittest name='MFP OpApproval RPAC Test' runcount='3' timestamp='2012-10-17
10:06:49' elapsedTime='13.72 seconds'>
</unittest>
```

Here is the unittest.xml output from the failing run:

```
<unittest name='MFP OpApproval RPAC Test' runcount='3' timestamp='2012-10-17
10:07:38' elapsedTime='6.42 seconds'>
  <failurelist>
    <failure>
      <testname> </testname>
      <message>Got Error: Expected measure &apos;ipopappenbb&apos; at ArrayKey(0,
0, 0, 0) is false;
                        false == true</message>
   <file>OpApproval.xml</file>
      <line>52</line>
    </failure>
    <failure>
      <testname> </testname>
      <message>Got Error: Expected measure &apos;ipopappenbb&apos; at ArrayKey(0,
0, 0, 0) is false;
                         false == true</message>
      <file>OpApproval.xml</file>
      <line>60</line>
    </failure>
  </failurelist>
</unittest>
```

Schema

1. Top Level Parent Tag

1.1 testscript

Top level tag, which contains all the rpac tags.

1.2 testsuite

Contains setup, teardown, and testcase tag elements.

1.3 testcase

Contains a list of tag elements to be executed.

1.4 setup

The setup tag is used to set any preconditions that must be true prior to each test case. These can include initializing measure values, setting the date, or running batch scripts to prepare the domain.

1.5 teardown

The teardown tag contains tag elements used to restore the environment to a default state after a test case is executed. This can include returning measures to their default values or running batch scripts to clean up the domain.

The sample implementation of the above tags are given below:

Sample Usage:

```
<testscript>
<testsuite name="MFP RPAC Test">
<setup>
<!--- setup code ->
 </setup>
 <testcase name="Measure Analysis Workbook : build">
<!---testcase code ->
</testcase >
<teardown>
<!--- teardown code ->
</teardown>
</testsuite>
</testscript>
```

2. Generic Tags

2.1 Attribute

Specifies the attributes of a Tag.

2.2 Description

Describes the functionality of the Tags or Attributes.

2.3 Fail

Creates a failure message.

Attribute	Description	Category
Msg	Message to be used in the failure text	Optional

2.4 key

Set the key specs value of the named key.

Attribute	Description	Category
Name	Key name	Required
keyspec	Key value spec	Required

2.5 rpas-today

Sets the RpasToday environmental variable to the value of this tag.

Sample usage:

```
<rpas-today>20070101</rpas-today>
```

Where: the date is given in yyyymmdd format.

2.6 shell

Start a shell process with the command provided as value of this tag.

Attribute	Description	Category
working-directory	Directory to use for shell; if not provided then the current working directory is used	Optional
check-status	Boolean to determine if a non-zero return status throws an exception	Optional
is-shell	Boolean to indicate that the process is a shell	Optional

Sample usage:

```
<shell check-status="true" is-shell="false">wbmgr -d [DOMAIN] -remove -all</shell>
```

2.7 register-measure

Register a measure with the provided properties.

Attribute	Description	Category
Name	Measure name	Required
Intx	Base intersection	Required
Type	Type of measure	Required
Naval	NA value of the measure	Required
Defagg	Default aggregation	Required

Sample usage:

```
<register-measure name="MeasureX" intx="WEEK_STR_DAY_" type="int"  defagg="total"
naval="0"/>
```

2.8 unregister-measure

Unregister a measure registered by rpac.

Attribute	Description	Category
Name	Measure name	Required

Sample usage:

```
<unregister-measure name="MeasureX" />
```

3. Workbook Operation Tags

3.1 Workbook Wizard Operations

3.1.1 build Build workbook based on provided tag data.

Attribute	Description	Category
template-name	Template name	Required
label	Label for workbook	Required
user	User used to access workbook	Required

Sample usage:

```
<build template-name="MeasureAnalysis" label="MEAS AN" user="adm">
```

3.1.2 wizard-page-settings Contains the settings for a wizard page.

Attribute	Description	Category
id	ID use to match up settings with wizard pages For configured templates, this follows the pattern: wiz_<wbt_name><base_dim> Example: For the EnableOp workbook, the wizard page that sets CHNL is: Fwiz_EnableOpchnl Note: For custom templates, get the name from the administrator.	Required

Sample usage:

```
<wizard-page-settings id="wiz_EnableOpchnl">
```

3.1.3 set-hier-selection Set the selection of a SingleHier Select control of a custom wizard page.

Attribute	Description	Category
control-name	Name of the control for a wizard page. This can be obtained from the template.cfg file	Required
dim-name	Name of the dimension containing the selection	Required
selection	Name of the position selected	Required

Sample usage:

```
<set-hier-selection control-name="loc_select" dim-name="chnl"  selection="1" >
```

3.1.4 set-selected Set the value for a Boolean control on a custom wizard page.

Attribute	Description	Category
control-name	Name of the control for a wizard page. This can be obtained from the configuration	Required
Selected	Set this to "TRUE" or "FALSE"	Required

Sample usage:

```
<set-selected control-name="UseExtraMeasures" value="TRUE">
```

3.1.5 set-selection Set the selection of a drop-down list control on a custom wizard page.

Attribute	Description	Category
control-name	Name of the control for a wizard page. This can be obtained from the configuration	Required
Selection	Text for the selection operation	Required

Sample usage:

```
<set-selection control-name="ForecastLevel" value="2">
```

3.1.6 set-selections Set the selections of a list box or tree control on a custom wizard page. This tag should contain a space-delimited list of the selections.

Attribute	Description	Category
control-name	Name of the control for a wizard page. This can be obtained from the configuration	Required

Sample usage:

```
<set-selections control-name="ExtraMeasures">
meas1 meas2 meas3 </set selections>
```

3.1.7 set-text Set the contents of an edit control on a custom wizard page.

Attribute	Description	Category
control-name	Name of the control for a wizard page	Required
test	Text for the operation	Required

Sample usage:

```
<set-text control-name="MyText" value="abc">
```

3.1.8 set-tree-selections Set the selections for a two-tree control on either a standard or custom wizard page.

Attribute	Description	Category
control-name	Name of the control. For two-tree controls, this should always be set to "tree1"	Required
rollup-name	The highest level of the wizard page	Required
dim-name	Name of the dimension on which the selections are made	Required

Sample usage:

```
<set-tree-selections control-name="tree1" rollup-name="chnl" dim-name="str">
str1,str2,str3
</set-tree-selections>
```

3.2 Workbook Operations

3.2.1 Workbook-operations Contains the workbook operations to be executed.

3.2.2 Calc Does workbook calculations.

Attribute	Description	Category
template-name	Template name	Required
label	Label for workbook	Required
user	User used to access workbook	Required

Sample usage:

```
<calc template-name="RptAdmin" label="Reporting Administration" user="adm"/
```

3.2.3 custom-menu Checks for the existence of, or executes, a custom menu.

Attribute	Description	Category
template-name	Template name	Required
label	Label for workbook	Required
user	User used to access workbook	Required
menu-label	Menu label for this custom menu. This can be obtained from the configuration	Required
menu-exist	Set this to "true" to see if the menu item exists, or leave it blank to execute the custom menu	Optional

Sample usage:

```
<custom-menu template-name="BottomUp" label="Botup" user="adm"
menu-label="menuItem83"  menu-exist="true"/>
```

3.2.4 refresh Does workbook refresh.

Attribute	Description	Category
template-name	Template name	Required
label	Label for workbook	Required
user	User used to access workbook	Required
rule-group	Set this to use a rule group other than the default refresh group	Optional

Sample usage:

```
<refresh template-name="BottomUp" label="Botup" user="adm" rule-group="refresh_
alt"/>
```

3.2.5 Close Does workbook close. Arguments may be provided for debug purposes.

Attribute	Description	Category
template-name	Template name	Optional
label	Label for workbook	Optional
user	User used to access workbook	Optional

Sample usage:

```
<close template-name="BottomUp" label="Botup" user="adm"/>
```

3.2.6 Does workbook commit Does workbook commit.

Attribute	Description	Category
type	Commit Type (now | later)	Required
template-name	Template name	Required
label	Label for workbook	Required
user	User used to access workbook	Required

Sample usage:

```
<commit type="now"  template-name="BottomUp" label="Bottom Up MFP Refresh"
user="adm" />
```

3.3 Workbook Assertions

3.3.1 assert-window-contain-measure Assert that the measures provided are visible (or not visible) on the window.

Attribute	Description	Category
name	Name of the window in the workbook configuration. User-generated windows cannot be used	Required
measure	Comma separated list of measures names	Optional
containflag	If true, will check that measures provided are visible. If false, will check that the measures provided are not visible.	Optional

Sample usage:

```
<assert-window-contain-measure  name=" chnldeptssn__W"  measures =" Ipopappenbb"/>
```

3.3.2 assert-window-exists Assert if a window exists in the current workbook.

Attribute	Description	Category
name	Window name of current workbook	Required

Sample usage:

```
<assert-window-exists name="chnldeptssn__W" />
```

3.3.3 assert-window-intersection Assert window intersection matches provided intersection.

Attribute	Description	Category
name	Window name of current workbook	Required

Attribute	Description	Category
baseint	Intersection string to compare. Leave blank if checking for scalar.	Optional

Sample usage:

```
<assert-window-intersection name="chnldeptssn__W" baseint"chnldeptssn_" />
```

3.3.4 assert-window-not-exists Assert if a window does not exist in the current workbook.

Attribute	Description	Category
name	Window name of current workbook	Required

Sample usage:

```
<assert-window-not-exists name="chnldeptssn__W" />
```

3.3.5 assert-window-measure-exists Assert a measure exists.

Attribute	Description	Category
name	Measure name	Required

Sample usage:

```
<assert-measure-exists name=" Ipopappenbb" />
```

3.3.6 assert-workbook-measure-intx Assert a workbook measure base intersection matches an expected value.

Attribute	Description	Category
name	Measure name	Required
intx	Intersection string to compare with	Required

Sample usage:

```
<assert-measure-intersection name=" Ipopappenbb" baseint"chnldeptssn_" />
```

3.3.7 assert-workbook-measure-value-eq Asserts that a workbook measure at the specified positions is equal to the provided value.

Parent tag(s): testcase

Child tag(s): None

Attribute	Description	Category
keys	The keys to be used to compare measure values. Can be either indices or position names . Each dimension spec in the keys-spec is delimited by a ":". Multiple position ranges for a dimension can be specified by using "," as separators. When using indices, a range of positions can also be used. Ex - "0-2:4-7,11-13:4" implies indices 0,1,2 for the first dimension, 4,5,6,7,11,12,13 for the second, and 4 for the third dimension of the measure. **Note:** Because indices are based on the workbook selections, they may not be consistent across all test cases. Care must be exercised to ensure that you are using the correct indices for your specific test case.	Required
namedkeys	Position names used to get keys of the measure array. Using indices instead of position names or positions that do not exist in the MeasureStore (Workbook) will be flagged as an error. Similar to "keys" above, but using position names only.	Optional
path	Path of the flat file used for comparison against the Measure array. This looks like pointing to a pre-set environment variable. "path" cannot be used along with "keys" or "namedkeys". "path" is used only to compare the measure data against a pre-existing flat file. If none of "path","keys" and "namedkeys" are specified, the comparison occurs for ALL the keys in the measure array of the Measure Store (Workbook).	Optional
start	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
width	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
value	The expected value of the measure. Comparison on equality.	Optional

Sample usage: (indices)

```
<assert-workbook-measure-value-eq name="drwppthrsclsp" keys="0-2"  value="0.0" />
```

Sample usage (names):

```
<assert-workbook-measure-value-eq name="drwppthrsclsp" namedkeys="pos1,pos2,pos3"
value="0.0" />
```

3.3.8 assert-workbook-measure-value-ge Assert a workbook measure value at the specified positions is greater than or equal to provided value.

Attribute	Description	Category
keys	The keys to be used to compare measure values. Can be either indices or position names. Each dimension spec in the keys-spec is delimited by a ":". Multiple position ranges for a dimension can be specified by using "," as separators. When using indices, a range of positions can also be used. Ex - "0-2:4-7,11-13:4" implies indices 0,1,2 for the first dimension, 4,5,6,7,11,12,13 for the second, and 4 for the third dimension of the measure. **Note:** Because indices are based on the workbook selections, they may not be consistent across all test cases. Care must be exercised to ensure that you are using the correct indices for your specific test case.	Required
namedkeys	Position names used to get keys of the measure array. Using indices instead of position names or positions that do not exist in the MeasureStore (Workbook) will be flagged as an error. Similar to "keys" above, but using position names only.	Optional
path	Path of the flat file used for comparison against the Measure array. This looks like pointing to a pre-set environment variable. "path" cannot be used along with "keys" or "namedkeys". "path" is used only to compare the measure data against a pre-existing flat file. If none of "path","keys" and "namedkeys" are specified, the comparison occurs for ALL the keys in the measure array of the Measure Store (Workbook).	Optional
start	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
width	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
value	The expected value of the measure. Comparison on equality.	Optional

Sample usage (indices):

```
<assert-workbook-measure-value-ge name="drwppthrsclsp" keys="0-2"  value="0.0" />
```

Sample usage (names):

```
<assert-workbook-measure-value-ge name="drwppthrsclsp" namedkeys="pos1,pos2,pos3"
value="0.0" />
```

3.3.9 assert-workbook-measure-value-gt Assert a workbook measure value is greater than a provided value.

Attribute	Description	Category
keys	The keys to be used to compare measure values. Can be either indices, or position names. Each dimension spec in the keys-spec is de-limited by a ":". Multiple position ranges for a dimension can be specified by using "," as separators. When using indices, a range of positions can also be used. Ex - "0-2:4-7,11-13:4" implies indices 0,1,2 for the first dimension, 4,5,6,7,11,12,13 for the second and 4 for the third dimension of the measure. **Note:** Because indices are based on the workbook selections, they may not be consistent across all test cases. Care must be exercised to ensure that you are using the correct indices for your specific test case.	Required
namedkeys	Position names used to get keys of the measure array. Using indices instead of position names or positions that do not exist in the MeasureStore (Workbook) will be flagged as an error. Similar to "keys" above, but using position names only.	Optional
path	Path of the flat file used for comparison against the Measure array. This looks like pointing to a pre-set environment variable. "path" cannot be used along with "keys" or "namedkeys". "path" is used only to compare the measure data against a pre-existing Flat file. If none of "path","keys" and "namedkeys" are specified, the comparison occurs for ALL the keys in the measure array of the Measure Store (Workbook).	Optional
start	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
width	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
value	The expected value of the measure. Comparison on equality.	Optional

Sample usage (indices):

```
<assert-workbook-measure-value-gt name="drwppthrsclsp" keys="0-2"  value="0.0" />
```

Sample usage (names):

```
<assert-workbook-measure-value-gt name="drwppthrsclsp" namedkeys="pos1,pos2,pos3"
value="0.0" />
```

3.3.10 assert-workbook-measure-value-le Assert a workbook measure value is greater than or equal to a provided value.

Attribute	Description	Category
keys	The keys to be used to compare measure values. Can be either indices, or position names. Each dimension spec in the keys-spec is delimited by a ":". Multiple position ranges for a dimension can be specified by using "," as separators. When using indices, a range of positions can also be used. Ex - "0-2:4-7,11-13:4" implies indices 0,1,2 for the first dimension, 4,5,6,7,11,12,13 for the second, and 4 for the third dimension of the measure. **Note:** Because indices are based on the workbook selections, they may not be consistent across all test cases. Care must be exercised to ensure that you are using the correct indices for your specific test case.	Required
namedkeys	Position names used to get keys of the measure array. Using indices instead of position names, or positions that do not exist in the MeasureStore (Workbook) will be flagged as an error. Similar to "keys" above, but using position names only.	Optional
path	Path of the flat file used for comparison against the Measure array. This looks like pointing to a pre-set environment variable. "path" cannot be used along with "keys" or "namedkeys". "path" is used only when we want to compare the measure data against a pre-existing flat file. If none of "path","keys" and "namedkeys" are specified, the comparison occurs for ALL the keys in the measure array of the Measure Store (Workbook).	Optional
start	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional

Attribute	Description	Category
width	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
value	The expected value of the measure. Comparison on equality.	Optional

Sample usage (indices):

```
<assert-workbook-measure-value-le name="drwppthrsclsp" keys="0-2"  value="0.0" />
```

Sample usage (names):

```
<assert-workbook-measure-value-le name="drwppthrsclsp" namedkeys="pos1,pos2,pos3"
value="0.0" />
```

3.3.11 assert-workbook-measure-value-lt Assert a workbook measure value is less than a provided value.

Attribute	Description	Category
keys	The keys to be used to compare measure values. Can be either indices, or position names. Each dimension spec in the keys-spec is de-limited by a ":". Multiple position ranges for a dimension can be specified by using "," as separators. When using indices, a range of positions can also be used. Ex - "0-2:4-7,11-13:4" implies indices 0,1,2 for the first dimension, 4,5,6,7,11,12,13 for the second, and 4 for the third dimension of the measure. **Note:** Because indices are based on the workbook selections, they may not be consistent across all test cases. Care must be exercised to ensure that you are using the correct indices for your specific test case.	Required
namedkeys	Position names used to get keys of the measure array. Using indices instead of position names, or positions that do not exist in the MeasureStore (Workbook) will be flagged as an error. Similar to "keys" above, but using position names only.	Optional

Attribute	Description	Category
path	Path of the flat file used for comparison against the Measure array. This looks like pointing to a pre-set environment variable. "path" cannot be used along with "keys" or "namedkeys". "path" is used only when we want to compare the measure data against a pre-existing flat file. If none of "path","keys" and "namedkeys" are specified, the comparison occurs for ALL the keys in the measure array of the Measure Store (Workbook).	Optional
start	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
width	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
value	The expected value of the measure. Comparison on equality.	Optional

Sample usage (indices):

```
<assert-workbook-measure-value-lt name="drwppthrsclsp" keys="0-2"  value="0.0" />
```

Sample usage (names):

```
<assert-workbook-measure-value-lt name="drwppthrsclsp" namedkeys="pos1,pos2,pos3"
value="0.0" />
```

3.3.12 assert-workbook-measure-value-ne Assert a workbook measure value is not equal to a provided value.

Attribute	Description	Category
keys	The keys to be used to compare measure values. Can be either indices, or position names. Each dimension spec in the keys-spec is de-limited by a ":". Multiple position ranges for a dimension can be specified by using "," as separators. When using indices, a range of positions can also be used. Ex - "0-2:4-7,11-13:4" implies indices 0,1,2 for the first dimension, 4,5,6,7,11,12,13 for the second, and 4 for the third dimension of the measure. **Note:** Because indices are based on the workbook selections, they may not be consistent across all test cases. Care must be exercised to ensure that you are using the correct indices for your specific test case.	Required

Attribute	Description	Category
namedkeys	Position names used to get keys of the measure array. Using indices instead of position names, or positions that do not exist in the MeasureStore (Workbook) will be flagged as an error. Similar to "keys" above, but using position names only.	Optional
path	Path of the flat file used for comparison against the Measure array. This looks like pointing to a pre-set environment variable. "path" cannot be used along with "keys" or "namedkeys". "path" is used only to compare the measure data against a pre-existing flat file. If none of "path","keys" and "namedkeys" are specified, the comparison occurs for ALL the keys in the measure array of the Measure Store (Workbook).	Optional
start	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
width	The field start value. If this does not exist in the XML, the value is taken from the Measure's Properties. Used in flat-file comparison.	Optional
value	The expected value of the measure. Comparison on equality.	Optional

Sample usage (indices):

```
<assert-workbook-measure-value-ne name="drwppthrsclsp" keys="0-2"  value="0.0" />
```

Sample usage (names):

```
<assert-workbook-measure-value-ne name="drwppthrsclsp" namedkeys="pos1,pos2,pos3"
value="0.0" />
```

3.3.13 assert-workbook-popcount-eq Assert a workbook measure popcount matches an expected value.

Attribute	Description	Category
name	Measure name	Required
pop-count	Expected popcount	Required

Sample usage:

```
<assert-workbook-popcount-eq name=" Ipopappenbb" pop-count="100" />
```

3.3.14 assert-workbook-popcount-eq-zero Assert a workbook measure is not populated.

Attribute	Description	Category
name	Measure name	Required

Sample usage:

```
<assert-workbook-popcount-eq-zero name=" Ipopappenbb"  />
```

3.3.14 assert-measure-popcount-ne-zero Assert a workbook measure is populated.

Attribute	Description	Category
name	Measure name	Required

Sample usage:

```
<assert-workbook-popcount-ne-zero name=" Ipopappenbb"  />
```

3.4 Workbook Data Edit Operations

3.4.1 clear-workbook-list Clear a list of workbook measures of all data.

Attribute	Description	Category
path	Path to a file containing a new-line delimited list of measures	Required

Sample usage:

```
<clear-workbook-list path="my_list.txt"/>
```

3.4.2 clear-workbook-measure Clear a workbook measure of all data.

Attribute	Description	Category
name	Measure name	Required

Sample usage:

```
<clear-workbook-measure name=" Ipopappenbb/>
```

3.4.3 edit-workbook-measure Edits the value of a workbook measure based on provided attributes. This tag closely models the action of editing a cell from the Client, as opposed to set-workbook-measure, which does a direct update of the workbook arrays.

This tag can be used in a variety of ways:

Set by loadmeasure file:

By specifying <path>, <start>, and <width>, you can set the measure values according to a supplied loadmeasure file. This file must be in fixed-width format.

Set by positions:

You can set individual positions by providing a <keys> tag. If you want, you can provide the list of positions at an aggregate level by specifying the intersection using the <keyint> tag.

Set everything:

If you do not supply a <path> or <keys> tag, the entire measure will be set to the supplied value.

Attribute	Description	Category
name	Measure name	Required
keys	List of comma separated position names	Optional
namedkeys	List of comma separated position names	Optional
path	Path to load measure type file to compare with	Optional
keyint	Intersection string	Optional
value	Expected value	Optional
start	Starting column of measure field	Optional
width	Width of measure field	Optional

Note: For Date type measures, RPAC supports setting the array cell using either of the following two Date & Time formats - "%m/%d/%Y" or "%m/%d/%Y-%H:%M:%S" where m = 2 digit month, d = 2 digit date, Y = 4 digit year, H = 2 digit hour, M = 2 digit min, and S = 2 digit sec.

Sample usage (using keys):

```
<edit-workbook-measure name="meas1" keys="0-2:5"  value="0.0" />
```

Sample usage (using keyint):

```
<assert-workbook-measure-value-ne name="meas2" namedkeys="chnl1:week1"
value="02/24/2012" />
```

Sample usage (load-measure file):

```
<edit-workbook-measure name="meas1" path="loadMeas1.dat" start="100" width="20"/>
```

Sample usage (set all):

```
<edit-workbook-measure name="meas1" value="2"
```

3.4.4 set-workbook-measure Sets the value of a workbook measure based on provided attributes. Unlike edit-workbook-measure, this performs a direct change of the workbook data and does not update the changearrays unless told to do so. This can have an impact on the calculation cycle since bypassing the change arrays results in a full evaluation rather than incremental.

This tag can be used in a variety of ways:

Set by loadmeasure file:

By specifying <path>, <start>, and <width>, you can set the measure values according to a supplied loadmeasure file. This file must be in fixed-width format.

Set by positions:

You can set individual positions by providing a <keys> tag. If you want, you can provide the list of positions at an aggregate level by specifying the intersection using the <keyint> tag.

Set everything:

If you do not supply a <path> or <keys> tag, the entire measure will be set to the supplied value.

Attribute	Description	Category
name	Measure name	Required
keys	List of comma separated position names	Optional
namedkeys	List of comma separated position names	Optional
path	Path to load measure type file to compare with	Optional
keyint	Intersection string	Optional
value	Expected value	Optional
start	Starting column of measure field	Optional
width	Width of measure field	Optional
incremental	True if evaluation is to be incremental	Optional

Note: For Date type measures, RPAC supports setting the array cell using either of the following two Date & Time formats - "%m/%d/%Y" or "%m/%d/%Y-%H:%M:%S" where m = 2 digit month, d = 2 digit date, Y = 4 digit year, H = 2 digit hour, M = 2 digit min, and S = 2 digit sec.

Sample usage (using keys):

```
<set-workbook-measure name="meas1" keys="0-2:5"  value="0.0" incremental="true"/>
```

Sample usage (using keyint):

```
<set-workbook-measure-value-ne name="meas2" namedkeys="chnl1:week1"
value="02/24/2012" />
```

Sample usage (load-measure file):

```
<set-workbook-measure name="meas1" path="loadMeas1.dat" start="100" width="20"/>
```

Sample usage (set all):

```
<set-workbook-measure name="meas1" value="2"
```

4. Domain Operation Tags

4.1 Assertions

4.1.1 assert-dimension-contain Assert a dimension contains or does not contain a position name.

Attribute	Description	Category
name	Dimension name	Required

Attribute	Description	Category
posname	List of comma separated position names	Required
containflag	If flag is true then position names are checked to see if they are contained in the dimension	Required

Sample usage:

```
<assert-dimension-contain name="DEPT" posname="dept1" containflag="true"  />
```

4.1.2 assert-dimension-exists Assert a dimension name exists.

Attribute	Description	Category
name	Dimension name	Required

Sample usage:

```
<assert-dimension-exists name="DEPT" />
```

4.1.3 assert-dimension-size Assert a dimension size is between a minimum and a maximum value.

Attribute	Description	Category
name	Dimension name	Required
min	Minimum value	Required
max	Maximum value	Required

Sample usage:

```
<assert-dimension-size name="DEPT" min="8" max="32"  />
```

4.1.4 assert-domain-measure-exists Assert a domain measure exists.

Attribute	Description	Category
name	Dimension name	Required

Sample usage:

```
<assert-domain-measure-exists name="r_ex_test"  />
```

4.1.5 assert-domain-measure-intx Assert a domain measure base intersection matches an expected value.

Attribute	Description	Category
name	Dimension name	Required
intx	Intersection string to compare with	Required

Sample usage:

```
<assert-domain-measure-intx name="r_ex_test" intx="str_sku_week"
```

4.1.6 assert-domain-measure-value-eq Assert a domain measure value is equal to an expected value. This tag can be run in a number of ways:

Check against a loadmeasure file:

By specifying <path>, <start>, and <width>, you can validate that the contents of a measure are equal to the contents of a fixed-with loadmeasure file.

Check against specified positions:

By providing <keys> or <namedkeys>, you can verify that all of the positions in your specification are equal to the provided value. You can specify <keyint> to provide positions at an alternate rollup.

Check entire measure:

If you do not supply a <keys>, <namedkeys>, or <path> tag, then rpac will assert that every logical cell in the measure is equal to the value provided.

Attribute	Description	Category
name	Measure name	Required
keys	The list of comma separated position names	Optional
namedkeys	The list of comma separated position names	Optional
path	Path to load measure type file to compare to	Optional
keyint	Intersection string	Optional
value	Expected value	Optional
start	Starting column of measure field	Optional
width	Width of measure field	Optional

Sample usage (using keys):

```
<assert-domain-measure-value-eq name="bulylagtx" keyint="year" namedkeys=" 2007"
value="Test1"/>
```

Sample usage (using path):

```
<assert-domain-measure-value-eq name="bulylagtx" path="loadmeas.dat" start="100"
width="20" />
```

Sample usage (check all):

```
<assert-domain-measure-value-eq name="bulylagtx" value="Test1" />
```

4.1.7 assert-domain-measure-value-ge Assert a domain measure value is greater than or equal to an expected value. This tag can be run in a number of ways:

Check against a loadmeasure file:

By specifying <path>, <start>, and <width>, you can validate that the contents of a measure are equal to the contents of a fixed-with loadmeasure file.

Check against specified positions:

By providing <keys> or <namedkeys>, you can verify that all of the positions in your specification are equal to the provided value. You can specify <keyint> to provide positions at an alternate rollup.

Check entire measure:

If you do not supply a <keys>, <namedkeys>, or <path> tag, then rpac will assert that every logical cell in the measure is equal to the value provided.

Attribute	Description	Category
name	Measure name	Required
keys	The list of comma separated position names	Optional
namedkeys	The list of comma separated position names	Optional
path	Path to load measure type file to compare to	Optional
keyint	Intersection string	Optional
value	Expected value	Optional
start	Starting column of measure field	Optional
width	Width of measure field	Optional

Sample usage (using keys):

```
<assert-domain-measure-value-ge name="bulylagtx" keyint="year" namedkeys=" 2007"
value="Test1"/>
```

Sample usage (using path):

```
<assert-domain-measure-value-ge name="bulylagtx" path="loadmeas.dat" start="100"
width="20" />
```

Sample usage (check all):

```
<assert-domain-measure-value-ge name="bulylagtx" value="Test1" />
```

4.1.8 assert-domain-measure-value-gt Assert a domain measure value is greater than a an expected value. This tag can be run in a number of ways:

Check against a loadmeasure file:

By specifying <path>, <start>, and <width>, you can validate that the contents of a measure are equal to the contents of a fixed-with loadmeasure file.

Check against specified positions:

By providing <keys> or <namedkeys>, you can verify that all of the positions in your specification are equal to the provided value. You can specify <keyint> to provide positions at an alternate rollup.

Check entire measure:

If you do not supply a <keys>, <namedkeys>, or <path> tag, then rpac will assert that every logical cell in the measure is equal to the value provided.

Attribute	Description	Category
name	Measure name	Required
keys	The list of comma separated position names	Optional

Attribute	Description	Category
namedkeys	The list of comma separated position names	Optional
path	Path to load measure type file to compare to	Optional
keyint	Intersection string	Optional
value	Expected value	Optional
start	Starting column of measure field	Optional
width	Width of measure field	Optional

Sample usage (using keys):

```
<assert-domain-measure-value-gt name="bulylagtx" keyint="year" namedkeys=" 2007"
value="Test1"/>
```

Sample usage (using path):

```
<assert-domain-measure-value-gt name="bulylagtx" path="loadmeas.dat" start="100"
width="20" />
```

Sample usage (check all):

```
<assert-domain-measure-value-gt name="bulylagtx" value="Test1" />
```

4.1.9 assert-domain-measure-value-le Assert a domain measure value is less than or equal to an expected value. This tag can be run in a number of ways:

Check against a loadmeasure file:

By specifying <path>, <start>, and <width>, you can validate that the contents of a measure are equal to the contents of a fixed-with loadmeasure file.

Check against specified positions:

By providing <keys> or <namedkeys>, you can verify that all of the positions in your specification are equal to the provided value. You can specify <keyint> to provide positions at an alternate rollup.

Check entire measure:

If you do not supply a <keys>, <namedkeys>, or <path> tag, then rpac will assert that every logical cell in the measure is equal to the value provided.

Attribute	Description	Category
name	Measure name	Required
keys	The list of comma separated position names	Optional
namedkeys	The list of comma separated position names	Optional
path	Path to load measure type file to compare to	Optional
keyint	Intersection string	Optional
value	Expected value	Optional
start	Starting column of measure field	Optional

Attribute	Description	Category
width	Width of measure field	Optional

Sample usage (using keys):

```
<assert-domain-measure-value-le name="bulylagtx" keyint="year" namedkeys=" 2007"
value="Test1"/>
```

Sample usage (using path):

```
<assert-domain-measure-value-le name="bulylagtx" path="loadmeas.dat" start="100"
width="20" />
```

Sample usage (check all):

```
<assert-domain-measure-value-le name="bulylagtx" value="Test1" />
```

4.1.10 assert domain-measure-value-lt Assert a domain measure value is less than an expected value. This tag can be run in a number of ways:

Check against a loadmeasure file:

By specifying <path>, <start>, and <width>, you can validate that the contents of a measure are equal to the contents of a fixed-with loadmeasure file.

Check against specified positions:

By providing <keys> or <namedkeys>, you can verify that all of the positions in your specification are equal to the provided value. You can specify <keyint> to provide positions at an alternate rollup.

Check entire measure:

If you do not supply a <keys>, <namedkeys>, or <path> tag, then rpac will assert that every logical cell in the measure is equal to the value provided.

Attribute	Description	Category
name	Measure name	Required
keys	The list of comma separated position names	Optional
namedkeys	The list of comma separated position names	Optional
path	Path to load measure type file to compare to	Optional
keyint	Intersection string	Optional
value	Expected value	Optional
start	Starting column of measure field	Optional
width	Width of measure field	Optional

Sample usage (using keys):

```
<assert-domain-measure-value-lt name="bulylagtx" keyint="year" namedkeys=" 2007"
value="Test1"/>
```

Sample usage (using path):

```
<assert-domain-measure-value-lt name="bulylagtx" path="loadmeas.dat" start="100"
```

```
width="20" />
```

Sample usage (check all):

```
<assert-domain-measure-value-lt name="bulylagtx" value="Test1" />
```

4.1.11 assert-domain-measure-value-ne Assert a domain measure value is not equal to an expected value. This tag can be run in a number of ways:

Check against a loadmeasure file:

By specifying <path>, <start>, and <width>, you can validate that the contents of a measure are equal to the contents of a fixed-with loadmeasure file.

Check against specified positions:

By providing <keys> or <namedkeys>, you can verify that all of the positions in your specification are equal to the provided value. You can specify <keyint> to provide positions at an alternate rollup.

Check entire measure:

If you do not supply a <keys>, <namedkeys>, or <path> tag, then rpac will assert that every logical cell in the measure is equal to the value provided.

Attribute	Description	Category
name	Measure name	Required
keys	The list of comma separated position names	Optional
namedkeys	The list of comma separated position names	Optional
path	Path to load measure type file to compare to	Optional
keyint	Intersection string	Optional
value	Expected value	Optional
start	Starting column of measure field	Optional
width	Width of measure field	Optional

Sample usage (using keys):

```
<assert-domain-measure-value-ne name="bulylagtx" keyint="year" namedkeys=" 2007"
value="Test1"/>
```

Sample usage (using path):

```
<assert-domain-measure-value-ne name="bulylagtx" path="loadmeas.dat" start="100"
width="20" />
```

Sample usage (check all):

```
<assert-domain-measure-value-ne name="bulylagtx" value="Test1" />
```

4.1.12 assert-domain-popcount-eq Assert a domain measure popcount matches an expected value.

Attribute	Description	Category
name	Measure name	Required
pop-count	Expected popcount	Required

Sample usage:

```
<assert-domain-popcount-eq name=" Ipopappenbb" pop-count="100" />
```

4.1.13 assert-domain-popcount-eq-zero Assert a domain measure is not populated.

Attribute	Description	Category
name	Measure name	Required

Sample usage:

```
<assert-domain-popcount-eq-zero name=" Ipopappenbb" />
```

4.1.14 assert-domain-popcount-ne-zero Assert a domain measure is populated.

Attribute	Description	Category
name	Measure name	Required

Sample usage:

```
<assert-domian-popcount-ne-zero name=" Ipopappenbb" />
```

4.1.15 assert-bitsize-value-eq Assert the dimInfo bitsize or dimregistry bitsize value of a dimension.

Attribute	Description	Category
name	List of comma separated dimension names	Required
category	Bit Size Category (dimInfo or dimRegistry)	Required
value	Comma separated Bit Size Values	Required

Sample usage:

```
<assert-bitsize-value-eq name="week,skup," category="dimInfo" value="12,14" />
```

4.1.16 assert-dim-registry-version Assert the value for dimension registry version.

Attribute	Description	Category
value	Registry version	Required

Sample usage:

```
<assert-dim-registry-version value="1"   />
```

4.1.17 assert-reindex-threshold Assert the Threshold value of the dimension.

Attribute	Description	Category
name	List of comma separated dimension names	Required
value	List of Comma separated threshold values	Required

Sample usage:

```
<assert-reindex-threshold name="scls,week" value="25,103" />
```

4.1.18 assert-reindex-required Assert whether Diminfo Bitsize is equal to DimRegistry Bitsize.

Attribute	Description	Category
bitsize-flag	Returns true if Bitsize mismatch occurs	Required
name	Dimension Name	Required

Sample usage:

```
<assert-reindex-required name="week" bitsize-flag="false" />
```

4.1.19 assert-reindex-in-progress Assert the status of reindex with the user input.

Attribute	Description	Category
flag	Flag to indicate whether Reindexing Domain is in progress	Required

Sample usage:

```
<assert-reindex-in-progress flag="false"  />
```

4.2 Domain Data Edit Operations

4.2.1 clear-all-measures Clears all measures used in the following tags: <edit-workbook-measure>, <set-workbook-measure>, <set-domain-measure>.

This sets all the contents of these measures to the NA value.

4.2.2 clear-domain-measure Clear a domain measure of all data. This sets all the contents of this measure to its NA value.

Attribute	Description	Category
name	Measure name	Required

Sample usage:

```
<clear-domain-measure name="r_ex_pick_real" />
```

4.2.3 clear-domain-measure-list Clear a list of domain measures of all data.

Attribute	Description	Category
path	Path to a file containing a new-line delimited list of measures	Required

Sample usage:

```
<clear-domain-measure-list path="measlist.txt" />
```

4.2.4 execute-expression Execute an expression.

Attribute	Description	Category
expression	The expression to execute/calculate. This is a full expression string,	Optional
incremental	True if this is to be an incremental evaluation	Optional

Sample usage:

```
<execute-expression expression="a = b+c" incremental="false" />
```

4.2.5 set-domain-measure Set the value of a domain measure based on provided attributes.

Attribute	Description	Category
name	Measure name	Required
keys	The list of comma separated position names	Optional
namedkeys	The list of comma separated position names	Optional
path	Path to load measure type file to compare to	Optional
keyint	Intersection string	Optional
value	Expected value	Optional
start	Starting column of measure field	Optional
width	Width of measure field	Optional
incremental	True if evaluation is to be incremental	Optional

Sample usage:

```
<set-domain-measure name="r_ex_pick_real" keyint="dept" keys="1" value="150"
type="real" />
```

Note: For Date type measures, RPAC supports setting the array cell using either of the following two Date & Time formats - "%m/%d/%Y" or "%m/%d/%Y-%H:%M:%S" where m = 2 digit month, d = 2 digit date, Y = 4 digit year, H = 2 digit hour, M = 2 digit min, and S = 2 digit sec.

Sample usage (using keys):

```
<set-domain-measure name="meas1" keys="0-2:5"  value="0.0" />
```

Sample usage (using keyint):

```
<set-domain-measure-value-ne name="meas2" namedkeys="chnl1:week1"
value="02/24/2012" />
```

Sample usage (load-measure file):

```
<set-domain-measure name="meas1" path="loadMeas1.dat" start="100" width="20"/>
```

Sample usage (set all):

```
<set-domain-measure name="meas1" value="2"
```

C

Appendix: Fusion Client Properties Files

Several properties files are located in the config folder of the Fusion Client installation. The majority of these files should not be altered. However, you can alter the "rpasConfig.properties File", "PivotTableStyles.properties File", "GraphConfig.xml File", and "cdtConfig.properties File" files. This appendix contains the following sections:

- Editing Property Files
- rpasConfig.properties File
- PivotTableStyles.properties File
- GraphConfig.xml File
- cdtConfig.properties File

Editing Property Files

To edit these files, perform the following steps:

1. Navigate to the following location in the RPAS Fusion Client installation directory:

   ```
   <RPAS_Fusion_Client_Installation>/config/rpas/
   ```

2. Open the file with a text edit and edit the relevant parameters.
3. Save and close the file.
4. For the changes to take effect, restart the WebLogic server.

rpasConfig.properties File

Use the rpasConfig.properties file to alter the look and feel of the Fusion Client.

The first two settings (Cell Data Caching Block Size Properties and PivotTable Fetch Size Properties) affect the responsiveness of the workbook from the point of view of the user.

Larger settings result in a bigger block of data being loaded, making the workbook slower to open and slower when refreshing. However, the user will be able to scroll further before a refresh operation is required. Smaller settings will ensure the workbook loads quicker, but will result in user scrolling to cells requiring a refresh operation more frequently.

There is no single optimum setting for these values, requirements changing from implementation to implementation depending on the expected size of the pivot table that users will be working with.

- Cell Data Caching Block Size Properties: The block size is set to be 33% more than the maximum table size in each dimension. The default setting accommodates 22 columns and 35 rows, which is typical for a 1280x1024 screen. This default setting ensures that the pivot table fetches no more than one block of cell values when refreshed, and at the same time allows for some nearby scrolling without causing a server fetch. Please set their values to be the same as the values for the pivottable.fetch.columns and pivottable.fetch.rows properties.

  > **Note:** A possible exception to the recommended defaults is if there is a fixed number of rows or columns, and you want to allow the users to freely scroll in that direction without causing a server fetch. The relevant row or columns property may then be set to that value - although this may result in significantly increased load and refresh times.

  As mentioned above, if you change the pivottable.fetch.column and pivottable.fetch.rows properties, ensure you set the celldatacache properties to be same as the former. The values shown are the default.

  - `celldatacache.fetch.columns=30`
  - `celldatacache.fetch.rows=47`
  - `celldatacache.enabled=true`

  > **Note:** These values are not used if the `block.celldata.fetch.and.cache.strategy property` is set to `false` in the Internal Configuration section.

- PivotTable Fetch Size Properties: Set these properties to match your largest pivot table fetch size, which is the best block size that the middle tier fetches from RPAS. (See the `celldatacache.fetch` property above.) In some cases, a smaller size provides better scrolling performance.

  You can adjust the following settings. The values shown below are the defaults.

  - `pivottable.fetch.columns=30`
  - `pivottable.fetch.rows=47`

  It is recommended that you set the Cell Data Caching Block Size Properties and PivotTable Fetch Size Properties to identical values. By setting the values in this manner, a single fetch from the pivot table (on the browser) to the application server is matched up with a single fetch from RPAS.

- Pivot Table Default Column Header Properties: Use these properties to configure the height and width of column headers. These sizes can override the autosize feature in the pivot table, allowing the administrator to set specific default sizes for headers that have not been sized by a user yet. The values provided for the properties are in pixel size.

  ```
  pivottable.default.col.header.row.height=-1
  pivottable.default.col.width=80
  ```

  If you set the value to -1 or commented it out, the column typically resizes so that the text before the first white space is visible.

- Pivot Table Default Row Header Properties: Use these properties to configure the height and width of row headers. These sizes can override the autosize feature in the pivot table, allowing the administrator to set specific default sizes for headers that have not been sized by a user yet. The values provided for the properties are in pixel size.

  ```
  pivottable.default.row.header.column.width=300
  pivottable.default.row.height=-1
  ```

  If you set the value to -1 or commented it out, the column typically resizes so that the text before the first white space is visible.

- Internal Configuration Property: Use this property to configure the cell data fetch and cache strategy. Set it to `true` to fetch and cache in fixed sized blocks the coordinates whose vertices are integer multiples of the block height (y-coord) and the block width (x-coord). Set to `false` to fetch blocks based on the displayed screen coordinates. All fetches retrieve only the data that is not already cached. The default setting is `true`.

  `block.celldata.fetch.and.cache.strategy=false`

- Xml Persistence Properties: Set the following properties as desired. The values shown below are the default.
  - `xmlpersistence.enabled=false`
  - `serverloglevel=error`

    This enables the RPAS Server logging levels. Logs are sent to the appropriate users' folder on the RPAS Server machine. The logs are named `rpas_D<login_date_time>`.

    Possible values are `all`, `profile`, `debug`, `audit`, `information`, `warning`, `error`, or `none`.
  - `datacontrol.enabled=true`

    This property is set to `true` to enable CDF DataControl. Do not change this value.
  - `printexport.maximum.cells=200000`

    If set to `0`, it runs the export for all the cells but it may cause an out-of-memory error on the server.
  - `insertmeasure.maximum.measures=10`

    Limits how many measures can be inserted at once.

- Real Time Alert Blocking Properties: These settings can be used to tune the how the real time alert hits are obtained from the server for the purpose of navigation and filtering. There are two settings for navigation, and one for filtering.

  - real.time.alerts.max.block.size=1000

  When fetching real time alert hits for alert navigation, get at most this amount at a time from the server. Default value is 1000, minimum value is 2.

  - real.time.alerts.max.blocks.per.view=3

- Number of blocks of real time alert hits (of the size above) to cache on a view for navigation purposes. Default value is 3, minimum value is 1.

  - real.time.alerts.max.filter.block.size=10000

When fetching real time alert hits for filtering a view, get at most this amount at a time from the server. Default value is 10000, minimum value is 2.

- Real Time Alert Count Display on the Tool Bar: For real time alerts, the count of the active alert is displayed on the toolbar. This setting defines the max width of the alert count on the toolbar before an overflow form is used instead. The overflow is defined via resourcesReal Time

  toolbar.alert.count.max.width=5

  This setting -1 is no limit, 0 is don't show the count. Default is 5.

- Restrict Size of Cell-based Operations: Extremely large selections can lead to memory problems in Cut, Copy, Paste, Fill, or Clear operations. This setting defines a limit.

  ccp.max.selection.size=10000

  Maximum selection size in number of cells. Default is 10000.

- Valid Image Hosts: Users can specify a web URL as the location of an image. This can be dangerous if left unchecked. Users will only be allowed to enter URLs from the hosts defined by this setting.

  valid.image.url.hosts=localhost

  List of valid hostnames separated by commas. Enter only hostname and not the http protocol. For example xyz.com

- Splash Screen Property: Use this property to alter the splash screen. Set this property to true `if` the splash screen should be displayed on a calculate function. Otherwise, set to `false`. The default is true.

  ```
  splash.screen.on.calculate=true
  ```

- Valid Image File Types Property: Use this property to specify the file type for images.

  ```
  valid.image.file.types=jpg,jpeg,bmp,png,gif,ico,tif
  ```

- Concurrent Session Property: Set this property to true if you want the Fusion Client to automatically create a new concurrent RPAS session. The default is false. When this property is set to false, users are asked if they want to create a new concurrent session or terminate the old session and create a new one.

  ```
  autocreate.concurrent.session=false
  ```

- Custom Menu Character Length Property: Use this property to set the number of characters the custom menu buttons should truncate to. The default is 20.

  ```
  custom.menu.number.of.characters=20
  ```

- Title Bar Color Properties: Use these properties to set the color of the title bar in the selected or deselected window. You can specific colors by using hex color codes (EBF1F9) or by entering common HTML colors (white, black, red, and so on).
  - `selected.window.background.color=EBF1F9`
  - `unselected.window.background.color=white`

- Expand Taskflow Property: Use this property to set the initial collapse or expand state of the taskflow after the first workbook is built or opened in a user session. The user’s updates after login remain throughout the session.

  ```
  minimize.workbook.taskflow=false
  ```

PivotTableStyles.properties File

The RPAS Fusion Client installs with the following default formatting styles for the cells that appear on the pivot table in the views:

- Read-only cells: Cell with a light grey background.
- Protected cells: Cell with a light grey background.
- Edited cells: Text within the cell appear in italics with no background color.

You can change the default formatting styles (set a different background color or font style) using the PivotTableStyles.properties file. To change the default pivot table cell formatting styles, edit the following parameters

- pivotTable.protected.style: Style for the protected cells in the pivot table.
- pivotTable.unprotected.style: Style for the un-protected cells in the pivot table.
- pivotTable.readonly.style: Style for the read-only cells in the pivot table.
- pivotTable.edited.style: Style for the edited cells in the pivot table.
- pivotTable.edited.text.style: Style for the edited text in a cell.
- pivotTable.unedited.style: Style for the unedited text in a cell.
- pivotTable.undefined.style: Style for the undefined cells.
- pivotTable.*ambiguous.char*: Character to display when an ambiguous aggregation occurs.

To set a background color, specify background-color:<hexadecimal code for the color>;. For example, background-color:DDDDDD;.

To set a font style, specify font-style:<text-style>;. For example, font-style:italic;.

The Header text size is configurable; the default value is 150 characters.

GraphConfig.xml File

Use the GraphConfig.xml file to set default graph type and individual settings for each graph type.

Global Settings

In the graphGlobal section of the file, define the following properties:

- `defaultGraphType`: Enter the name of the graphic that you want to be the default graph type.
- `highlighting`: The default setting is highlight. Do not change this property.
- `animate`: The default setting is `animationOn`. Do not change this property.
- `maxGroups`: Enter the maximum number of groups. The value must be between 1 and 5000. The default is 100.
- `maxSeries`: Enter the maximum number of series. The value must be between 1 and 5000. The default is 100.
- `maxIntersections`: Enter the maximum number of intersections. The value must be between 1 and 25000. The default is 1000.

Note: For `MaxGroups`, `maxSeries`, and `maxIntersections`, the default values are recommended. Setting these values to be larger may have a performance impact.

Individual Graph Settings

You can also alter the properties of individual graph types. The individual graph properties are the same as the properties in the Global Settings section, but with one addition:

- `defaultGraphSubtype`: Enter the graph type for each graph's subtype.

cdtConfig.properties File

Use the cdtConfig.properties file to configure the CDT Editor for consumer decision trees. The following properties are available:

- cdt.template.name: The template name for the CDT Editor workbook.
- cdt.info.view.name: The name of the worksheet view that contains the CDT XML, user description, and isvalid status (CDT INFO).
- cdt.info.xml.meas.name: The name of the measure that stores CDT as XML in the CDT INFO worksheet view.
- cdt.info.desc.meas.name: The name of the measure that holds the user description in the CDT INFO worksheet view.
- cdt.info.is.valid.meas.name: The name of the measure that indicates the validity of the version at a Category, Trading Area, and Consumer Segment intersection.
- cdt.category.dimension.name: The name of the CDT dimension that contains Categories.
- cdt.trading.area.dimension.name: The name of the CDT dimension that contains Trading Areas.
- cdt.consumer.segment.dimension.name: The name of the CDT dimension that contains Consumer Segments.
- cdt.attr.view.name: The name of the worksheet view that contains the attribute names and values information (ATTR).
- cdt.attr.value.meas.name: The name of the measure that hold the attribute values for each SKU and attribute name in the ATTR worksheet view.
- cdt.prod.attr.dimension.name: The name of the product attributes dimension that contains attribute-names.
- cdt.approval.view.name: The name of the worksheet view that contains the Accept XML status for processing (APPROVAL).
- cdt.accept.xml.measure.name: The name of the measure that indicates if any new or modified CDT XML has been processed (accepted) in the APPROVAL worksheet view.

D

Appendix: 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 value, numeric value, or date and time. 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 &
> ```

This appendix describes the following environment variables used for RPAS:

- Required Settings
- Database Settings:
  - RPAS_ARRAY_TAG
  - RPAS_CACHE
  - RPAS_PAGE_SIZE
  - RPAS_PAGE_SPLIT_PERCENTAGE
  - RPAS_LOCK_TIMEOUT
  - RPAS_USER_MODE
- Java Special Expression Settings:
  - RPAS_JAVA_CLASSPATH
  - RPAS_JAVA_DEBUG_PORT
  - RPAS_JAVA_MAXHEAPSIZE
- Log Settings:
  - RPAS_CACHE_LM_MULTIPILER / RPAS_DISABLE_LM_DIRECT
  - RPAS_CLIENT_TIMEOUT / RPAS_SERVER_TIMEOUT

  - RPAS_COMMIT_FAILED_MAX / RPAS_COMMIT_SUCCESS_MAX
  - RPAS_JAVA_MAXHEAPSIZE
  - RPAS_LOCK_DETAIL_PATH
  - RPAS_LOCK_LOGGING
  - RPAS_LOCK_LOG_PATH
  - RPAS_LOCK_TRACK_MODE
  - RPAS_LOCK_TRACK_PATH
  - RPAS_LOCK_TRACK_DATABASES
  - RPAS_LOCK_ACCESS_LIMIT
  - RPAS_LOCK_MSG_PATH
  - RPAS_LOG_BACKUPS
  - RPAS_LOG_FD
  - RPAS_LOG_FLUSH
  - RPAS_LOG_LEVEL
  - RPAS_LOG_PATH
  - RPAS_REQUEST_TIMEOUT
  - RPAS_SHELL
  - RPAS_SPLIT_BUF_MB
  - RPAS_STATS_PATH
  - RPAS_TCPNODELAY
  - RPAS_TRACE_SSL
  - TMPDIR
  - TODAY_RETEK
- Parallelization Settings:
  - RPAS_EXPORT_MEASURES_MAX
  - RPAS_MIN_MEAS_PER_PROC
  - RPAS_PROCESSES
- Profiling Logging:
  - RPAS_PROFILING_ENABLE
  - RPAS_PROFILING_PATH
- Date and Time Setting:
  - RPAS_TODAY
  - RPAS_TODAY_STATIC
- Numeric Precision:
  - RPAS_INCAGGPRC
  - epsilon
- RPAS_SSL_WALLET

- LDR_CNTRL=NAMEDSHLIB=RPASZONE
- Nagle's Algorithm

Required Settings

The following environment variables are required.

- RPAS_HOME
- RIDE_HOME
- RIDE_OPTIONS
- JAVA_HOME
- RPAS_JAVA_CLASSPATH
- PATH

Since these environment variables are required at the time of installation, and since they vary according to the operating system that RPAS is running on, these environment variables are described in the "Installing on UNIX and Linux Environments" chapter of the *Oracle Retail Predictive Application Server Installation Guide*. See that chapter for detailed instructions for those environment variables.

Database Settings

These variables are used for RPAS B-tree storage performance.

RPAS_ARRAY_TAG

RPAS BTree arrays contain a field named array tag. Whenever a BTree array is written, the value of the RPAS_ARRAY_TAG environment property, if it is set, is written as the value of the array tag field. By using this property, users can set a record in an array to provide information about the last time the array was modified.

RPAS_CACHE

RPAS_CACHE determines the number of pages of a BTree array file that are contain in memory. This variable can be set to have values between 16 and 512 (including 16 and 512). The larger you set the cache value, the more memory usage and less disk access you have.

If you set the cache size from 4 to 64, the performance of the BTree arrays should level out around 16 pages. In tests, more pages did not provide better performance. However, for other access patterns, more cache pages may improve performance. Therefore, a larger cache and page size may increase performance due to less disk access. However, having a larger cache and page size increases the memory image of an open BTree file. It is likely that if the number of cache elements is 512 and the page size is 256K, then each open BTree file could use up to 131,072K of memory.

```
export RPAS_CACHE=16
```

Values: 4, 8, 16, 32, 64, 128, 256, 512

RPAS_PAGE_SIZE

RPAS_PAGE_SIZE determines the size of a single BTree array page if the logical size of the BTree array is greater than 200,000 cells. A BTree array is compose of a number of pages and page types. Each page of a BTree array file is the same size. In the trunk, there are five pages types: header, branch, leaf, data, and free. A larger page size implies more memory usage, less fragmentation, and less disk access.

The page size also affects the disk overhead that each BTree file requires. A large page size may provide better disk usage if the array is densely pack. However, large page size with a loosely pack array uses more disk space. For example, consider the effect of having 1,000,000 cells with only one cell populated on a 256K page versus one cell populated on a 16K page. The first case requires 256,000,000K of disk space, while the second requires 16,000,000K of disk space.

Therefore, RPAS_PAGE_SIZE and RPAS_CACHE both affect performance based on the access pattern. RPAS_PAGE_SIZE affects disk space and memory usage, and RPAS_CACHE affects the memory usage.

```
export RPAS_PAGE_SIZE=32K
```

Values: 4K, 8K, 16K, 32K, 64K, 128K, 256K

RPAS_PAGE_SPLIT_PERCENTAGE

This variable sets the page split percentage. The page split percentage determines the percentage of low order keys to keep in the low order page of the B-Tree when a full page is being split to accommodate data for a new key. It is expressed as a number between 1 and 100. RPAS stores non-NA values only. If an operation (load or calculation) causes a value to be non-NA, RPAS tries to store it on a page and, if the page is full, it causes it to split. Page splitting is an expensive operation; that is, it takes a significant amount of processing time and if it happens too often, it can significantly slow down the load or calculation process causing the splits. For efficient storage and to prevent pages from being split very often, it is recommended that the value be kept between 50 and 90. If this environment variable is not set, RPAS uses a page split percentage of 90.

RPAS_LOCK_TIMEOUT

This environment variable sets the maximum number of milliseconds to wait for a database lock. The default value, 60,000, forces a database lock to wait for a maximum of one minute before throwing a database lock exception.

```
export RPAS_LOCK_TIMEOUT=60000
```

Values: 60000, 90000, 120000, and so on

When performing certain operations, it is possible for two or more users to be contending for access to the same database, which happens most commonly when two users attempt to simultaneously commit and save the same data back to the domain. This can also occur when running RPAS utilities in parallel to prevent database locking during batch operations. 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 environment 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 line below indicates 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
```

On the client side, the exception sent from the server appears as a warning dialog box only after some new action is initiated. With the client, additional warning dialog boxes are displayed to the user.

RPAS_REQUEST_TIMEOUT

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

The environment variable RPAS_REQUEST_TIMEOUT should be set to a value that is the number of seconds of idle time that 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 time out.

```
export RPAS_REQUEST_TIMEOUT=600
```

Values: 600, 900, 1200, and so on

RPAS_USER_MODE

This environment variable determines how the DbCloser class behaves. It controls whether the DbCloser aggressively closes databases. The “single” value puts the DbCloser is single user mode. This is useful for running batch processes when no users are logged in, because in such a situation there is no need to rapidly close databases.

```
export RPAS_USER_MODE="single"
```

Value: single

Java Special Expression Settings

The following environment variables can be used for Java special expression settings.

RPAS_JAVA_CLASSPATH

When using the Java Special Expression functionality, the RPAS_JAVA_CLASSPATH environment property contains the location of the Java expressions. The value of RPAS_JAVA_CLASSPATH should be a set of paths to the jar files that contain the compiled Java expressions. The list of paths must also contain the path to the jar file oracleRpasUtils.jar located within the lib sub-directory of RPAS_HOME.

RPAS_JAVA_DEBUG_PORT

The RPAS_JAVA_DEBUG_PORT environment variable can be used to specify to RPAS what port the JVM, created to execute a Java Special Expression, should listen to in order for a debugger to attach to the JVM. It can be used to debug Java Special Expressions during their development. The value of the RPAS_JAVA_DEBUG_PORT property should be an integer within the range of available port addresses.

RPAS_JAVA_MAXHEAPSIZE

The RPAS_JAVA_MAXHEAPSIZE environment property can be used to specify to RPAS that the JVM, created to execute a Java Special Expression, should request more memory from the OS than the default. The value of RPAS_JAVA_MAXHEAPSIZE should be the number of megabytes of memory to request followed by the letter 'm' (for example, 1024m).

Log Settings

The following environment variables can be used for log settings.

RPAS_CACHE_LM_MULTIPILER / RPAS_DISABLE_LM_DIRECT

The RPAS loadmeasure application uses these environmental variables. The RPAS_DISABLE_LM_DIRECT environmental variable may be "yes" or "1" to force the loadmeasure application to run in legacy mode which disables the use of the RPAS_CACHE_LM_MULTIPILER variable.

The RPAS_CACHE_LM_MULTIPILER variable must be between 2 and 10 and is used to compute the amount of data the loadmeasure application will cache base on its value and the value of the RPAS_CACHE variable.

RPAS_CLIENT_TIMEOUT / RPAS_SERVER_TIMEOUT

These environmental variables determine the amount of time a TCP/IP read or write call waits for data to be processed. If no data has been processed in the amount of time provided, RPAS throws an exception.

RPAS_COMMIT_FAILED_MAX / RPAS_COMMIT_SUCCESS_MAX

These environmental variables determine the number of failed or success entries that are kept in the success or failed directories for commit ASAP.

RPAS_JAVA_MAXHEAPSIZE

This environment parameter, if set, applies only to the RPAS Java Special Expression functionality, and is passed along as the -Xmx argument to the Java JVM instance.

RPAS_LOCK_DETAIL_PATH

This environment property gives a directory path to which detailed file lock history data will be written, in binary form, using one log file per RPAS process. The lock detail files, which will not be "human readable", are constructed to be extremely compact so that a large amount of file history can be written efficiently. These details consist only of the file's inode (internal operating system identifier for the file), the operation (open for read, open for write, close), and a date/time code. The utilities for analyzing this information are not currently a part of the official RPAS release, so the data would need to be sent to the RPAS Development staff for analysis. Because of this, the environment variable should only be set when requested by RPAS Development or Support in cases of an ongoing support need related to file lock contention.

RPAS_LOCK_LOGGING

This environment property causes RPAS binaries to log details about the locking and unlocking of files. The value for RPAS_LOCK_LOGGING should be set to the logger level (error, warning, debug, profile, information) at which the file lock details will be output. So, for instance, with RPAS_LOCK_LOGGING=debug, if you run an RPAS binary with -loglevel debug, then you will see the file lock details, but if you run the binary with -loglevel warning, you will not see them.

RPAS_LOCK_LOG_PATH

This environment property gives a file path to which details of file locks will be written, in human-readable text form. The details match those which are output to the screen when RPAS_LOCK_LOGGING is set to an appropriate level.

RPAS_LOCK_TRACK_MODE

This Boolean environment property, if set to "true", turns on tracking of file lock statistics. When enabled, any RPAS binary reports the minimum, maximum, average, and count of read and write locks on each file that it opens. These details are logged to the screen unless RPAS_LOCK_TRACK_PATH is also used (see RPAS_LOCK_TRACK_PATH).

RPAS_LOCK_TRACK_PATH

This string environment property gives a file path/name to which the file lock statistics (as described under RPAS_LOCK_TRACK_MODE) will be written, rather than displayed to the screen. Note that, if this value is set, you do not need to set RPAS_LOCK_TRACK_MODE, as it is implied

RPAS_LOCK_TRACK_DATABASES

This environmental variable will filter the logged file lock information in order to limit it to one or more databases of interest. It should be set to a comma-separated list of database names, for example "admin,measdata". This would cause lock information to be logged only for files in the admin.db and measdata.db databases.

This setting works in conjunction with these other environment variable settings:

- RPAS_LOCK_LOGGING
- RPAS_LOCK_ACCESS_LIMIT
- RPAS_LOCK_LOG_PATH
- RPAS_LOCK_TRACK_PATH

RPAS_LOCK_ACCESS_LIMIT

Set this environmental variable to the maximum number of seconds a lock file can be locked before the name of the lock file is logged. The logging occurs at the level provided by the environmental variable RPAS_LOCK_LOGGING. This allows you to filter out brief lock times from the log, showing only those files which are locked for a longer time.

RPAS_LOCK_MSG_PATH

This environment setting determines the location of a file that can contain messages to be added to the log. This is primarily utilized in debugging situations, when the SE team needs to note a particular occurrence in the log.

RPAS_LOG_BACKUPS

The RPAS_LOG_BACKUPS environment variable sets the maximum number of log files to keep.

```
export RPAS_LOG_BACKUPS=20
```

Values: 2, 3, 4, 10, 20, 30, and so on

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 variable 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.

RPAS_LOG_FD

This environment property causes RPAS binaries to log the opening and closing of all file handles. The value for RPAS_LOG_FD should be set to the logger level (error, warning, debug, profile, information) at which the file open/close details will be output. So, for instance, with RPAS_LOG_FD=debug, if you run an RPAS binary with -loglevel debug, then you will see the file lock details, but if you run the binary with -loglevel warning, you will not see them.

RPAS_LOG_FLUSH

Setting this environmental variable to anything other than empty or the string "false" (case-insensitive) causes the RPAS logger to flush the output after each line of output. This may cause a tiny performance impact, but ensures that all logger output has been sent before any error or crash condition. It should generally only be needed in cases of debugging.

RPAS_LOG_LEVEL

The RPAS_LOG_LEVEL environment variable sets the default logging level to use when an application starts.

```
export RPAS_LOG_LEVEL="debug"
```

Values: all, profile, debug, audit, information, warning, error, none

RPAS_LOG_PATH

The RPAS_LOG_PATH environment variable contains the path of a directory where RPAS daemon applications creates log files. The default is the current working directory where the application was started from.

```
export RPAS_LOG_PATH=" C:/RPAS/Domains/Temp"
```

Value: The absolute path to the directory where RPAS daemon applications creates log files

RPAS_REQUEST_TIMEOUT

This environmental variable specifies an amount of time (in seconds) that a TCP/IP read or write call will wait for data. This functions similarly to the RPAS_SERVER_ TIMEOUT environment setting, however, this variable only applies to the RpasDbServer application. The intent of this environmental variable is to handle the issue of fire walls. A fire wall may be setup to kill any TCP/IP connection that is idle for 60 minutes. If someone using the client has connected to an RpasDbServer behind a fire wall with this setup, they could run into this issue. If the RPAS Client user is connected, but leaves the connection idle for 60 minutes (for example, over lunch hour), the fire wall would then kill the connection. The Client appears connected, but will not be functional. If the user attempts to reconnect to the server, the RPAS DomainDaemon will not allow the connection since the earlier RpasDbServer process is still running. Setting the RPAS_REQUEST_TIMEOUT to 30 minutes (or 1800 seconds) using the export command before starting the DomainDaemon:

```
export RPAS_REQUEST_TIMEOUT=1800
```

Now the RpasDbServer will timeout after 30 minutes of idle TCP/IP time. The Client user will then be able to reconnect to the server since the previous session will have properly disconnected.

RPAS_SHELL

This environment setting applies only to running the RPAS Server on Windows in the Cygwin environment, and is needed only if your Cygwin installation is in a non-standard directory location. RPAS Server code will expect to find the Cygwin BASH shell executable at the path C:/cygwin/bin/bash.exe. If your Cygwin BASH shell is located at a different path, set the RPAS_SHELL environment variable to the correct location.

RPAS_SPLIT_BUF_MB

This environment parameter optionally adjusts the size (in megabytes) of the file buffer used in the internal measureFileSplitter utility. If needed, it should be set to a positive integer value; if not present, the default of 8 is used.

RPAS_STATS_PATH

This environmental variable is used to determine where to write the stats information gathered during a workbook build.

RPAS_TCPNODELAY

The RPAS Client/Server communications use a TCP/IP interface. Built in to most TCP/IP implementations is an efficiency improvement known as Nagle's Algorithm. A complete discussion is beyond the scope of this document (see http://en.wikipedia.org/wiki/Nagle's_algorithm), but it relates to added efficiency when sending many small packets of data by grouping them. However, this can cause delays in certain communication patterns, an issue that some customers detect when using RPAS. To disable Nagle's Algorithm, you may set environment variable RPAS_ TCPNODELAY to "on". Conversely, to enable Nagle's Algorithm, the value of RPAS_ TCPNODELAY may be set to "off". If the environment variable is unset, or set to anything besides on or off, the TCP/IP communication behavior is determined by your operating system settings.

RPAS_TRACE_SSL

This environmental variable is used to create a log file of the SSL handshake. This variable must be set to two values with a comma separator. The first value must be one of these strings: "fatal", "error", "warn", "user", "debug", "ssl", or "memory". The second value can be either an RPAS loglevel string such as "basic", "none", "error", "warning", "information", "audit", "profile", "debug", or "all"; or an absolute path to a log file.

To set the ssl trace to "user" level and log to the file "/home/larry/ssltrace.log" use:

```
Export RPAS_TRACE_SSL=user,/home/larry/sslTrace.log
```

To set the ssl trace to "debug" and log the output to the RPAS logging level "debug" use:

```
Export RPAS_TRACE_SSL=debug,debug
```

TMPDIR

The RPAS Server sometimes needs to create temporary files. By default, these are placed into the /tmp directory. If the RPAS administrator wishes to set a different location for temporary files, the TMPDIR environment variable should be set to the full path to this directory.

TODAY_RETEK

See RPAS_TODAY in the Date and Time Setting section.

Parallelization Settings

The following environment variables can be used for parallelization settings.

RPAS_EXPORT_MEASURES_MAX

The RPAS command-line utility "exportMeasure" runs sub-processes in order to support greater efficiency by parallel processing. A positive integer value assigned to RPAS_EXPORT_MEASURES_MAX sets an upper limit on the number of measures to be handled by each invocation of the sub-process. If the environment variable is not set, the default value of 100 is used.

RPAS_MIN_MEAS_PER_PROC

The RPAS command-line utility "loadmeasure" runs sub-processes in order to support greater efficiency by parallel processing. A positive integer value assigned to RPAS_MIN_MEAS_PER_PROC sets a lower limit on the number of measures being loaded in each sub-process. If the environment variable is not set, the default value of 100 is used.

RPAS_PROCESSES

This variable is used when performing batch measure registration. It sets the maximum number of processes used to stage data. The default is 1, which causes all processes to be run serially.

If a user specifies a value less than 1, it will override to 1. If a user specifies a value greater than the number of subdomains, it will be override to the number of subdomains in the global domain. If this variable is set to a number greater than 1 when doing a measure registration in a global domain, RPAS spawns that number of parallel processes to register or unregister measures in each local domain.

```
export RPAS_PROCESSES=4
```

Values: 1, 2, 3, 4, and so on

Profiling Logging

The following two environment variables may be set up 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 a 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.

Date and Time Setting

The following environment variables can be used to set the date and time.

RPAS_TODAY

RPAS_TODAY tells RPAS what day that it should think today is.

```
RPAS_TODAY=20090530
```

Note: TODAY_RETEK is an older, deprecated synonym. It is allowed for historical reasons, but RPAS_TODAY is preferred.

RPAS_TODAY_STATIC

The RPAS_TODAY_STATIC environment variable affects how date and time and the RPAS_TODAY environment variable are handled. The setting of RPAS_TODAY_STATIC affects the date and time that is returned by the DateTime::now() function.

- If RPAS_TODAY_STATIC is set to true, all calls to DateTime::now() return the same date and time as the first call made to the DateTime::now() function. The same date and time is returned no matter how many times the DateTime::now() function is called.

  For example, RPAS_TODAY_STATIC is set to true and RPAS_TODAY=20090530. If a call is made to DateTime::now() at 10:30 pm, it returns a date and time of 5/30/2009 10:30:00 pm. If another call is made 10 minutes later to DateTime::now(), a date and time of 5/30/2009 10:30:00 pm is also returned.

- Set the HHMMSS portion of RPAS_TODAY (YYYYMMDDHHMMSS) when setting the RPAS_TODAY_STATIC to TRUE. The function DateTime::now() or keyword 'now' also include the time of the day in its value. If RPAS_TODAY_

STATIC is set to TRUE and if YYYYMMDDHHMMSS format is not used but YYYYMMDD format is used while setting RPAS_TODAY, then the day portion of 'now' will remain the same but the HHMMSS portion of 'now' changes.

- If RPAS_TODAY_STATIC is set to false, a call to DateTime::now() returns the current date and time.

  For example, RPAS_TODAY_STATIC is set to false and RPAS_TODAY=20090530. If a call is made to DateTime::now() at 10:30 pm, it returns a date and time of 5/30/2009 10:30:00 pm. If another call is made 10 minutes later to DateTime::now(), a date and time of 5/30/2009 10:40:00 pm is returned.

- User level environment variable TZ (time zone) needs to be set for the user who starts the RPAS server processes (DomainDaemon, RPAS ODBC Server) or runs RPAS utilities (like printMeasure). A missing TZ causes the system to misinterpret the date value stored in RPAS. For example, on a Linux system, the default DATE type measure NA value of 0001/01/01 would be interpreted as 7295/12/31 23:00:00.000 if TZ were not set.

  If the user is in time zone EST5EDT, use the command `export TZ=EST5EDT` to set TZ. It is suggested that the command be added to user's login script (for example, .profile).

Numeric Precision

Two variables affect the level of numeric precision that is displayed in the RPAS clients:

- `RPAS_INCAGGPRC`: An environment variable.
- `epsilon`: The smallest difference that is allowed between any two numbers.

RPAS_INCAGGPRC

Use the `RPAS_INCAGGPRC` environment variable to set how a number's precision is displayed. Set this variable to the precision level you want displayed. The default value of this variable is 1e-06. Unless this variable is altered, very small values, such as 0.00000001, are displayed as 0.

Following is an example of setting this variable to a value smaller than the default:

```
export RPAS_INCAGGPRC=0.0000000001
```

epsilon

The epsilon variable also affects the level of precision that is displayed. This variable is returned as a value by a public static function in the MathUtilities.h file. It is hardcoded to 0.0000000001 (1e-09).

RPAS uses the epsilon value to consider any two numbers as different if the difference is greater than or equal to epsilon. If the difference between any two numbers is less than epsilon, then those numbers are considered as equal. Indirectly, the epsilon value affects the display as it sets a limit to the number precision.

Note: Reducing the epsilon value may impact performance.

RPAS_SSL_WALLET

This variable may be used as an alternative to the -wallet argument to DomainDaemon. If the RPAS_SSL_WALLET environment variable is defined, its value can be used by DomainDaemon in place of specifying the information as a parameter to the -wallet argument.

Please refer to the "DomainDaemon" section of the guide for more information on the use of the -wallet argument.

LDR_CNTRL=NAMEDSHLIB=RPASZONE

This environment setting performs two functions:

- It creates a special named zone in the memory to load shared libraries for RPAS processes to avoid version conflict.

  By default, AIX loads shared libraries in a single global zone. This could potentially cause conflict between the libraries on the system and RPAS versions. For example, RPAS has its own version of libz. If a non-RPAS program has loaded a different version of libz from the system into the global zone, RPAS will use that one instead of loading its own.

- It may also reduce the memory footprint for shared libraries in the private memory of RPAS processes.

  The global zone can be filled up if the system is very busy. In that case, the shared libraries are loaded into private space of RPAS processes, which would reduce the amount of memory available for RPAS data. A named zone is difficult to fill up since it is used only by RPAS processes.

```
export LDR_CNTRL=NAMEDSHLIB=RPASZONE
```

Nagle's Algorithm

Nagle's algorithm is used to improve the efficiency of TCP/IP networks. The algorithm combines a group of small outgoing messages and sends them all at once. This reduces the number of packets that are sent over the network.

Two settings are used for Nagle's algorithm, RPAS_TCPNODELAY for the Server and FC_TCPNODELAY for the Fusion Client.

Server

Setting RPAS_TCPNODELAY to **on** disables Nagle's algorithm on the server side when it sends the data back to the client. This setting is useful for improving the performance of the Fusion Client in some cases.

Setting RPAS_TCPNODELAY to **off** or not setting it enables Nagle's algorithm.

Fusion Client

Setting FC_TCPNODELAY to **true** disables Nagle's algorithm on the Fusion Client side when it sends the data from the client to the server. This setting is useful in improving the performance of the Fusion Client in some cases.

Setting FC_TCPNODELAY to **false** or not setting it enables Nagle's algorithm.

E

Appendix: RPAS Diagnostic Tool

This appendix covers a diagnostic tool useful for identifying the causes of problems in the software.

Diagnostic Tool Description

When investigating defects observed at a customer in the RPAS Fusion Client or Server, it is useful to get detailed information about the transactions executed during the bug. At higher logging levels, RPAS can be set to routinely collect a substantial amount of information. Unfortunately, this information does not include the actual transaction requests and response data that the customer is seeing.

This appendix describes a diagnostic tool that enables the customer to record their transactions and send those recordings to Oracle. This recording enables Oracle's support engineers to replay the transactions in the exact sequence they occurred and diagnose the problem. This diagnostic tool consists of two modules.

Recorder

This module is part of the fusion client code. When the fusion client runs in recording mode, all transactions are written to a log file for later analysis.

Replayer

This is a standalone tool. Providing the user has a copy of the same domain that the information was recorded in uses the same key store files and key store password, it can be used with the log file path as argument and replays the transactions in exact order as they originally occurred.

Using the Tools

The two can be used in combination by:

1. Using **Recorder** at the customer to generate a log file for later analysis.
2. Sending a copy of the domain and Recorder log file to Oracle Support.
3. Using **Replayer** to reproduce the fault at Oracle and allow the problem to be diagnosed by the RPAS development team.

Using Recorder

The Fusion Client has the capability to produce a recording of RPAS transactions at the customer's location in the form of a log file. To use this functionality, use the following settings in the *rpas/rpasconfig.properties* file in the Fusion Client configuration directory (<fc_install_folder>/config):

```
# Enable transaction logging (default is false)
txnlogging.enabled=true
# Comma-separated list of users to track transactions for
# If not defined, then no users are tracked
# i.e. you must specify a user
txnlogging.users=user1,user2
```

The logging option only applies to specific users. These users must be specified as a comma-separated list of *userids* against the property *txnlogging.users*. In order for these uses to generate logs, the property *txnlogging.enabled* must also be set to true.

Once the *rpas/rpasconfig.properties* file has been configured, the selected users can record their actions and submit the generated logs. These are stored in the directory *txnlogs* in the Fusion Client configuration directory, with the name *txnlog.<username>*. As an example the recording for user1 would be stored in *txnlog.user1*.

A fresh recording is started each time a user logs in. The resulting log is only available for copying after the user session ends. This session could end due to a logout, a session timeout or the application server being brought down.

Because each user session generates a file with the standard name *txnlog.<username>*, prior sessions by the same user will be overwritten unless the previous file is renamed or moved to another location.

Using Replayer

This section describes configuring and using Replayer.

Configuring Realplayer for Use

Replayer enables the RPAS development team to reproduce the issue in an Oracle test environment and diagnose the problem. For this the conf folder will be the default config root. This folder contains the *FCkey.jks*, *FCTrust.jks* and *Replayer.properties* files that are needed by Replayer. This conf folder can be in any location but the system environment folder.

```
#domain daemon port number
replayer.db.daemon.port=1800
#The server where RPAS is running.
replayer.db.server.host=localhost
#domain directory path. the domain is the copy of the customer #domain where the
scenario has to be reproduced.
replayer.domain.dir=D:\\Domain\\PlatformConfigGD
#id stores location. The path which contains fckey.jks and #fctrust.jks. This will
conf directory by default.
replayer.id.store.loc=
#key store password.
replayer.key.store.pwd=clientjkspwd1
```

Replayer Use

There are replayer.bat and replayer.sh files present in the conf Replayer/scripts. Either of these can be used to replay the recorded transaction log. Run replayer.bat or replayer.sh with the path to the log file as argument. These scripts run the Replayer.jar file into which the stand alone Java program for replaying the transactions is bundled.

The data available in the log file is:

- Request objects and their parameters
- Response objects

- Transaction time

Using the Transaction Log as Data Store

The transaction log file can be configured as the data store in lieu of the solution domain configured in Foundation.xml. This allows the support team to avoid having to collect the entire domain data from the customer. This is the equivalent of performing a manual replay of the customer scenario.

In order to do this, the support team must know the exact UI operations performed by the customer during the session. These operations can then be performed in the identical sequence. The data store logic will return the canned responses as long as the request is of the expected type. The request would be saved to a file. This can then be used for later comparison with the original customer actions that led to the reported problems.

If at any point in the manual replay a request does not match the recorded one, an error message is returned to the user. No request can be skipped. The user can continue only if he/she performs a UI operation that results in the expected result.

Working with Oracle Support

Customers may get the most out of these tools by working in cooperation with Oracle support.

Design

There are two mechanisms in the bridge code to execute RPAS transactions.

The most frequently used type involves constructing an instance of some *bridge* classes, populating the *request* fields of the bridge object, passing to *Session.execute()*, and processing the response fields in the bridge object which is modified in place by the *execute()* method.

The other kind of transaction is the proxy kind. A *WorkbookProxy* object from the bridge layer is instantiated and invokes the varying methods required.

For both types of transaction the recording mechanism intercepts the method call and logs the objects.

Data Stored in Log File

The following data is stored in the log file:

Regular Transactions

- ServerRequest prior to executing the transaction (as a Java-serialized object).
- Same ServerRequest. This is similar to the above, but post-execution so it has the response fields.
- The time taken by the transaction.

Proxy Transactions

- Method Name (as Java-serialized String).
- Method (as Java-serialized String).
- Method parameters (as Java-serialized objects of various types).

- Response object (if any), as Java-serialized object. If a response is void, then a "NULL" string is written out in Java-serialized form.
- The time taken by the transaction.

Replay

Replayer reads in and processes all the data stored for each transaction in the log. Apart from printing out to the console text that describes the transaction and the recorded transaction time, the transaction is also executed against a local domain, and the time taken by the replay call is also printed out to the console.

Replaying - Real Time Scenarios

When using the diagnostic tool, the RPAS front ended is not need to reproduce the defect. The debug tool can be used to stop Replayer at various points so that the sustaining engineer can examine the request and response in the debugger. The code also allows diagnosis of whether the defect is server side or client side.

> **Note:** In order to facilitate diagnosis of the problem, the Replayer source code can be changed to skip certain requests. The source code is present in the Replayer jar file.

Once a solution has been found and implemented, the fix can be verified by running Replayer and verifying that no exceptions occur.

Example 1: Percent of parent measure creation error

This is associated with the following error message:

PARENT OF PERCENTAGE MEASURE CREATION ERROR IN MEASURE ANALYSIS WKBK

Steps to Reproduce

This error can be created by the following steps in RPAS:

1. Within the Taskflow, go to Administration>Analysis and open the Measure Analysis workbook wizard.
2. Select a numerical measure; for example Cleansed Demand (OoS).
3. Select any Business Unit Positions.
4. Select any Location Position.
5. Select any Product Positions.
6. Select any Calendar Positions.
7. Open the workbook.
8. Right click on the measure name.
9. Select Create percent-of-parent measure.
10. Click Create.
11. 11.The error should appear *StyleException: ../../Libraries/MeasureStore/rpas/StyleWorkbook.cpp:6104*

Identifying the Error

When this scenario is recorded and replayed in the development or support environment with the customer domain, the transactions are executed in same order. The output will have a log of all the transactions that occurred. This will show the exception and its associated details:

```
In Login method...
Login status : true
regularTxn:UserData (-1):0
regularTxn:GetAccessibleTemplates (-1):0
regularTxn:AlertResults (-1):0
regularTxn:WizardInitialize (-1):0
regularTxn:WizardGetNext (-1):0
regularTxn:WizardGetNext (-1):0
regularTxn:WizardCancel (-1):0
regularTxn:WizardInitialize (-1):0
regularTxn:WizardGetNext (-1):0
regularTxn:GetWizardLayout (-1):0
regularTxn:WizardCancel (-1):0
regularTxn:WizardCancel (-1):0
regularTxn:WizardInitialize (-1):0
regularTxn:WizardGetNext (-1):0
regularTxn:WizardGetNext (-1):0
regularTxn:WizardCancel (-1):0
regularTxn:WizardInitialize (-1):0
regularTxn:WizardGetNext (-1):0
regularTxn:GetWizardLayout (-1):0
regularTxn:SetWizardData (-1):0
regularTxn:WizardGetNext (-1):0
regularTxn:GetWizardLayout (-1):0
regularTxn:LoadAttributeDef (-1):0
proxyTxn:getSavedDynamicAttributeslud2(-1):7
proxyTxn:getSavedDynamicAttributeslud1(-1):8
proxyTxn:getSavedDynamicAttributesdens(-1):3
proxyTxn:getSavedDynamicAttributesstat(-1):4
proxyTxn:getSavedDynamicAttributesstrd(-1):5
proxyTxn:getSavedDynamicAttributesclim(-1):6
proxyTxn:getSavedDynamicAttributesgeog(-1):4
proxyTxn:getSavedDynamicAttributesvol(-1):3
proxyTxn:getSavedDynamicAttributescnc(-1):4
proxyTxn:getSavedDynamicAttributescmp2(-1):4
proxyTxn:getSavedDynamicAttributeschnl(-1):11
proxyTxn:getSavedDynamicAttributescmpy(-1):3
proxyTxn:getSavedDynamicAttributesrgn(-1):3
proxyTxn:getSavedDynamicAttributesdist(-1):3
proxyTxn:getSavedDynamicAttributesstr(-1):13
regularTxn:SetWizardData (-1):0
regularTxn:WizardGetNext (-1):0
regularTxn:GetWizardLayout (-1):0
regularTxn:LoadAttributeDef (-1):0
proxyTxn:getSavedDynamicAttributeskdpt(-1):4
proxyTxn:getSavedDynamicAttributeskcls(-1):3
proxyTxn:getSavedDynamicAttributeskitm(-1):4
proxyTxn:getSavedDynamicAttributesvend(-1):4
proxyTxn:getSavedDynamicAttributesfabr(-1):3
proxyTxn:getSavedDynamicAttributespl2(-1):3
proxyTxn:getSavedDynamicAttributespl1(-1):4
proxyTxn:getSavedDynamicAttributescmpn(-1):6
proxyTxn:getSavedDynamicAttributesdvsn(-1):3
proxyTxn:getSavedDynamicAttributespgrp(-1):4
```

```
proxyTxn:getSavedDynamicAttributesdept(-1):3
proxyTxn:getSavedDynamicAttributesclss(-1):4
proxyTxn:getSavedDynamicAttributesscls(-1):12
proxyTxn:getSavedDynamicAttributesstyl(-1):7
proxyTxn:getSavedDynamicAttributessku(-1):3
regularTxn:SetWizardData (-1):0
regularTxn:WizardGetNext (-1):0
regularTxn:GetWizardLayout (-1):0
regularTxn:LoadAttributeDef (-1):0
proxyTxn:getSavedDynamicAttributeswk2(-1):6
proxyTxn:getSavedDynamicAttributeswk1(-1):4
proxyTxn:getSavedDynamicAttributesevnt(-1):4
proxyTxn:getSavedDynamicAttributesflst(-1):3
proxyTxn:getSavedDynamicAttributeshldy(-1):6
proxyTxn:getSavedDynamicAttributesyear(-1):6
proxyTxn:getSavedDynamicAttributesssn(-1):3
proxyTxn:getSavedDynamicAttributesqrtr(-1):4
proxyTxn:getSavedDynamicAttributesmnth(-1):4
proxyTxn:getSavedDynamicAttributesweek(-1):4
regularTxn:SetWizardData (-1):0
regularTxn:WizardFinish workBookIDt0000000038
(-1):0
regularTxn:WizardCancel (-1):0
regularTxn:OpenWorkbookMega setting workBookID in
ClientOpenWorkbookMega:t0000000038
(-1):0
regularTxn:AllDimSplitDef (-1):0
regularTxn:GetPositionContext (-1):0
regularTxn:GetPositionContext (-1):0
regularTxn:GetPositionContext (-1):0
proxyTxn:isWindowConfiguredstr_cmpnweek_W(-1):6
proxyTxn:isWindowConfiguredstr_scls_W(-1):6
proxyTxn:isWindowConfiguredstr_sclsweek_W(-1):6
proxyTxn:isWindowConfiguredstr_sku_week_W(-1):5
proxyTxn:getVisibleMeasures{}(-1):19
regularTxn:GetAllAlerts (-1):0
regularTxn:GetRTAlertHits (-1):0
regularTxn:GetAccessibleWorkbooks (-1):0
regularTxn:GetWindowData (-1):0
regularTxn:GetAllPositions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
```

```
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:DynamicAttrDefinitions (-1):0
regularTxn:GetAttributeValues (-1):0
regularTxn:GetMultiAttributeValues (-1):0
regularTxn:GetWindowData (-1):0
regularTxn:GetAllPositions (-1):0
regularTxn:GetAttributeValues (-1):0
regularTxn:GetMultiAttributeValues (-1):0
regularTxn:GetWindowData (-1):0
regularTxn:GetAllPositions (-1):0
regularTxn:GetAttributeValues (-1):0
regularTxn:GetMultiAttributeValues (-1):0
regularTxn:GetWindowData (-1):0
regularTxn:GetAllPositions (-1):0
regularTxn:GetAttributeValues (-1):0
regularTxn:GetMultiAttributeValues (-1):0
regularTxn:RuleBasedProtection (-1):0
regularTxn:GetCommitStatus (-1):0
regularTxn:Fetch (-1):0
Exception in thread "main" com.retek.mdap.bridge.ServerException: StyleException:
c:\rpas\svn\14.0.0.0.741\libraries\measurestore\rpas\styleworkbook.cpp:6156
at com.retek.mdap.comm.RequestBroker.readException(RequestBroker.java:382)
at com.retek.mdap.comm.RequestBroker.readObject(RequestBroker.java:357)
at com.retek.mdap.comm.RequestBroker.execute(RequestBroker.java:475)
at com.retek.mdap.comm.Session.execute(Session.java:297)
at txn.ExecTransactionLog.executeRegularTxn(ExecTransactionLog.java:158)
at txn.ExecTransactionLog.main(ExecTransactionLog.java:83)
regularTxn:CreateExtendedMeasure Process exited with exit code 1.
```

From the exception, it can be seen that the server exception is thrown during execution of the *createExtendedMeasure* transaction. This leads to the conclusion that either the parameters passed to the bridge object are incorrect or it is not handled in the right way on the server side. In this case the exception was because the wrong worksheet name was passed and the server was not able to find it. The problem can therefore be addressed on the Fusion Client side.

Example 2: Dimension Exception when Overwriting Level Split on Dimension

This scenario occurs if a user tries to apply a second split on a dimension that already has a split applied. Each dimension can only have one split applied at one time. As a result, the first split is apparently overwritten but this will result in a Refresh Error *dimensionException*.

Steps to Reproduce

This error can be created by the following steps in RPAS:

1. Create a new Aggregation and Spreading Workbook; select multiple subclasses.
2. Show levels SKU and Subclasses in outline mode.
3. Move measure *upbd* to 4-5th place on column axis from last.
4. Enter some value on SKU under *upbd* measure, so that the sku has value 0, 1, 2, 3, 5, 6, 7, 8 in one of the subclasses.
5. Create a new attribute on SKU (for example *upbd-sku*) and select measure *upbd*; choose clustering and number of cluster 3 and then click on auto fill.
6. Create a new attribute on subclass (for example *upbd-subclass*), and select measure *upbd*; choose clustering and number of cluster 3 ad then click on auto fill.
7. Create level split on SKU first and select attribute *upbd-sku* (created in step 5).
8. Create a new level split on Subclass and select the attribute *upbd-subclass* (created in step 6).
9. Select overwrite with new split option and click ok.
10. User will see refresh error *DimensionException: Dimension 'd005' encode failed 15 > 10. Edits are disabled.A temporary method of preventing the error is to hide the problem measure(s) from the worksheet*.

This may take a few iterations reproduce. This is because:

- The steps are complex.
- Many RPAS transactions happening and its difficult to identify the exact step where this exception occurs.

Identifying the Error

The diagnostic tool recording makes it possible to reproduce it easily, and shift focus to fixing the issue.

- The first stage is to check the Replayer log. This shows which transaction caused the exception.
- The next stage is to put in a break point and check the values of the members of transaction object when the failure occurred.
- This enables an investigator to determine if the data sent from the client side is correct or not. If it is valid this shows it is a server side issue.
- The Server team can look at the particular transaction and debug it to arrive at the source of the error.

Example 3: Performance Issues

The final example is that of performance issues.

Problem Statement

The RPAS Fusion Client workbook screen rendering shows performance is excessively slow when there are large quantities of data to return. For this test case there are 70,000 items. The customer reports that it takes several minutes to return the data each time it is refreshed and this is affecting the productivity of their staff.

Problem Resolution

In a test environment, the same workbook opens in a few seconds. The next stage is to determine where the problem lies. Most of the workbook open time is spent in RPAS transactions. The Recorder keeps track of transaction times. By using Replayer it is possible to set out the customers and the test times side by side. For example:

```
proxyTxn:getVisibleMeasures{}(20):19
```

This allows an investigator to determine whether:

- A specific transaction is slow at the customer side.
- Whether every transaction is slow at the customer side.
- Whether the transaction times are similar and the problem lies elsewhere.

This enables the broad nature of the problem to be determined and further investigation to begin.