Oracle® Retail Predictive Application Server

Administration Guide
Release 13.1.1

September 2009

ORACLE

Oracle® Retail Predictive Application Server Administration Guide, Release 13.1.1

Primary Author: Bernadette Goodman

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

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

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

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

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

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

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

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

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

Preface

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

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

Audience

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

Related Documents

For more information, see the following documents in the Oracle Retail Predictive Application Server Release 13.1.1 documentation set:

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

Customer Support

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

https://metalink.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

If you are installing the application for the first time, you install either a base release (for example, 13.0) or a later patch release (for example, 13.0.2). If you are installing a software version other than the base release, be sure to read the documentation for each patch release (since the base release) before you begin installation. Patch documentation can contain critical information related to the base release and code changes that have been made since the base release.

Oracle Retail Documentation on the Oracle Technology Network

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

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

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

Conventions

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

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

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

A hyperlink appears like this.

1

Introduction

Overview

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

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

System Administration Workbooks

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

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

Workbook and Wizard Descriptions

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

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

General Workbook Procedures

Change a Workbook's Calculation Method

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

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

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

Set the Workbook to Deferred Calculation Mode

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

Return the Workbook to Automatic Calculation Mode

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

Send the Queue of Data Changes to the Server

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

Refresh and Export Data

Refresh the Data in a Worksheet

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

Export the Current Worksheet View to an Output File

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

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

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

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

Insert Measures into an Open Worksheet

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

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

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

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

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

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

Global Domain

Overview

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

Domains can be built in one of two methods:

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

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

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

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

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

Measure Data

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

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

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

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

Consider a global domain environment where the partition-level is based on the Department dimension in the Product hierarchy. Data for measures that have a base intersection in the Product hierarchy at or below Department are stored in the local domain based on the Department to which the underlying position in the Product hierarchy belongs. Other hierarchies are irrelevant for this discussion.

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

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

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

2

Domain Administration

DomainDaemon

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

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

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

Usage

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

Argument	Description
`-version`	Prints the RPAS version, revision, and build information of the utility.
`-start`	Starts a DomainDaemon on the specified port.
`-port` *portNum*	The *portNum* must be between 1025 and 65535 (inclusive). If *portNum* is set to `auto`, it will find any free port.
`-loglevel`	This option enables additional logging. **Note**: The `-debug` option has been deprecated. Instead of using `-debug`, please 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.

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

Starting the DomainDaemon

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

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

Example:

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

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

Monitoring the DomainDaemon

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

Example:

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

Stopping the DomainDaemon

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

Example:

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

Losing a Client-Server Connection

The connection between the RPAS Client and the RPAS Server can be lost for any number of reasons, but 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 choose to gracefully terminate the existing session and log in to a new session. Graceful termination includes the completion of any pending processes and custom menu or rule group processing in the existing session, followed with an auto-save of the workbook and subsequent termination of the server process.

The number of logins of a user to a particular domain is limited to five. If for any reason the connection is lost again before the new login occurs, RPAS will still continue to gracefully terminate the previous session. However, it won't start up another session for the user until the user regains connectivity and tries to log in again.

Graceful termination of an existing session could take an arbitrarily long time. This time is equal to the sum of the time taken to complete any running calculations and then 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.

Autosave of 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 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 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.

Environment Variables

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

The common syntax for setting these variables is as follows:

```
Export ENVIRONMENT_VARIABLE=XXXXXX
```

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

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

Lock Timeout Variable

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

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

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

```
Export RPAS_LOCK_TIMEOUT=120000
```

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

The environmental variable `RPAS_REQUEST_TIMEOUT` should be set to a value that is the number of seconds of idle time 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.

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

Log File Backups

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

The environment is set by executing the following command:

```
Export RPAS_LOG_BACKUPS=X
```

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

Profiling Logging

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

- `RPAS_PROFILING_ENABLE`

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

- `RPAS_PROFILING_PATH`

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

Date and Time Setting

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.
- 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.

Database Settings

These variables are used for RPAS B-tree storage performance.

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 unavailable, RPAS uses a page split percentage of 90.

RPAS_PAGE_SIZE

This variable sets the page size for the RPAS B-Tree. Valid values are 4K, 8K, 16K, 32K, and 64K. RPAS automatically uses a page size of 2K for arrays where the logical size of the dimension space is less than 1000 cells, and a page size of 4K for arrays where the logical size of the dimension space is between 1000 and 200,000 cells. For all other dimension space sizes, RPAS uses a page size of 8K or the value of this environment variable.

Since RPAS caches page data during all its processes, it is recommended that large page sizes are used when dimension spaces are large and when it is known that most processing happens in an unraveling order of positions. This is always the case when processing is driven by the innermost dimension. In most RPAS domains, dimensions of the Calendar hierarchy are the innermost dimension for most arrays.

Centralized Administration

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

Administrative Workbook Templates and Wizards

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

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

RPAS Utilities

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

Domain Upgrade

Upgrade a Simple or Global Domain Environment

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

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

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

Upgrade Process Overview

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

1. Acquire the new version of the platform, which includes the following key components:
   - RPAS Server
   - RPAS Client
   - RPAS Configuration Tools
2. Create a backup copy of any domains that will be upgraded.
3. Commit any existing workbooks that have not yet been committed (if commit is required).

   RPAS does not guarantee the upgrade of existing saved workbooks or styles as part of the upgrade process from one major release or patch to the next. If users are unable to build workbooks after upgrading a domain, system administrators must delete all the existing style and user/selections folders to correct the problem.
4. Remove from the auto-workbook queue (`wbbatch` utility) any `-commit` and `-refresh` scheduled batch jobs. If upgrading an 11.0.x environment to 12.1.x (or later), all `-build` scheduled batch jobs should also be removed from the queue.

   > **Note:** Skip this step if upgrading a 12.1 or newer environment.
5. Run the `convertDomain` utility on existing domains. In global domain environments this utility is only executed from the master domain.

   > **Note:** Skip this step if upgrading a 12.1 or newer environment.
6. Run the `upgradeDomain` utility on existing domains. In global domain environments this utility is only executed from the master domain.

7. Upgrade existing configurations (see the following section, "Upgrade Configurations" for more information on this step).

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

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

   Note: When using the `-updatestyles` flag, deleting existing styles is not necessary.

convertDomain

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

Usage

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

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

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

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

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

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

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

upgradeDomain

Usage

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

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

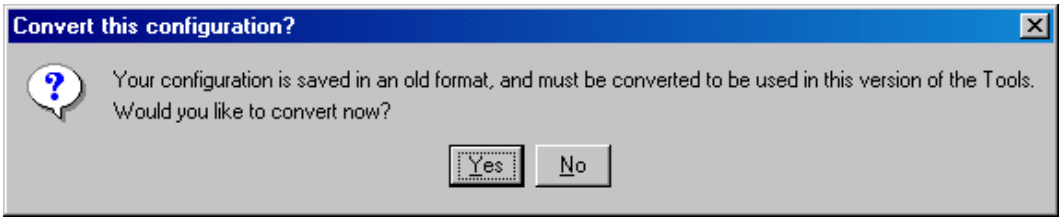
Argument	Description
`-d` *path*	Path to the domain being upgraded.
`-verbose`	Optional parameter to show the detail about each change that is applied to the domain.
`-n`	Optional parameter to report which changes would be applied without applying the changes.
`-purgeWorkbooks`	Required parameter that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

> **Note:** An administrator can also run the Domain Information utility to verify the upgrade process as shown below.
>
> ```
> domaininfo -d pathtodomain -domainversion
> ```

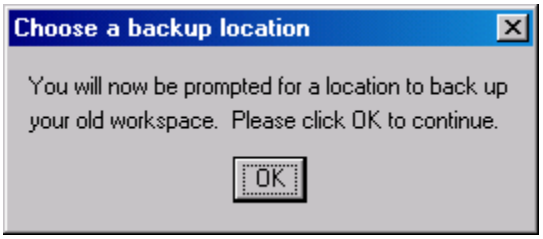
Upgrade Configurations

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

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

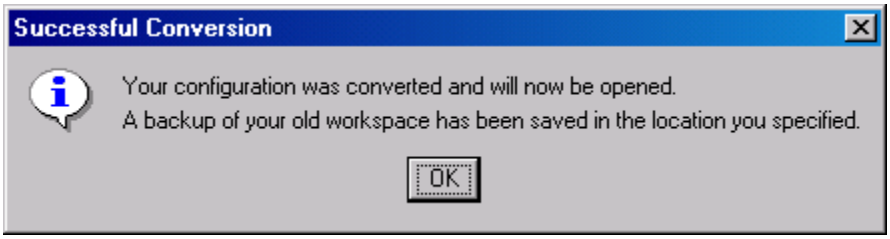


2. Click **Yes**. The Backup Location message box appears. Click **OK** to continue.



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

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



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

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

Command-line Support for Configuration Upgrading

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

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

Upgrade Application Configurations

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

rpasinstall

Usage

```
rpasInstall <arguments>
```

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

Argument	Description
`-fullinstall`	Required argument.
`-patchinstall`	Required argument.
`-testinstall`	Required argument.
`-ch` *`config_home`*	Required argument. Path to the directory containing the configuration file.
`-cn` *`config_name`*	Required argument. The name of the configuration.
`-in` *`input_home`*	Required argument. Path that includes the directory containing the input files for the domain to be created.

Argument	Description
-log *log_name*	Required argument. Path that includes the name of the log file to be created or updated.
-configdir *config_directory*	The path to the directory containing the xml files used by RPAS. This is a required argument if the user wants to supply globaldomainconfig.xml or calendar.xml.
-dh *domain_home*	The path to directory in which the domain will be created. Use if and only if a globaldomainconfig.xml is not used.
-p *dim_name*	The partitioning dimension. Use if and only if global domain is being implemented without the use of globaldomainconfig.xml.
-rf *function_name*	The filename of the function to be registered. This pairing may be repeated for multiple functions. This argument is not required if there are no functions to be registered.
-updatestyles	This argument will enable the update of styles based on the styles set in the Configuration Tools.

3

Security and User Administration

Functional Overview

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

This chapter also describes user administration and security administration.

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

User Logon Security

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

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

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

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

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

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

Account Lockout may be enabled for a domain.

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

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

Password expiration may be enabled for a domain.

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

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

Password history may be enabled for a domain.

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

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

Measure Level Security

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

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

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

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

Position Level Security

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

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

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

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

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

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

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

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

Workbook Security

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

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

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

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

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

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

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

User Administration

Overview

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

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

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

Access the User Administration Tab

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

Add a User

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

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

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

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

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

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

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

Add a User Group

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

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

Delete a User

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

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

Delete a User Group

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

> **Note:** 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.

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

Edit a User

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

Security Administration Workbook

Overview

The Security Administration workbook is only available to system administrators. After users and user groups are created, the administrator may set up and maintain access permissions to workbook templates and measures within those workbook templates. This workbook allows the administrator to determine which templates individual users can access, as well as the measures that users can access while manipulating workbooks in the system. 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 one or more worksheets for each of the following:

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

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

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

Workbook Template Rights Worksheet

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

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

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

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

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

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

Workbook Template Measure Rights Worksheet

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

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

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

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

Measure Rights Worksheet

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

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

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

Selecting a security option does not affect any measures already included in the workbook. The selected security option is used only while inserting the measure into an already built and open workbook.

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

Dimension Modification Rights Worksheet

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

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

Position Level Security Worksheets

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

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

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

Workbook Template Limits Worksheets

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

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

Max Domain Session Limit Worksheet

The Max Domain Session Limit worksheet 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 due to this limit being reached.

Max User Session Limit Worksheet

The Max User Session Limit worksheet 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 admin 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 due to this limit being reached.

Using the Security Administration Workbook

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

Access Security Administration

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

Set or Modify Users' Access to Workbook Templates

1. From the File menu, select **New**.
2. Click the **Administration** tab.
3. Select **Security Administration**.
4. Click **OK**.
5. On the Workbook Template Rights worksheet, select each template for which a user's access rights require modification. Set to Denied, Read-only or Full Access.

6. Changes must be committed to the master database before they take effect. To commit the changes, select **Commit Now** from the File menu.
7. Save the workbook by selecting **Save** from the File menu, if desired.
8. To close the workbook, select **Close** from the File menu.

Set Measure Availability for Workbook Templates

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

Assign or Restrict User Access to Measures

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

Change a User's Ability to Modify Dimensions

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

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

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

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

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

Limit the Number of Workbooks that a User Can Save

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

Set or Modify the Maximum Domain Session Limit

1. From the File menu, select **New**.
2. Select the **Administration** tab.
3. Select **Security Administration**.
4. Click **OK**.
5. On the Max Domain Session Limit worksheet, modify the scalar measure Maximum Domain Session Limit value with a valid integer value.
6. Changes must be committed to the master database before they take effect. To commit the changes, select **Commit Now** from the File menu.
7. To save the workbook, select **Save** from the File menu.
8. To close the workbook, select **Close** from the File menu.

Set or Modify the Maximum User Session Limit

1. From the File menu, select **New**.
2. Select the **Administration** tab.
3. Select **Security Administration**.
4. Click **OK**.
5. On the Max User Session Limit worksheet, modify the Maximum User Session Limit measure per user.
6. Changes must be committed to the master database before they take effect. To commit the changes, select **Commit Now** from the File menu.
7. To save the workbook, select **Save** from the File menu.
8. To close the workbook, select **Close** from the File menu.

4

Hierarchy Maintenance

Overview

Hierarchy Maintenance Workbook

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

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

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

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

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

Hierarchy Maintenance Example

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

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

Hierarchy Maintenance Wizard

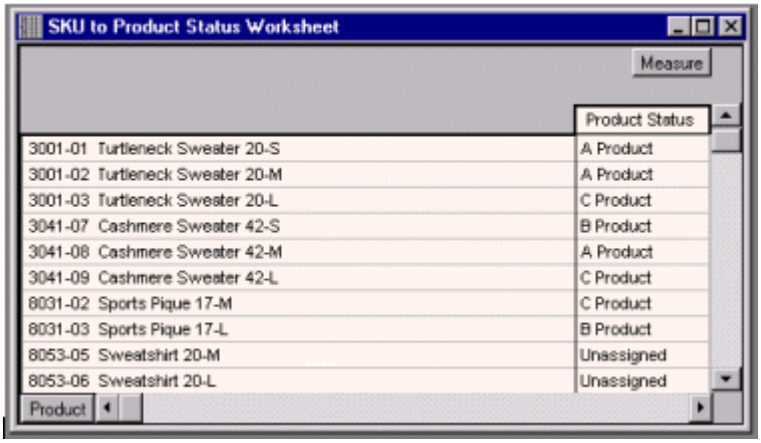
The first step in maintaining hierarchies is to access the Hierarchy Maintenance Wizard. In this wizard, select the SKUs that will be mapped to the various positions of the user-defined dimension. Responses to prompts in the wizard are used to format a new Hierarchy Maintenance workbook.

Hierarchy Maintenance Worksheet

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

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

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

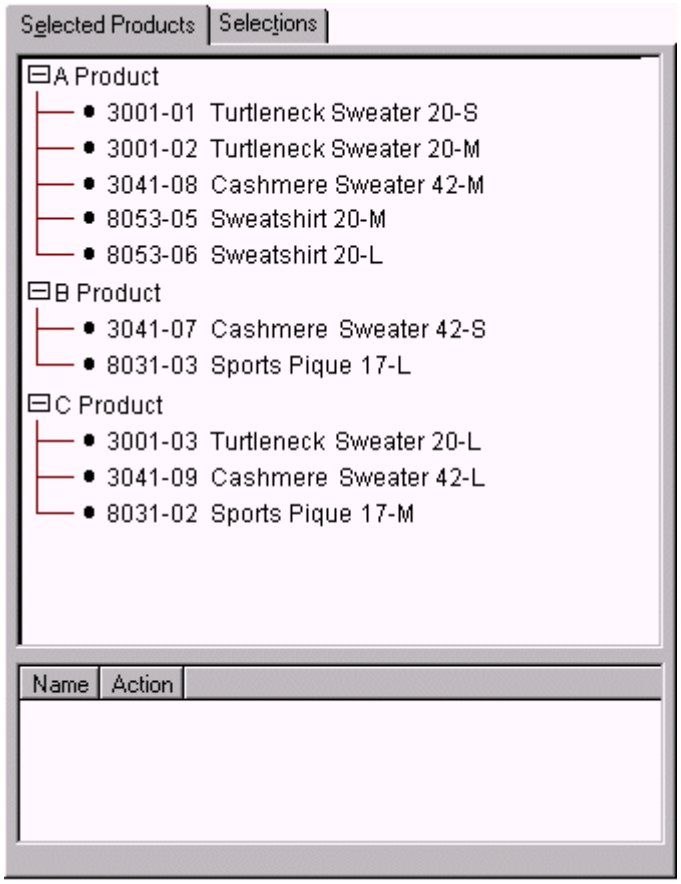


Note: The Oracle Retail Predictive Solutions system is case-sensitive when a new position name (label) is entered in the Hierarchy Maintenance workbook. After the workbook is committed, the typing of the group name is not case-sensitive. For example, "B Product" can later be entered as "b product" after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

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

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

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



Access Hierarchy Maintenance

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

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

Maintain a User-Defined Dimension within a Hierarchy

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

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

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

5

Measure Analysis

Overview

Measure Analysis Workbook

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

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

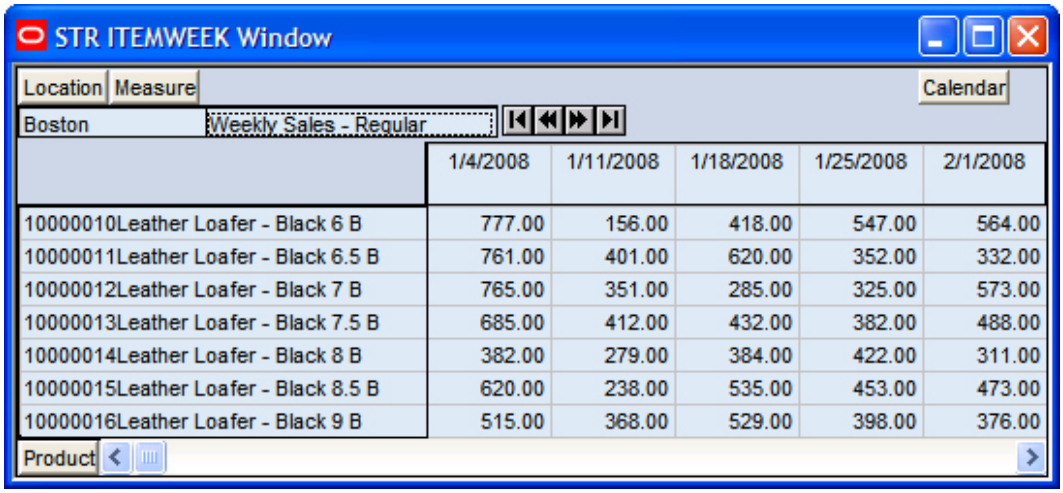
> **Note:** Due to its dynamic nature, formatting settings cannot be saved in the Measure Analysis workbook.

Measure Analysis Wizard

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

Measure Analysis Worksheet

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



	1/4/2008	1/11/2008	1/18/2008	1/25/2008	2/1/2008
10000010Leather Loafer - Black 6 B	777.00	156.00	418.00	547.00	564.00
10000011Leather Loafer - Black 6.5 B	761.00	401.00	620.00	352.00	332.00
10000012Leather Loafer - Black 7 B	765.00	351.00	285.00	325.00	573.00
10000013Leather Loafer - Black 7.5 B	685.00	412.00	432.00	382.00	488.00
10000014Leather Loafer - Black 8 B	382.00	279.00	384.00	422.00	311.00
10000015Leather Loafer - Black 8.5 B	620.00	238.00	535.00	453.00	473.00
10000016Leather Loafer - Black 9 B	515.00	368.00	529.00	398.00	376.00

Example of Measure Analysis Worksheet

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

Access Measure Analysis

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

Review and Edit Sales or Other Registered Measure Data

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

   OR

   To open a new workbook, select **New** from the File menu.
2. On the Analysis tab, select **Measure Analysis**.
3. Click **OK**.
4. The Measure Analysis Wizard opens and prompts the user to select the measures to be displayed in the new workbook. Use Ctrl-Click and/or Shift-Click to select multiple measures.
5. Click **Next**.

6. For each hierarchy specified in the base intersection of the measure(s) selected, the user will see a hierarchy wizard from which to select positions to view. Repeat this step for each hierarchy wizard.
7. Click **Next**.
8. Click **Finish** to open the Measure Analysis workbook.
9. On the Measure Analysis Worksheet(s), view the stored data associated with the measures and hierarchy positions selected in the wizard. Make any changes as required. Administrative users may commit changes.

6

Workbook Auto Build Maintenance

Overview

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

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

Workbook Auto Build Maintenance Wizard

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

Accessing the Auto Build Maintenance Workbook

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

Add a Workbook to the Auto Build Queue

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

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

Delete a Workbook from the Auto Build Queue

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

7

Internationalization

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

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

Translation

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

1. Graphical user interface (GUI)
2. Error messages

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
- Demo data
- Training Materials

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

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

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

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

   The **foundation.ini** file is found in one of the following places. When RPAS searches for the file, it searches in the following order:

   a. Look in the user's 'Application Data' folder:
      - For Microsoft Windows XP: C:\Documents and Settings\<*user*>\Application Data\Oracle Predictive Solutions
      - For Microsoft Windows Vista: C:\Users\<*user*>\AppData\Local\Oracle Predictive Solutions

   b. Look in the same folder as the currently loaded **foundation.exe** file.

   c. Look in the current working folder, which can be one of the following:
      - The debugger's setting, if started by debugger
      - The same folder as the **foundation.exe** file, if started manually
      - ./winnt/system32, if started from automation

   d. C:\Windows

      If the file is not found, it is created at the location in step a.

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

Language	Code
Brazilian Portuguese	22
English	9
French	12
German	7
Italian	16

Language	Code
Japanese	17
Korean	18
Russian	25
Simplified Chinese	2052
Spanish	10
Traditional Chinese	1028

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

   Note: The files will be removed from your 'input' folder when they are loaded. So, any file that is left in this directory still needs to be loaded.

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

   Note: The translated .ovr files are released in UTF-8 encoding. Any changes to these files should be made with a UTF-8 capable editor and saved without BOM (Byte Order Mark).

   The .ovr files consist of three data columns, as shown below.

   ```
   Gub_mean_                SPANISH              Media máxima
   ```

   The data is populated on a specific column and has definite width. When making any change to this data, you must ensure that the new data abides by the predefined width and position for that data for the file.

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

Translation Administration Workbook

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

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

Hierarchy Labels Worksheet

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

Dimension Labels Worksheet

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

Workbook Template Group Labels Worksheet

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

Workbook Template Labels Worksheet

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

Measure Labels Worksheet

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

Measure Descriptions Worksheet

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

User Group Labels Worksheet

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

Message Labels Worksheet

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

RGRP Labels Worksheet

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

8

Commit as Soon as Possible

Overview

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

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

Using Commit ASAP

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

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

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

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

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

> **Note:** If a user attempts to commit a workbook ASAP that already has a process in the queue, the original processes will be removed from the queue. That means that there can only ever be one pending commit ASAP in the queue for a given workbook/user/template name combination.
>
> **Note:** Workbooks must have been saved at least once before attempting a Commit ASAP. A workbook has not been saved if the label says "untitled."

Managing the Workbook Queue – showWorkbookQueues

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

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

Usage

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

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

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

Commit ASAP Settings – configCommitAsap

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

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

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

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

Usage

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

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

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

Logging and Technical Information

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

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

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

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

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

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

9

Batch Processes and RPAS Utilities

Overview

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

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

RPAS Utilities Logging Options

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

Log Levels

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

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

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

- E indicates 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 gets recorded in rpas.log under each user's working directory. Information related to domain activities, such as user sign-on and sign-off, gets recorded in DomainDaemon.log.

Utilities with Standard Logging

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

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

Scripts

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

convertDomain

All output to the screen.

createRpasDomain

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

Utilities with Special Logging

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

DomainDaemon

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

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

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

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

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

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

RpasDbServer

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

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

loadHier

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

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

If there are any problems 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-partition hierarchies are reported as shown in the following:

<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.

positionBufferMgr

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

reconfigGlobalDomainPartitions

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

renamePositions

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

regmeasureServer

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

Utilities with Multi-Process Logging

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

These utilities are as follows:

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

domainprop

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

hierarchyMgr

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

configCommitAsap

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

Using Shell Scripts to Run Batch Processes

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

A Sample Shell Script

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

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

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

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

Common Information and Parameters for RPAS Utilities

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

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

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

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

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

Each 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** sub-directory of the Tools directory. If a problem with the configuration tools is encountered, send these two files to Oracle Retail Customer Care along with a description of the problem.

10

Hierarchy Management

Overview

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

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

Placeholder Positions in the Domain

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

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

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

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

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

Position Repartitioning

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

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

Loading RDF and Curve Parameters after Repartitioning

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

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

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

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

Syntax

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

Example:

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

Enabling Dummy Positions

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

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

Configuring and Scheduling the Rebuffering Process

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

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

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

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

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

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

Loading Hierarchies – loadHier

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

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

Example of `loadHier` using a CSV File:

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

`loadHier` allows the use of a simple compression technique to reduce the size of the input file and to reduce the file I/O time. CSV input files can be prepared to substitute a column's value with a '?' character if it matches the value from the same column in the previous row. The '?' character can be used for multiple consecutive rows to reuse the value from the row preceding the first use of '?'. Note that `loadHier` assumes that the '?' character will only be used to take advantage of RPAS' proprietary CSV compression. Therefore if a single '?' is meant to be the value of a column, it should be enclosed between opening and closing double quotes; e.g., "?".

Due to the compression algorithms used in the CSV format, the `–nosort` argument is automatically used when CSV is detected. This argument is used to prevent the reordering of rows so that RPAS will not lose track of which column value is to be cached for a subsequent row. If the order of records is important, CSV files should be sorted before passing them to `loadHier`.

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

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

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

RPAS allows for multiple input files to be loaded for the same hierarchy. The extra input files should be named with a secondary extension (for example, ' msgs.dat.1'). The extra input files can be loaded only with the main input file. For example, you cannot load 'msgs.dat.1' in a separate `loadHier` call. 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.

Position Label Translation

To enable translation of position labels for the desired dimension(s) 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 that will be 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 will be named `r_skulabel.csv.ovr` and will have 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,白色耐克跑鞋大小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
```

The following table lists the supported languages and their identifiers.

Language ID	Language
CHINESE_SIMPLIF	Simplified Chinese
CHINESE_TRADITI	Traditional Chinese
ENGLISH	English
FRENCH	French
GERMAN	German
HUNGARIAN	Hungarian
ITALIAN	Italian
JAPANESE	Japanese
KOREAN	Korean
PORTUGUESE	Portuguese
RUSSIAN	Russian

Language ID	Language
SPANISH	Spanish

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. Once committed, these translations are available for every new workbook.

It is possible that due to errors when preparing translation files, 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.

Usage

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

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

Argument	Description
`–d domainPath`	The domain in which to load the hierarchy data.
`–load hierName`	The name of the hierarchy to load and refresh.
`–loadAll`	Loads all hierarchy input files (with a .dat file extension) that are located in the input directory of the domain. Including this argument disables the reshaping process until all files have been loaded.
`–checkParents`	Verifies the one-to-one mapping of a child position being loaded to its parent position.
`–purgeAge` *purgeage*	Specifies the `purgeage` during `loadHier`.If not specified, `loadHier` gets `purgeage` from domain. In Global Domains, `-purgeAge` supports the purge of partition positions when the *purgeage* is reached.
`–noClean`	This is a switch that prevents the removal of input files and temporary data files that are generated during the hierarchy load process. Input files will remain in the “input” directory of the domain after the process is completed. This option is often used for debugging or troubleshooting purposes.

Argument	Description
-noSort	Use this argument for instances when the RPAS process of sorting the hierarchy file prior to loading the values is not needed due to preprocessing (external to RPAS). This flag will prevent the input <hier.dat> file from being sorted while loading the hierarchy. This argument is often needed when loading calendar hierarchy data. This option can be eliminated if an alphabetic sort arranges calendar data in chronological order. Due to the compression algorithms used in the CSV file format, the -noSort option will automatically be used when CSV is detected. It is recommended that you presort the data before exporting it into a CSV format.
-logLoadedPositions	This argument will enable the logging of successfully loaded input file lines into a "loaded[HIERNAME].dat" file under the processed directory.
-maxProcesses *count*	If specified, some parts of loadHier will run in parallel, meaning that it will use a maximum of the defined processes, which are specified by *count*.
-forceInputRollups	This argument enforces new hierarchy roll-up changes. New roll-up changes will override or dominate existing hierarchy roll-ups in the event they conflict with the rollups specified in the input file. This allows you to load a hierarchy file that reclassifies one or more upper level positions while removing one or more discontinued base-level positions that roll-up to the reclassified position.
-forceNAConsistency	Use this argument to force NA consistency when the current NA value is different from the originally defined NA value for the measure.
-defaultDomain ldom#, ldom#,	Use to specify comma separated default domain paths that will be used for accommodating new partitions. The domain paths can point to existing local domains or to new (non-existing) local domain. The local domain names are specified by a fully qualified path. To specify more than one local domain, separate local domain paths with a comma. **Example:** loadHier -defaultDomain ldom1, ldom2, ldom3
-returnReshapeFailure	Returns a non-zero value if one or more arrays fail to reshape. The default behavior is to log a warning with the array name and continue. The loadHier process still runs to completion even if a non-zero value is returned.

Notes

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

Example:

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

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

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

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

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

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

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

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

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

FilterHier utility

Sometimes, a retailer has a master file of hierarchy data that needs to be loaded into multiple domains. Some of these domains may be missing one or more levels from the master hierarchy, mostly because the planning levels in these domains are higher than the lowest level in the master and the domains don't 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 would have their relevant hierarchy load data in one master file, but using loadhier, the retailer would not be able to load just what is relevant to the domain from the master. System integrators would 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 ridding them of the need to write 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 will be a .csv.dat file that can be subsequently used by the loadHier utility.

Usage

`filterHier -d domainPath -input inputPath [COMMAND] {OPTIONS}`

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

Argument	Description
`-d domainPath`	The domain in which to load the hierarchy data.
`-input inputpath`	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.
`-compress`	Creates a compressed .csv.dat output file. RPAS has introduced 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 will combine one or more input files into an output file that can be imported into the target domain using loadHier. The input files must be csv data containing a comma separated header line listing the name of each column. Column names should 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 will be arranged to match the hierarchy format of the target domain. Any dimensions from the input files that are not present in the output domain will be filtered out.

The output file will be a properly formatted .csv.dat file, and will be 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.

Reconfiguring the Partitions of a Global Domain – reconfigGlobalDomainPartitions

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

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

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

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

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

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

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

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

Usage

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

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

Argument	Description
`-d` *pathToMasterDomain*	Specifies the path to the master domain in a Global Domain environment.
`-add` *posName1, posName2, …*	Adds one or more positions at the partition level to a specified local domain. The path to the local domain must follow the list of positions to add, using the `-sub` argument. If the specified path is to a local domain that does not yet exist, the system will create a new local domain with the specified positions at the partition level. This argument cannot be used with `-remove` or `-input`.
`-remove` *posName1, posName2, ...*	Removes the designated positions from the local domain to which the positions belong. The path to the local domain does not need to be specified with this argument. The local domain will be deleted if all the positions at the partition level in a local domain are removed. This argument cannot be used with `-add` or `-input`.
`-move` *posName1, posName2, ...*	Moves the specified positions at the partition level from the current domain in which the positions are located to the specified local domain. This argument requires specification of the `-sub` argument. To move positions, all dimensions below the partition level must be enabled for dummy positions.
`-sub` *pathToSubDomain*	Specifies the path to the local domain to which positions are being added or the destination local domain for positions being moved. When a new domain path is specified using `-sub` option, a new local domain will be created. This argument is required for the `-add` argument and `-move` argument.

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

Using an Input File

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

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

Below is the required format of the input file:

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

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

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

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

Notes, Assumptions, and Limitations

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

Updating or Reporting the Dummy Position Buffer – positionBufferMgr

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

Usage

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

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

Argument	Description
`-d` *pathToDomain*	Specifies the path to the domain.
`-hier` *hiername*	The hierarchy that will be rebuffered. If no hierarchy is specified, the utility will rebuffer or report on all hierarchies configured to have dummy positions other than calendar.
`-partitionPositions` *"pos1, pos2, …"*	By specifying positions at the partition level, the utility will only rebuffer local domains to which the specified positions belong. Positions specified that are not at the partition level will be ignored.
`-rebuffer`	Adjust all dimensions of the specified hierarchy to have the number of dummy positions based on the configuration of the dimension for the environment. If a hierarchy is not specified with this argument, the utility will run on all hierarchies other than calendar configured to have dummy positions.
`-report`	Report the number of buffer positions for all dimensions of the provided hierarchy. The minimum buffer size is also displayed if applicable. If a hierarchy is not specified with this argument the utility will run on all hierarchies configured to have dummy positions other than calendar.
`-maxProcesses` *max*	If specified, some parts of `positionBufferMgr` will run in parallel, utilizing up to the given number of processes.

Notes

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

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

Example of `positionBufferMgr -report`:

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

Renaming Positions – renamePositions

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

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

Usage

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

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

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

Notes:

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

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

Input File

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

Old position names must be an existing position name.

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

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

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

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

Setting Properties for Dimensions – dimensionMgr

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

Usage

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

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

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

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

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

Notes

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

Multiple command arguments are allowed.

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

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

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

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

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

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

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

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

Deleting a DPM Position

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

Setting Informal Positions to Formal – updateDpmPositionStatus

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

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

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

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

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

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

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

Usage

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

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

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

Exporting Hierarchy Data – exportHier

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

Usage

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

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

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

Note

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

Managing Position Lists as PQDs – pqdMgr

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>
```

Notes

- Input files are vailidated 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 specifying an access level of user, 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>*

Usage

```
pdqMgr -d domainPath -load xmlFile
pdqMgr -d domainPath -delete xmlFile
pdqMgr -d domainPath -deleteAll
pdqMgr -d domainPath -validate xmlFile
pdqMgr -d domainPath -export outFile [-user userName|-world]
```

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

Argument	Description
`-d` *domainpath*	Specifies the path to the domain.
`-load` *xmlFile*	Use this argument to load position lists from an input XML file. Position lists with an operation attribute of "delete" are ignored.
`-delete` *xmlFile*	Use this argument to delete 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`	Use this argument to delete all PQD lists from the domain.
`-validate` *xmlFile*	Use this argument to validate the XML file and report any errors. No impact on the existing PQD files in the domain.
`-export` *outFile* `[-user userName|-world]`	Use this argument to export 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.

11

Data Management

Loading Measure Data – loadmeasure

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

`loadmeasure` 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.

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 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 overwrite, 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; e.g., "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.
- .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 will be switched to NA.
- .inc – (Increment) Increment mode should only be used with numeric measures where 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, the new value for the cell would be 5 (2 incremented with 3).

- .clr – (Clear) Clear mode is a variation of replace mode. It is meant to be used when measure data is loaded in parts or staggered in time, such 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.

  For example, assume that there are four regions, each with several stores, and we load the data region by region or for a subset of regions at a time. When loading data, we want to 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, but 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 would be 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.

`loadmeasure` 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.

Note: `loadmeasure` does not guarantee any specific ordering of loads based on the appended extensions.

`loadmeasure` also allows multiple types of load files to be present in the input directory at the same time. RPAS loads .rpl files first, then .ovr, .clr and .inc files. If fixed width and CSV files are present, then fixed width files of a particular load type are loaded before CSV files of that type; all files of one load type are loaded before moving on to the next load type. Since .rpl files would 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

`loadmeasure` allows multiple measures to be loaded from a single file.

Note: Please refer to the Data Interface Tool section of the *RPAS Configuration Tools User Guide* for more information.

If a CSV file is used for loading multiple measures, `loadmeasure` uses 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.

Loading Data from Below the Base Intersection of the Measures

`loadmeasure` supports loading measure data from an intersection lower 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. Please 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 fault in 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 substitutes 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 loading multiple measures in a single call, only the preprocessing script for the first listed measure will have 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 N/A 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 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 will load one or more input files that can contain data from one or all of the local domains within the given global domain environment. The utility will then split the input files and load them into the required domain, which is the local domain to which the position belongs, or the master domain if the measure has a base intersection above the partition level. The split will only occur once in the case of multiple measures. Local domains will be checked for files even if there is no file in the global domain. The utility can be run in parallel in a global domain environment.

Usage

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

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

Argument	Description
–d *pathToDomain*	Specifies the domain in which to load the measure.
–measure *measureNames*	Specifies the name of the measure(s) to load. Measure names must be lowercase (for example, measurename1, measurename2, measurename3). If more than one measure is specified, all the measures must be in the same input file.

Argument	Description
-applyloads	Use this argument to apply 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 will apply all loads that have been queued up in the staging area. It will clear out the staged loads unless the measure's loadsToKeep property has been set to a non-zero number. In that case, it will not clear out the latest loadsToKeep loads. Note that only .ovr, .inc, and .clr loads can be staged. .rpl loads cannot be staged. Additionally, staging is only allowed for base intersection loads. RPAS cannot stage loads where 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 loadmeasure processes to run concurrently across the local domains in a global domain environment. If this argument is omitted or if less than two processes are specified, the application will do all processing in a single process and no child processes will be created. This only specifies the number of child processes and the controlling process is not included (*max* + 1 is the actual number of processes).
-noClean	Use this option to prevent the input files from being moved to the processed directory. This option is meant for use when a single file is used to load multiple measures, but not all measures from the file are loaded at once. The use of this option instructs loadmeasure to leave the load file behind for subsequent loading of unloaded measures. The user might want to use this option to perform intermediate processing between loads of measures available from the same file.

Argument	Description
–forcePurge	Force 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 N/A 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 5, at data load time all data that is more than 5 days before RPAS_TODAY will be purged. This option does not require you to load any new data.
–splitonly	This argument 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 specifically useful if a multi-measure file is being used such 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.
–noSplit	Use this argument to load the pre-split input files (created by –splitOnly) into the local domains. This option should only be used in global domain environments.
–defrag	This argument can be used to defragment the domain at the end of the measure loading process to reduce the physical size of the domain. This space-saving is achieved by replacing the existing fragmented pages with copied, fully populated BTree database pages.
–loglevel *level*	Use this argument to set the logger verbosity level. Possible values: all, profile, audit, information, warning, error, or none.

Argument	Description
-recordLogLevel *level*	This argument is used to set 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 with the use of 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.

Loading Image Paths for Positions

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

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

Example

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

<domain path> is the path to the domain.

Exporting Measure Data – exportMeasure

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

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.

Usage

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

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

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

Argument	Description
-meas "*measSpec, measSpec …*"	Must specify one. measSpec is measName.modifier. The -meas argument may be repeated to export multiple measure arrays to the same output file. modifier include the following: ▪ .precision<double>, specify the precision for numeric measure ▪ .format<formatString>, specify the user defined export format The examples below provide valid measure specifications given MeasNameA is a valid real type measure. **Examples:** -meas MeasNameA -meas MeasNameA.precision(0.0001) -meas MeasNameA.format("%13.2f").precision(0.01) -meas MeasNameA.precision(0.01).format("%13.2f") 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 %b – 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")
-wb *wbname*	If specified, exportMeasure exports data from the specified workbook (wbname). A valid workbook name must be used.
-intx *intxString*	Specifies the export intersection. Measures will be exported at this intersection. If measure's baseint is higher than export intx, measure's defaff method will be used for aggregation. If measure's baseintx is lower than export intersection, Replication will be used to spread measure down to export intersection.
-mask *measureName*	Specifies a mask measure which must be a valid Boolean measure registered. In current measure store, its baseintx must be at same export intx.

Argument	Description
-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 current output file. If not specified, current output file will be erased and replaced with new data.
-nomerge	If run in a global domain environment and exporting intersection below partition dimension, and have processes set greater than 1, specifying nomerge will stop exportMeasure from merging multiple output files created from each local domain to the master output file. Output files created from local domain will be stored at masterdomain/output/exportMeasure[TS] folder, where [TS] represents a timestamp. Files are named as out000X.txt, where 000X is the index of the local domain.
-compress	Specifies the output file should be in the compressed CSV format.

Exporting Measure Data – exportData

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

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

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

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

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

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

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

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

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

Usage

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

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

Argument	Description
`-d` *domainPath*	Specifies the domain that contains the data that to export.
`-out` *outputFile*	Specifies the file that will contain the exported data. The *outputFile* is relative to the domain unless the full path is specified.

Argument	Description
`–meas \"measSpec\"`	Specifies the measure(s) to export. *measSpec* must be quoted, and the format is `\"measName cellFormat naValue naFormat\"` The `–meas` argument can be repeated to export multiple measure arrays to the same output file. Measures are exported at the base intersection.
`–array \"arraySpec\"`	Specifies the array to export. *arraySpec* must be quoted, and the format is `\"dbName|arrayName cellFormat naValue naFormat\"` ▪ `dbName` can be a path to the database (relative paths are relative to the domain root). ▪ Both `cellFormat` and `naFormat` use `printf` format commands. See the `printf` function for more information on the possible values. The `–array` argument can be repeated to export multiple arrays to the same output file. The order in which arrays are listed is the order in which they will be exported. **Note:** This argument cannot be used in a Global Domain environment and can only be used in simple domains. This argument cannot be used with `–useLoadFormat`.
`–params db array`	Instead of specifying all parameters on the command line, this parameter allows the parameters to be read from an array. ▪ *db* specifies the name of a Gem file where the array of parameters is stored. ▪ *array* specifies the name of an array in the specified database that has the above parameters.

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`*	Controls whether a line of data is exported based on having NAs in a cell. ▪ *`always`* exports data regardless of whether or not it contains NAs. ▪ *`allna`* does not export a row of data if all columns are NA (default). ▪ *`anyna`* does not export a row of data if any cell contains a NA value. ▪ *`arrayna`* does not export a row of data if the value in the given array name is NA (requires `-naArray`).
`-naArray` *`arrayName`*	When *`arrayna`* is specified using the `-skipNA` parameter, this option specifies the export array that is checked to determine if data is exported.
`-index` *`arrayName`*	Controls whether arrays are indexed by looking at a specified array. Only export the non-NA cells in the given array and each cell in the other arrays that have the same position names. If another array is at a higher dimension level, translate the given arrays cell index to the other arrays.
`-append`	Specifies that output is appended at end of output file. The default is to overwrite output file.

Argument	Description
-wide	This parameter causes the data to be exported wide, which means the innermost dimension will go across the row instead of each cell on a separate line. This is most useful when the innermost dimension is time. 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*	Used to limit 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.
-precision *precisionValue*	This parameter causes the utility to avoid exporting values that differ from the NA value by the specified value. Any values smaller than the precision value are not exported. For example, consider a measure with the NA value of zero and a precision value of 0.01. A value of 0.0034 would not be exported while a value of 0.34 would be exported. The precision value must be less than one. If a value greater than one is provided the utility returns a warning.
-processes *max*	Defines the maximum number of processes to run in parallel.
-useArrayNaValue	This argument will enable the use the NA value of the array instead of the NA value specified in measSpec or arraySpec.
-useLoadFormat	This argument enables the use of the format as specified by the measure property. The level at which the data is stored in the domain will be used. The -dim parameter is not allowed with the -useLoadFormat.

-useLoadFormat Parameter

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

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

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

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

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

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

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

Mapping Data Between Domains – mapData

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

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

Note: This utility does not update buffer positions.

Usage

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

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

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

Moving Data between Arrays – updateArray

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

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

Note: This utility does not update buffer positions.

Usage

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

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

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

`–destRange first:last`	Optional argument.
	Default is no range.
	Defines range along innermost dimension of destination array.
	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`

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

Scan Domain Data – 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 extermely 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 attempting to perform a repair of the databases, the utility has 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.

Usage

```
scanDomain –version
scanDomain –d domainPath [–detectDataLoss ] [–detectCorruption] [–loglevel level]
[–noheader]
scanDomain –d domainPath –repairCorruption [–backup]
[–processes maximumNumberOfProcesses] [–loglevel level] [–noheader]
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 could run two consecutive commands for detecting corruption and data loss, although this would be less efficient.

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

Argument	Description
`-version`	Prints the version of the utility.
`-d` *`domainPath`*	Required argument to specify the path to a global, non-partitioned, or local domain.
`-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`	Optional argument to back up database files before attempting to repair them. **Note:** This argument can only be used with `-repairCorruption`.
`-processes` *`maximumNumberOfProcesses`*	Optional argument to specify the maximum number of processes to be started to repair Btree database corruptions. **Note:** This argument can only be used with `-repairCorruption`.
`-loglevel` *`level`*	Optional argument to set the logger verbosity level. Possible values: `all`, `profile`, debug, audit, `information`, `warning`, `error`, or `none`.
`-noheader`	Optional argument to disable the timestamp header.
`-?|-help|-usage`	Optional argument to get the usage text.

12

Operational Utilities

Find Alerts – alertmgr

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

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

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

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

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

Usage

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

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

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

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

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

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

Copying Domains – copyDomain

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

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

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

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

Usage

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

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

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

Argument	Description
-maxProcesses count	If this argument is specified, some parts of copyDomain will run in parallel, utilizing up to the given number of processes.
-noSubDomains	Do not copy any local domains in the source domain.
-export	Export each database (.gem file) from the source domain into a format that can be used on a UNIX platform. This argument cannot be used when specifying an -xmlConfigFile.
-gzip	Compress the copied domain into a gzip format. This argument cannot be used when specifying an -xmlConfigFile.

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

Format of the XML configuration file

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

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

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

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

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

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

Move a Domain – moveDomain

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

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

Usage

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

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

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

Format of the XML Configuration File

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

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<rpas>
    <globaldomain>
        <srcPath>C:\usr\Rpas\Domains\GlobalDomain</srcPath>
        <dstPath>C:\usr\Rpas\Domains\GlobalDomain2</dstPath>
        <subdomain>
            <srcPath>C:\usr\Rpas\Domains\GlobalDomain\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 intended to be moved; otherwise an error condition will be incurred.

"srcPath" and "dstPath" tags for local domains are required ONLY if the local domain is intended to be moved; otherwise, the lack of tags for a specific local 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; i.e. when moving global domains, all local domains that reside under the global domain folder and are not included in the XML file, will be moved to the destination global domain folder. Other local domains with a specified destination location will be moved according to the configuration.

Assumptions and Requirements

The following rules apply to the XML configuration settings:

- All source and destination paths must be absolute.
- All source paths must correspond to existing directories.
- All destination paths must be valid, in the sense that:
- The parent of the destination directory must exist
- The parent directory must be writable by the user.
- The destination directory itself must not exist.
- The source and destination master domain paths are required.
- The source and destination sub domain paths are required only for the domains that need to be moved.
- The sub domains that need to be moved can be specified or those sub domains that will remain under the master domain can be left out. If a sub domain 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) master domain plus the size of the largest local domain.

Setting Miscellaneous Domain Properties – domainprop

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

Usage

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

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

Argument	Description
`–d` *`pathToDomain`*	Specifies the domain path.
`–expirePassword` *`days`*	Used to set or view the number of days a password is valid. Specify a valid integer after the argument to set the number of days for which the password will be valid. If no number is provided the utility prints the current setting. **Note:** Do not use this argument with `–property`.
`–passwordHistory` *`oldPasswordCount`*	Used to set or view the number of previous passwords that are kept to ensure that a user does not repeat a password too often. If a number follows `–passwordHistory`, the property is set to that number. Otherwise, the current setting is printed. **Note:** Do not use this argument with `–property`.
`–property` *`propertyname=value`*	Used to specify 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. **Note:** Do not use this argument with `–expirePassword`, `–passwordHistory`, or `–lockAccount`.
`–lockAccount` *`{#failedLogins}`*	Used to set or view the number of failed login attempts that can occur before the account is locked out. If a number follows `–lockAccount`, the property is set to this value. Otherwise, the current setting is printed. **Note:** Do not use this argument with `–property`.
`–reportSubDomains`	Use this option to show property values for subdomains (should all match).

Available Properties

Domain Property Name	Type	Description
disable_commit_asap	Boolean	If this property is set to TRUE, the ability to use "Commit ASAP" in the File menu of the RPAS Client is disabled. Furthermore, the "Commit ASAP" radio button on the workbook close dialog is disabled. By default, this property is set to FALSE.
disable_commit_later	Boolean	If this property is set to TRUE, the ability to use "Commit Later" in the File menu of the RPAS Client is disabled. Furthermore, the "Commit Later" radio button on the workbook close dialog is disabled. By default, this property is set to FALSE.
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 defines the path to the help files. It is set when using custom help files for the RPAS Client instead of the default help files that are provided with the RPAS Client installation. These files need to be in "WebHelp" format. This path would normally be a path to a network folder where the common help files are located. The folder must have a file named "Start.htm". This file is the starting point for the custom help. The default RPAS help has several pages for context sensitive help. Help searches for these folders in the preset relative location from the root folder of WebHelp. If an implementation uses custom help, it must locate the context help files in the expected default locations relative to that folder. Otherwise, context sensitive help will not be available and an error will be raised every time the user tries to invoke context sensitive help for a particular function. If this property is not set, the path specified in the foundation.ini file of the Client machine is used. If a path is not specified in that file, help looks in the default location in the RPAS Client installation for help files. If the default files have been removed, help is not available. For the location of the foundation.ini file, see the "Translation Administration" section.
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.

Domain Property Name	Type	Description
meas_fillclr_precedence	Boolean	By default, when deciding which color to fill a particular cell with, 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.
measure_locking_disabled	Boolean	If this property is set to FALSE, the user can lock a measure on a work sheet. By default, this property is set to FALSE. To disable measure locking, set this property to TRUE.
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 that 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.

Calculation Engine – mace

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

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

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

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 running `mace` 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 transitting over a list of rule groups
- -print: print all specific rules and groups
- -validate: validate rule groups

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
```

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

Argument	Description
`-d` *domainPath*	Specifies the domain in which to load the measure.
`-find` *string*	Use this argument to search all expressions for the specified string, printing all the rules and rule groups that have these expressions.
`-newRule {-ruleName` *ruleName}*	Use this argument to create a new empty rule. If desired, use the *-ruleName* argument to specify a name for the rule.
`-label {`*ruleLabel* | *groupLabel}*	Use this argument to specify the label of the rule with the `-newRule` argument or label of the group with the `-newGroup` argument.
`-processes` *numProcesses*	Use this argument to specify the number of child processes to be run in parallel.
`-delRule` *ruleName*	Use this argument to remove the specified rule.
`-addRule` *groupName:ruleName*	Use this argument to add the specified rule to the group specified by *groupName*.
`-removeRule` *groupName:ruleName*	Use this argument to remove the specified *ruleName* from the group specified by *groupName*.
`-newGroup` *groupName*	Use this argument to create a new rule group with the specified name.
`-removeGroup` *groupName*	Use this argument to remove the specified group and non-shared rules in it.

Argument	Description
–addExpression *ruleName*	Use this argument to add an expression to the specified rule. –expression should be used with this argument.
–check	Use this argument to validate the specified expression. –expression should be used with this argument.
–run	Use this argument to evaluate the specified expression or rule group. –expression should be used with this argument.
–resolve *groupName*	Use this argument to order (does not evaluate) expressions within rule group. Requires a comma-separated list of edited measures.
–transit *workbookName*	Use this argument to run a calc engine by transitioning over a list of rule groups. Requires the name of an existing workbook and a comma-separated list of rule-group names.
–print *{ruleList | groupList | true}*	Use this argument to print all the specified rules and rule groups. The *ruleList* is a comma-separated list of rule names. The *groupList* is a comma-separated list of group names. If "true" is supplied for either *ruleList* or *groupList*, all rules or rule groups are printed.
–purgeRules	Use this argument to remove all rules not contained in any rule groups.
–removeAllRuleData	Use this argument to remove all rule groups and all rules.
–validate {calc | general | refresh}	Use this argument to validate rule groups. Use calc to validate a calc rule group. To validate a refresh rule group, use the refresh parameter along with –calcRuleGroup to specify the corresponding calc rule group. For all other types of rule groups, use general.
–debugRuleEngine	Use this argument to generate a file "mace.log" in the working directory for logging RuleEngine specific debug information.
–expression *exprString*	Use the argument to specify the expression. This argument is used in conjunction with the –addExpression, –check, and –run arguments.
–group *groupName*	Use this argument to specify the rule group to evaluate using the –run argument.
–measures *measureList*	Use this argument to specify the measures to resolve.

Argument	Description
`-group groupList`	Use this argument to specify a list of group names, separated by commas. Use this argument in conjunction with the `-transit` and `-print` arguments.
`-rule ruleList`	Use this argument to specify a list of rule names, separated by commas. Use this argument in conjunction with the `-print` argument.
`-allGroups`	Use this argument in conjunction with the `-print` argument to print all rule groups.
`-addGroup`	Use this argument to create a new rule group with the specified name

Managing Users – usermgr

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

Usage

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

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

Argument	Description
`-d domainPath`	Specifies the path to a domain to add, remove, or get information about a user.
`-add userName`	Use this argument to add a user with a specified name. Use the other arguments specified in the usage to add those attributes for that user.
`-label label`	Use this argument to specify the label of the user to add to the domain.
`-password psw`	Use this argument to specify the password of the user to add to the domain.
`-group grp`	Use this argument to specify the user group of the user to add to the domain.
`-admin`	Use this argument to specify that the user to add to the domain has administrative rights.
`-remove userName`	Use this argument to remove the user with the specified name from the domain.

Argument	Description
-removeLabel *label*	Use this argument to Remove all users with this label.
-list	Use this argument to list all the users registered to the specified domain.
-print	Use this argument to print the specified user or group information.
-user *username*	Use this argument to specify the user name in the specified domain to print. This argument is only applicable to -print option.
-group *groupname*	Use this argument to specify the group in the specified domain name to print. This argument is only applicable to –print option.
-addUsers	Adds users from begin to end using userFormat argument as a printf format string to form both the user name and the password. For example the arguments: "-addUsers 'usr$02d'-begin 1 -end 3" would result in usr01, usr02, and usr03 created with the password matching the userName.

Managing the Workbook Batch Queue – wbbatch

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

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

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

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

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

Update Auto-Workbook Build Tasks when Using PNI Functionality

If PNI positions in a domain are changed, any auto-workbook build tasks in the workbook batch queue for that domain need to be removed and recreated. If this is not done, the positions in the domain hierarchies that were changed will not be included in the workbook. The following steps illustrate this situation:

1. An auto-workbook build task is in the workbook batch queue.
2. Changes are made to the domain hierarchies. For example, PNI positions are deleted and then reloaded.
3. An auto-workbook build task is created using the wbbatch utility.
4. The workbook will not include the PNI positions that were deleted and then reloaded.

Usage

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

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

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

Workbook Manager – wbmgr

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

Usage

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

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

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

Register Measure – 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 are optional, such as default aggregation and spreading method. 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 will cause 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 can be modified, such as type, base intersection, and database name. 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.

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
```

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

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.
`–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.

Argument	Description
-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 navalue must be the same type as the measure. For date, the navalue 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.

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 will be 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 will be 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.

Argument	Description
-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 will not be 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 inlcude 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 which 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.
-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.

Argument	Description
`-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 'mmddyyyy:mmddyyyy', where the first date is the starting date of the range and the second date is the ending date of the range. 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.

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.

Argument	Description
-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 Client. The actual value of the picklist is specified by the -range option of the measure. -nopicklist is only available with the -modify option. It means the measure should not be displayed as a picklist measure in the RPAS Client.
-materialized *(persistent|display)*	Specifies whether the measure is 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 Only 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.

Argument	Description
-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*	Specifes 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 specifed.
-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 meta data 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 navalue as a regular value.
-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.

Argument	Description
-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.

Register Token Measure – 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 will be evaluated against the value measure. A token measure name cannot be used in expression without the prefixing "@".

In the following example, TM1 is a token measure registered with the value measure name VM1.

The following expression:

@TM1 = a + b

Will be evaluated as:

VM1 = a + b

The following expression is not valid, because TM1 is used without prefixing it with "@":

TM1 = "sth"

If evaluated using mace, mace 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.

Usage

```
regTokenMeasure -version
regTokenMeasure -d pathToDomain -add tokenMeasure=valueMeasure {-fnhbi}
regTokenMeasure -d pathToDomain -list
regTokenMeasure -d pathToDomain -remove tokenMeasure=valueMeasure
```

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

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 its 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.

13

Informational Utilities

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

Retrieving Domain Information – domaininfo

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

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

Usage

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

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

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

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

Checking the Validity of a Domain – checkDomain

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

Usage

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

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

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

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

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

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

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

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

Determining RPAS Server Version – rpasversion

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

Usage

```
rpasversion -l pathToLibrary
```

List Contents of a Database – listDb

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

Usage

```
listDb pathToDb*
listDb -row -db pathToDb*
listDb -row -pageUsage -db pathToDb*
listDb -row -standardOptions -db pathToDb*
listDb -standardOptions -db pathToDb*
listDb -version
```

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

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

Printing Data from Arrays – printArray

Use `printArray` to print the contents of an array.

Usage

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

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

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

Printing Data from Measures – printMeasure

Use the `printMeasure` utility to print measure information.

Usage

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

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

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

14

RPAS 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 driver presents OREDB as a relational database to ODBC and JDBC client applications. The RPAS ODBC/JDBC Driver enables ODBC 3.51 Oracle Corporation 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*.

ODBC Configuration

Overview

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.

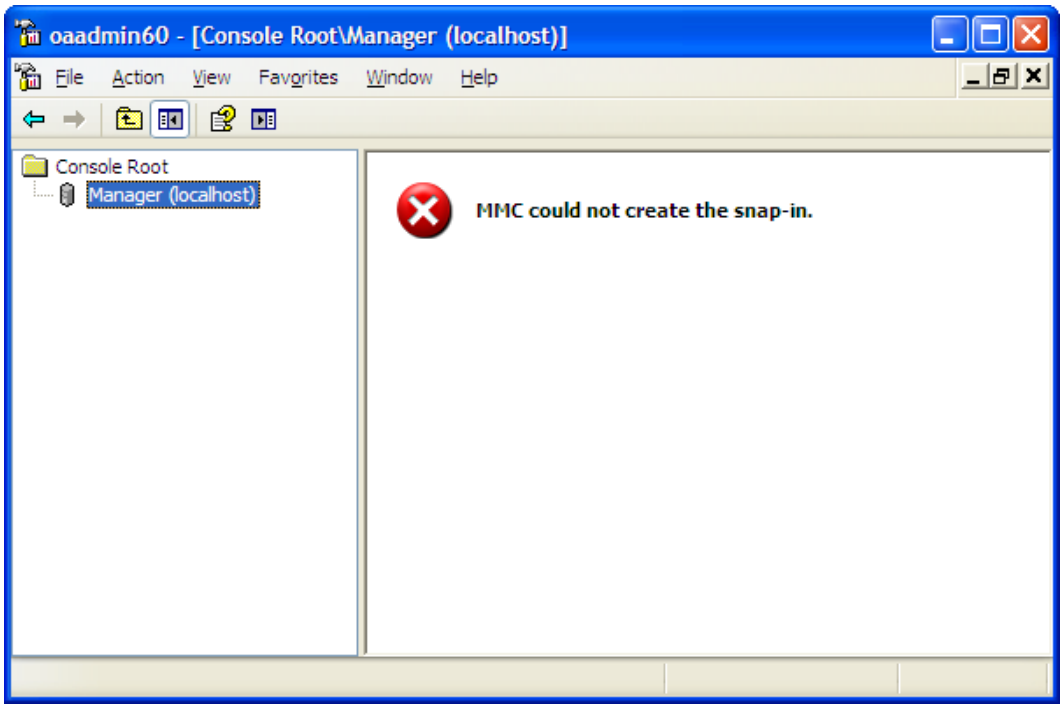
Defining the ODBC Server Configuration Settings

On UNIX/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. A command line version of the Management Console named "oacla" is also available. Oracle Retail strongly suggests that users use the Management Console GUI for ODBC server configuration.

To define the ODBC Server configuration settings:

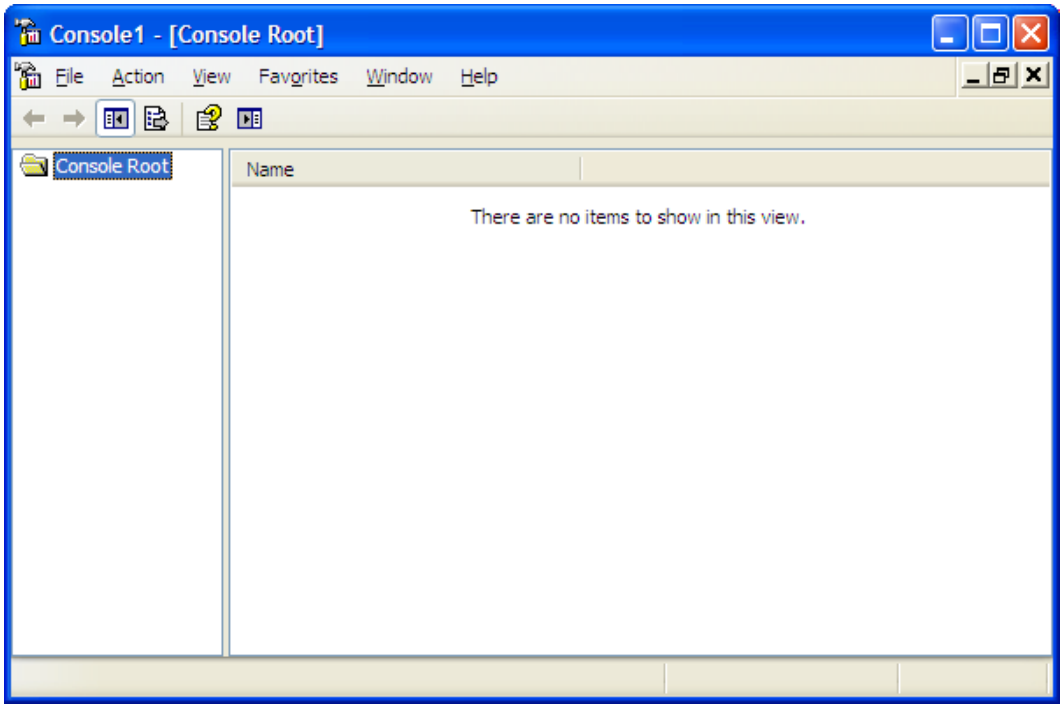
1. Add RPAS ODBC Manager in the Management Console.
   a. From the **Start** menu, select **Oracle RPAS ODBC Server** and then **Management Console**. The Oracle RPAS ODBC Management Console appears.

> **Note:** When you start the Management Console for the first time, an error message appears indicating the snap-in is not registered.



Console Manager Window Opened for First Time

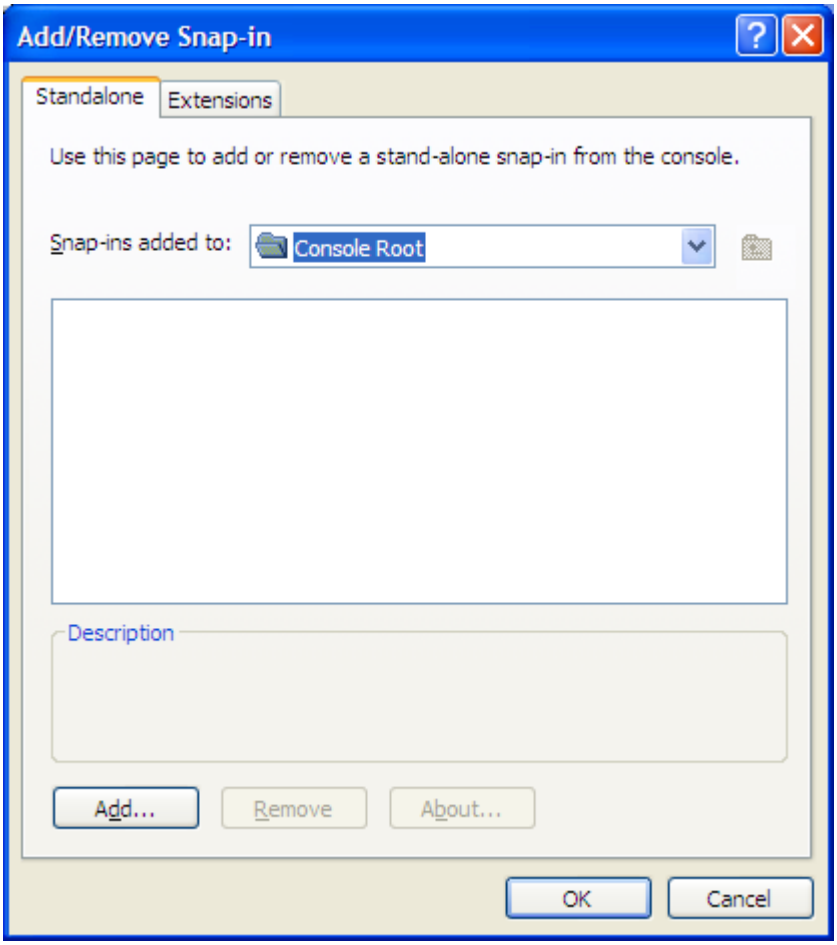
   b. To create a new work space, select **File** and then **New.**



Console Manager Window Showing Console Root Directory

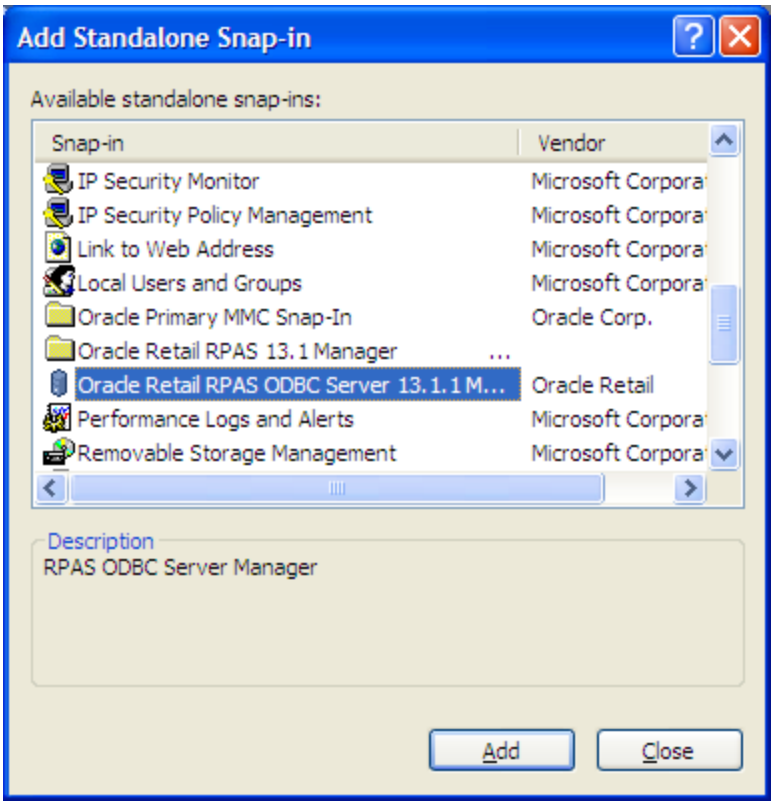
To connect to a remote host, the Management Console is installed on a Windows PC and the Agent service is installed on a remote server.

c. To add a new snap-in, select **File** and then **Add/Remove Snap-in** to start the wizard to add a new snap-in.



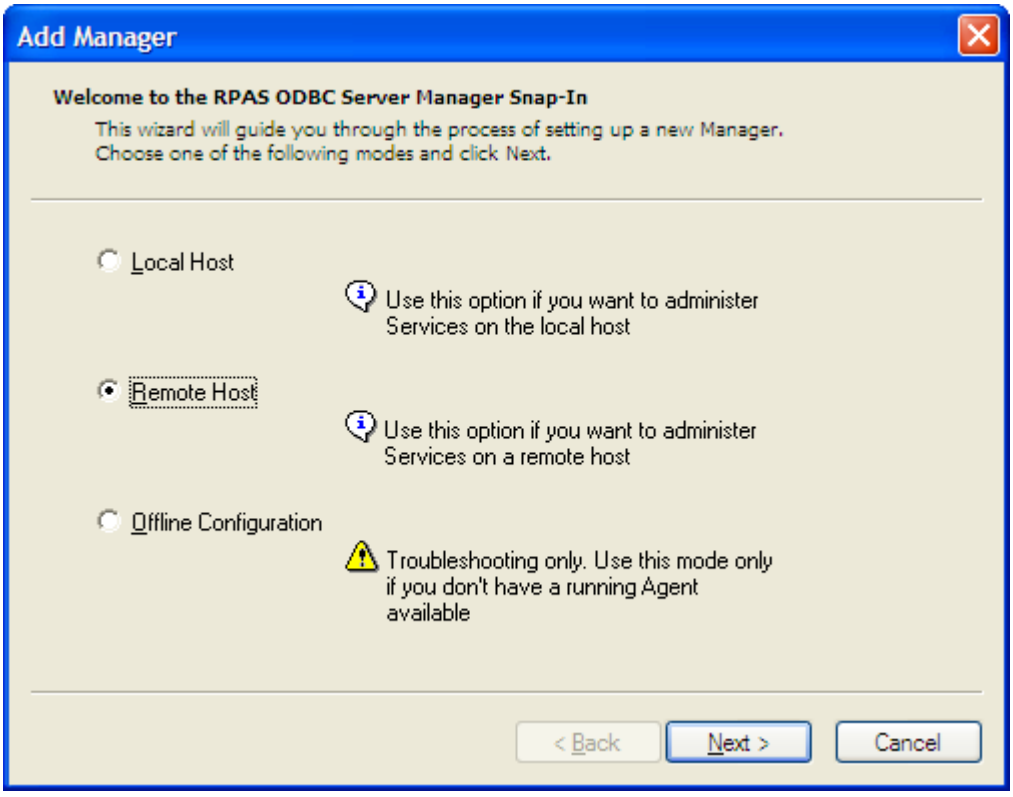
Add/Remove Snap-in Dialog Box

d. To add a new snap-in, click **Add**.



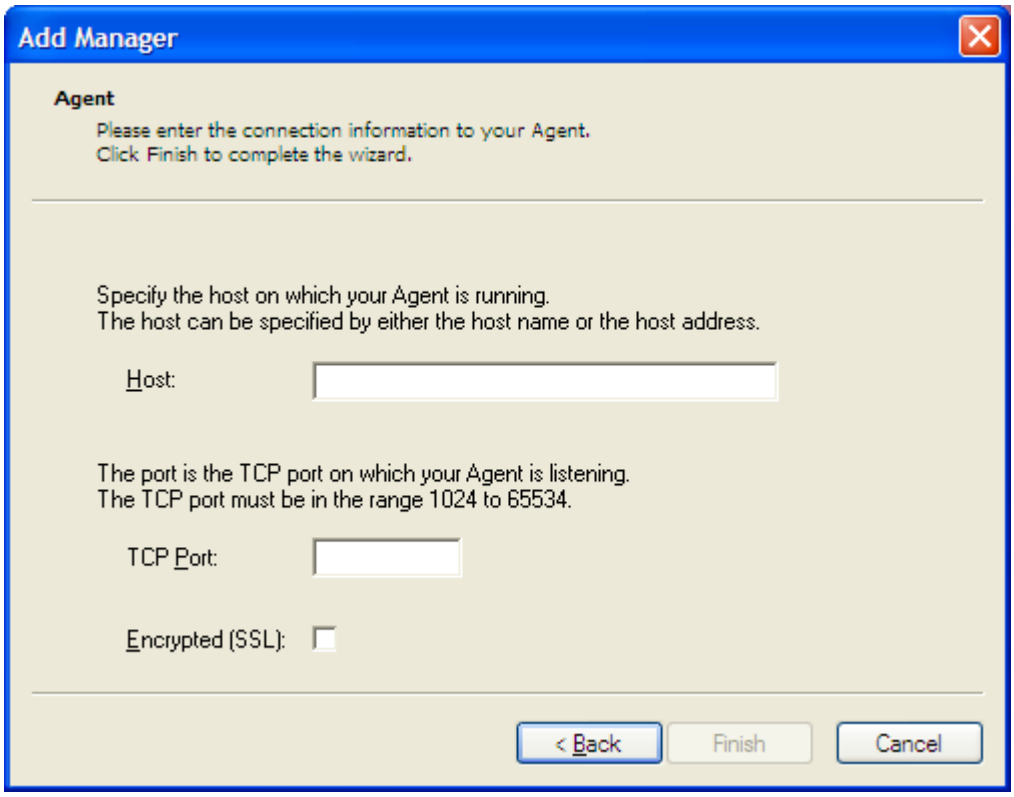
Add Standalone Snap-in Dialog Box

e. Select **Oracle Retail RPAS ODBC Server 13.1.1 Manager** and click **Add**.



Add Manager Dialog Box Showing Selection of the Mode for a New Manager

f. Select Local Host or Remote Host depending on the location of the server you want to manage and then click **Next**.



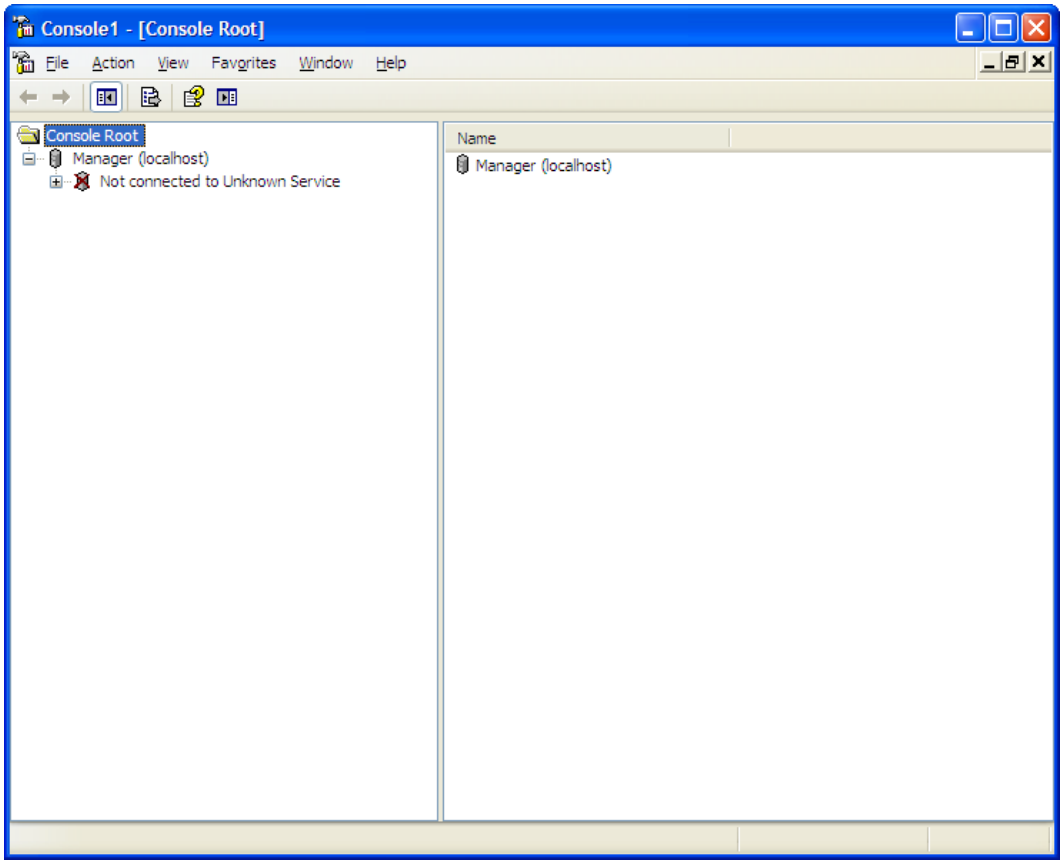
Add Manager Dialog Box Showing Connection Information to be Added for the Agent

g. 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.

h. If the Agent is not SSL enabled (which is the default), uncheck "Encrypted(SSL)". If the Agent is SSL enabled, check "Encrypted(SSL)".

i. Click **Finish**. You are returned to the previous window. You can click **Close** or **OK** to close the windows.

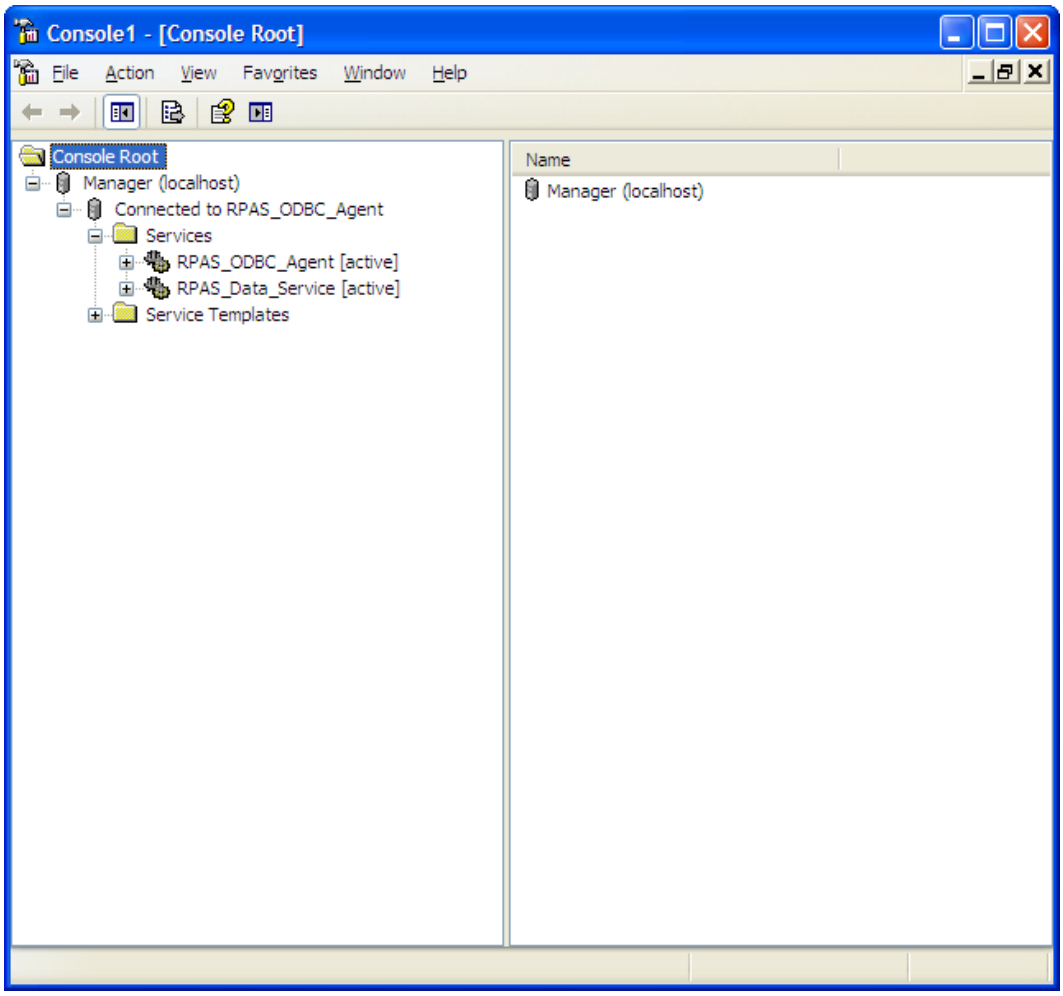
2. From the Management Console, use the ODBC Manager to perform the required configuration.
   a. To configure for a remote host, click + to expand the ODBC Manager you just added.



Console Manager Window Showing Expanded Console Root Directory

   b. 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.

c. Expand Services by clicking +.



Console Manager Window Showing Expanded Connected to RPAS_ODBC_Agent Directory

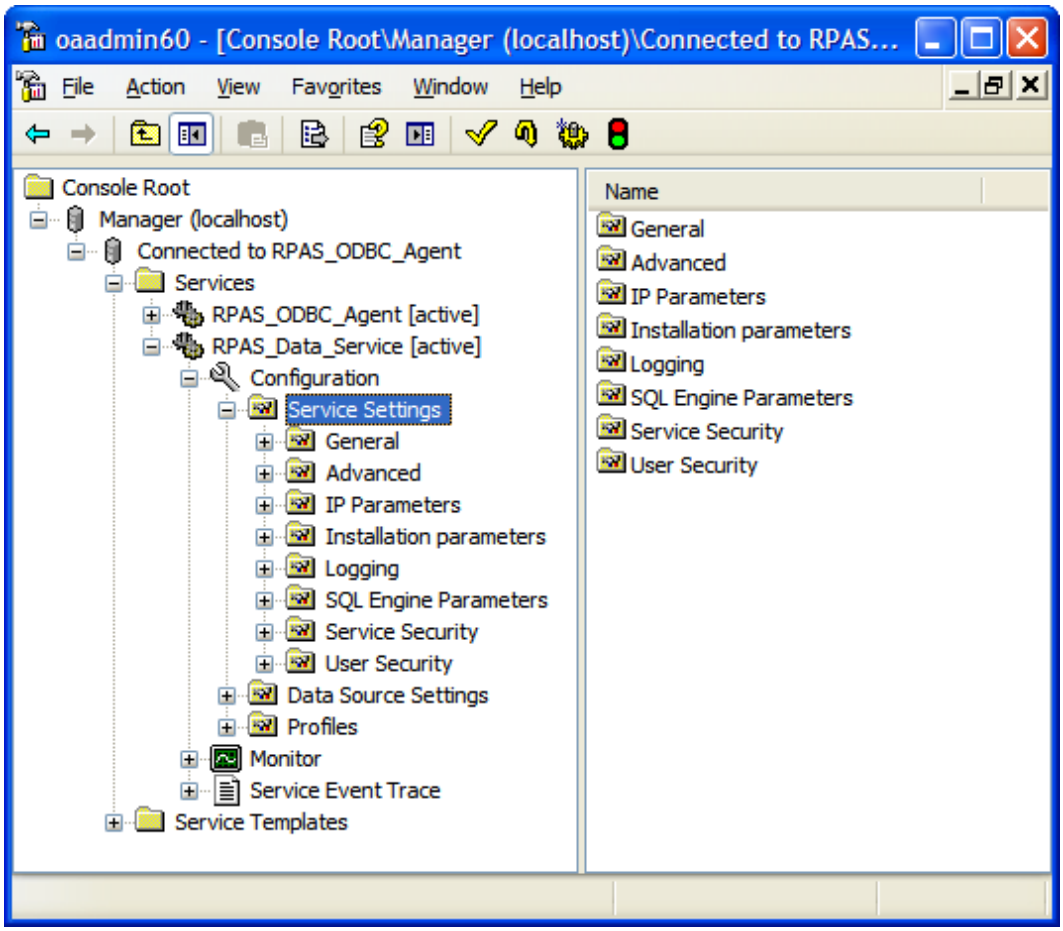
d. Set up the environment variables.

> **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 RPAS home library
- Path to the ODBC server binary
- Path to the ODBC server IP binary

i. Navigate to the following screen. The following figure shows the exact screen you should see.



Console Manager Window for Configuration

ii. Right-click on the blank space of the right panel.

iii. Select **New/Attribute** in the menu.

iv. Select "ServiceEnvironmentVariable" from the list for "Attribute". In the "Value" field, enter the following:
PATH={pathToRpasHomeLib};{pathToODBCServerBin};{pathToODBCServerIPBin}

In the sample shown in the following figure, 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

New Attribute Dialog Box

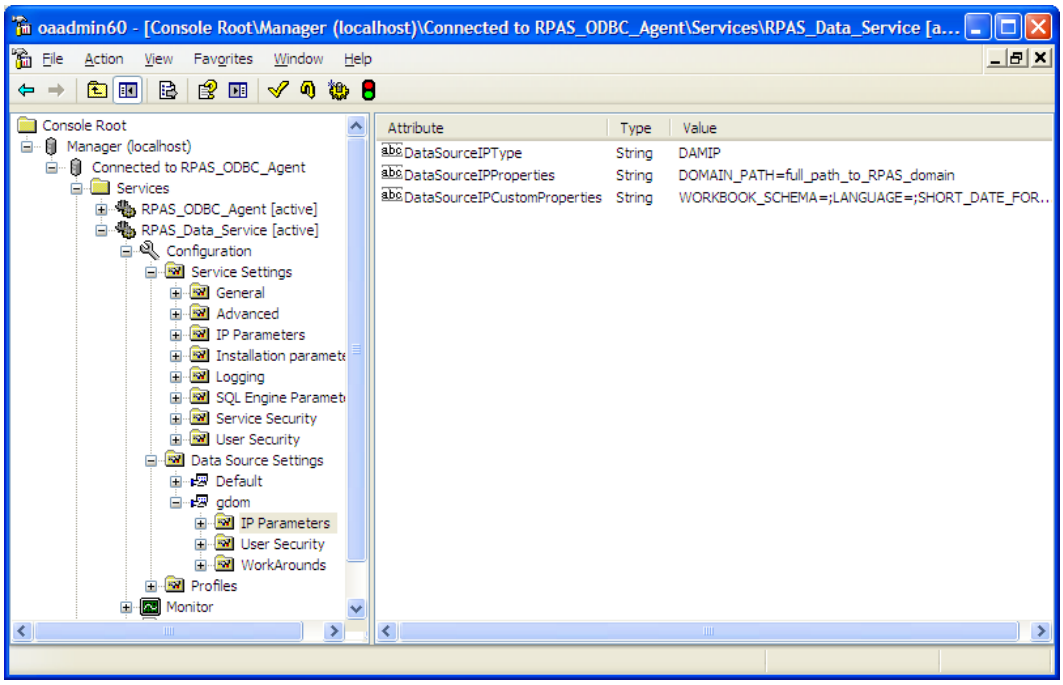
v. Click **OK**.

e. 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.

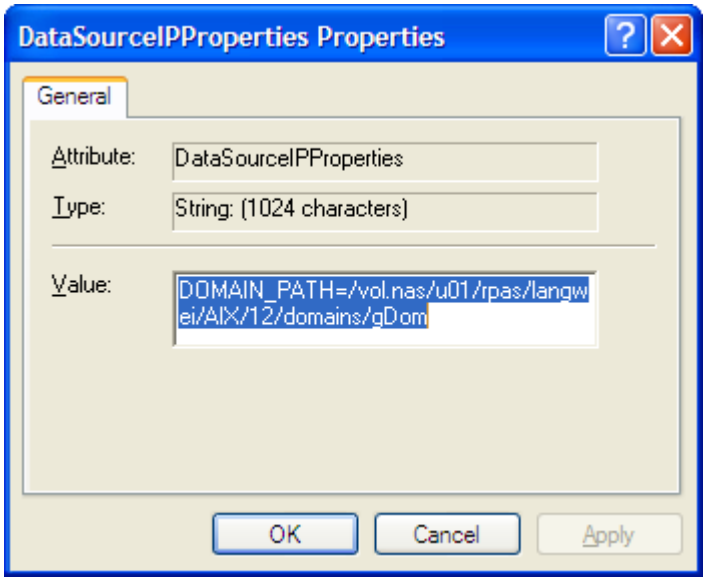
The DataSourceIPCustomProperties attribute contains pre-registered property names, which are NOT to be modified.

f. Under gdom, select IP Parameters.



Console Manager Window with IP Parameters Selected under gdom

g. On the right panel of the window, double click DataSourceIPProperties attribute. The following window appears.

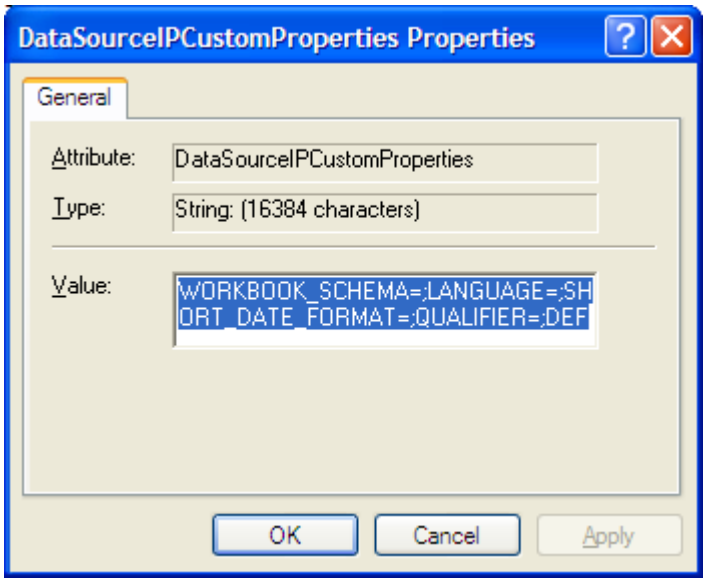


DataSourceIPProperties Properties Dialog Box

Make sure the Value field has the keyword "DOMAIN_PATH=", and the string after the "=" sign is the absolute path to the RPAS domain you want connect to on the server side.

To connect to a domain on the local host, the following is an example of the path: "DOMAIN_PATH=C:\RPAS\11\Test\ODBC\nt_testGlobalDomain"

h. On the right panel of the window, double click the DataSourceIPCustomProperties attribute. The following window appears.



DataSourceIPCustomProperties Properties Dialog Box

Copy the value from the DataSourceIPCustomProperties of the sample "gdom" data source to make sure all pre-registered properties are included.

3. Import the configuration changes:
   a. To save the configuration, right click "Services" in the Console Manager window. In the menu, select **All Tasks** and then **Save Configuration**.
   b. To save the snap-in configuration, select **File** and then **Save as**. Select <*installdir*>\admin\oaadmin60.msc for the file name. This will overwrite the original file (which was basically empty).
   c. 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.

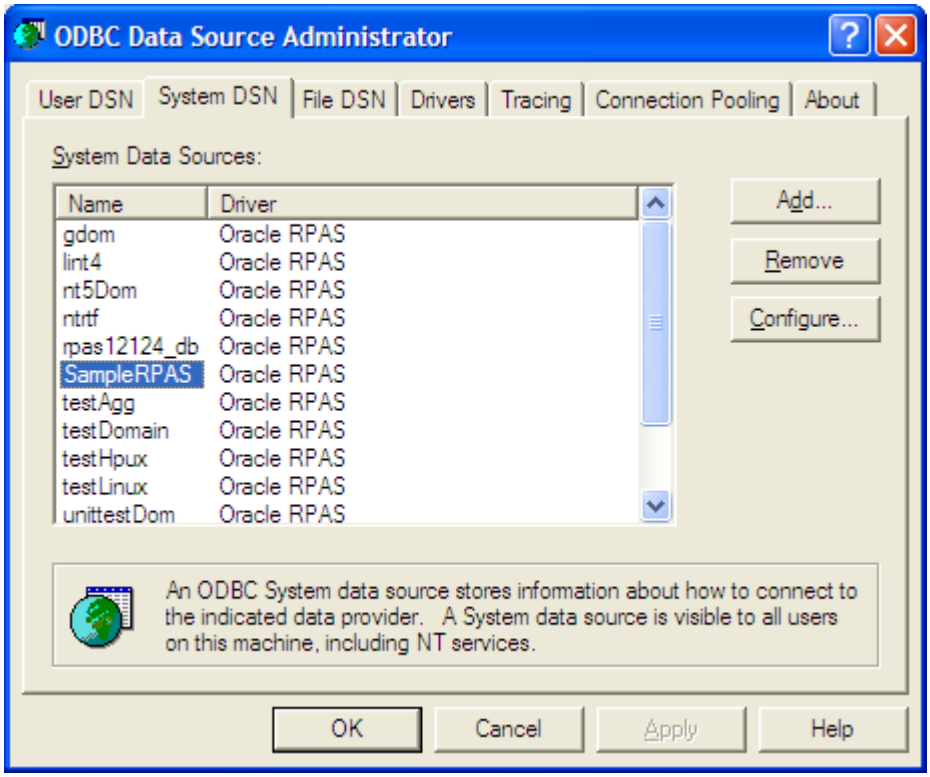
Defining the ODBC Client Configuration for Windows

To define the ODBC Client configuration settings:

1. From the **Start** menu, select "ODBC Administrator" under the "Oracle RPAS odbc driver" menu item.

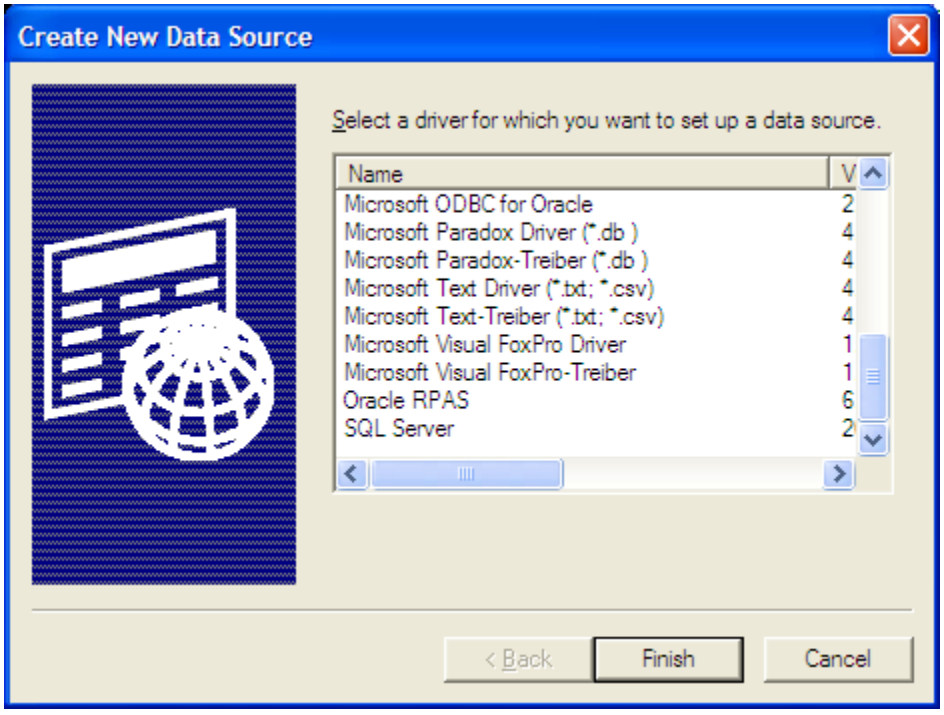
> **Note:** Upon successful installation of RPAS ODBC Client, a sample DSN named "SampleRPAS" is automatically created and configured to connect to the data source "gdom" on the server side. To create a new DSN, see the following steps.

2. In the ODBC Data Source Administrator window, select **Add** to add a data source.



ODBC Data Source Administrator Dialog Box

3. In the "Create New Data source" window, select "Oracle RPAS".



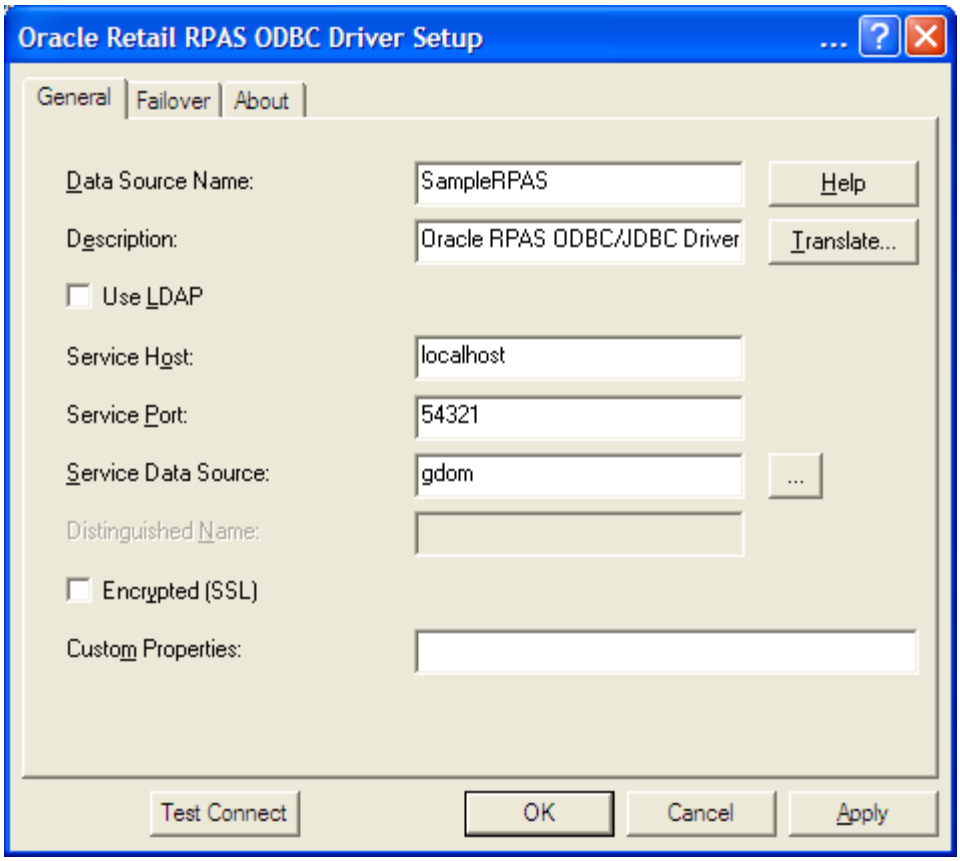
Create New Data Source Dialog Box

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.

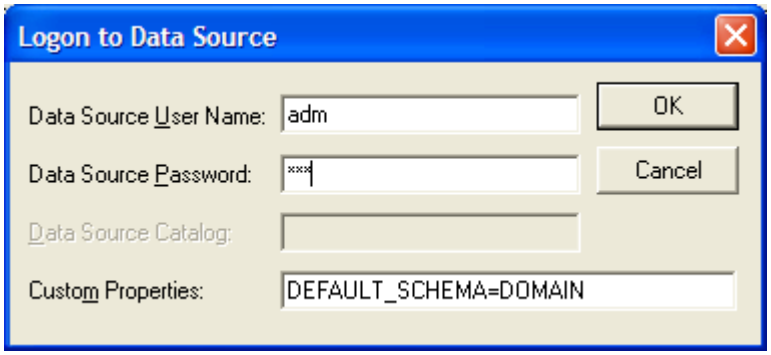
   - In the Service Data Source field, enter the name of the service data source that has been configued for the data service. The default for this field is "gdom".
   - If the data service is not SSL enabled, uncheck "Encrypted(SSL)". If the data service is SSL enabled, check "Encrypted(SSL)".
   - Enter 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):

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 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 will present 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.



Oracle Retail RPAS ODBC Driver Setup Dialog Box

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.



Logon to Data Source Dialog Box

If the connection is successful, the following dialog box is displayed.

Oracle Retail RPAS ODBC Driver Setup Dialog Box for Successful Connection

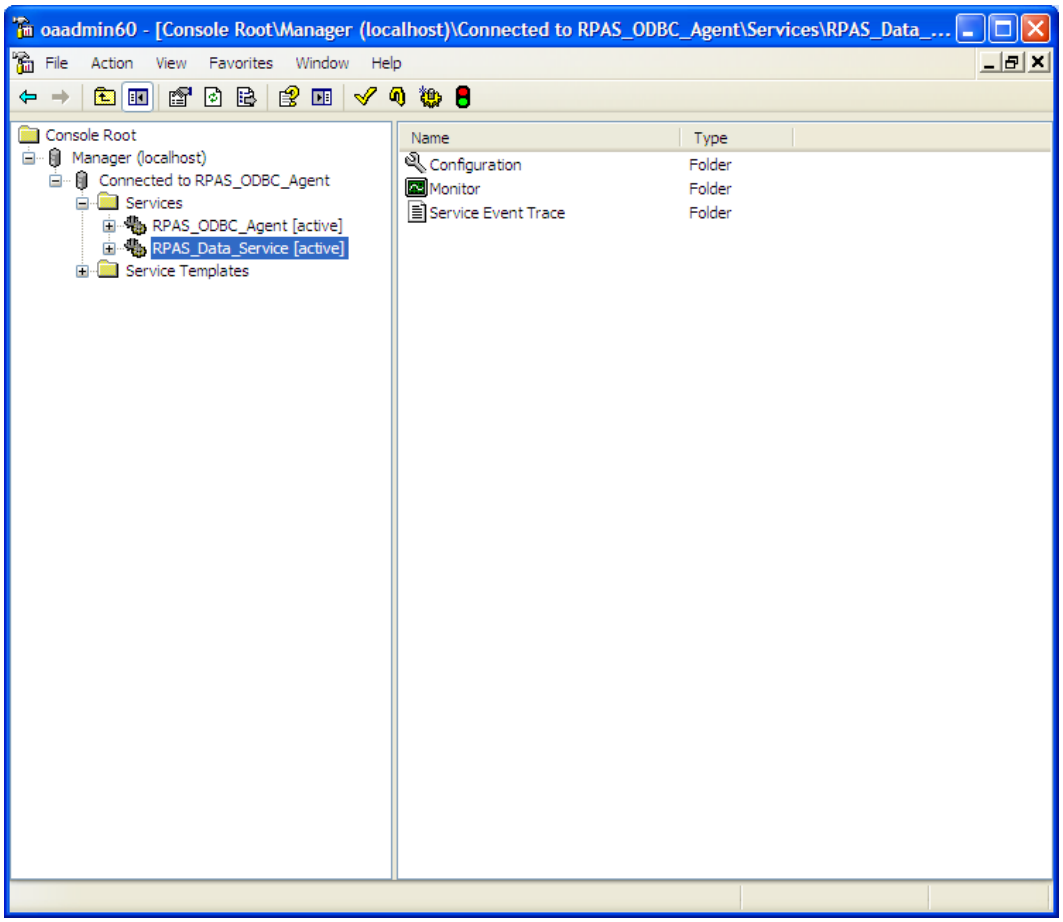
6. Click **OK**.

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.



Console Manager Window with Data Service Selected

Testing the Connection Using Interactive SQL

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

1. Select **Start**, **All Programs**, **Oracle RPAS ODBC 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

Overview

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 and 64-bit ODBC Client.

> **Note:** For the remainder of this chapter, the 32 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
      ```

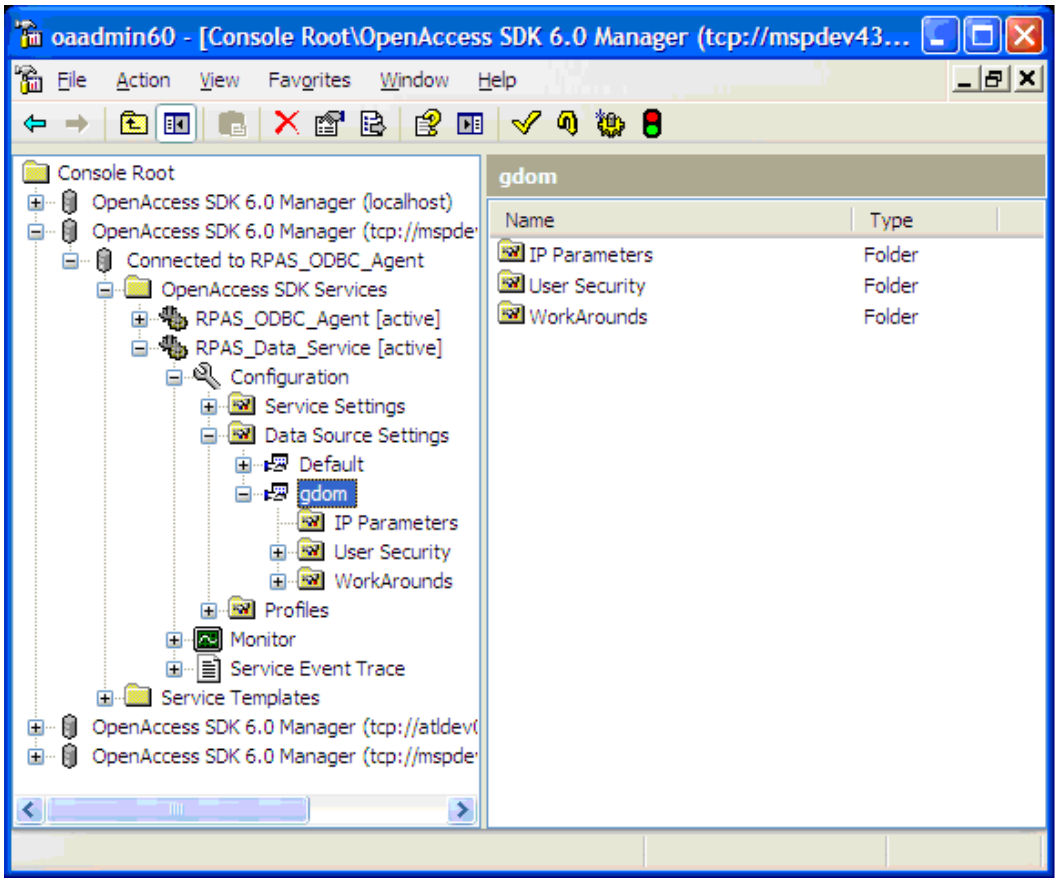
   c. Create a new [MyRPASDataSource] section: The [SampleRPAS] section can be copied and modified. In the new [MyRPASDataSource] section:

      – 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.



Console Manager Window with gdom Selected

Testing the Connection

Once 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, 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 the previous section. 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

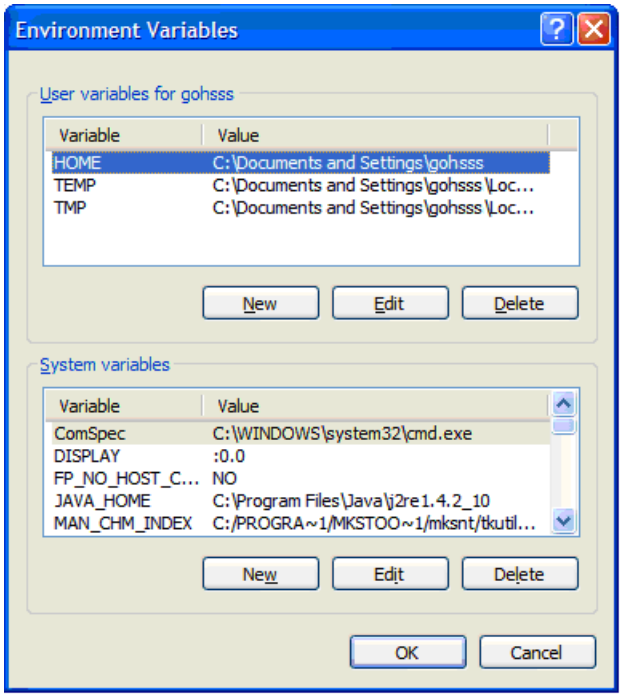
This section describes how to install, set up, and use the RPAS JDBC driver on all UNIX/Linux and Windows platforms.

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*.

Once 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.

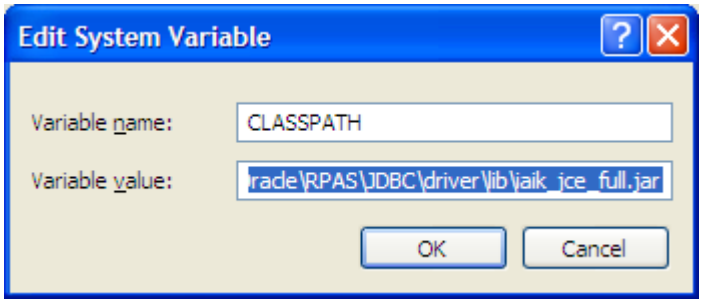
Updating Environment Variables for the JDBC Driver on Windows

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.



Environment Variables Dialog Box

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



Edit System Variable Dialog Box

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**.

   Where `driver_home` is the location where the jdbcclient was installed. If your jdbcclient was installed in C:/jdbcclient, you would have 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.

   Once 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 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:

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*	The maximum number of characters, specified by *numberofchars*, that Spy will log 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 –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.Open
AccessDriver;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.*;
import jdbc.sql.*;

public class RPASDriverTest {
    public RPASDriverTest() {
    }

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

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

            Connection conn = null;

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

            DatabaseMetaData dma = conn.getMetaData();

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

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

        } catch (SQLException ex) {
            System.out.println ("\n*** SQLException caught ***\n");
            while (ex != null) {
                System.out.println ("SQLState: " +
```

```
                                ex.getSQLState ());
                    System.out.println ("Message:  " + ex.getMessage ());
                    System.out.println ("Vendor:   " +
                                ex.getErrorCode ());
                    ex = ex.getNextException ();
                    System.out.println ("");
                }
            }
            catch (java.lang.Exception ex) {
                //Got some other type of exception.  Dump it.
                ex.printStackTrace ();
            }
        }
}
```

Data Query

Limitations

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

Contention

All domain data queries are allowed only in batch mode. The driver should not be expected to query a domain while users are connected and commit changes to the domain using the RPAS client. This limitation is due to the fact that the RPAS platform implements an exclusive lock policy that allows either a single write process to a measure or one or more read processes to a measure. This lock policy prohibits querying from a domain during online operations.

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

Workbook Queries

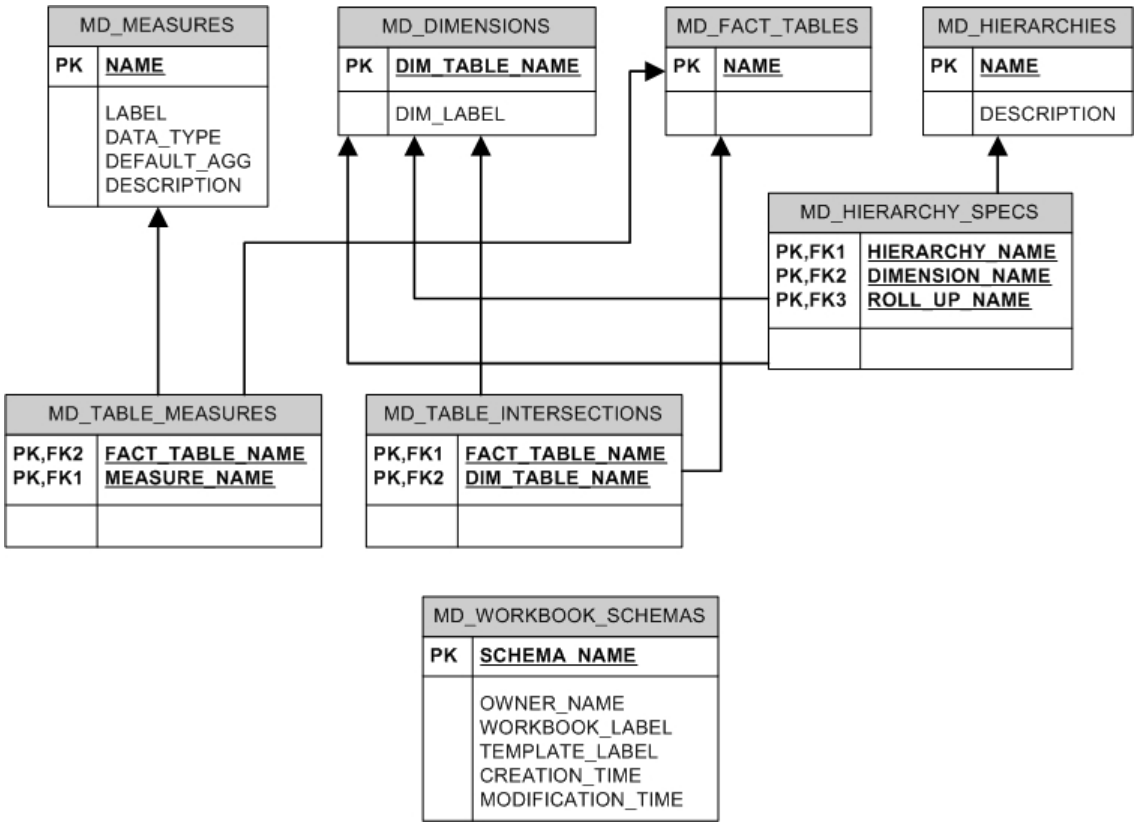
All workbook queries must be performed against saved workbooks. The workbook can be open or closed. The user has to save the workbook before reporting.

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.



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

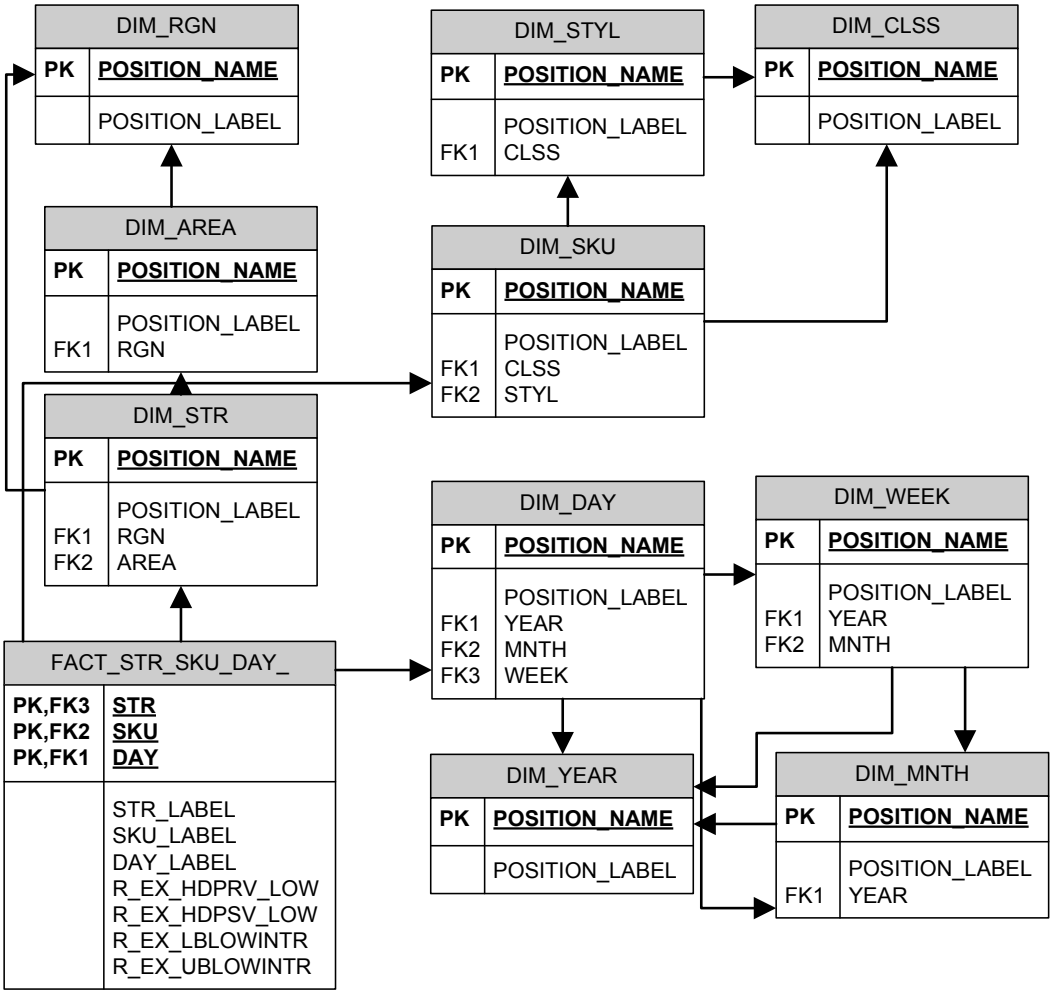
> **Note:** The MD_WORKBOOK_SCHEMAS table is not included in workbooks.

Fact and Dimension Tables

The following figure shows an example of the structure of fact and dimension tables and the relationships between them. A fact table represents an intersection where one or more measures' data is stored. Each measure is represented by a column in the table. Additionally, each dimension on the intersection is represented by a column. A record in the fact table is uniquely identified by a unique combination of position names for the intersecting dimensions.

A dimension table represents a dimension. It includes a column to list all position names, their labels, and their rollup mapping to each dimension at higher levels in the hierarchy.

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



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

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

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

For information on limitations when performing queries, see the "Limitations" section.

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 worksheet in the Security Administration workbook) and all measures to which the users have explicitly been given access rights using the Measure Rights worksheet in the Security Administration workbook. All other measures are not accessible to the users.

When 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

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

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

Querying Fact Data

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

```
Select
    *
From
    FACT_STR_SKU_DAY_
```

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

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

Note: The optional "`Hint`" clause in the above SQL statement is not ANSI SQL standard, but the 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** fram, 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', the value of this custom property would be 'AGG_TABLE_NAMES=FACT_DEPT, FACT_DEPT_YEAR'.
4. Click **OK**.

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

   WORKBOOK_SCHEMA=DOMAIN_T0;AGG_TABLE_NAMES= FACT_DEPT, FACT_DEPT_YEAR

Clients

This section lists some sample ODBC/JDBC client applications that can connect to the RPAS 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 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.

To connect OBIEE to a predefined DSN:

1. Make sure the following Windows services are running:
   - Oracle BI Java Host
   - Oracle BI Server
2. Start the OBIEE Administration Tool. 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. Select **ODBC 3.5** for Connection Type, enter the RPAS user name and password, and then click **OK**.

   The RPAS schemas and tables appear in a new window.
7. Select the objects you want to import, and then click **import**. Once the import is complete, click **Close**.

   A new physical model is created and listed in the **Physical** panel of the Admin Tool window.
8. Expand the physical model. Double-click on **connection** to open "connection" properties. Make sure the **Connection Type** is set to **ODBC 3.5**. 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.

Configuring the ODBC Client for OBIEE

The following example provides a sample of configuring the ODBC client for OBIEE. This example was developed for OBIEE on AIX, but the process is the same for other environments.

1. Open $BIEE_HOME/setup/odbc.ini file where $BIEE_HOME is the directory where OBIEE is installed.
2. Set the TraceDll to odbctrac.so that comes with RPAS odbcclient, and 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`.

   Example:

   ```
   [rpas_domain]
   Driver=absolute_path_to_odbc_client/oaodbc.so
   DriverUnicodeType=1
   Description=Rpas domain data source
   ```
5. Save your changes to the file.

Microsoft Access

To connect using Microsoft Access:

1. Start Microsoft Access.
2. Create a new (or open an existing) Access file (.mdb file).
3. From 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/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 Microsoft Access displays a list of the fields in the linked table. Select a field, or a combination of fields, that will uniquely identify each record, and then click **OK**.

JDeveloper

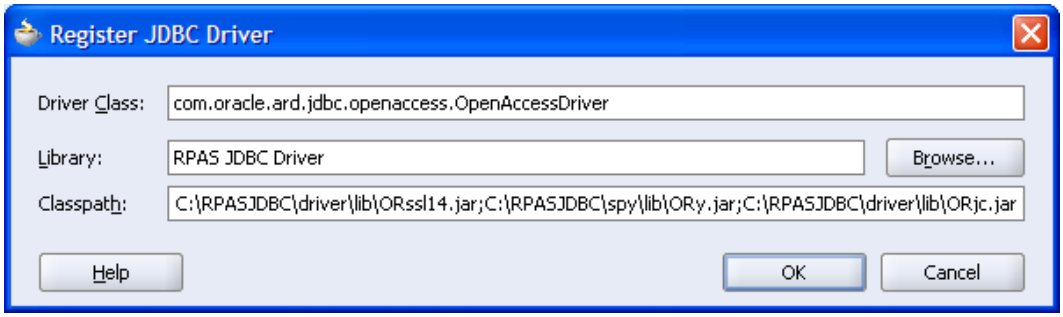
JDeveloper works best with a native JDBC driver, which is included in the RPAS ODBC/JDBC Driver package.

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 which is configured on the ODBC server.



Register JDBC Driver Dialog Box

7. Follow the instructions to finish creating the connection.

   Once the connection is established, a user can expand the Connection (and the nodes under the Connection) to browse the objects in the RPAS datastore. A user can also open a SQL worksheet (by selecting **SQL Worksheet** from the Tools menu) to write/execute SQL statements.

XML Publisher

This section describes how to make the connection from XML Publisher to RPAS using 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 which 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 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 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
```

Once connected, users 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 functionalities 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

Function	Description
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 3 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.

Function	Description
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.
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.

Function	Description
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.
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.

Function	Description
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

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.
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.

Function	Description
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'.
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'.

Function	Description
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.

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'.

Function	Description
RTRIM(*string_exp*)	Returns the characters of string_exp with trailing blanks removed. For example: SELECT RTRIM('abc ') FROM emp; This returns 'abc'.
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 / 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
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*: Is 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*: Is 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:* Is an expression that returns date or timestamp value or a character string in a date-time format.

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*: Is 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.

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

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.

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 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.

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
CHAR, BINARY	255
DATE	10
DOUBLE	15

Keyword	Value
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.

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

Keyword	Precision
SQL_BINARY	255
SQL_BIT	1
SQL_CHAR	255

Keyword	Precision
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

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 un-supported 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 will become 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 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/Time Functions

The OpenAccess SDK SQL engine does not support the following Date/Time functions:

- CURRENT_TIME[(*time-precision*)] - The SQL engine does not support time-precision argument.
- CURRENT_TIMESTAMP[(*time-precision*)]
- EXTRACT(*extract-field FROM extract-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.

15

Using Oracle Data Integrator with RPAS

Introduction

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

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

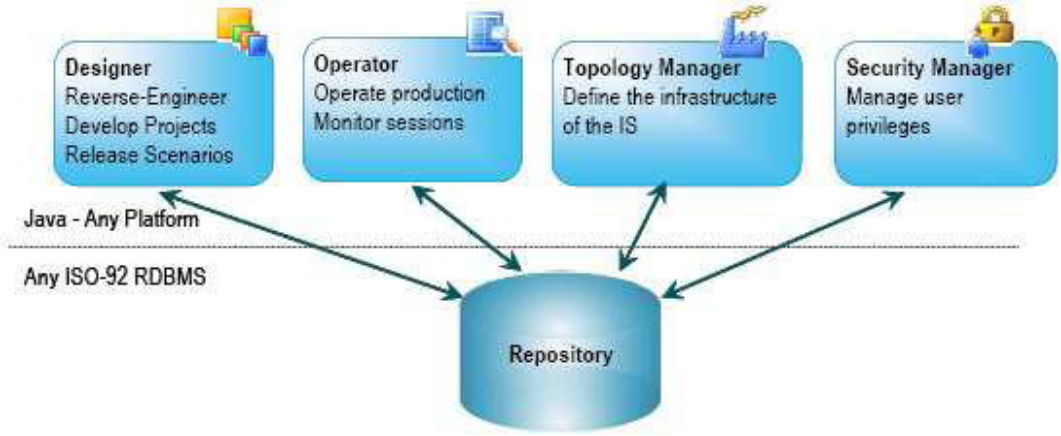
> **Note:** In Release 13.1.1, ODI-RPAS integration includes batch data integration only. Support for real-time data transfer/replication is not included.

ODI Architectural Overview

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

Graphical Modules

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



Graphical Modules in ODI

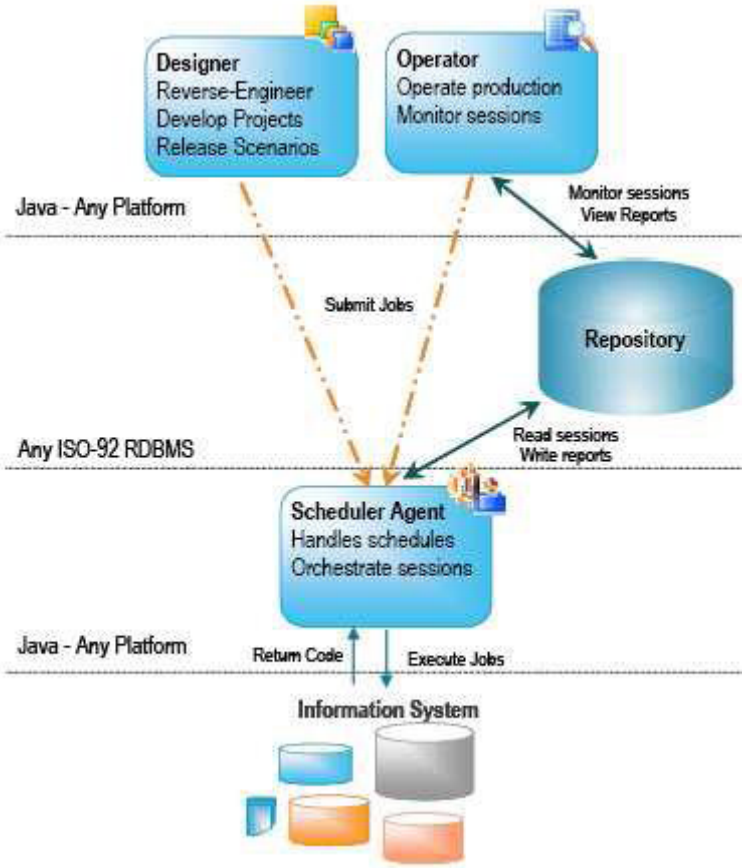
- **Designer** defines declarative rules for data transformation and data integrity. All project development takes place in this module; this is where database and application metadata are imported and defined. The Designer module uses metadata and rules to generate scenarios for production. This is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

All modules store their information in the centralized repository.

Run-time Components

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

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

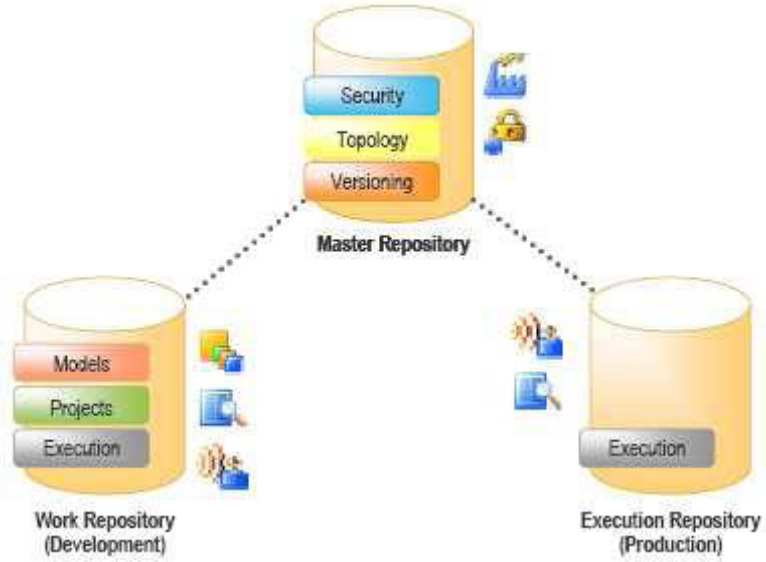


Run-time Components in ODI

The Repositories

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

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



Repositories

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

A work repository stores information for the following:

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

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

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

Using ODI with RPAS

ODI Installation

The ODI graphical modules need to be installed on the users' working computers (Windows PCs or UNIX work stations). Here, the term "users" refers to individuals creating, managing, and running ODI projects. All required JDBC drivers (including RPAS JDBC driver) must be installed in the ODI driver directory in order for the graphical modules to connect to the ODI repositories, source, and target databases.

The ODI Agents should be installed on the machines hosting the source or target databases if the data integration jobs involve running database-specific (non-SQL) utilities such as RPAS loadMeasure.

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

ODI Configuration

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

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

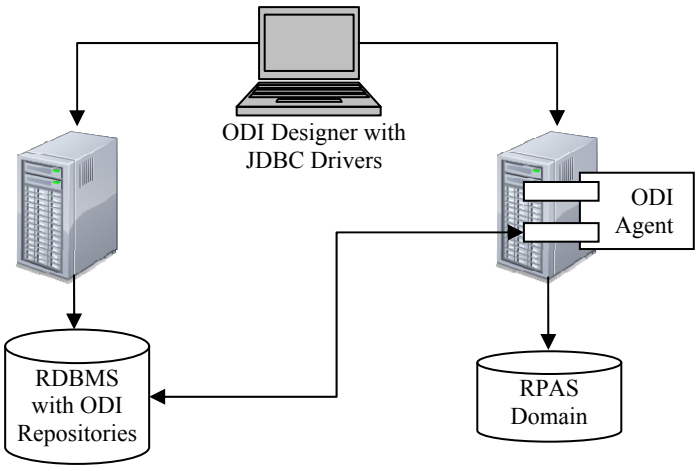
Deployment of ODI and RPAS

ODI and RPAS can be deployed in two different ways. In one deployment, the data transfer is supported between a relational database and RPAS. In a second type of deployment, the data transfer is between two RPAS domains.

ODI Agent must be installed and run on the RPAS server if the RPAS domain is a target datastore. ODI Agent is not required, but is recommended at the RDBMS server (in order for the data transfer job to take advantage of database-specific utilities).

Data Integration between a Relational Database (such as Oracle DBMS) and the RPAS Domain

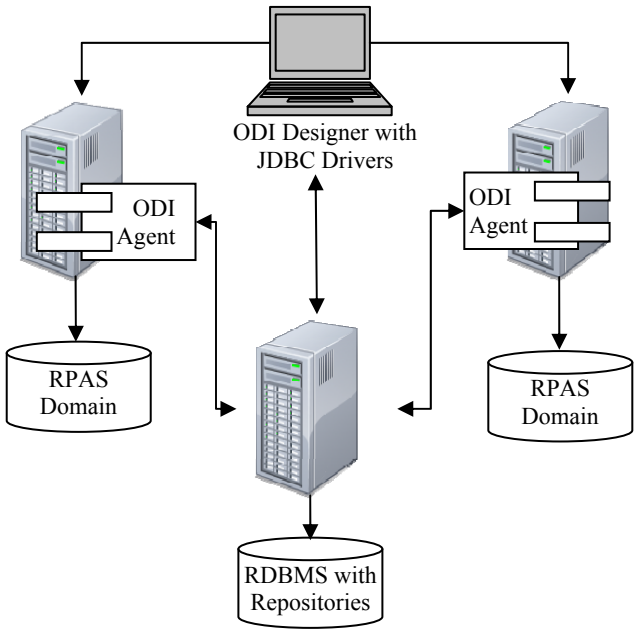
In this configuration, one of the data sources is an RPAS domain, the other is a RDBMS, which can serve as ODI repositories. ODI Agent is required to at least be installed on the RPAS server.



Data Integration between Relational Database and RPAS Domain

Data Integration between Two RPAS Domains

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



Data Integration between Two RPAS Domains

Installing ODI Agents with RPAS

As mentioned before, the ODI Agents need to be installed on the server running data transfer utilities.

In an ODI-RPAS deployment, RPAS load utilities are used for data loading. ODI Agent must be installed and run on the server where the RPAS domain is located. The agent must have "write" access to the input directory of the domain. The agent executes the RPAS load utilities for importing data to the domain.

Installation of the ODI Agent is as easy as copying directories. For more information, refer to the "Manual Installation of Oracle Data Integrator" section in the *ODI Installation Guide*.

The listener agent can be started from ODI's bin directory:

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

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

Installing RPAS Technology and Knowledge Modules

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

1. Loading measure data using loadMeasure utility
2. Loading hierarchy data

Note that there is no RPAS-specific knowledge module for exporting data from RPAS since the SQL interface utilizing the RPAS JDBC driver is used.

ODI and RPAS installation requires installation of RPAS technology and knowledge modules. In order to install RPAS technology, copy the definition file "TECH_RPAS JDBC.xml" to ODI's import/export directory (named "impexp") on the machine where the ODI graphical modules are installed. In a follow up step, RPAS technology must be imported into Topology Manager by using ODI's "import" functionality.

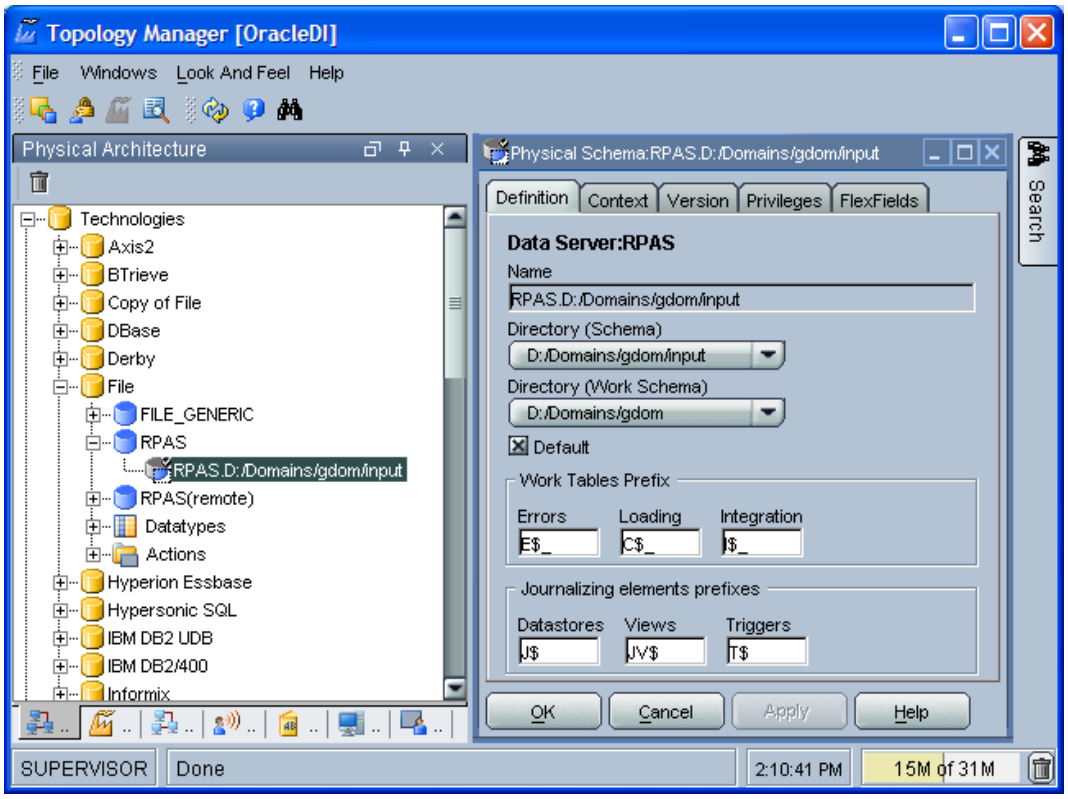
To install the RPAS Knowledge Modules, copy the knowledge module XML files to ODI's import/export directory on the machine where the ODI graphical modules are installed.

Creating Agents and RPAS Data Servers/Schemas

In ODI Topology Manager, logical and physical agents need to be created to connect to the listener agents that run on the machines where RPAS domains are located.

Two types of physical data servers should be created for RPAS: one under the pre-existing FILE technology, and another under the RPAS JDBC Technology.

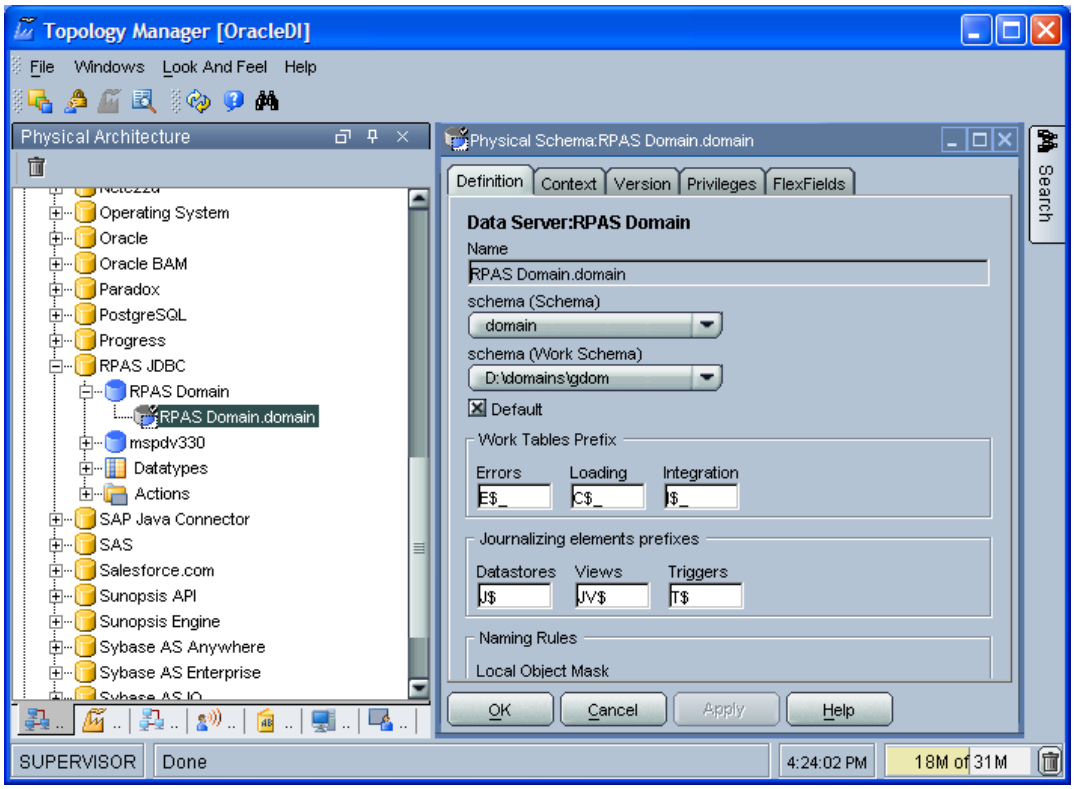
The FILE/RPAS data server is used to receive data in flat file format that serves as input to the RPAS domain. This is the data server needed for the two RPAS integration knowledge modules (IKM) that use RPAS load utilities. The "Directory(Schema)" of this data server should be set to the RPAS domain's input directory, as shown in the following figure.



Topology Manager Screen Showing RPAS Data Server Definition

The RPAS JDBC data server is primarily used for the following purposes:

1. Reverse engineering RPAS schemas (for creating data models in ODI Designer).
2. Data source is an RPAS domain which implies using RPAS JDBC driver.



Topology Manager Screen Showing RPAS Domain Data Server Definition

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

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

It is recommended that multiple Contexts be created in Topology Manager: one for development environment, one for QA, and probably one for production.

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

Development Considerations

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

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

   In this scenario, it is required that the development, QA, and production domains must have conforming RPAS measures and hierarchies (same structure and dimensions). Other non-RPAS databases must also have conforming schema structures across development, QA, and production environments. For best portability, Oracle Retail recommends that the development environment and the production environment use the same names for ODI logical schemas.
2. The development team has access to production databases, meaning a production Context can be set up in the ODI instance in which the developers work.

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

Recommendations for Designing Data Integration Interfaces

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

The RPAS JDBC Driver

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

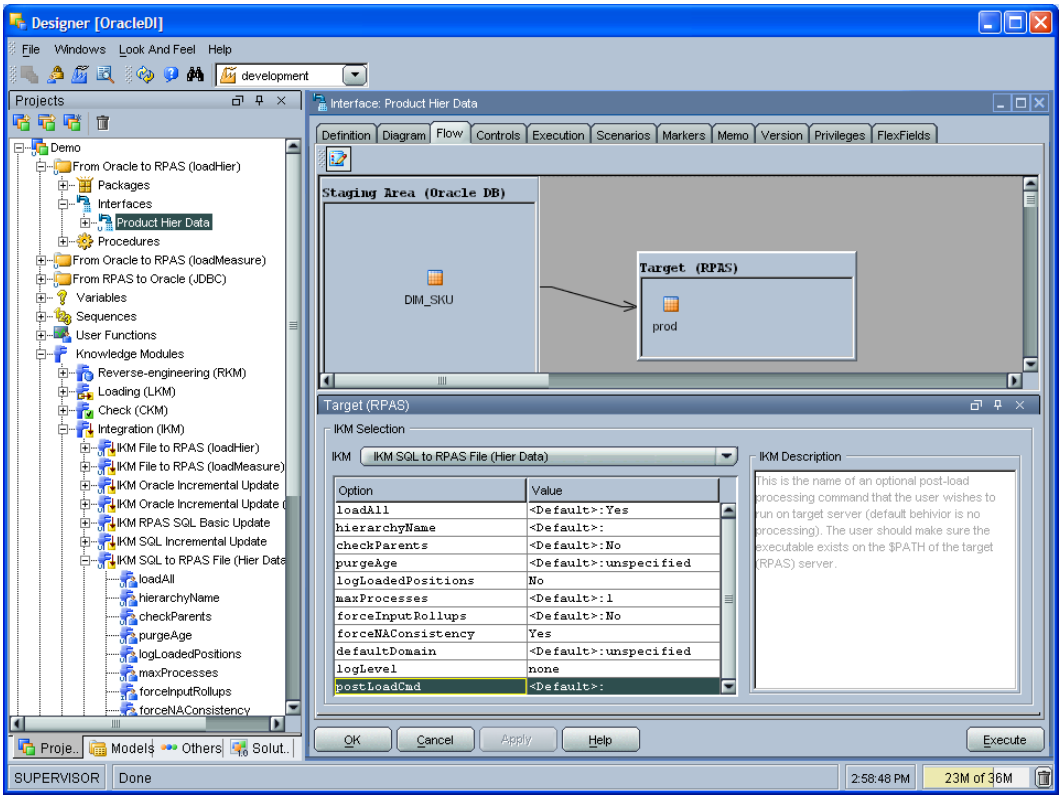
The RPAS Specific Knowledge Modules

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

Loading Hierarchy Data

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

This knowledge module implements options to support all the parameters of the RPAS loadHier utility, and a "postLoadCmd" option to support the last optional step, as shown in the following figure. For description of loadHier parameters, refer to the "Loading Hierarchies – loadHier" section in this guide.

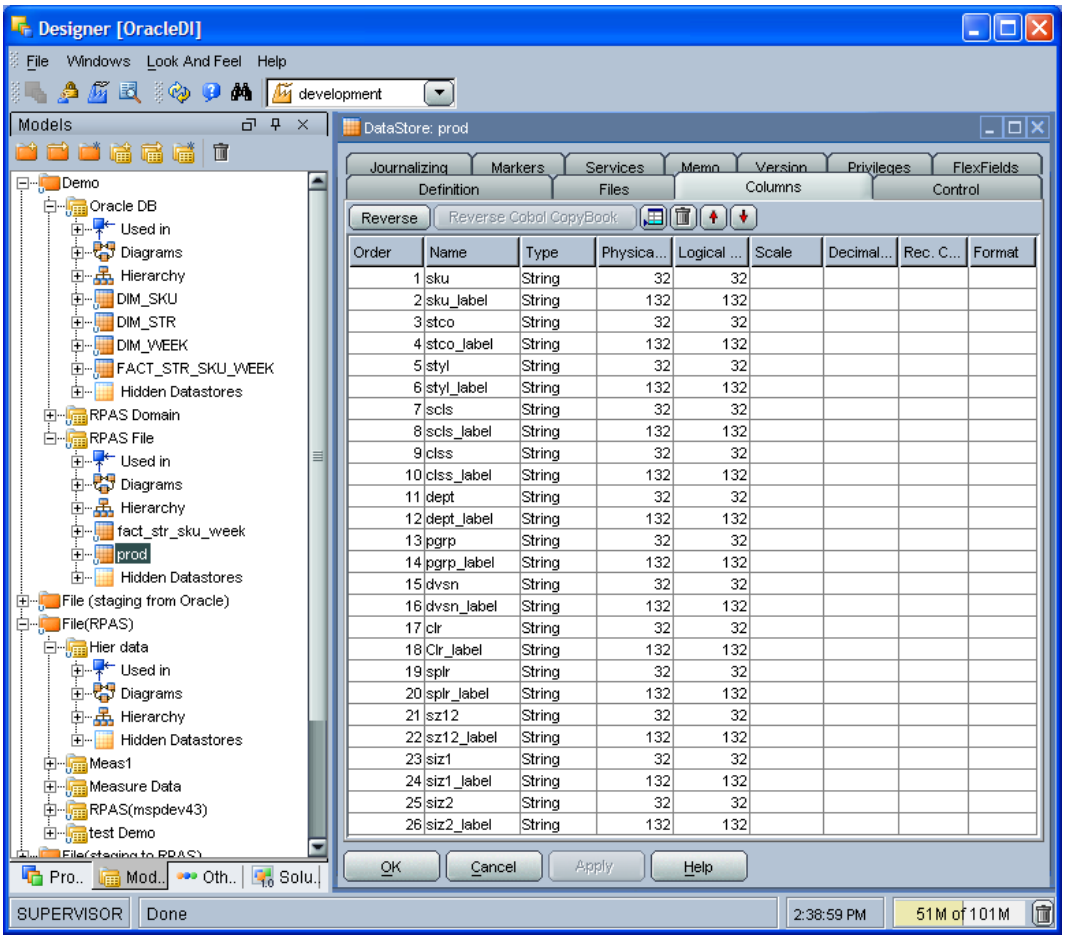


Designer Screen Showing loadHier Options

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

When creating the data model for the target datastore, the table must be named after the hierarchy name defined in RPAS. The table must have one header line, and must contain columns for all dimensions of the hierarchy being loaded with two columns for each dimension: one for position name and one for position label. The position name columns must be named after the dimension names of the hierarchy, the name of label columns should be the name of dimension appended with "_label".

The following figure shows a column definition of “prod” table (“prod” is the name of product hierarchy in a test RPAS domain).



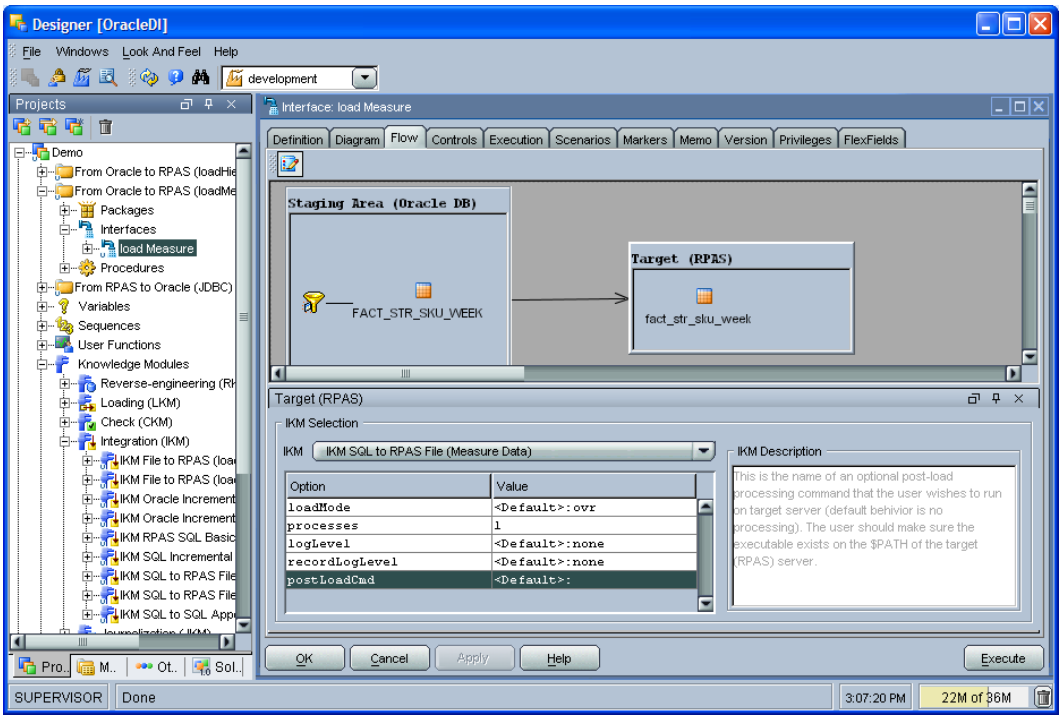
Designer Screen Showing Column Definition of prod Table

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

Loading Measure Data

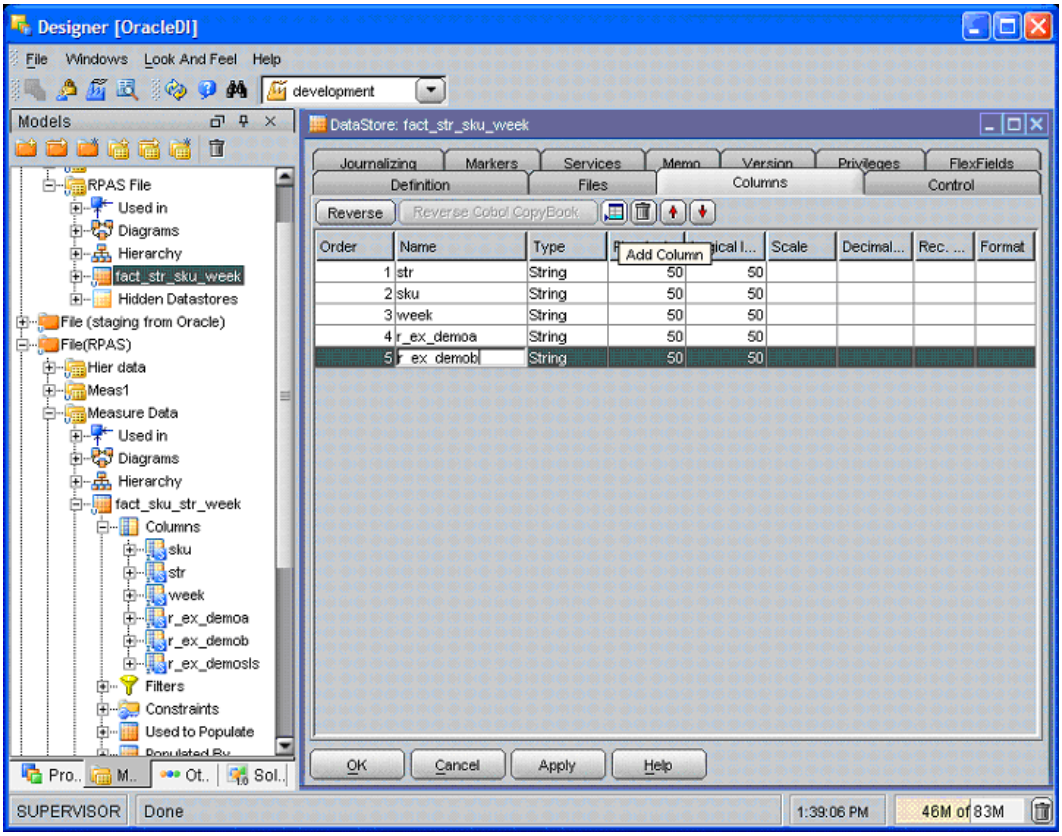
An integration knowledge module “IKM SQL To File (Measure data)” is used for batch measure data transfer. The SQL To File knowledge module was designed for bulk data transfer, with source being an RPAS domain or a relational database and target being an RPAS domain. It uses SQL query to extract data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

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



Designer Screen Showing loadMeasure Options

In order to use this IKM, the data model of the target database should be a “File” type table. The CSV file format must be used. There is no restriction on the name of the file. The file must have a header line (for column names). The file (or table) should contain dimension columns and measure columns. The name of the dimension columns should be consistent with the RPAS dimension names, and the name of the measure columns should be the same as RPAS measure names. The combination of the dimension columns should correspond to the load intersection or the base intersection of the measures contained in this table.



Designer Screen Showing Column Definitions of Target Measure Data Table

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

Other Recommendations

When reading from partitioned RPAS domain, to achieve optimal performance, Oracle Retail suggests multiple interfaces be developed, each interface reading from different partitions. At the execution time, all of the interfaces can run in parallel. The number of interfaces should be determined based on the number of partitions, number of processors of the server, and other loading on the server.

New knowledge modules can be developed based on "IKM SQL To File (measure data)" and "IKM SQL To File (hier data)" to take advantage of the performance-optimized database specific export utilities of source databases. For example, if the source database is Oracle DBMS, instead of using SQL query, the developer could consider Oracle's export utility if suitable, and a new IKM could be created by slightly modifying the existing RPAS specific IKM.

Limitations and Potential Issues

- Release 13.1.1 of RPAS does not support real-time data integration with RPAS domains.
- ODI repositories cannot be created in an RPAS domain. For data integration across multiple RPAS domains, an additional relational database is required for storing the repository.
- RPAS JDBC Driver does not support SQL INSERT. RPAS Data/hierarchy load utilities must be used when the target database is an RPAS domain.
- When loading measure data into an RPAS domain using the "IKM SQL to File" knowledge module, data can be loaded at the load intersection or base intersection only.
- If a data transfer (to RPAS) job fails, manual cleanup might be needed in the RPAS domain input directory.

16

Publishing Measure Change Events

Introduction

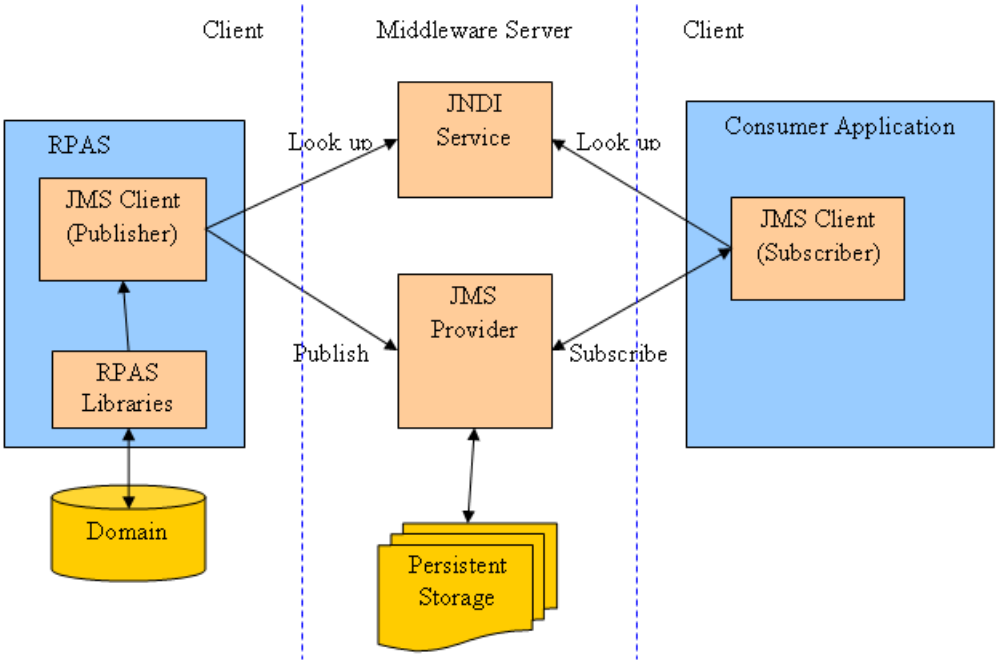
Event driven planning requires the ability to identify events when they arise. This includes events that result from changes in plans and those that arise because of advancement in the planning activity, for example, approval of a plan or creation of new items. The ability to get notification of the 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 to 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:

1. MACE: all left hand side measures in the expression or expressions of the rule group specified in the command line.
2. Workbook Commit: all left hand side measures in the commit rule group.
3. loadMeasure: all successfully loaded measures.
4. loadHier: measures whose base intersection contains at least one affected dimension. "Affected" means adding and deleting one or more **logical** positions.
5. 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.

The following diagram demonstrates a JMS system in the context of RPAS. The JMS system consists of four basic components:

- JMS Provider: The central daemon which 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.



JMS System in RPAS

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
```

- "true" means inclusion in monitoring.
- "false" is used to exclude a subject/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, consists 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 their 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 which can contain any of the macros listed in the following table. The macros are replaced by actual values at run time.

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.

Following 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.

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 …`	name/value pairs. If the configuration file is not specified, the name/value pairs 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:

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 needs to be copied to the config directory of the domain.
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, 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"

Setting Name	Description	Required	Value
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.

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.

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

Property Name	Description	Required	Value
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.
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 [...]]]

- {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 the "Configuring the RPAS JMS Publisher"section for details.

The following table lists all the 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.

Property Name	Description	Required	Value
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
```

17

In-Context Launch

In-Context Launch (ICL) is an RPAS feature that provides retailers with a planning centered, integrated UI solution for responding to events by adjusting the affected plans within the context of the changed circumstances and interacting plans. A planner's workspace can be developed where 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.

RPAS launches a planning application 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. Launching the planning application involves the following steps:

1. Start up the RPAS client, if it is not already running.
2. Log in the user, if the user is not already logged in.
3. Create the workbook, if a new workbook is being built.
4. Open the new or existing workbook, 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 Oracle Retail Workspace or other Web portal.

Launching RPAS

A system integrator is responsible for creating a Workspace or Web-based application that can launch RPAS. Creating a Workspace application involves the following steps:

1. Configure the navigational pane in Workspace 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 launch RPAS in-context.
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 in-context.

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 ICL is implemented as an extension of the RPAS Web Launch feature. As a prerequisite, 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 "RPAS Web Deployment" chapter in the *RPAS Installation Guide*.

System integrators must design the report page to have a link, or other web GUI widget, which invokes client-side JavaScript code to initiate an RPAS ICL. The page embeds a Java applet from RPAS Web Launch (RWL Applet).

Note: Java 1.6_10 or above is required.

The applet performs the following operations:

1. Find the required RPAS client installation on the user's workstation, if it was installed by the applet in a previous Web session.

   Note that a standalone RPAS client installation is not recognized because the installation is tracked by a browser cookie. However, the applet can be forced to use a preinstalled client by setting a parameter for the applet.
2. Download and install the appropriate version of the RPAS client if the required RPAS client installation cannot be found or a new build of the client is available on the Web server.
3. Start up the RPAS client.
4. Route the ICL request to the RPAS server through the RPAS client and wait for a response. The response can be a success or failure code, and a corresponding message.

   The applet makes a call back to a JavaScript function to return the response. It is the responsibility of the system integrators to provide code to handle the response in the JavaScript function.

Embedding an RWL Applet on a Web Page

System integrators can embed the applet in a Web page using the regular HTML `<Applet>` tag. The following attributes are required in the `<Applet>` tag:

Attribute	Description	Value
code	Applet class name	`com.retek.mdap.client.launch.LaunchApplet.class`
codebase	URL location to download the applet archive	The URL must point to the RPAS Web Launch Web server instance, for example: `"http://mspdev43.us.oracle.com:7777/RPAS4/"`
archive	Applet archive file name	`mdap.jar`
mayscript	Enable JavaScript interaction	NA

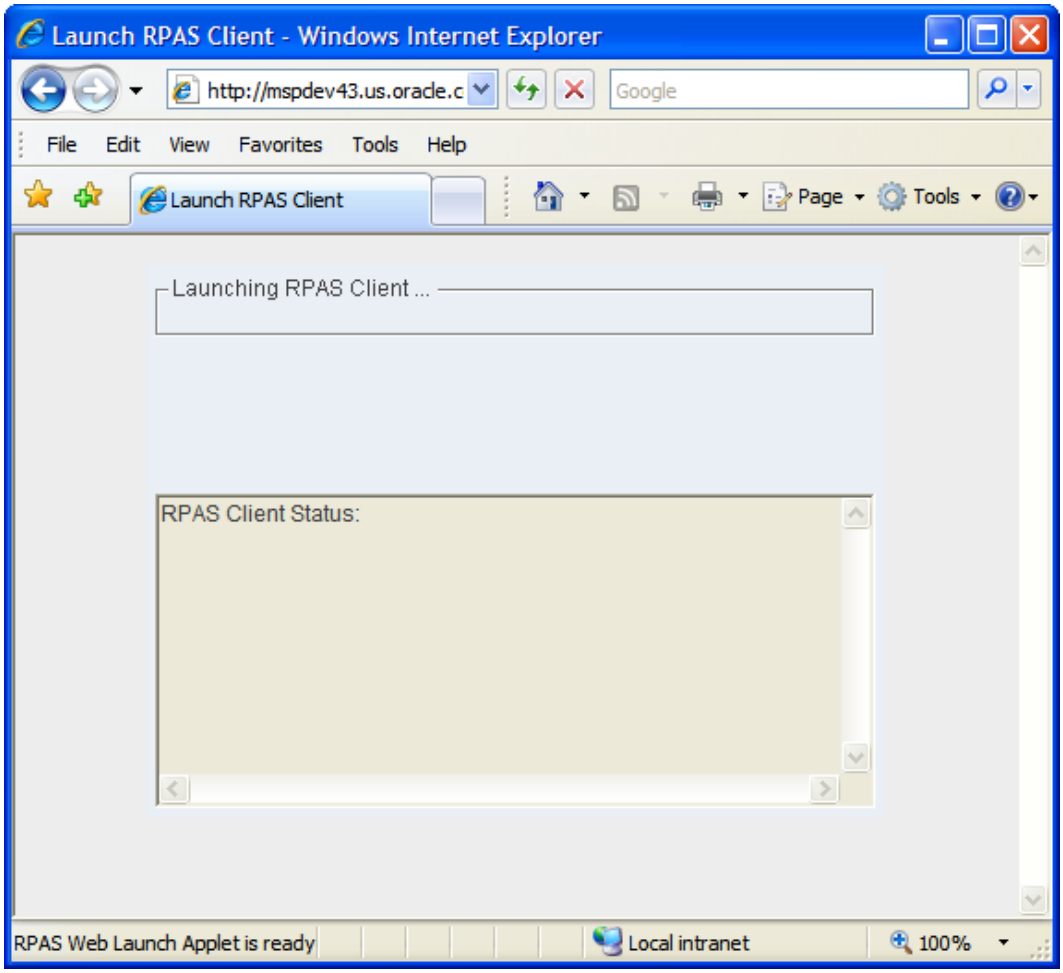
The following parameters are used to control the behavior of the RWL Applet:

Parameter	Description	Required	Example
AutoStart	Launch the client automatically when the applet starts. Must be set to false for ICL.	Yes	Must be set to false if the launch is initiated by JavaScript code.
defaultInstallDir	Default installation location for the RPAS client. The client version number is automatically appended to the end of this string.	No	`C:\Oracle RPAS Client`
Debug	Debug flag. If true, more logging information is printed in the Java Console of the Web browser.	No	true
launchPreinstalledOnly	Launch the preinstalled RPAS client only. Do not install an RPAS client. The preinstalled path is determined by the server side setting in the configuration file `clientPath.txt`. For more information, refer to the *RPAS Installation Guide*.	No	false
supportMultipleVersions	Support multiple RPAS client versions on the same workstation. Must be set to true for ICL.	Yes	Must be set to true.
WebLaunchBase	The URL for the Web Launch Servlet. If not specified, the applet will generate a URL by appending "/web" to it's codebase property.	No	`http://mspdev43.us.oracle.com :7777/RPAS4/web`

It is recommended that the applet is embedded with the height and width set to zero. When the height and width are set to non-zero values, a text area is shown that gets updated with status information from the RPAS client.

The following figure shows the applet UI if the RWL Applet is embedded in the Web page with non-zero width and height. The message window in the applet is updated with status information from the RPAS client. Status information includes the success or failure of building or opening a workbook and any error messages returned by RPAS.

It is assumed that when using non-zero height and width, system integrators appropriately size the applet in order for the UI to be user-friendly. For example, a width and height of 5 is unrealistic for any user interface returning messages to the end-user.



UI of the RWL Applet when Height and Width Values are Non-zero

Using JavaScript with the RWL Applet

System integrators embed the RWL Applet in an HTML web page. The launch context can be passed to the applet by the JavaScript-enabled active content of the Web page. Real-time status is passed back. The communication between the applet and the active content is achieved by a JavaScript invocation of an applet method.

To launch an instance of an RPAS client, the active content makes a JavaScript call:

```
document.appletName.launchRPASClient(id, xmlstring [, user, password]);
```

- `id` - RPAS client process identifier, which can be any string.
- `xmlstring` - launch context in XML format encoded by Base64 scheme. Base64 encoding is a popular scheme used in e-mail contents, readily available in JavaScript public domain source code.

  For purposes of quick manual testing or prototyping, the MIME Tools plug-in in Notepad++ 4.8.5 and above may be used to encode any text to Base64 and back.
- `user` and `password` - optional arguments. For details on ICL supported mechanisms for user authentication, see the "User Authentication" section.

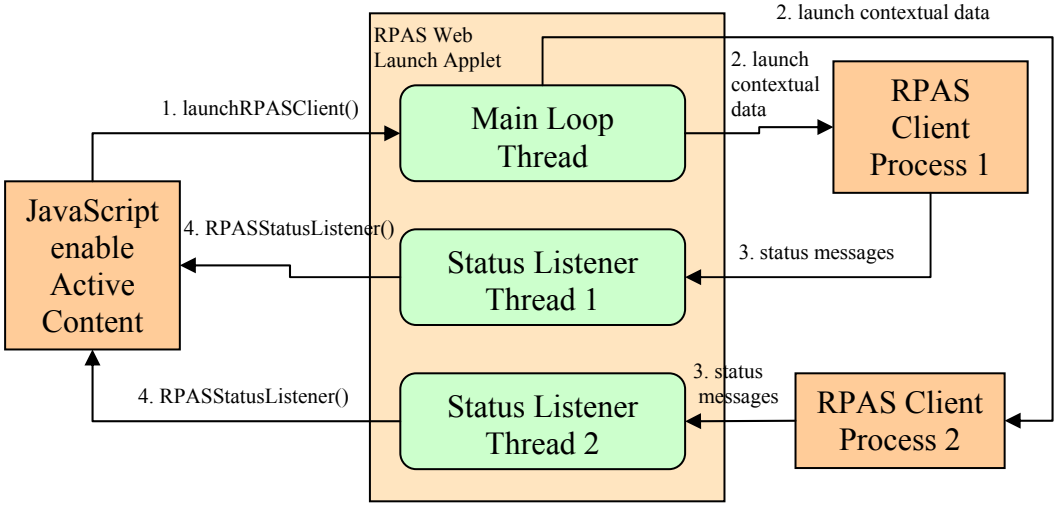
The applet notifies active content regarding the status of the RPAS client through a JavaScript callback function. The following JavaScript function may be defined if a custom callback is desired:

```
function RPASStatusListener(id, code, message)
{
   …
}
```

- `id` - same RPAS client process identifier previously passed by the launchRPASClient()call.
- `code` - status code (0 – success, 1 – failure, -1 – applet exception)
- `message` - a descriptive status message.

If the callback function is not defined, RWL Applet displays status messages on the status bar of the browser in the following format: "id – [OK | Error | Exception] – message".

The following figure shows the interactions between Active Content, RWL Applet, and RPAS client processes. Note that one embedded RWL applet can be invoked multiple times using JavaScript to launch multiple RPAS client processes using different context XML data. The figure shows the case where two RPAS client processes are launched.



Interaction between RWL Applet, Active Content of the Web Page, and RPAS Client

Since the status listener function `RPASStatusListener(id, code, message)` can be invoked multiple times by multiple status listener threads in the applet, the JavaScript developer should use the function `id` argument to identify where the status message came from and act accordingly.

The Web page must stay active to enable the communication between the RPAS client processes and RWL Applet. The Web page can become inactive, for example, if the user clicks the Back button of the Web browser or closes the active tab or window.

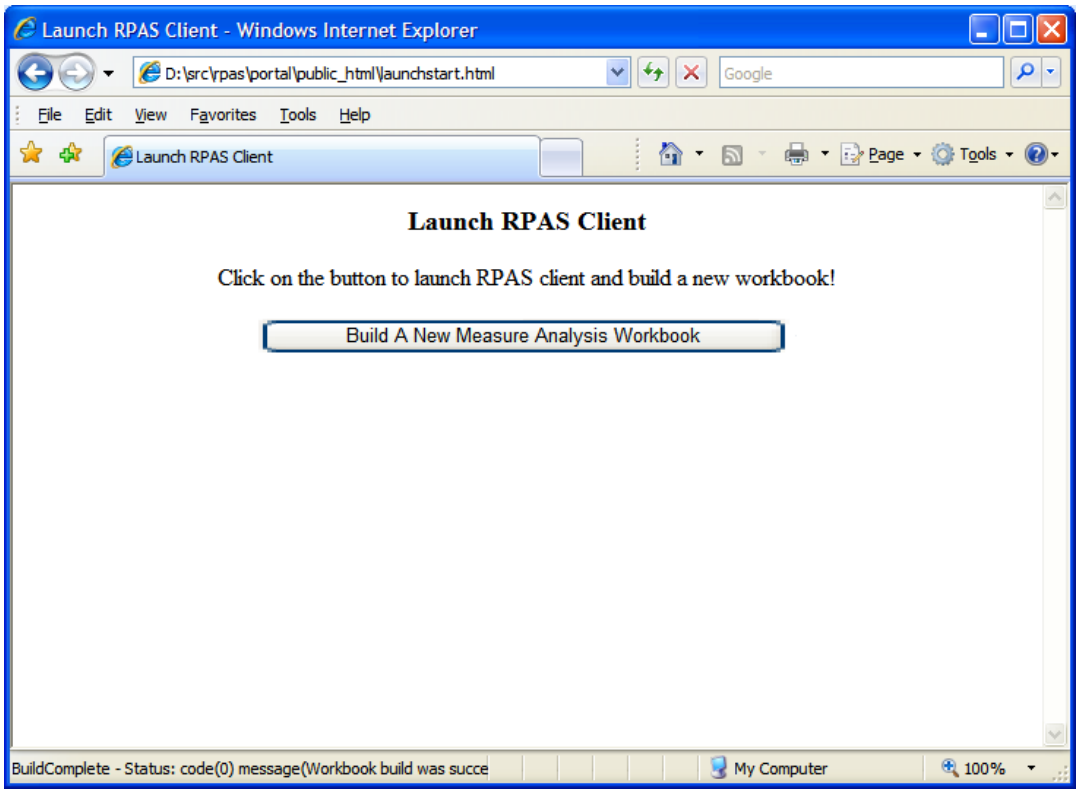
If the Web page becomes inactive, the communication channel between the applet and client process is terminated and cannot be recovered. If this happens, the RPAS client process becomes orphaned and then acts like a regular RPAS client. It will continue to function, but its status will not be displayed on the Web page.

The following sample code demonstrates the integration of RWL Applet and JavaScript-enabled active content. Note that the width and height of the applet are set to zero so that the applet is hidden on the web page. The JavaScript function `launchRPASClient()` simply invokes the corresponding applet method. The status callback function `RPASStatusListener()` is implemented to display the status information in the Web browser status bar which is generally at the bottom of the browser window.

```
<HTML>
<HEAD>
<META http-equiv=Content-Type content="text/html; charset=UTF-8">
<TITLE>Launch RPAS Client</TITLE>
</HEAD>
<SCRIPT>
var idcomplete = "BuildComplete";
var xmlcomplete =
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiID8+DQo8UlBBU0xh\
dW5jaENvbnRleHQ+ICAgDQo8U2VydmVyPg0KCTxkZXNjcmlwdGlvbj5UZXN0RG9t\
YWluPC9kZXNjcmlwdGlvbj4NCiAgICA8cHJvZmlsZT5wvcHJvZmlsZT4N\
Cgk8ZGVidWc+ZmFsc2U8L2RlYnVnPg0KICAgIDxkYkRhZW1vblBvcnQ+MTIzNDg8\
L2RiRGFlbW9uUG9ydD4NCgk8ZGJTZXJ2ZXJIb3N0TmFtZT5tc3BkZXY0My51cy5v\
cmFjbGUuY29tPC9kYlNlcnZlckhvc3ROYW1lPg0KCTxkYlNlcnZlclBvcnRdGFy\
dD4xMDI1PC9kYlNlcnZlclBvcnRdGFydD4NCgk8ZGJTZXJ2ZXJQb3J0RW5kPjY1\
NTM1PC9kYlNlcnZlclBvcnRFbmQ+DQoJPGRlZmF1bHREYlVzZXIgLz4NCgk8ZGVm\
YXVsdERiUGFzc3dvcmQgLz4NCgk8ZG9tYWluUm9vdERpcmVjdG9yeSAvPg0KPC9T\
ZXJ2ZXI+DTxDbGllbnQ+DSAgIDx2ZXJzaW9uPjEzLjEuMDwvdmVyc2lvbj4NPC9D\
bGllbnQ+DQo8RG9tYWluPg0KCTxkZXNjcmlwdGlvbj5UZXN0RG9tYWluPC9kZXNj\
cmlwdGlvbj4NCgk8ZGlyZWN0b3J5TmFtZT4vdm9sLm5hcy9ycGFzX3NlL2h1YW5n\
bWEvZG9tYWlucy9UZXN0RG9tYWluPC9kaXJlY3RvcnlOYW1lPg0KICAgPHVzZXI+\
YWRtPC91c2VyPg0KCTxwYXNzd29yZD5hZG08L3Bhc3N3b3JkPg0KPC9Eb21haW4+\
DQo8V29ya2Jvb2s+DQoJPGJ1aWxkPg0KCQk8dGVtcGxhdGVtcGxhdGVOYW1lPk1FQVNVUkVB\
TkFMWVNJUzwvdGVtcGxhdGVOYW1lPg0KCQk8V2l6YXJkUGFnZXM+DQoJCQk8V2l6\
YXJkUGFnZSBpZD0iRXh0cmFNZWFzdXJlc1BhZ2UiIGZhbGxiYWNrPSJub25lIj4N\
CgkJCQk8TGlzdGJveCBuYW1lPSJsYjEiPg0KCQkJCQk8dmFsdWU+DQoJCQkJCQk8\
c2VsZWN0aW9uPlfRVhfREVNT0E8L3NlbGVjdGlvbj4NCiAgICAgICAgICAgICAg\
ICAgIDxzZWxlY3Rpb24+Ul9FWF9ERU1PQjwvc2VsZWN0aW9uPg0KICAgICAgICAg\
ICAgICAgICAgPHNlbGVjdGlvbj5SX0VYX0RFTU9DPC9zZWxlY3Rpb24+DQoJCQkJ\
CTwvdmFsdWU+DQoJCQkJPC9MaXN0Pg0KCQkJPC9XaXphcmRQYWdlPg0KCQk8\
PFdpemFyZFBhZ2UgaWQ9IkRBWSIgZmFsbGJhY2s9Im5vbmUiPg0KCQkJCTxTdXBl\
clRyZWUgbmFtZT0idHJlZTEiIGZhbGxiYWNrPSJhbGwiPg0KCQkJCQk8dmFsdWU+\
DQoJCQkJCQk8ZGltIG5hbWU9IldFRUsiPg0KCQkJCQkJCTxzZWxlY3Rpb24+VzAx\
XzE5OTc8L3NlbGVjdGlvbj4NCgkJCQkJCTxkc2VsZWN0aW9uPlcwMV8xOTk4PC9z\
ZWxlY3Rpb24+DQoJCQkJCQk8JPHNlbGVjdGlvbj5XMDFfMTk5OTwvc2VsZWN0aW9u\
Pg0KCQkJCQkJCTxzZWxlY3Rpb24+VzAxXzIwMDA8L3NlbGVjdGlvbj4NCgkJCQkJ\
CTwvZGltPg0KCQkJCQk8L3ZhbHVlPg0KCQkJCTwvU3VwZXJUcmVlPg0KCQkJCTxT\
CQkJCQk8ZGVmYXVsdD4NCgkJCQkJCTxkaW0gbmFtZT0iREFZIj4NCgkJCQkJCQk8\
CQkJCQk8c2VsZWN0aW9uPjE5OTcwMTA4PC9zZWxlY3Rpb24+DQoJCQkJCQk8L2Rp\
bT4NCgkJCQkJPC9kaW0gbmFtZT0iREFZIj4NCgkJCQkJPC9kaW0gbmFtZT0iREFZ\
PC9YWx1ZT4NCgkJCQkJPC9kaW0gbmFtZT0iREFZIj4NCgkJCQkJPC9kaW0gbmFt\
CTxXaXphcmRQYWdlIGlkPSJTS1UiIGZhbGxiYWNrPSJub25lIj4NCgkJCQk8U3Vw\
ZXJUcmVlIG5hbWU9InRyZWUxIj4NCgkJCQkJPHZhbHVlPg0KCQkJCQkJPGRpbSBu\
YW1lPSJDTFNTIj4NCgkJCQkJCQk8c2VsZWN0aW9uPjAyMTwvc2VsZWN0aW9uPg0K\
CQkJCQkJPC9kaW0+DQoJCQkJCTwvdmFsdWU+DQoJCQkJPC9TdXBlclRyZWU+DQoJ\
```

```
CQk8L1dpemFyZFBhZ2U+DQoJCTwvV2l6YXJkUGFnZXM+DQoJCTxzYXZlPg0KCQkJ\
PGxhYmVsPlRlc3R3YjwvbGFiZWw+DQoJCQk8YWNjZXNzPkdyb3VwPC9hY2Nlc3M+\
DQoJCQk8Z3JvdXA+QURNSU48L2dyb3VwPg0KCQk8L3NhdmU+DQoJPC9idWlsZD4N\
CjwvV29ya2Jvb2s+DQo8L1JQQVNMYXVuY2hDb250ZXh0Pg0K";
function launchRPASClient(id, xmlstring)
{
   document.LaunchApplet.launchRPASClient(id, xmlstring);
}

function RPASStatusListener(id, code, message)
{
   status = id + " - Status: code(" + code + ") message(" + message + ")";
}
</SCRIPT>
<BODY>
<CENTER>

<H3>Launch RPAS Client</H3>
Click on the button to launch RPAS client and build a new workbook!
<P>
<INPUT type="button" value="Build A New Measure Analysis Workbook"
   onClick = "launchRPASClient(idcomplete, xmlcomplete)">

<APPLET name="LaunchApplet"
        code="com.retek.mdap.client.launch.LaunchApplet.class"
        codebase="http://mspdev43.us.oracle.com:7777/RPAS4/"
        archive="mdap.jar"
        align="center" width="0" height="0" mayscript VIEWASTEXT>

<PARAM name="AutoStart" value="false">
<PARAM name="supportMultipleVersions" value="true">
<PARAM name="launchPreinstalledOnly" value="false">
<PARAM name="Debug" value="true">
<PARAM name="defaultInstallDir" value="C:\Oracle RPAS Client">
Your web browser is not currently Java-enabled.
</APPLET>
</CENTER>
</BODY>
</HTML>
```

The following figure shows this sample code rendered by a Web browser. Note that the status bar shows the current status of the RPAS client.



Example HTML Code Rendered by a Web Browser

User Authentication

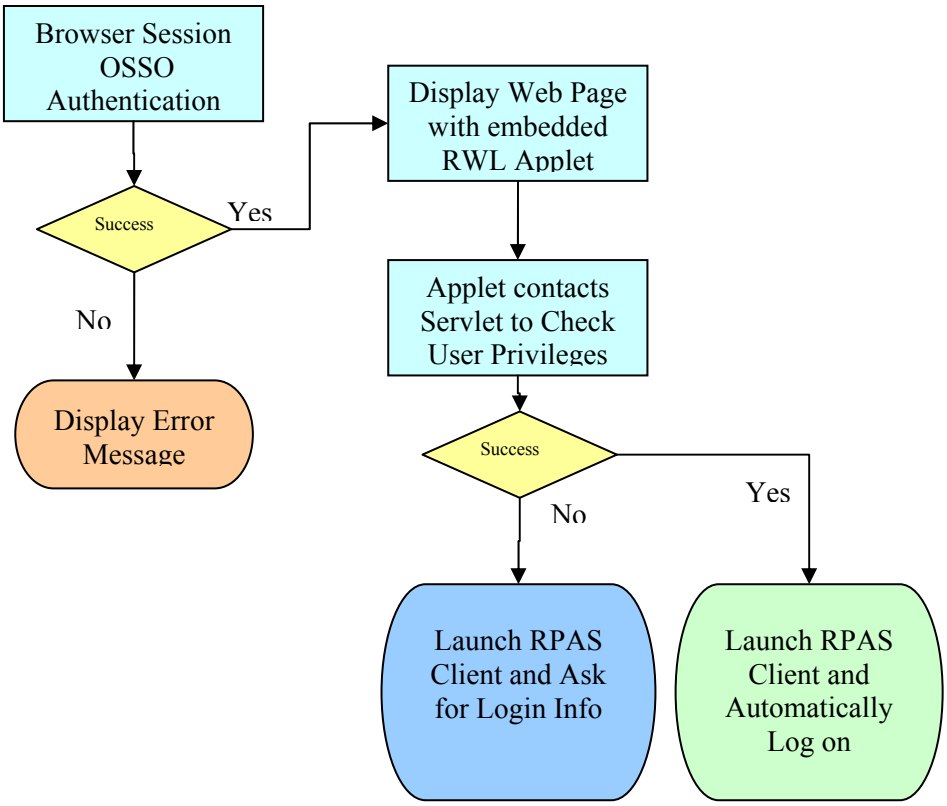
This section describes Oracle Single Sign-On (OSSO) and non-OSSO authentication of users.

OSSO-Based Authentication

If the HTML active content is protected by OSSO, it can be assumed that the user has already been authenticated when the applet is started. However, it cannot be assumed that the user has the privileges to launch the RPAS client. The privileges can only be confirmed by RWL Servlet. The applet, therefore, makes a query to the servlet to confirm the privileges of the user before launching the RPAS client.

When OSSO is being used, a user name and password is not needed and should not be used for the call to `document.appletName.launchRPASClient()`. System integrators should rely on OSSO credentials and mechanisms to authenticate the user.

The following figure shows the flow of OSSO support for ICL.



OSSO Support Flowchart for ICL

Non-OSSO Authentication

When OSSO is not used, the authentication for the RPAS client login is done by RPAS itself. There are several ways to pass user credentials, user name and password, to the RPAS client via the RWL Applet:

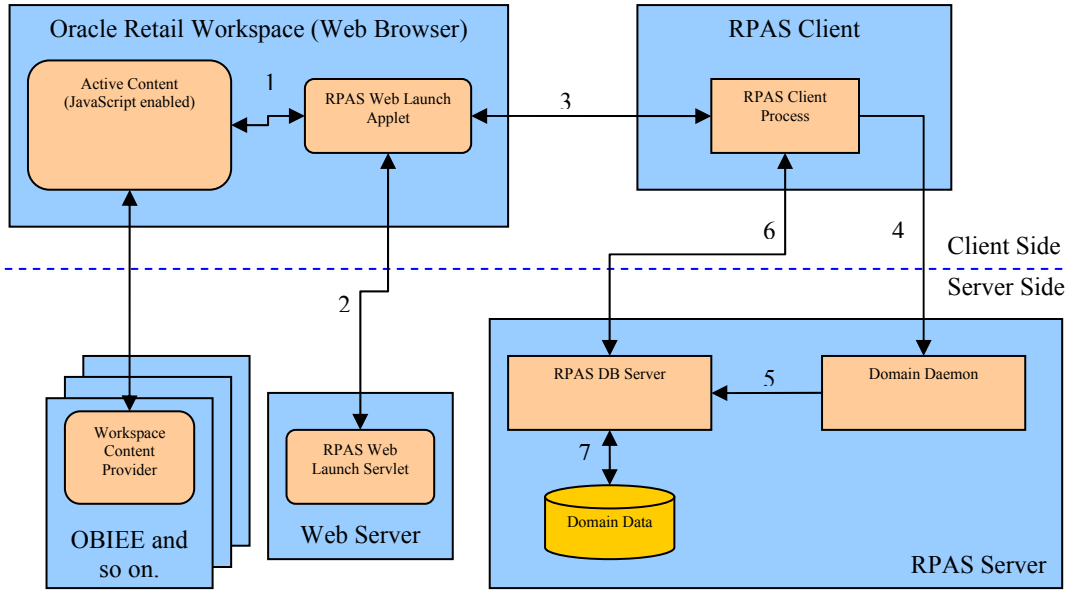
- Directly embed the user name and password in the XML data. The domain section of launch context can contain user and password information.

  Note: This may pose a security issue depending on the lifecycle of the launch contextual data.

- Do not pass the user name and password to the applet. The RPAS client will prompt the user to log in every time the client starts. This may be inconvenient, but may be acceptable if only one RPAS session is needed.
- Use JavaScript code to pass in the user name and password in the applet method call `launchRPASClient(id, xmlstring, user, password)`. The user name and password can be obtained from the user input fields of the Web page. The user only needs to enter the user name and password once for one browser session. Multiple RPAS client sessions can then be launched without entering the user name and password again.

Deploying In-Context Launch

The following diagram shows how an RPAS In-Context Launch enabled application is deployed. Note that 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 Workspace or for any reporting, dashboard or BAM applications. In this release, RPAS ICL 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, ICL can be achieved from any Web-based application.



ICL Deployment Diagram

Use Cases

Launching without Context

A system integrator may want to launch RPAS without any context. This means starting up an RPAS client 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 Workspace 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 an in-context launch. This includes launching an RPAS client 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 in-context 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 datetime of the workbook. These parameters can be specified in the XML context specification to have RPAS open the workbook. For information, see the "Open Workbook" section.

To enable opening the latest workbook from a nightly auto-workbook-build batch, system integrators can provide the label and template name. ICL will automatically default 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 depending on what the end-user is or has been doing in the Workspace or other 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 a report and had drilled down to identify 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 the "Build Workbook" section.

RPAS Launch Context Format

The launch context is described in XML format. It consists of four components: client, server, domain, and workbook.

- The client component identifies the RPAS client to be launched. It is needed to support multiple client versions on the same workstation.
- The server component identifies the server to connect to.
- The domain component identifies the domain to log in to.
- The workbook component contains information for opening or building a workbook.

Client Settings

Setting	Description	Type	Required
version	RPAS client version number. RPAS Web Launch supports multiple client versions. This version number is used to determine which client binary will be launched. If not specified, the default client for RPAS Web Launch is used.	String	No

Setting	Description	Type	Required
allowConcurrentSessions	Allow the same user to open multiple concurrent RPAS client sessions connecting to the same domain without prompting for confirmation.	Boolean	No

Server Settings

Setting	Description	Type	Required
description	Descriptive name for this server configuration.	String	No
profile	Enable profiling for the RPAS client.	Boolean	No
debug	Enable debug logging for the RPAS client.	Boolean	No
dbDaemonPort	The port number of DomainDaemon.	Integer	Yes
dbServerHostName	The host name where DomainDaemon is running.	Integer	Yes
dbServerPortStart	The starting port number of the RpasDbServer listening port range.	Integer	No
dbServerPortEnd	The ending port number of the RpasDbServer listening port range.	Integer	No
defaultDbUser	The default user ID for RpasDbServer.	String	No
defaultDbPassword	The encrypted password for the default user ID.	String	No
domainRootDirectory	The root directory of RPAS domains. If specified, it is prepended to the directoryName in Domain Settings.	String	No

Domain Settings

Setting	Description	Type	Required
description	Descriptive name for this domain configuration. It is displayed as the title of the RPAS client main window.	String	Yes
directoryName	The domain path.	String	Yes
user	The user ID used to log in. If not specified, a login dialog prompts for user input.	String	No
password	The encrypted password for the user ID. If not specified, a login dialog prompts for user input.	String	No

Workbook Settings

The Workbook section has two options to open or build 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 the "Open Workbook" section. For information on building workbook, see the "Build Workbook"section.

XML Schema

The following is the full schema for RPAS Launch Context.

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <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="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="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="StringDataType" />
  <xs:element name="Tree" type="StringSetDataType" />

  <xs:element name="SuperTree">
    <xs:complexType>
      <xs:choice>
      <xs:sequence>
        <xs:element name="value" type="DimDataType"/>
        <xs:element name="default" type="DimDataType" minOccurs="0" />
      </xs:sequence>
      <xs:sequence>
        <xs:element name="value" type="PQDDataType"/>
        <xs:element name="default" type="PQDDataType" minOccurs="0" />
      </xs:sequence>
      </xs:choice>
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="fallback" type="xs:string" use="optional" />
    </xs:complexType>
  </xs:element>

  <xs:element name="open">
    <xs:complexType>
        <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:complexType>
</xs:element>

<xs:element name="build">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="templateName" type="xs:string" />
      <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 réf="WizardPages" />
      <xs:element réf="save" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<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:element name="Client">
  <xs:complexType>
    <xs:sequence>
```

```
        <xs:element name="version" type="xs:string" />
        <xs:element name="allowConcurrentSessions" type="xs:boolean"
minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Server">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="description" type="xs:string" />
        <xs:element name="profile" type="xs:boolean" />
        <xs:element name="debug" type="xs:boolean" />
        <xs:element name="dbDaemonPort" type="xs:integer" />
        <xs:element name="dbServerHostName" type="xs:string" />
        <xs:element name="dbServerPortStart" type="xs:integer" />
        <xs:element name="dbServerPortEnd" type="xs:integer" />
        <xs:element name="defaultDbUser" type="xs:string" />
        <xs:element name="defaultDbPassword" type="xs:string" />
        <xs:element name="domainRootDirectory" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Domain">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="description" type="xs:string" />
        <xs:element name="directoryName" type="xs:string" />
        <xs:element name="user" type="xs:string" minOccurs="0"/>
        <xs:element name="password" type="xs:string" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Workbook">
    <xs:complexType>
      <xs:choice>
        <xs:element ref="open" />
        <xs:element ref="build" />
      </xs:choice>
    </xs:complexType>
  </xs:element>

  <xs:element name="RPASLaunchContext">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Client" />
        <xs:element ref="Server" />
        <xs:element ref="Domain" />
        <xs:element ref="Workbook" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Open Workbook

Description

In any retail planning scenario, a particular workbook or, more likely, 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 a Workspace application can configure and code an in-context opening of the specific workbook.

System integrators can programmatically specify the parameters that RPAS needs to open a pre-built workbook. Programmatically does not imply the use of an API, but implies program application logic that constructs the XML specification that ICL can use to identify and open the desired workbook.

System integrators can identify the target workbook by either of the following ways:

- By the RPAS name of the workbook.
- By 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 alphabetical ordering of the template name.

Specifications

There are two options that can be specified when a workbook is opened.

Option 1

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

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 launch context for opening a workbook by RPAS name:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RPASLaunchContext>
<Server>
    <description>TestDomain</description>
     <profile>false</profile>
    <debug>false</debug>
<dbDaemonPort>12348</dbDaemonPort>
    <dbServerHostName>mspdev43.us.oracle.com</dbServerHostName>
    <dbServerPortStart>1025</dbServerPortStart>
    <dbServerPortEnd>65535</dbServerPortEnd>
    <defaultDbUser />
    <defaultDbPassword />
    <domainRootDirectory />
</Server>

<Client>
     <version>13.1.0</version>
</Client>

<Domain>
    <description>TestDomain</description>
    <directoryName>/vol.nas/rpas_se/huangma/domains/TestDomain</directoryName>
     <user>adm</user>
    <password>…</password>
</Domain>

<Workbook>
    <open>
         <name>t1</name>
    </open>
</Workbook>
</RPASLaunchContext>
```

Here is sample launch context for opening a workbook by label and template name:

```
<RPASLaunchContext>
… …  (Same as above, omitted here.)
<Workbook>
    <open>
         <label>wb2</label>
         <templateName>MEASUREANALYSIS</templateName>
    </open>
</Workbook>
</RPASLaunchContext>
```

Build Workbook

Description

RPAS users generally build workbooks through a manual process using RPAS Wizard Pages from workbook templates. This process is automated in ICL by specifying the contents and selections of the wizard pages in the launch context.

The automated process enables system integrators to programmatically specify the parameters that RPAS needs to build a new workbook. The integrator can program RPAS web launch to build virtually any workbook from any workbook templates simple or complex, by constructing an XML string according to the launch context 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 workbook start from a workbook template, which contains a list of wizard pages. 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 on 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 on 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 "allowManualOverride" option is 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.

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

Wizard Page Specifications

Setting	Description	Type	Required
id (attribute)	Wizard page internal identifier.	String	Yes
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 fall back 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

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

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

Listbox and Tree			
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. ▪ "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 Set type.	XML section	No

SuperTree			
Setting	**Description**	**Type**	**Required**
name (attribute)	Widget name.	String	Yes
fallback (attribute)	Wizard page level fallback option. Can be one of the following values: ▪ "default": use value from the default setting. ▪ "all": select all available positions. ▪ "intersection": 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.

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
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>
<Client>
  <version>13.1.0</version>
</Client>
<Server>
  <description>mspdev43-65535</description>
  <profile>false</profile>
  <debug>false</debug>
  <dbDaemonPort>12348</dbDaemonPort>
  <dbServerHostName>mspdev43.us.oracle.com</dbServerHostName>
  <dbServerPortStart>1025</dbServerPortStart>
  <dbServerPortEnd>65535</dbServerPortEnd>
  <defaultDbUser />
  <defaultDbPassword />
  <domainRootDirectory />
</Server>
<Domain>
  <description>TestDomain</description>
  <directoryName>/vol.nas/rpas_se/domains/TestDomain</directoryName>
</Domain>
<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>
```

A

Appendix: Integration Guide

Integration with Oracle Retail Workspace

The Oracle Retail Workspace installer prompts you to enter the URL for your supported Oracle Retail applications. However, if a client installs a new application after Oracle Retail Workspace is installed, the retail-workspace-page-config.xml file needs to be edited to reflect the new application.

The file as supplied comes with all appropriate products configured, but the configurations of non-installed products have been "turned off". Therefore, when "turning on" a product, locate the appropriate entry, set "rendered" to "true", and enter the correct URL and parameters for the new application.

The entry consists of the main URL string plus one parameter named "config". The value of the config parameter is inserted by the installer. Somewhere in the installer property files there is a value for the properties "deploy.retail.product.rms.url" and "deploy.retail.product.rms.config".

The entry consists of the main URL string plus one parameter named "config". The value of the config parameter will be inserted by the installer. Somewhere in the installer property files there will be a value for the properties "deploy.retail.product.rms.url" and "deploy.retail.product.rms.config".

For example, suppose RMS was installed on mycomputer.mycompany.com, port 7777, using a standard install and configured with the application name of "rms121sedevhpsso". If you were to access RMS directly from your browser, you would type in:

```
http://mycomputer.mycompany.com:7777/forms/frmservlet?config=rms121sedevhpsso
```

The entry in the retail-workspace-page-config.xml after installation would resemble the following:

```
<url>http://mycomputer.mycompany.com:7777/forms/frmservlet</url>
    <parameters>
        <parameter name="config">
            <value>rms121sedevhpsso</value>
        </parameter>
    </parameters>
```

Oracle Single Sign-on Overview

What is Single Sign-On?

Single Sign-On (SSO) is a term for the ability to sign onto multiple Web applications via a single user ID/Password. There are many implementations of SSO – Oracle currently provides three different implementations: Oracle Single Sign-On (OSSO), Java SSO (with the 10.1.3.1 release of OC4J) and Oracle Access Manager (provides more comprehensive user access capabilities).

Most, if not all, SSO technologies use a session cookie to hold encrypted data passed to each application. The SSO infrastructure has the responsibility to validate these cookies and, possibly, update this information. The user is directed to log on only if the cookie is not present or has become invalid. These session cookies are restricted to a single browser session and are never written to a file.

Another facet of SSO is how these technologies redirect a user's Web browser to various servlets. The SSO implementation determines when and where these redirects occur and what the final screen shown to the user is.

Most SSO implementations are performed in an application's infrastructure and not in the application logic itself. Applications that leverage infrastructure managed authentication (such as deploying specifying "Basic" or "Form" authentication) typically have little or no code changes when adapted to work in an SSO environment.

What Do I Need for Oracle Single Sign-On?

The nexus of an Oracle Single Sign-On system is the Oracle Identity Management Infrastructure installation. This consists of the following components:

- An Oracle Internet Directory (OID) LDAP server, used to store user, role, security, and other information. OID uses an Oracle database as the back-end storage of this information.
- An Oracle Single Sign-On servlet, used to authenticate the user and create the OSSO session cookie. This servlet is deployed within the infrastructure Oracle Application Server (OAS).
- The Delegated Administration Services (DAS) application, used to administer users and group information. This information may also be loaded or modified via standard LDAP Data Interchange Format (LDIF) scripts.
- Additional administrative scripts for configuring the OSSO system and registering HTTP servers.

Additional OAS servers will be needed to deploy the business applications leveraging the OSSO technology.

Can Oracle Single Sign-On Work with Other SSO Implementations?

Yes, OSSO has the ability to interoperate with many other SSO implementations, but some restrictions exist.

Oracle Single Sign-on Terms and Definitions

Authentication

Authentication is the process of establishing a user's identity. There are many types of authentication. The most common authentication process involves a user ID and password.

Dynamically Protected URLs

A "Dynamically Protected URL" is a URL whose implementing application is aware of the OSSO environment. The application may allow a user limited access when the user has not been authenticated. Applications that implement dynamic OSSO protection typically display a "Login" link to provide user authentication and gain greater access to the application's resources.

Identity Management Infrastructure

The Identity Management Infrastructure is the collection of product and services which provide Oracle Single Sign-on functionality. This includes the Oracle Internet Directory, an Oracle HTTP server, and the Oracle Single Sign-On services. The Oracle Application Server deployed with these components is typically referred as the "Infrastructure" instance.

MOD_OSSO

mod_osso is an Apache Web Server module an Oracle HTTP Server uses to function as a partner application within an Oracle Single Sign-On environment. The Oracle HTTP Server is based on the Apache HTTP Server.

Oracle Internet Directory

Oracle Internet Directory (OID) is an LDAP-compliant directory service. It contains user ids, passwords, group membership, privileges, and other attributes for users who are authenticated using Oracle Single Sign-On.

Partner Application

A partner application is an application that delegates authentication to the Oracle Identity Management Infrastructure. One such partner application is the Oracle HTTP Server (OHS) supplied with the Oracle Application Server. OHS uses the MOD_OSSO module to configure this functionality.

All partner applications must be registered with the Oracle Single Sign-On server. An output product of this registration is a configuration file the partner application uses to verify a user has been previously authenticated.

Realm

A Realm is a collection users and groups (roles) managed by a single password policy. This policy controls what may be used for authentication (for example, passwords, X.509 certificates, and biometric devices). A Realm also contains an authorization policy used for controlling access to applications or resources used by one or more applications.

A single OID can contain multiple Realms. This feature can consolidate security for retailers with multiple banners or to consolidate security for multiple development and test environments.

Statically Protected URLs

A URL is considered to be "Statically Protected" when an Oracle HTTP server is configured to limit access to this URL to only SSO authenticated users. Any attempt to access a "Statically Protected URL" results in the display of a login page or an error page to the user.

Servlets, static HTML pages, and JSP pages may be statically protected.

What Single Sign-On is not

Single Sign-On is NOT a user ID/password mapping technology.

However, some applications can store and retrieve user IDs and passwords for non-SSO applications within an OID LDAP server. An example of this is the Oracle Forms Web Application framework, which maps OSSO user IDs to a database logins on a per-application basis.

How Oracle Single Sign-On Works

Oracle Single Sign-On involves a couple of different components. These are:

- The Oracle Single Sign-On (OSSO) servlet, which is responsible for the back-end authentication of the user.
- The Oracle Internet Directory LDAP server, which stores user IDs, passwords, and group (role) membership.
- The Oracle HTTP Server associated with the Web application, which verifies and controls browser redirection to the OSSO servlet.
- If the Web application implements dynamic protection, then the Web application itself is involved with the OSSO system.

Statically Protected URLs

When an unauthenticated user accesses a statically protected URL, the following occurs:

1. The Oracle HTTP server recognizes the user has not been authenticated and redirects the browser to the Oracle Single Sign-On servlet.
2. The OSSO servlet determines the user must authenticate, and displays the OSSO login page.
3. The user must sign in via a valid user ID and password. If the OSSO servlet has been configured to support multiple Realms, a valid realm must also be entered. The user ID, password, and realm information is validated against the Oracle Internet Directory LDAP server.
4. The OSSO servlet creates and sends the user's browser an OSSO session cookie. This cookie is never persisted to disk and is specific only to the current browser session. This cookie contains the user's authenticated identity. It does NOT contain the user's password.
5. The OSSO servlet redirects the user back to the Oracle HTTP Server, along with OSSO specific information.
6. The Oracle HTTP Server decodes the OSSO information, stores it with the user's session, and allows the user access to the original URL.

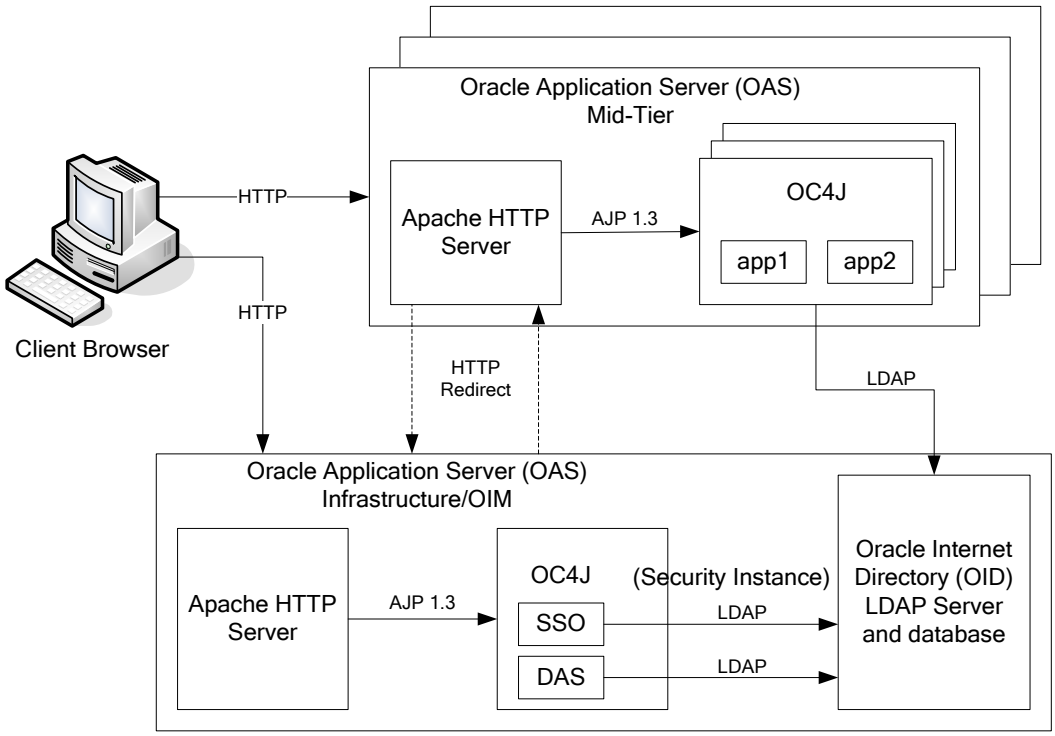
Dynamically Protected URLs

When an unauthenticated user accesses a dynamically protected URL, the following occurs:

1. The Oracle HTTP server recognizes the user has not been authenticated, but allows the user to access the URL.
2. The application determines the user must be authenticated and sends the Oracle HTTP server a specific status to begin the authentication process.
3. The Oracle HTTP Server redirects the user's browser session to the OSSO Servlet.
4. The OSSO servlet determines the user must authenticate, and displays the OSSO login page.
5. The user must sign in via a valid user ID and password. If the OSSO servlet has been configured to support multiple Realms, a valid realm must also be entered. The user ID, password, and realm information is validated against the Oracle Internet Directory LDAP server.
6. The OSSO servlet creates and sends the user's browser an OSSO session cookie. This cookie is never persisted to disk and is specific only to the current browser session. This cookie contains the user's authenticated identity. It does NOT contain the user's password.

7. The OSSO servlet redirects the user back to the Oracle HTTP Server, along with OSSO specific information.
8. The Oracle HTTP Server decodes the OSSO information, stores it with the user's session, and allows the user access to the original URL.

Single Sign-on Topology



Installation Overview

Installing Oracle Single Sign-On consists of installing the following components:

1. Installing the Oracle Internet Directory (OID) LDAP server and the Infrastructure Oracle Application Server (OAS). These are typically performed using a single session of the Oracle Universal Installer and are performed at the same time. OID requires an Oracle relational database and if one is not available, the installer will also install this as well.

   The Infrastructure OAS includes the Delegated Administration Services (DAS) application as well as the OSSO servlet. The DAS application can be used for user and realm management within OID.
2. Installing additional midtier instances (such as OAS 10.1.2) for the Oracle Retail applications, such as RMS, that are based on Oracle Forms technologies. These instances must be registered with the Infrastructure OAS installed in step 1.
3. Installing additional application servers to deploy other Oracle Retail applications and performing application specific initialization and deployment activities.

Infrastructure Installation and Configuration

The Infrastructure installation for OSSO is dependent on the environment and requirements for its use. Deploying an Infrastructure OAS to be used in a test environment does not have the same availability requirements as for a production environment. Similarly, the Oracle Internet Directory (OID) LDAP server can be deployed in a variety of different configurations. See the *Oracle Application Server Installation Guide* and the *Oracle Internet Directory Installation Guide* for more details.

OID User Data

Oracle Internet Directory is an LDAP v3 compliant directory server. It provides standards-based user definitions out of the box.

The current version of Oracle Single Sign-On only supports OID as its user storage facility. Customers with existing corporate LDAP implementations may need to synchronize user information between their existing LDAP directory servers and OID. OID supports standard LDIF file formats and provides a JNDI compliant set of Java classes as well. Moreover, OID provides additional synchronization and replication facilities to integrate with other corporate LDAP implementations.

Each user ID stored in OID has a specific record containing user specific information. For role-based access, groups of users can be defined and managed within OID. Applications can thus grant access based on group (role) membership saving administration time and providing a more secure implementation.

OID with Multiple Realms

OID and OSSO can be configured to support multiple user Realms. Each realm is independent from each other and contains its own set of user IDs. As such, creating a new realm is an alternative to installing multiple OID and Infrastructure instances. Hence, a single Infrastructure OAS can be used to support many development and test environments by defining one realm for each environment.

Realms may also be used to support multiple groups of external users, such as those from partner companies. For more information on Realms, see the *Oracle Internet Directory Administrators Guide*.

User Management

User Management consists of displaying, creating, updating or removing user information. There are two basic methods of performing user management: LDIF scripts and the Delegate Administration Services (DAS) application.

OID DAS

The DAS application is a Web-based application designed for both administrators and users. A user may update their password, change their telephone number of record, or modify other user information. Users may search for other users based on partial strings of the user's name or ID. An administrator may create new users, unlock passwords, or delete users.

The DAS application is fully customizable. Administrators may define what user attributes are required, optional or even prompted for when a new user is created.

Furthermore, the DAS application is secure. Administrators may also what user attributes are displayed to other users. Administration is based on permission grants, so different users may have different capabilities for user management based on their roles within their organization.

LDIF Scripts

Script based user management can be used to synchronize data between multiple LDAP servers. The standard format for these scripts is the LDAP Data Interchange Format (LDIF). OID supports LDIF script for importing and exporting user information. LDIF scripts may also be used for bulk user load operations.

User Data Synchronization

The user store for Oracle Single Sign-On resides within the Oracle Internet Directory (OID) LDAP server. Oracle Retail applications may require additional information attached to a user name for application-specific purposes and may be stored in an application-specific database. Currently, there are no Oracle Retail tools for synchronizing changes in OID stored information with application-specific user stores. Implementers should plan appropriate time and resources for this process. Oracle Retail strongly suggests that you configure any Oracle Retail application using an LDAP for its user store to point to the same OID server used with Oracle Single Sign-On.

B

Appendix: Curve Administration Guide

curvevalidate

`Curvevalidate` automatically executes during the domain install, and it can also be run at any time against a Master or one subdomain. If run against the Master Domain, it checks the master and all subdomains. If run against a subdomain, it checks the Master and only the subdomain (not all other subdomains). This function verifies that:

- Profile and Source intersections and source data are properly defined
- Profile intersections respect the partition dimension

Usage

```
curvevalidate -d domainpath [-s]
```

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

Argument	Description
`-d` *domainpath*	Path to the domain.
`-s`	Set defaults.
`-debug`	This argument causes temporary measures to be retained for debugging purposes.
`-h`	Standard information argument to display information about `curvebatch` in the terminal screen.
`-version`	Standard argument that displays version information.
`-loglevel`	Used to set the logger verbosity level. Available verbosity levels are as follows: ▪ `all` ▪ `profile` ▪ `information` ▪ `warning` ▪ `error` ▪ `none`
`-noheader`	Standard argument to disable the use of timestamp header.

1. Each Profile must have at least one Source Level.
2. For each Profile:
   a. For global domains, ALL intersections {Data Intersection, Profile Intersection, Stored Intersection, Aggregation Intersection, and Approval Intersection} must be below the partition (NOT HBI).
   b. Data Intersection (if a data source is specified) must conform to X in {Profile Intersection, Stored Intersection, Aggregation Intersection, and Approval Intersection}.

c. Profile Intersection must conform to the Stored Intersection.
d. Aggregation Intersection must conform to the Approval Intersection.
e. Aggregation Intersection must not be below the Approval Intersection.
f. Aggregation Intersection must be above the Data Intersection (if data source specified).
g. If the Aggregation Intersection conforms to Profile Intersection:
    i. The Profile Type must NOT be diff(8).
    ii. The Aggregation Intersection must be above the Profile Intersection.
    iii. The Aggregation Intersection must be above the Stored Intersection.
h. If Aggregation Intersection does not conform to Profile Intersection:
    i. The Profile Type must be Diff (8).
    ii. There must be at least one common hierarchy between the Aggregation Intersection and X in {Profile Intersection, Stored Intersection}.
    iii. For each common non-PROD hierarchy H of Aggregation Intersection and X in {Profile Intersection, Stored Intersection}:
    Aggregation Intersection's H dimension must not be below X's H dimension.

3. For each Source Level:
    a. For global domains, ALL intersections {Profile Intersection, Stored Intersection, and Aggregation Intersection} must be below the partition (NOT HBI).
    b. Parent Profile's Data Intersection (if data source specified) must conform to X in {Profile Intersection, Stored Intersection, and Aggregation Intersection}.
    c. Profile Intersection must conform to Stored Intersection.
    d. Aggregation Intersection must be above parent Profile's Data Intersection (if data source specified).
    e. If Aggregation Intersection conforms to Profile Intersection:
        i. The Profile Type must NOT be diff(8).
        ii. The Aggregation Intersection must be above the Profile Intersection.
        iii. The Aggregation Intersection must be above the Stored Intersection.
    f. If Aggregation Intersection does not conform to Profile Intersection:
        i. The Parent Profile Type must be Diff (8).
        ii. There must be at least one common hierarchy between the Aggregation Intersection and X in {Profile Intersection and Stored Intersection}.
        iii. For each common non-PROD hierarchy H of Aggregation Intersection and X in {Profile Intersection and Stored Intersection}:
        Aggregation Intersection's H dimension must not be below X's H dimension.

curvebatch

Usage

```
curvebatch -d domainpath [-level # ] [-debug] | -h | -version
```

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

Argument	Description
`-d` *domainpath*	Path to the domain.
`-level #`	The # signifies the profile ID. When using level, a valid profile ID must be provided.
`-debug`	This argument causes temporary measures to be retained for debugging purposes.
`-h`	Standard information argument to display information about `curvebatch` in the terminal screen.
`-version`	Standard argument that displays version information.
`-loglevel`	Used to set the logger verbosity level. Available verbosity levels are as follows: ▪ `all` ▪ `profile` ▪ `information` ▪ `warning` ▪ `error` ▪ `none`