Oracle® Retail Predictive Application Server Cloud Edition

Configuration Tools User Guide

Release 23.1.201.0
F80322–01
April 2023

Oracle Retail Predictive Application Server Cloud Edition Configuration Tools User Guide, Release 23.1.201.0

F80322–01

Primary Authors: Judith Meskill, Eric Bloemeke, Judy Zhou

Contents

H Appendix – RPASCE Rule Writing Tips

Send Us Your Comments

Oracle Retail Predictive Application Server CE Configuration Tools User Guide, Release 23.1.201.0.

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

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

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

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

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

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

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

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

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

Preface

This guide describes the RPASCE Configuration Tools. It provides step-by-step instructions to complete most tasks that can be performed through the user interface.

Audience

This Configuration Tools User Guide is for users and administrators of Oracle RPASCE. This includes merchandisers, buyers, business analysts, and administrative personnel.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

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

- *Oracle Retail Predictive Application Server Cloud Edition Administration Guide*
- *Oracle Retail Predictive Application Server Cloud Edition Release Notes*
- *Oracle Retail Predictive Application Server Cloud Edition Security Guide*
- *Oracle Retail Predictive Application Server Cloud Edition User Guide*

Customer Support

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

`https://support.oracle.com`

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)

- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Improved Process for Oracle Retail Documentation Corrections

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

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

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

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

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

Oracle Retail Documentation on the Oracle Help Center (docs.oracle.com)

Oracle Retail product documentation is available on the following web site:

`https://docs.oracle.com/en/industries/retail/index.html`

(Data Model documents can be obtained through My Oracle Support.)

Conventions

The following text conventions are used in this document:

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

1

Introduction

The Oracle Retail Predictive Application Server Cloud Edition (RPASCE) Configuration Tools provide a flexible means to configure and build RPASCE-based applications with retailer-specific business parameters. The configuration tools provide a streamlined, user-friendly interface to leverage RPASCE functionality. Once a configuration is created, an installer script is used to build an RPASCE Planning Data Storage (PDS).

The Configuration Tools consist of an integrated set of task-specific configuration aids that are used to configure a solution template or to modify an existing solution template.

RPASCE functionality is exposed to the Configuration Tools through Application Programming Interfaces (APIs).

A configuration is typically created and maintained by an application administrator or solution expert. Details of the configuration are stored locally on the administrator's PC or on the network. Once the configuration is complete, the administrator uses the configuration to create a new PDS or update an existing PDS.

Users of the configured solution will access the RPASCE PDS by using the RPASCE Client that is installed on their machines. The PDS accessed represents the business process and environment that was configured in the solution by the configuration administrator together with the appropriate data.

Once the PDS is created, administrative RPASCE processes (such as hierarchy maintenance and user administration) are accomplished by an RPASCE administrator using the utilities on the server.

Note:

For more information on RPASCE administration, refer to the *Oracle Retail Predictive Application Server Administration Guide for the RPASCE Client*.

Customizing Configuration Tools Font Type and Size

Depending on the underlying Windows sytem and the monitor resolution, the font size of Configuration Tools might appear too small to some users. Users may also have their own preferable font type. A settable preference for font type and size is desired. Thus, RPASCE allows users to use environment variables in order to control the font type and font size used in the Configuration Tools.

The environment variables must be exported before the configuration tools start to take effect. Before starting the Configuration Tools, users can set the environment variable OVERRIDE_FONT_NAME for the font type and DELTA_FONT_SIZE for the delta value of the font size to offset their system default size.

In the following example, the Tools uses the Serif font type and increases their system default font size by 8 for their Tools display. Users may adjust the values to achieve the desired effect.

export OVERRIDE_FONT_NAME="Serif"export DELTA_FONT_SIZE=8

If these two environment variables are not explicitly configured, the Configuration Tools is defaulted to use the Serif font type, that is, the Times Roman font, and a delta font size 4.

Configuration Tools Business Process

The Configuration Tools business process follows these steps:

1. Set up system properties
2. Create a project
3. Create solutions
4. Configure hierarchies and dimensions
5. Configure measures and measure components
6. Configure rules sets, rule groups, and rules
7. Configure workbooks, workbook tabs, and worksheets
8. Configure wizards
9. Configure dashboards
10. Configure contextual help [optional]
11. Define interfaces used to import data
12. Build an RPASCE PDS instance

Sample Configurations

Some examples in this guide use the sample configuration, which is delivered with the RPASCE platform and can be installed along with the RPASCE software and Configuration Tools. The sample configuration may not match every illustration because RPASCE software and Configuration Tools software versions might vary between users. The examples are meant to provide a context to the reader. For information about the sample configuration provided with the RPASCE platform, refer to the Starter Kit Guide for the version of the Starter Kit you have installed.

Using the Configuration Tools Online Help

This Help site provides step-by-step procedures and other information about using RPASCE Configuration Tools. We have implemented some tools to assist your navigation of this Help site. This page explains these tools.

About the Online Help

The online Help system uses JavaScript for some of its functionality. Make sure you have enabled JavaScript for your Web browser. Refer to the online Help in your Web browser for instructions on enabling JavaScript.

Formatting Conventions

This section provides information about the documentation conventions used in the online Help.

Note:

Notes are displayed using this convention. Notes contain additional information about the process or procedure that you are performing.

Navigate: The navigation sections of a procedure provide information about how to access the window that is the starting point of a procedure.

Navigate the Online Help

This Help site provides several ways for you to navigate to your topic.

Use the Table of Contents

The table of contents is the most common way that you will navigate to your topic.

1. Select the **Table of Contents** tab to display the table of contents on the left side of your screen.
2. Select the **+** sign in front of a book to expand it and view the topics.
3. Select a topic from the table of contents to view it.

Using the Search Feature

Use the search feature to explore the contents of your topics and find matches to queries that you define. There are some basic rules for making queries in full-text searches.

- You can type your search in uppercase or lowercase characters. Searches are not case sensitive.
- You can search for any combination of letters (a-z) and numbers (0-9).
- Punctuation marks such as the period, colon, semicolon, comma, and hyphen are ignored during a search.
- Group the elements of your search using double quotes or parentheses to set apart each element.
- You cannot search for quotation marks.

Use the following procedure to search the online Help:

1. Select the **Search** tab to display the search feature on the left side of your screen.
2. In the Search field, enter the word or words that you want to find.
3. Press **Enter**. Topics that match your search criteria display in the left pane.
4. Select a topic to view it.

Using the Business Process

The business process typically provides links to procedures that you need to perform to complete a task. You can select any link in the business process to view that topic.

Using the Index

Some Help sites may have an index. The index provides another way for you to navigate to information. There are two ways to use the index to search.

Browse the Index Entries

Follow these steps to browse the index entries:

1. Select the Index tab. Words and phrases that are listed in the index display in alphabetical order.
2. Scroll up or down to find a word or phrase.
3. Select the word or phrase to view additional information.

Search the Index

Follow these steps to search the index:

1. Select the Index tab. Words and phrases that are listed in the index display in alphabetical order.
2. In the keyword field, type the word or phrase. Words and phrases that match your entry are displayed.
3. Select the word or phrase to view additional information.

Using Links

There may be two different types of links in this online Help.

- Some topics contain hyperlinks that open a new page.
- Many topics have additional information that is available from a link.

Drop-down text typically provides additional steps or sub-steps for a process or procedure and displays under the linked word or phrase.

- Select the link once to view the text.
- Select the link again to hide the text.

Using Hyperlinks

Hyperlinks bring you to another page in the online Help or to a Web page on the Internet. There are two things to remember when using hyperlinks:

- Hyperlinks always display a brief description of where the hyperlink takes you.
- If your browser controls are turned off, follow these steps to return to the previous page:

1. Display the shortcut menu by perform one of the following actions:

   - Right-click with your mouse.
   - Press the Application key.
2. From the shortcut menu, click **Back** to return you to the previous page.

Navigating the Configuration Tools

This section includes information about:

- Starting the Program
- About the Configuration Tools Windows

Starting the Program

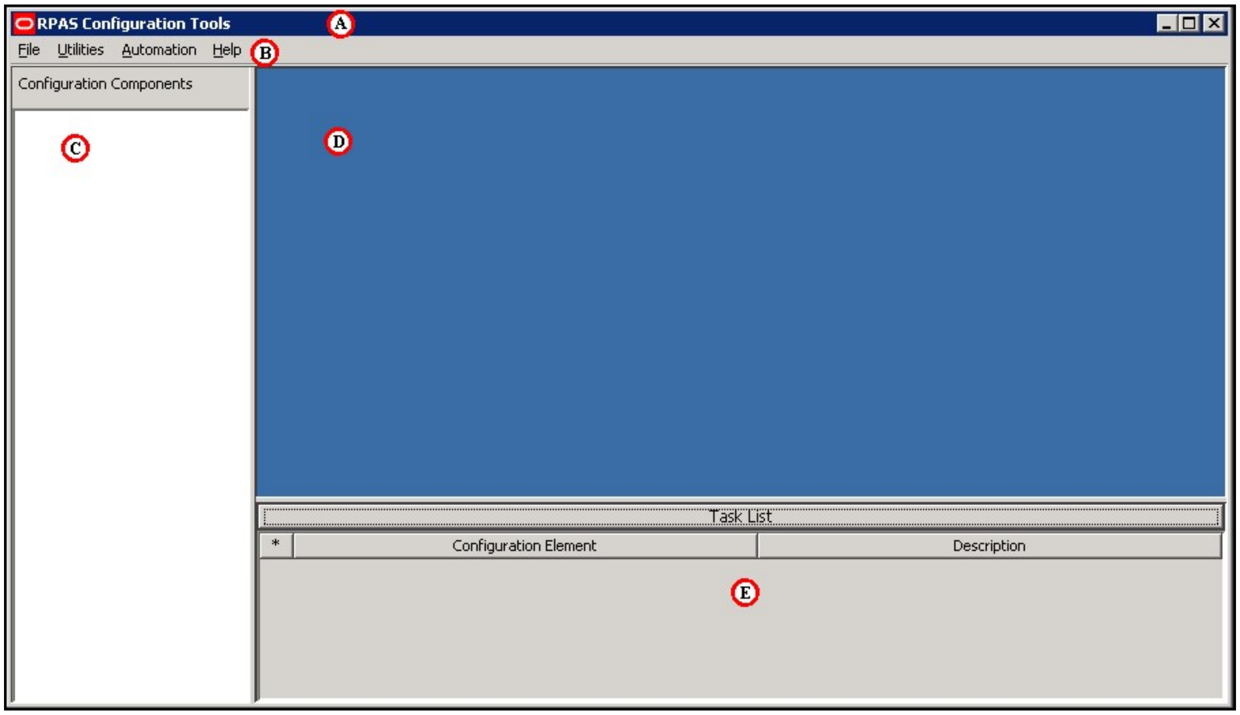
The Configuration Tools are distributed for the Cloud as the Starter Kit. That guide, which must be reviewed prior to using this guide, describes the installation and set-up processes. (For information for a specific Cloud Service, see the Starter Kit Guide for that Cloud Service.) The Configuration Tools can be run from the Cygwin shell anywhere by calling **`configTools`** once the RPASCE Starter Kit has been installed and set up.

About the Configuration Tools Windows

All tasks are performed through the RPASCE Configuration Tools window, which provides the following features:

- Drop-down menus
- Toolbars
- Active buttons
- Right-click functionality

Figure 1-1 Configuration Tools Window



The primary elements in the application window are as follows:

Table 1-1 Configuration Tools Window Elements

Key	Element	Purpose
A)	Title Bar	• Displays the product name • The three buttons at the far right on the title bar allow for the application window to be minimized, restored, maximized, and closed
B	Menu Bar	• Contains the menus that are used in the configuration tools • Each menu contains a set of commands that allow the configuration administrator to operate the configuration tool
C)	Configuration Components	• Displays information about configurations, projects, and solutions that are currently in use • Configuration information is not displayed until a configuration is opened.
D)	Workspace	As different elements of a configuration are selected in the Configuration Components pane, the related windows are displayed in the workspace.
E)	Task List	Displays errors and warnings within the configuration

A Note About RPASCE Configurability and Extensibility

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

2 Configuration Components Pane

This chapter includes information about the Configuration Components pane.

Know the Configuration Components

The Configuration Components pane is the starting point for creating a new configuration or for opening an existing configuration. It provides a high-level view of all the components that are necessary to configure an RPASCE application, and it is used to navigate to the various tools that are used to configure those components.

The Configuration Components pane is the core of the RPASCE Configuration Tools, and it provides an overall view of the configuration components. Each configuration contains one project and one or more solutions.

As a new configuration is created the configuration administrator assigns a name to the configuration, the project, and the solution. The configuration administrator can drill down through the configuration to work in specific areas. The icons in the Configuration Components assist the configuration administrator to intuitively navigate features within the Configuration Tools. If an area of the configuration has been modified, its icon will contain a modification flag icon. The configuration must be saved if the modifications are to be retained.

Figure 2-1 RPAS Configuration Tools Configuration Components Pane

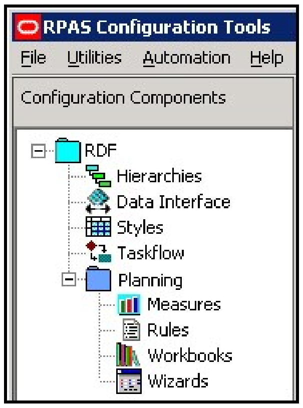


Table 2-1 Configuration Components Pane Icons

Icon Name	Description	Icon
Project	Starts a new project	📁

Table 2-1 (Cont.) Configuration Components Pane Icons

Icon Name	Description	Icon
Hierarchies	Opens the Hierarchy Definition window	☷
Data Interface	Opens the Data Interface Manager window	⇔
Styles	Opens the Style Definition Window	▦
Taskflow	Opens the Taskflow Manager	⇄
Solution	Starts a new solution	📁
Measures	Opens the Measure Manager window	📊
Rules	Opens the Rule Definition window	📄
Workbooks	Opens the Workbook Designer window	📚
Wizards	Opens the Wizard Designer window	▤

Configuration Components Pane Overview

This section describes the items in the Configuration Components pane.

Projects

Each project represents a single, logical RPASCE application. The hierarchies, dimensions, and styles are defined within a project and are available for use within all solutions in the project. The RPASCE Configuration Tools allows for multiple projects to be viewed and modified (the limit is three projects).

Solution

Each solution represents a grouping of measures, rules, and workbooks to support a business process as defined by the retailer. A project may have multiple solutions and a solution may use a subset of the hierarchies and dimensions defined within the project.

Hierarchy

The user may access the Hierarchy Definitions window by selecting the Hierarchy icon in the Configuration Components pane. For each project, one single or multiple hierarchies may be created and dimensions are defined within each hierarchy. Hierarchies are the structures used by an organization to describe the relationships that exist between the dimensions. The following hierarchies are automatically created when a new project is defined:

- Calendar
- Product
- Location

Users may create and define the individual dimensions for these hierarchies and for any additional hierarchies that may be desired.

> **Note:**
>
> The RPASCE (system) names for the default hierarchies (CLND, PROD, LOC, and ADMU) cannot be changed, but the default labels (Calendar, Product, and Location) can be changed. The ADMU label cannot be changed.

Data Interface

The user may access the Data Interface Manager by selecting the Data Interface icon in the Configuration Components pane. The data interface tool is used to define the format of data interface files and provide some data interface parameters, such as directions to RPASCE on how to deal with data that is sourced lower than its base intersection. The tool sets measure attributes that are referenced when loading measure data into the application. The information entered into the data interface will be referenced when the loadmeasure utility is used to load data for a measure.

Styles

The user may access the Style Definition window by selecting the Styles icon in the Configuration Components pane. The style tool is used to define styles that specify how the data for a measure is to be displayed within the RPASCE Client. Styles consist of a number of attributes, such as text font, size, and color as well as specifications of precision, alignment of text within the cell. These styles may then be assigned to measures within the Measure and Workbook Tools.

Color Palette

When selecting colors during configuration, you should select a color from the accessible colors listed in the Format window. These colors have been chosen by the UI designers for proper contrast and for compatibility with the Oracle Retail look-and-feel standards The Format window displays a set for the text color and a set for the background color, which differ slightly. If you use a color that is not listed as part of this palette, the UI will display the alternate color in the cells or header and will preview the color in the Format window; however, that color will not be listed in the color picker in the window and so cannot be selected.

The text colors are:

Code	Color
#000000	Black
#999999	Grey
#81bb5f	Green
#e95b54	Red
#309fdb	Blue
#ed813e	Orange
#2ebfbc	Teal

The background colors are:

Code	Color
#ffffff	White
#000000	Black
#f5f5f5	Light Grey
#dle7cc	Green
#fcedc8	Yellow
#ebf5fc	Light Blue
#f9ddc0	Orange

Taskflow

The user can access the Taskflow Manager window by selecting the Taskflow icon in the Configuration Components pane. Users can use the Taskflow tool to configure an activity taskflow for use within the RPASCE Client. This Taskflow directs users through various steps that have been setup to help them accomplish their business process. Taskflows contain activities that the user needs to complete. Each activity contains one or more tasks, which are mapped to certain workbooks. Within each task are one or more steps that are mapped to worksheets within the task's workbook. The activities can also be organized in an activity group. An activity group is a single, integrated taskflow that represents a business process and can include activities from multiple solutions.

Users can use the preconfigured taskflow that is delivered with the RPASCE solution, modify that preconfigured taskflow, or configure a customized one.

Solutions

A solution corresponds to an application configuration (for example, Financial Planning or Item Planning). For each solution, the following are configured:

- Measures
- Rule sets/rule groups/rules
- Workbooks/worksheets
- Wizards (optional)

Measures

Measures (multidimensional variables) are any item of data that can be represented on a grid in a worksheet. Measures are the data points used in the retailer's business process.

Rule Sets, Rule Groups, and Rules

Rules are collections of expressions (the basis of all calculations) that describe the relationships between measures. They are evaluated by the RPASCE calculation engine during a calculation. Rules can consist of multiple expressions as the following example represents:

- Expression 1: ReceiptUnits = ReceiptValue / ReceiptPrice
- Expression 2: ReceiptValue = ReceiptUnits * ReceiptPrice
- Expression 3: ReceiptPrice = ReceiptValue / ReceiptUnits

The collection of expressions represents a rule. These three expressions state the relationship between ReceiptUnits, ReceiptValue, and ReceiptPrice. Each expression solves for a different measure.

A rule group is a collection of rules that are treated as a unit by the calculation engine. The rules in the rule group must be considered together to satisfy the calculation requirements for a specific business process. The sequence of rules within a rule group affects the calculation sequence. For more information on rule sequencing and how it affects the calculation process, see Appendix: Calculation Engine User Guide

A rule set is a collection of rule groups that is used for organizational purposes by the Configuration Tools.

Workbooks and Worksheets

A workbook is the multidimensional framework that is used to perform specific business functions, such as creating a merchandise plan and reviewing available data. Workbooks are easily viewed and manipulated.

A workbook can contain any number of multidimensional spreadsheets (called worksheets) to present data. Measures and rules are used to define and calculate the measure data. All components work together to facilitate the viewing and analysis of business functions. The Configuration Tools allow the configuration administrator to configure workbook templates incorporating these various components.

Wizards

A wizard is a feature that guides the user through the process of building a new workbook. A wizard displays successive windows that require the user to answer a sequence of questions or enter information regarding the content of the workbook. Responses to these questions are used to format and populate the workbook. The layout of these wizard windows could be defined using the wizard tool. However, each workbook may use a standard wizard configuration, eliminating the need for the configuration administrator to access the Wizard Designer.

The main purpose of the wizard is to allow the end user to make choices regarding the scope of the workbook. For example, the first wizard will ask the user to select the SKUs to include in the workbook. The second will ask the user to select the stores to include in the workbook, and the third wizard will ask the user to select the dates to include in the workbook. At the end of the series of wizards, a workbook will be created that has data for the SKUs, stores, and dates that the user selected.

Task List

The Task List provides a centralized view of errors and warnings that are issued as a result of information input by the user. Use the information in the Task List as a guide for correcting errors or omissions in the Project.

The Task List title bar serves as a status indicator. If the title ***Task List*** is displayed in red, the Task List contains items that need the user's attention. If there are no errors or warnings, the title will be displayed in black.

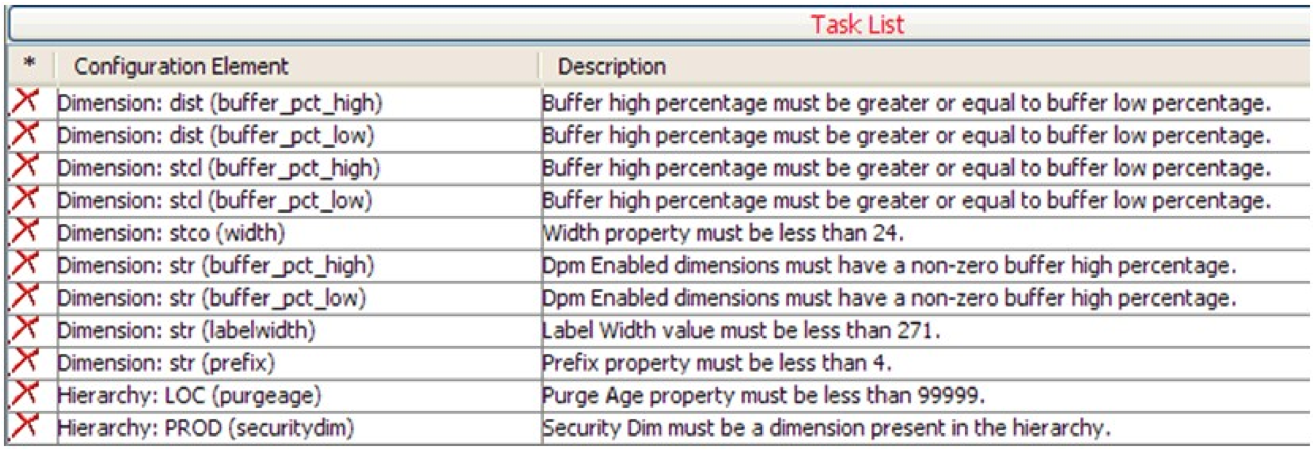
The title bar can also be used to show or hide the Task List. If the Task List is visible, click anywhere on the title bar to hide the list and move the title bar to the bottom of the window. If it is hidden, click anywhere on the title bar to display the Task List. The amount of space used by the task list sub-pane can also be changed by dragging the separator located above the task list title bar.

The Task List has three columns:

- The first column indicates the nature of the Task List item.
  - Indicates an error.
  - Indicates a warning.
- The second column identifies the configuration element involved.
- The third column provides a detailed description of the issue.

Figure 2-2 shows a sample task list.

Figure 2-2 Sample Task List



Task List

*	Configuration Element	Description
✗	Dimension: dist (buffer_pct_high)	Buffer high percentage must be greater or equal to buffer low percentage.
✗	Dimension: dist (buffer_pct_low)	Buffer high percentage must be greater or equal to buffer low percentage.
✗	Dimension: stcl (buffer_pct_high)	Buffer high percentage must be greater or equal to buffer low percentage.
✗	Dimension: stcl (buffer_pct_low)	Buffer high percentage must be greater or equal to buffer low percentage.
✗	Dimension: stco (width)	Width property must be less than 24.
✗	Dimension: str (buffer_pct_high)	Dpm Enabled dimensions must have a non-zero buffer high percentage.
✗	Dimension: str (buffer_pct_low)	Dpm Enabled dimensions must have a non-zero buffer high percentage.
✗	Dimension: str (labelwidth)	Label Width value must be less than 271.
✗	Dimension: str (prefix)	Prefix property must be less than 4.
✗	Hierarchy: LOC (purgeage)	Purge Age property must be less than 99999.
✗	Hierarchy: PROD (securitydim)	Security Dim must be a dimension present in the hierarchy.

Errors typically indicate definite validation problems, which are shown in red in the tool where the configuration setting is made. When the user fixes the erroneous condition, the Task List automatically removes the error listing for that condition. Warnings indicate the possibility of a problem occurring, and the user is advised to inspect the suspected element to ensure that everything is in order. Since warnings are more general than errors the Task List will not remove them automatically. The user is provided with options to remove errors and warnings through a right-click menu.

How RPASCE Uses Solution Configurations

RPASCE uses the following components for solution configurations:

The RPASCE Calculation Engine

The RPASCE calculation engine is a very powerful and flexible engine that is built to support On-Line Analytical Processing (OLAP) type calculations against a multi-dimensional model.

In the OLAP model individual pieces of data (called cells) correspond to a single position in one or more hierarchies or dimensions. Cells typically reference:

- A measure
- A calendar or time hierarchy
- Other hierarchies, such as product and location

The measure is fundamentally different to the other hierarchies because measures represent the events or measurements that are being recorded. The positions in the other hierarchies provide a context for the measurement: where, when, what, and so on. Measures relate to one another through rules and expressions. Positions in all the other hierarchies relate to each other through hierarchical relationships.

Aggregation and Spreading

The RPASCE calculation engine is designed to be robust and extensible, but in complete control of the calculation process. It enforces integrity of the data by ensuring that all known relationships between cells are always enforced. Much of the logic of the processing of rules and rule groups depends on this basic principle. RPASCE supports two different forms of relationships between cells:

- Hierarchical relationships that require aggregation and spreading
- Measure relationships that require rules and expressions

Aggregation and spreading are basic capabilities of the engine that do not require coding by the implementer, other than the selection of aggregation and spreading types to use for a measure. Hierarchical relationships, such as weeks rolling up to months or stores rolling up to regions, require the aggregation of data values from lower levels in a hierarchy to higher levels by using a variety of methods as appropriate to the measure.

To enable such data to be manipulated at higher levels, RPASCE supports spreading the changes, which also uses a variety of methods.

The inherent relationships between measures can be modeled through a rich rule and expression syntax. Modeling these relationships takes most of the effort in configuring an application model.

RPASCE Functions

RPASCE Functions are mechanisms for performing operations within an expression that are controlled and executed by the calculation engine.

Most functions have only one output.

The calculation engine controls and runs the evaluation of a function.

Functions may be used in long expressions with other functions and keywords.

The data that can be referenced is limited to the scope of the workbook.

Note:

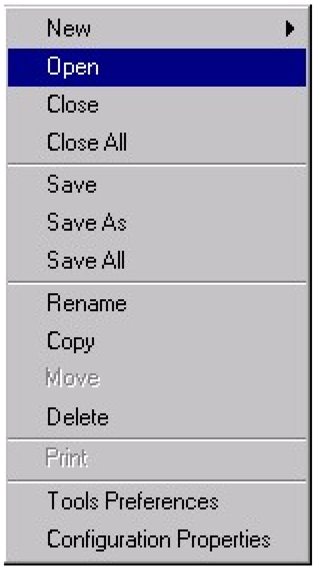
See Appendix – Calculation Engine User Guidefor a comprehensive definition of the RPASCE calculation engine and how it is used when configuring a solution.

See Appendix – Rules Function Reference Guide for details about standard RPASCE functions.

Right-Click Menus in the Configuration Components Pane

The right-click menu may be accessed by right-clicking in any location within the Configuration Components pane. Each available selection from the right-click menu is described in the following sections.

Figure 2-3 Example: Right-Click Menu

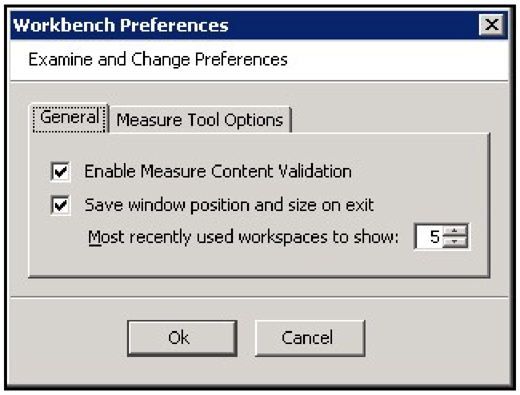


Setting Tools Preferences

Settings made here apply to all configuration projects created or viewed with the tool.

Navigate: From the File menu, select Tools Preferences. The Workbench Preferences window opens.

Figure 2-4 Workbench Preference Window


Workbench Preferences

Examine and Change Preferences

General | Measure Tool Options

[✓] Enable Measure Content Validation

[✓] Save window position and size on exit

Most recently used workspaces to show: 5

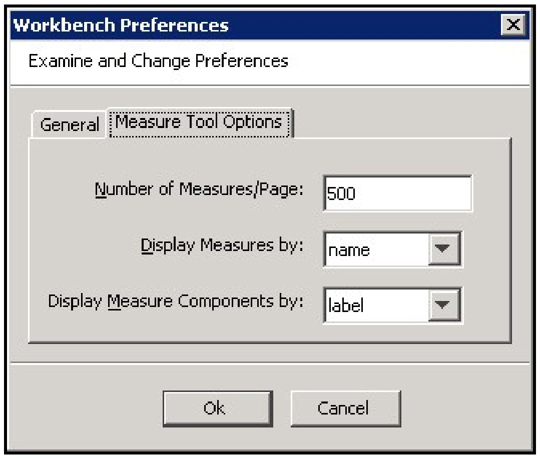
Perform the following steps to modify the Workbench Preferences.

1. Select the **General** tab.
   - **Enable Measure Content Validation** – Activating this check box enables the immediate validation of measure properties when configuration measure information is created or modified. This process can impact the performance of the RPASCE Configuration Tools. If this box is not checked, the manual Measure Content Validation icon is enabled on the Rule Definition toolbar. See the Rule Definition Tool for details.

- **Save window position and size on exit** – Activating this check box enables the RPASCE Configuration Tools to be launched in either full or minimized view based on the status in which the configuration administrator last exited the application. If the application is exited in a minimized view, the size of the window is also maintained when the RPASCE Configuration Tools is re-launched.
- **Most Recently used workspaces to show** – Use the up and down arrows to specify the number of configurations to be displayed in the Most Recently Used list displayed in the File menu window. This list allows the configuration administrator easy selection of a recently viewed configuration.

2. Select the **Measure Manager Options** tab.

Figure 2-5 Workbook Preferences - Measure Tool Options


Workbench Preferences
Examine and Change Preferences
General | Measure Tool Options
Number of Measures/Page: 500
Display Measures by: name
Display Measure Components by: label
Ok | Cancel

- **Number of Measures/Page** – Select the number of measures to display per page in the Measure Manager Tool. The default is 500.
- **Display Measures by** – Displays measures either by their name or label in various locations of the Tools. The default setting is *name*.
- **Display Measure Components by** – Display measure components (in the Measure Manager tool) either by name or by label. The default setting is *label*.

3. Click **OK** to save any changes and close the window.

3
Projects

Note:

This chapter contains descriptions of features that are not supported in the current version of RPASCE. However, in order to assist in the process of migrating from a pre-Version 21 solution to the Version 21 CE platform, the configuration components associated with these deprecated features are still available within the RPASCE Configuration Tools and are still described within this document. For additional information, see Appendix – Note on Deprecated Functionality in this document.

This chapter describes information for:

Working with Projects

A project is used to configure the structure of an application. Each project represents a single, logical RPASCE application; however, one or more applications may reside on one PDS instance. The hierarchies, dimensions, styles, and taskflow are defined within a project and are available for use within all solutions in a given project.

Note:

A solution can use a subset of the hierarchies and dimensions defined within the project. Within a project, you can additionally define certain properties (which are part of the Data Interface Tool) that describe how measures will be loaded into the application.

The RPASCE platform data is stored in a PDS instance. Only during PDS initial creation and patching, a simple domain is created from the application configuration. This simple domain is also referred as “sandbox” and is used for further building/patching the PDS instance. After the PDS instance is built/patched, RPASCE data is manipulated directly through the PDS instance instead of the sandbox.

Hierarchies are application-specific, which means that they are defined at the project (application) level and can be used by all solutions that are defined within that project (application). There is no requirement that each solution use all of the hierarchies defined in the project.

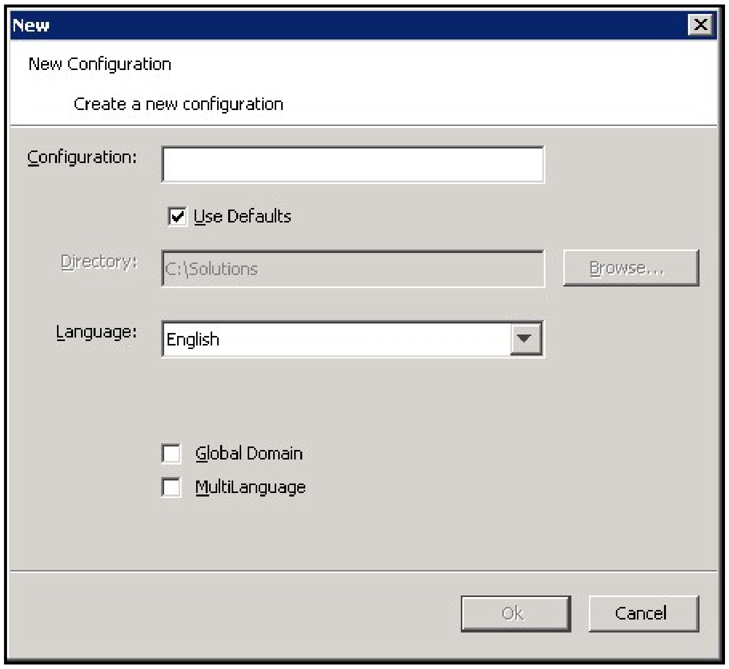
For example, a project may contain five hierarchies and three solutions, but each solution might only use four of those hierarchies in the base intersections of its measures, so even though five hierarchies exist, each solution may not use all of them.

Create a Project

Complete the following steps to create a project.

Navigate: From the File menu, select **New** and then **Project**, or right-click in the Configuration Manager and select **New** and then **Project**. The Figure 3-1 opens.

Figure 3-1 New Window



1. In the **Configuration** field, enter the name of the configuration/project.

   The Use Defaults option under the Configuration field points the user to a default path for storing the configuration. Clear the **Use Defaults** check box to enable the user to navigate to the appropriate directory using **Browse**.
2. Select the default language for the application in which this configuration will be used to create. The default is English.
3. Deprecated. Select the options for Global Domain and MultiLanguage as necessary. The possible settings for these boxes are as follows:
   - **Global Domain** – Deprecated. The Global Domain setting is always hardcoded to false since the starting sandbox is created as a simple domain. The change of this setting has no effect in RPASCE.
   - **MultiLanguage** – Deprecated. The MultiLanguage setting is always hardcoded to true since the starting sandbox and the PDS instance always contain multi-language translations. A Multi-lingual PDS allows for most data elements (measures, labels, and so on) in an RPASCE PDS to be translated into other languages. The change of this setting has no effect in RPASCE.
4. Click **OK** to save any changes and close the window.

Save Changes to a Project

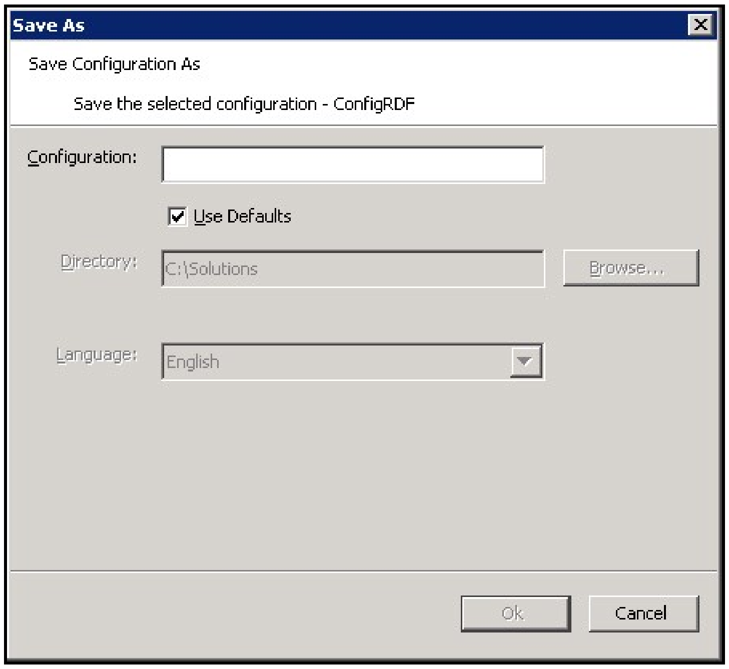
Navigate: From the File menu, select **Save**, or right-click in the Configuration Components pane and select **Save**. This saves the project under the same name that was used to create it and within the same directory.

Using the Save As Option to Save a Project Using a Different Name

Complete the following steps to use the Save As option to save a project using a different name.

Navigate: From the File menu, select **Save As**, or right-click in the Configuration Components pane and select **Save As**. The Save As window opens.

Figure 3-2 Save As Window



Save As

Save Configuration As

Save the selected configuration - ConfigRDF

Configuration:

☑ Use Defaults

Directory: C:\Solutions

Language: English

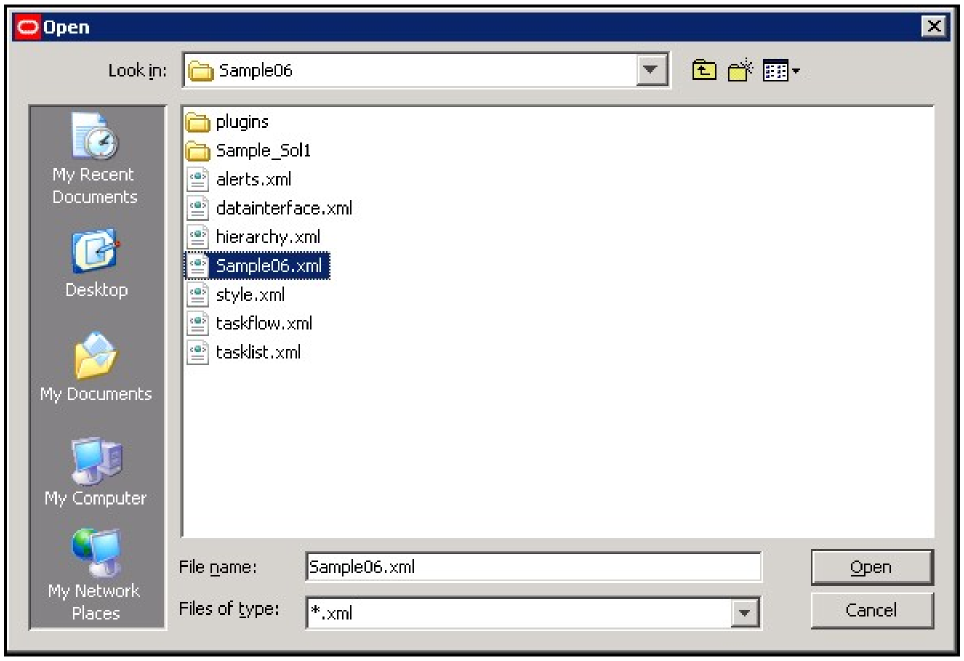
1. In the Configuration field, enter the new project name.
2. If **Use Defaults** is selected, click **OK** to save the project to the path displayed in the **Directory** field. If **Use Defaults** is selected and you want to save the project to a different location, clear **Use Defaults** and either enter the appropriate path in the Directory field or click **Browse** to navigate to the appropriate location where you want the project saved.
3. Click **OK** to save the project.

Open an Existing Project

Complete the following steps to open an existing project.

Navigate: From the File menu, select **Open** or right-click in the Configuration Components pane and select **Open**. The Open window opens.

Figure 3-3 Open Window



1. Choose one of the following methods:
   - Browse to the directory where your project is saved. Select the file whose name is the same as the project with an `.xml` extension and click **Open**. The project displays in the Configuration Components pane.
   - To open a project that was recently opened, select the project from the recently used projects list in the File menu. These projects display as a numbered list where the most recently used project is first in the list. Select the project to open in the Configuration Components pane. The number of projects that display in the most recently used list may be changed from the Workbench Preferences window, which is accessed by selecting **Tools Preferences** from the **File** menu.

Open an Existing Project from an Older Version of the Configuration Tools

If you attempt to open a project saved in a previous version of the RPASCE Configuration Tools, a window may open that allows you to convert the configuration to the new version.

Complete the following steps to open an existing project from an older version of the Configuration Tools.

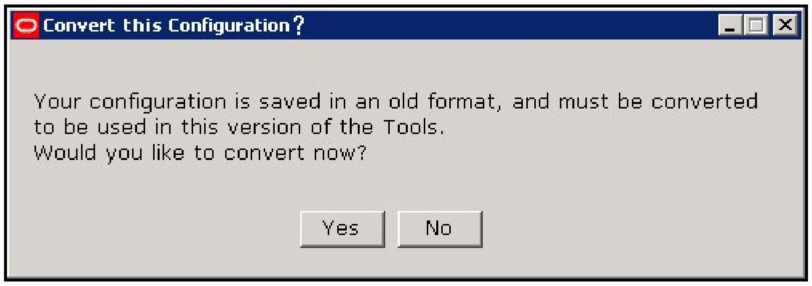
Navigate: From the File menu or right-click from the Configuration Components pane, select Open. The Figure 3-3 opens.

1. Choose one of the following methods:
   a. Browse to the directory where your project is saved. Select the file whose name is the same as the project with an `.xml` extension and click **Open**. The project opens in the Configuration Components pane.
   b. To open a project that was recently opened, select the project from the recently used projects list in the File Menu. These projects will be in a numbered list where the most recently used project is first in the list. Select the project, and it opens in the

configuration manager. The number of projects that opens in the most recently used list may be changed by using the Tools Preferences option of the File menu.

A message box opens:

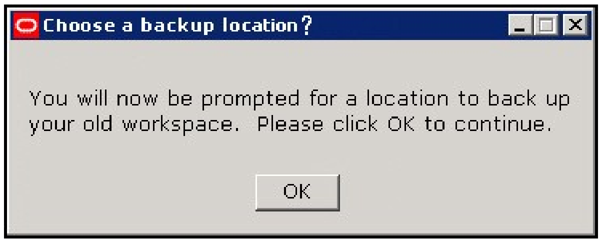
Figure 3-4   Convert This Configuration Message Box



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

2. To convert at a later time without currently viewing or modifying the project, click **No**. Click **Yes** to convert the project so it can be viewed or modified. If **Yes** is selected, the Figure 3-5 opens.

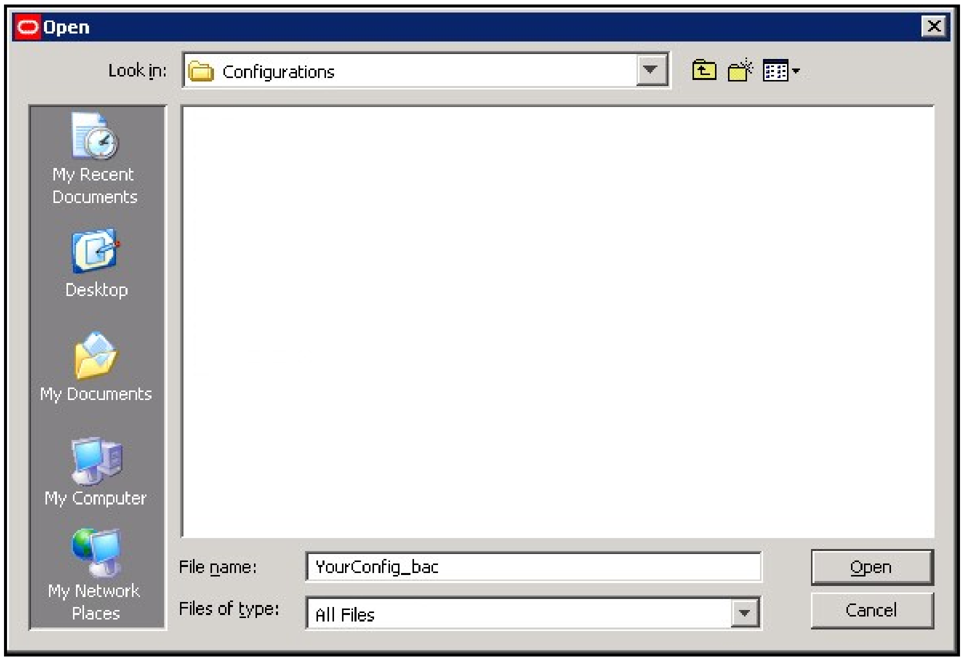
Figure 3-5   Choose a Backup Location Message Box



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

3. Click **OK**. The Figure 3-6 opens and prompts you for a location to store a backup of the project that is to be converted.

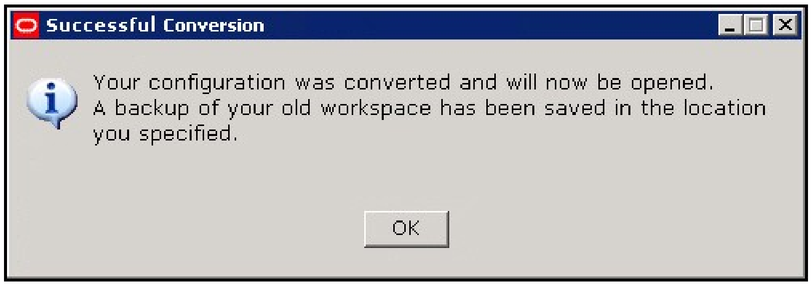
Figure 3-6 Open Window for Saving Configuration Backup



> **Note:**
>
> You may either rename the original configuration that is to be backed up or specify a new directory to store the original.

4. Select the directory to store the backup and click **Open**. The conversion process begins. If the conversion successfully completes the following message displays. Click **OK** to continue and view the project.

Figure 3-7 Successful Conversion Window



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

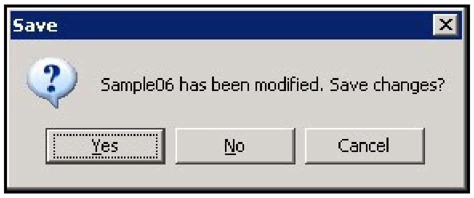
An error message opens if this process fails. The original project remains untouched and it will not open.

Close a Project

Complete the following steps to close a project.

Navigate: From the File menu or right-click from the Configuration Components pane, select **Close**. If no changes were made to the Project, the project is closed. If changes were made, the Figure 3-8 opens.

Figure 3-8 Save Window



Sample06 has been modified. Save changes?

Complete one of the following options:

- Click **Yes** to save the changes made to the project and close it.
- Click **No** to discard the changes made to the project since the last save and close it.
- Click **Cancel** to return to the configuration manager without closing the project.

Understanding Hierarchies

A hierarchy is a top-to-bottom set up of parent-child relationships between elements of the same type. Hierarchies provide a means to define relationships between dimensions (aggregates, roll-ups, and alternate roll-ups) and groups belonging to the same entity (for example, Time = years, months, weeks, and days).

The following hierarchies are automatically created and cannot be deleted within the Configuration Tools:

- CLND (Calendar)
- PROD (Product)
- LOC (Location)
- ADMU (User)
- LNGS (Languages)

These hierarchies are required by RPASCE-based solutions and cannot be removed, but additional hierarchies can be added to support the required business process.

Hierarchies define the path of data aggregation and spreading. In a workbook, the configuration administrator can view data at any required level of detail by drilling down or rolling up through dimensions in the hierarchy.

Note that ADMU and LNGS are not configurable hierarchies; therefore, they cannot be created or modified. ADMU and LNGS are built by RPASCE, and the RPASCE Configuration Tools makes them available for use in configurations. ADMU is the user hierarchy, and it exists to allow a measure to use the user dimension as part of its base intersection. LNGS is the language hierarchy, and it exists to support translation in multi-language applications. It is also available so that you can create a measure with the language dimension as part of its base intersection.

You can create and define dimensions for each of these hierarchies and for any additional hierarchies that are added to the project.

Note:

The names for the automatically generated hierarchies (CLND, PROD, LOC, ADMU, and LNGS) cannot be changed, but the default user labels for CLND, PROD, and LOC (Calendar, Product, and Location) can be changed. The user labels of ADMU and LNGS cannot be changed.

The CLND, ADMU, and LNGS hierarchies must exist in all applications, but PROD and LOC are not mandatory. If there are no dimensions created for the PROD and LOC hierarchies, the hierarchies are not created in the resulting PDS.

RPASCE does not impose any limit on the number of hierarchies that can be configured in a project. .

Configuration of Attribute Hierarchy Information

Dimension Attributes provide the ability to specify information about attributes that describe the positions within a hierarchy. For instance, Product Hierarchy positions may be represented by attributes describing color, brand, material, and so on.

Some applications that support processes that are heavily attribute driven make use of a construct often called an Attribute Hierarchy. An attribute hierarchy commonly contains two dimensions: a base dimension whose positions correspond to defined values for attributes and a parent dimension whose positions represent the attributes used by the application.

Using the color attribute as an example, an application containing an attribute hierarchy for Product would define positions for blue, green, red, and so on, on the attribute value dimension, which would then roll-up to the color position of the attribute position.

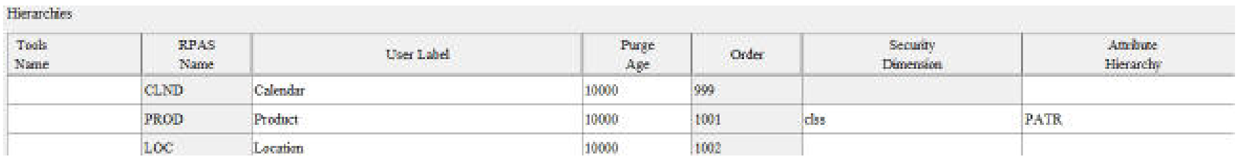
The existence of the attribute hierarchy allows applications to provide configured content that is aware of attributes and can use the values of attribute as an input into application processes. Examples of such functionality include rules whose execution logic depends upon the attribute values of the positions being evaluated or the ability to allow users to select attribute values from a drop-down list (as opposed to typing the attribute values into a text field.)

In order to leverage the existence of attribute hierarchies and to allow platform functionality to make use of the information contained within attribute hierarchies to simplify and improve the workflow of users, the hierarchy definition tool supports associating an attribute hierarchy with the hierarchy for which it provides attribute information.

In the previous example, an application contains an attribute hierarchy PATH (for Product Attribute) that contains information about attributes of the product hierarchy. By specifying PATH in the attribute hierarchy property of the product hierarchy, the configurer can provide this information to the platform so that core platform features can make use of the attribute information contained within PATH.

Figure 3-9, Figure 3-10, and Figure 3-11 illustrate how to configure the dimension attribute in Configuration Tools.

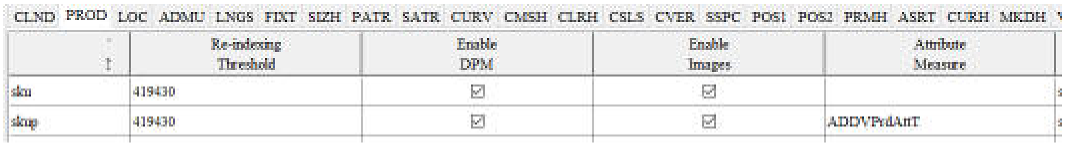
Figure 3-9 Associate Hierarchy to Attribute Hierarchy



Hierarchies

Tools Name	RPAS Name	User Label	Purge Age	Order	Security Dimension	Attribute Hierarchy
	CLND	Calendar	10000	999		
	PROD	Product	10000	1001	clss	PATR
	LOC	Location	10000	1002		

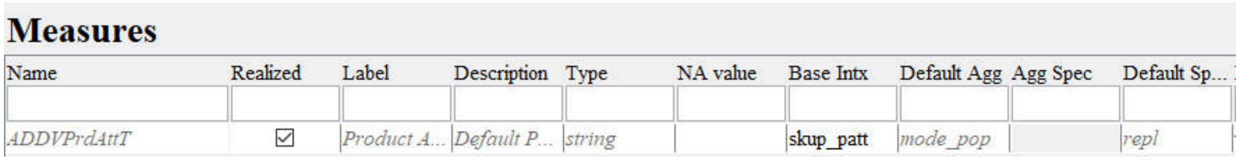
Figure 3-10 Associate Attribute Measure to Dimension



CLND PROD LOC ADMU LNGS FIXT SIZH PATR SATR CURV CMSH CLRH CSLS CVER SSPC POS1 POS2 PRMH ASRT CURH MKDH

	Re-indexing Threshold	Enable DPM	Enable Images	Attribute Measure
sku	419430	☑	☑	
skup	419430	☑	☑	ADDVPrdAttT

Figure 3-11 Defining Measure Attribute



Measures

Name	Realized	Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spec	Default Sp...
ADDVPrdAttT	☑	*Product A...*	*Default P...*	*string*		skup_patt	*mode_pop*		*repl*

Configuration of Virtual Hierarchy Information

Applications such as RDF sometimes require mirrored hierarchies and dimensions replicating other hierarchies and dimensions for their business logic. For example, RDF contains both PROD and PROR hierarchies, while PROR mirrors the PROD hierarchy. Measures can be registered at an intersection that combines the PROR and the original PROD hierarchies. Workbooks can also have worksheets defined on the PROR hierarchy.

The mirrored hierarchy is called the virtual hierarchy of the original hierarchy, and the dimension within the mirrored hierarchy is called the virtual dimension of the original dimension. In the RDF example, PROR is the virtual hierarchy of PROD, and the dimensions along the PROR hierarchy are the virtual dimensions of the dimensions along the PROD hierarchy.

With Virtual Hierarchy, the mirrored hierarchy can be implemented as a simple alias of the original hierarchy instead of as a complete individual hierarchy. Whenever any dimension of the virtual hierarchy is accessed, the accessor is automatically directed to the original dimension. Since the virtual hierarchy is simply an aliased version of the original hierarchy, what happens to the original hierarchy is immediately reflected in the virtual hierarchy. Virtual Hierarchy configuration avoids loading the essentially same hierarchy twice and improves performance in many ways.

The user must configure the virtual hierarchy and the virtual dimension in the Hierarchy Definition window (more details are provided in later sections). In a built RPASCE PDS, a virtual hierarchy is simply an alias of the original hierarchy, including all the dimensions. The virtual hierarchy and all its dimensions do not maintain their own position set and dimension dictionary in an RPASCE PDS. They always reference the original hierarchy's position set and dimension dictionary. Any changes made to the original hierarchy/dimension will then be automatically reflected in the virtual hierarchy/ dimension. In the built workbook, the virtual hierarchy will be materialized and become a real hierarchy as part of the workbook build, so there will be no virtual hierarchy within the workbook. This is because the workbook may require the original and the virtual hierarchies to have different ranges. In the case of RDF, the two ranges are complimentary, so it is possible that there is no common intersection in positions.

The positions in the virtual hierarchy and the virtual dimension inherit the translation from the original hierarchy and dimension, and there is no need to load the translated labels for the virtual hierarchy and the virtual dimension.

Virtual Hierarchy and Dimension Constraints

The following constraints apply:

- Cannot be defined for RPASCE internal hierarchies.

  The virtual hierarchy can be defined for regular loaded hierarchies such as product, location, and calendar, but cannot be defined for RPASCE internal hierarchies such as Meta, Data, Rgps, and so on.

- Virtual in PDS only and not in the workbook.

  The virtual hierarchy and dimension are only virtual in the PDS. They cannot be virtual in the workbook because the mirrored dimension may have a different range than the original one. In the workbook, when the workbook property Sync DPM to Virtual Dimension is not set, if the user adds a DPM position to the original dimension, the position will not be available in the same workbook, although it will be immediately available in the PDS’s virtual dimension. To pull the new position into the mirrored dimension in the workbook, the user must rebuild the workbook.

  Users can set Sync DPM to Virtual Dimension property to true to synchronize DPM positions created on the original dimension to the virtual dimension at the end of the DPM creation process in the same workbook. Refer to Sync DPM Positions to Virtual Dimension section in Chapter 4 for details.

- No DPM support for virtual hierarchies in workbook.

  Users must only do DPM on the original hierarchy.

- Cannot be partitioned.

  RPASCE only allows one partitioning dimension in one PDS configuration. So even though the user can define a virtual hierarchy for PROD, the virtual hierarchy will not be partitioned.

- Cannot be considered as Calendar.

  This is the same as the as partitioning issue. Within the RPASCE application, only one hierarchy can be considered as Calendar. Note that the hierarchy name must be **`clnd`** for it to be considered as Calendar. Even though the user can define a virtual hierarchy mirroring the calendar, the virtual hierarchy cannot be considered as Calendar nor used for calendar functions.

- Cannot share dimension attributes, images, or any measure-based data with its original hierarchy in the workbook.

  A measure registered against the original hierarchy is not accessible for the virtual hierarchy in the workbook. The original dimension and the virtual dimension may have different ranges in the workbook, so data stored for the original hierarchy will not be available for the virtual hierarchy in the workbook. The virtual hierarchy must define and manage its own data, such as attribute, images, and so on, as long as they are stored in measures.

- Dynamic hierarchies and hier mods not supported for the virtual hierarchy in Workbook.

  In the workbook hierarchy pane of the Workbook Designer, the dynamic hierarchies and hier mods are unavailable for virtual hierarchies. If such functions are required, they must be provided on the original hierarchy. It is preferable that the virtual hierarchy be presented in its original topology.

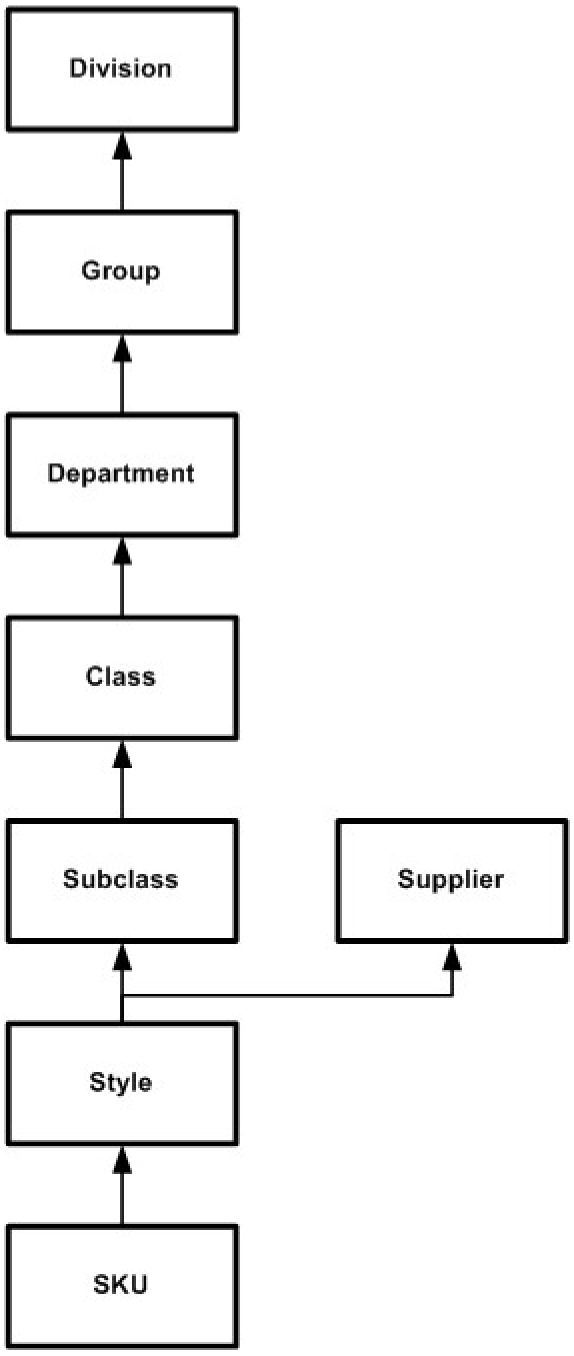
The Hierarchy Definition Window

The Hierarchy Definition window allows you to define and construct hierarchies, dimensions for each hierarchy, and the relationships between dimensions. It also offers the following features:

- Provides a visual representation of a hierarchy and its dimensions
- Provides a means to define the hierarchy data load file
- Allows existing hierarchies/dimensions to be reused in a new solution in the same project

The Figure 3-12 diagram represents a typical structure of an organization's product hierarchy.

Figure 3-12 Example of Product Hierarchy

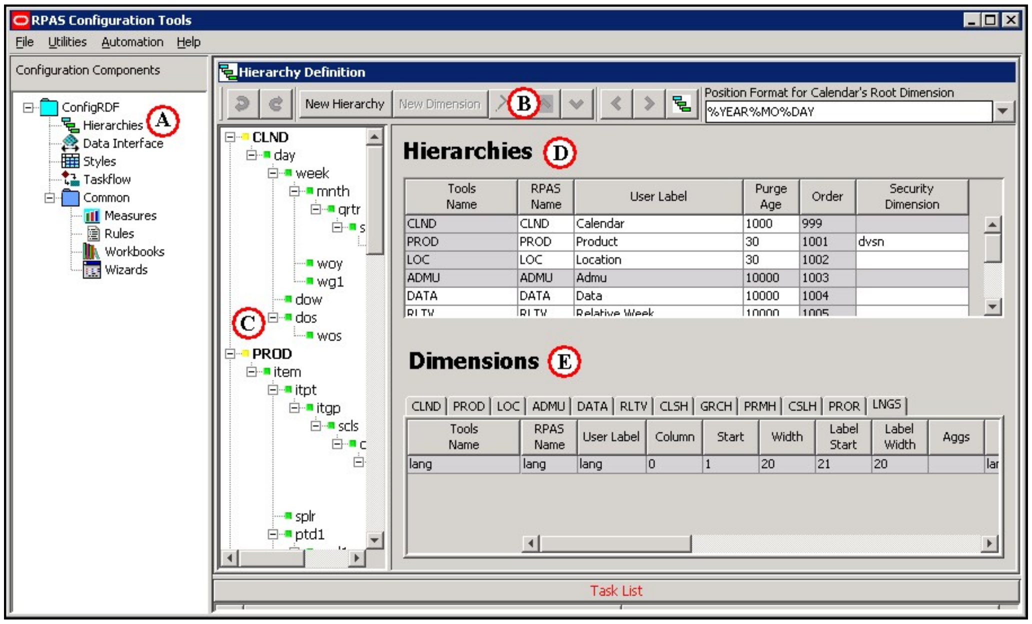


In the Figure 3-12 example, the Style dimension has two parents: Subclass and Supplier. Each position in the Style dimension will have a parent position in both the Subclass and Supplier dimension.

About the Hierarchy Definition Window

To access the Hierarchy Definition window, select Hierarchies (A) from the Configuration Components pane. The Hierarchy Definition window opens in the workspace.

Figure 3-13 Example of Hierarchy Definition Window



Tools Name	RPAS Name	User Label	Purge Age	Order	Security Dimension
CLND	CLND	Calendar	1000	999	
PROD	PROD	Product	30	1001	dvsn
LOC	LOC	Location	30	1002	
ADMU	ADMU	Admu	10000	1003	
DATA	DATA	Data	10000	1004	
RLTV	RLTV	Relative Week	10000	1005	

Tools Name	RPAS Name	User Label	Column	Start	Width	Label Start	Label Width	Aggs	
lang	lang	lang	0	1	20	21	20		lar

The Hierarchy Definition window contains the following elements:

Table 3-1 Hierarchy Definition Window Elements

ITag	Element	Description
A	Configuration Components pane	• Displays information about configurations, projects, and solutions that are currently in use • Configuration information is not displayed until a configuration is opened.
B	Hierarchy Definition toolbar	This toolbar displays options that can be performed. Buttons are enabled or unavailable based on the item selected on screen.
C	Hierarchy navigation tree	The navigation tree provides a visual representation of your hierarchies. Bold elements at the top of the tree structure represent the hierarchies. The items listed underneath each bold hierarchy are the dimensions defined in that hierarchy. Click **Plus** or **Minus** to expand the tree. The Hierarchy tree is also used to select a hierarchy or hierarchy dimension. Once an item is selected, you can modify its properties from the Dimension region in the Hierarchy Definition window. The Hierarchy navigation tree also provides a context menu when you right-click a tree item. The available options in the context menu depend on whether a hierarchy or dimension is selected. This context menu can be used to create a new hierarchy or dimension at the selected level. It also allows you to rename the selected item. When an item is renamed from the tree, it is the Tools Name that is being modified, which displays in the Dimensions region of the window.

Table 3-1 (Cont.) Hierarchy Definition Window Elements

ITag	Element	Description
D	Hierarchies region	This area displays the defined hierarchies and their properties.
E	Dimensions region	This area contains hierarchy tabs and allows you to define the dimension properties for your hierarchies. The tabs represent the hierarchies defined. Select the appropriate hierarchy tab to display its dimensions and modify dimension properties.

Gray fields in the Hierarchy Definition window indicate fields that cannot be modified. Any elements that display in red indicate problems or issues must also display in the Task List pane along with a brief description of the issues identified.

Working with Hierarchies

This section describes how to:

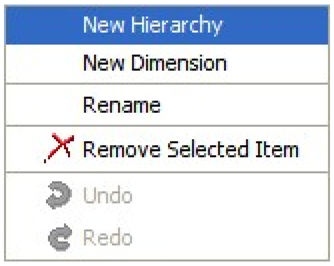
- Create a New Hierarchy
- Specify Hierarchy Properties
- Delete a Hierarchy
- Copy (Clone) Hierarchies
- Working with Position Formats
- Specify Hierarchy Properties
- Delete a Hierarchy
- Copy (Clone) Hierarchies
- Working with Position Formats
- Specifying the Position Format

Create a New Hierarchy

When a new project is created, following default hierarchies are automatically created: Calendar (CLND), Product (PROD), Location (LOC), User (ADMU), and Languages (LNGS). Additional hierarchies and dimensions can be created to meet your business needs.

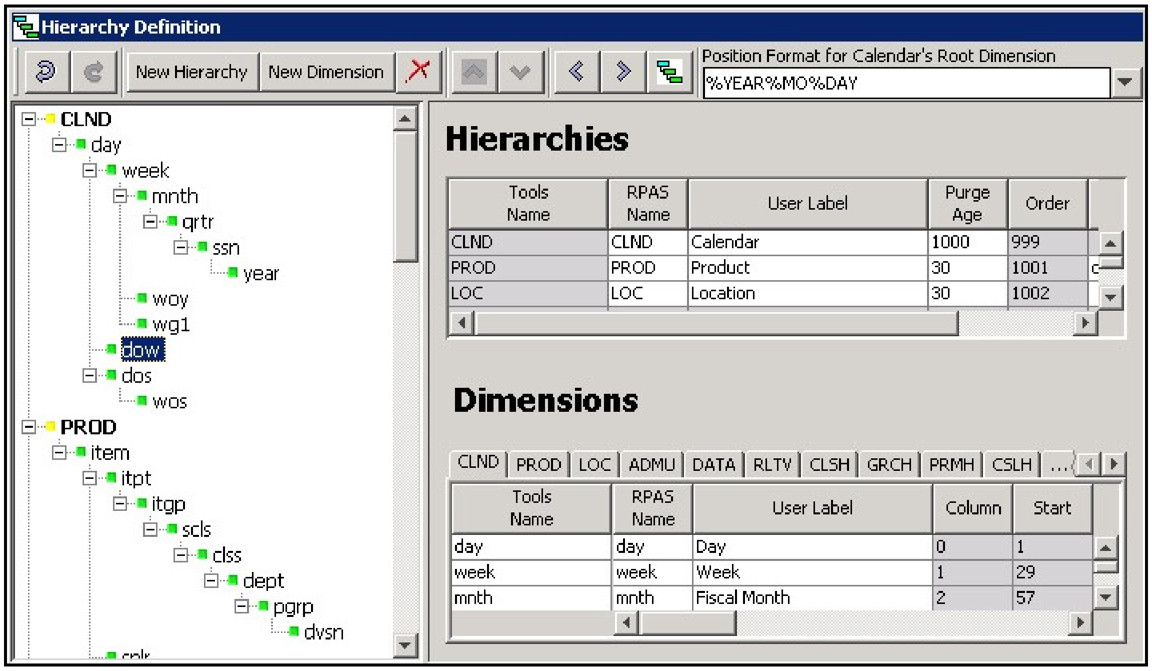
Navigate: Select **New Hierarchy** from Hierarchy Definition toolbar, or from the Hierarchy Definition tree, right-click and select **New Hierarchy** from the menu.

Figure 3-14 Select New Hierarchy



> **Note:**
>
> If multiple projects are open, make sure you are working from the desired project before adding a new hierarchy.

Figure 3-15 Hierarchy Definition Window



Tools Name	RPAS Name	User Label	Purge Age	Order
CLND	CLND	Calendar	1000	999
PROD	PROD	Product	30	1001
LOC	LOC	Location	30	1002

Tools Name	RPAS Name	User Label	Column	Start
day	day	Day	0	1
week	week	Week	1	29
mnth	mnth	Fiscal Month	2	57

Modify the Tools Name

To change the Tools Name of the newly created hierarchy in the Hierarchy navigation tree of the Hierarchy Definition window.

1. Choose one of the following methods:
   - Right-click on the hierarchy name and select **Rename**.
   - Double-click the hierarchy name.

2. Enter the new name.

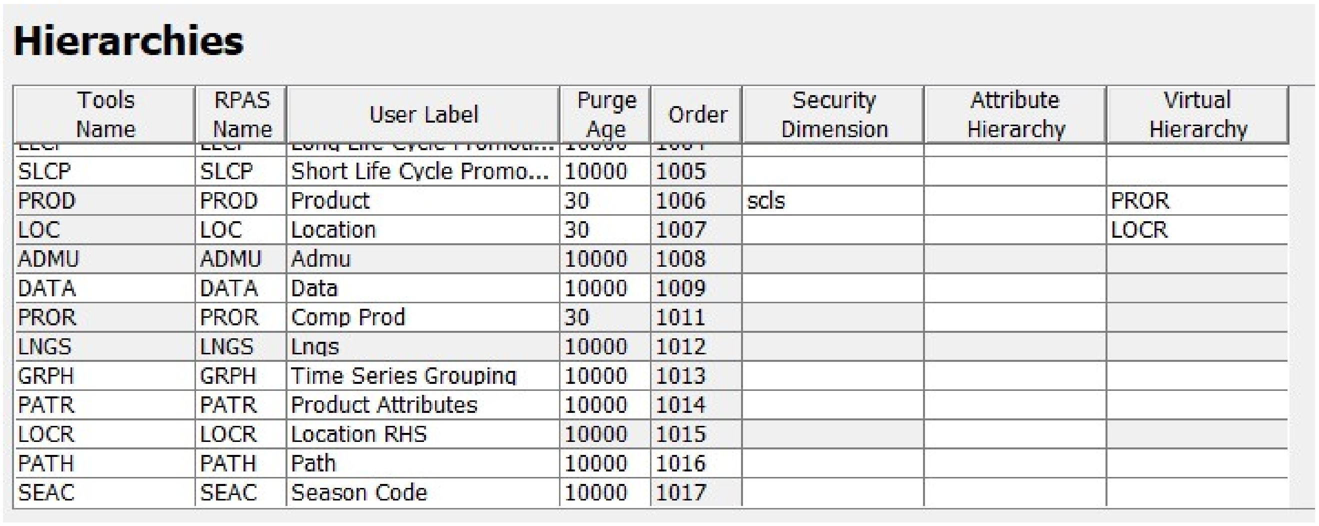
> **Note:**
>
> The RPASCE name can only be up to four (4) characters long.

3. Press **Enter** or click outside the hierarchy name.

Specify Hierarchy Properties

Hierarchy properties are defined from the Hierarchies region on the Figure 3-15.

Figure 3-16 Hierarchy Properties Region



Hierarchies

Tools Name	RPAS Name	User Label	Purge Age	Order	Security Dimension	Attribute Hierarchy	Virtual Hierarchy
SLCP	SLCP	Short Life Cycle Promo...	10000	1005			
PROD	PROD	Product	30	1006	scls		PROR
LOC	LOC	Location	30	1007			LOCR
ADMU	ADMU	Admu	10000	1008			
DATA	DATA	Data	10000	1009			
PROR	PROR	Comp Prod	30	1011			
LNGS	LNGS	Lngs	10000	1012			
GRPH	GRPH	Time Series Grouping	10000	1013			
PATR	PATR	Product Attributes	10000	1014			
LOCR	LOCR	Location RHS	10000	1015			
PATH	PATH	Path	10000	1016			
SEAC	SEAC	Season Code	10000	1017			

> **Note:**
>
> Adding new hierarchies and new dimensions to an existing hierarchy are supported during patching. The new dimension can be either a root dimension, a leaf dimension or a dimension in between.

From this location you can modify the following hierarchy properties:

- Tools Name
- RPAS Name
- User Label
- Purge Age
- Order
- Security Dimension

- Attribute Hierarchy
- Virtual Hierarchy
- Position Filter Measure

Tools Name

The name of the hierarchy that displays within the RPASCE Configuration Tools. This field is less restrictive than the RPAS Name field, allowing you to view and select a meaningful label for hierarchies and dimensions while working with the configuration rather than using the RPAS Name.

RPAS Name

The RPAS internal name of the hierarchy. This hierarchy name is used only by RPASCE (not the user) within the application.

> **Note:**
>
> The RPAS Name of a hierarchy cannot be edited if it is shaded gray; however, you can change other properties, such as User Label.
>
> CLND is always the innermost dimension. The order of the other hierarchies (PROD, LOC, and so on) can be changed.

User Label

The hierarchy label that is displayed to RPASCE users within the application.

Purge Age

The purge age determines when a position and its corresponding measure data are removed from a PDS instance. Specifically, it represents the number of days before the data is purged from the last time the position was included in the hierarchy input file that is loaded with the loadHier utility during a batch run (most commonly on a nightly or weekly basis). Setting this value to zero means that a position and all of its data will be immediately purged if it is not included in the hierarchy file.

> **Note:**
>
> The value set in this field serves as the default value to use when loading the corresponding hierarchy. This value can be overwritten by one of the arguments of the `loadHier` utility each time the utility is called. See the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide* for more information on the `loadHier` utility.

Example 1: A purge age of *0* will purge positions the first night they are not in the input file.

Example 2: A purge age of *1000* will purge the positions the 1000th night after they are last seen on the input file.

Order

Hierarchy order determines the ordering of dimension fields in the physical storage of data in the RPASCE PDS instance. This ordering is the traversal order of data for calculations, which relates to how RPASCE iterates over data when performing calculations. Data in the PDS is stored in Oracle database tables that have columns representing dimensions, with each dimension belonging to a different hierarchy.

To change the order of a hierarchy, select the hierarchy from the Hierarchies region or from the Hierarchy navigation tree and use the up and down arrow buttons located on the Hierarchy Definition toolbar to move the hierarchy to the desired location.

The hierarchy can also be arranged by dragging and dropping in the Hierarchy navigation tree. The order numbers are automatically changed and generated regardless of the utilized reordering technique.

For performance reasons, the Calendar hierarchy (and therefore all of its dimensions) is always the innermost dimension and defaults to an unedited number of 999. The ordering of any hierarchy can be changed with the exception of Calendar (CLND). The lower the order number, the nearer the hierarchy is to the innermost dimension.

Consider the following example for the Calendar, Product, and Location hierarchies:

- CLND order = 999
- PROD order = 1001
- LOC order = 1002
- Two products: P1 and P2
- Two locations: L1 and L2
- Two calendar periods: C1 and C2

The sequence of physically storing and iterating over the data with calendar as the innermost dimension and location as the outermost dimension is:

C1,P1,L1

C2,P1,L1

C3,P1,L1

C1,P2,L1

C2,P2,L1

C3,P2,L1

C1,P3,L1

C2,P3,L1

C3,P3,L1

C1,P1,L2

C2,P1,L2

C3,P1,L2

C1,P2,L2

C2,P2,L2

C3,P2,L2

With Calendar being the innermost dimension, data is first processed for all positions in the Calendar hierarchy and for the first position of the other hierarchies. In this example, data would be processed for all calendar positions for the first product and first location. This is followed by all calendar positions for the second product and first location, and so on.

It is recommended that retailers order their hierarchies with Calendar as the innermost dimension (required), followed by other hierarchies in their order of importance/ traversal – most commonly Product, Location, and then other hierarchies (if applicable).

Note:

Certain RPASCE-based solutions (such as Advanced Inventory Planning and Demand Forecasting) have additional hierarchies that are in a predefined order that should not be changed.

The Order column also indicates the order in which the hierarchy information is expected in the file used for measure data loading purposes.

Note:

The values *1000* or *1020* are not used as a hierarchy order as they are used internally by RPASCE.

CLND is always the innermost dimension and ADMU is always the outermost dimension.

The order of the other hierarchies (PROD, LOC, and others created by the configuration administrator) can be changed within the ConfigTools.

The relative hierarchy order in PDS is patchable.

Security Dimension

Selecting a security dimension for a hierarchy enables position-level security in the application for the corresponding hierarchy. Any dimension along any hierarchy except the Calendar hierarchy is valid. For example, if the security dimension for the product hierarchy is set to *Dept* (Department Level Security) within the application, access to departments can be granted or denied by the administrator for individual users, user groups, or all users. If position-level security is to be enabled in RPASCE, select the security level. Refer to the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide* for additional information about position-level security.

Note:

The Security Dimension hierarchy can be changed and patched. It will not adversely affect the results if changed.

Attribute Hierarchy

Specifying an attribute hierarchy for a hierarchy instructs the system that the information contained within the attribute hierarchy should be used to assist operations involving attributes for the hierarchy so specified. For example, specifying the value PATH for the Product hierarchy specifies that a hierarchy named PATH exists in the application whose positions contain attribute information about the Product hierarchy.

Note:

The Attribute hierarchy can be changed and patched.

Virtual Hierarchy

Specifying a virtual hierarchy for this hierarchy. This is an optional property and can be left blank when no virtual hierarchy exists for this hierarchy.

The hierarchy order of the virtual hierarchy must be different than its original hierarchy.

Both the Security Dimension and Purge Age columns are not editable for the virtual hierarchy since the virtual hierarchy inherits them from its original hierarchy.

The attribute hierarchy of the virtual hierarchy can be configured by the user. The virtual hierarchy can have the same or a different attribute hierarchy than its original hierarchy.

A virtual hierarchy cannot have another hierarchy defined as its virtual hierarchy as its virtual hierarchy column is unavailable from editing.

All dimensions within a virtual hierarchy are virtual dimensions and should be configured so in the Dimensions table. If a hierarchy has virtual hierarchy, not all dimension within the original hierarchy must have a corresponding virtual dimension. This might be a common case in a multi-application situation. For example, both MFP and RDF have PROD hierarchy, and PROD in RDF has virtual hierarchy PROR configured. In the merged PROD hierarchy in PDS, the dimensions that are only configured inside MFP application configuration do not have corresponding virtual dimension in PROR hierarchy. See Multi-Application Support for more details.

Note:

The virtual hierarchy can be changed and patched in the RPASCE application. The virtual hierarchy must sync with the changes in the Virtual Dimension table. If a virtual hierarchy is changed from set to unset in the Hierarchies table, then all its virtual dimensions must be unset as well in the Dimensions table, and vice versa.

Position Filter Measure

Specify a position filter measure for a hierarchy to support position visibility filtering (PVF). This is an optional property and can be left blank when the hierarchy is not filtered. Once the filter measure is specified, for any hierarchy except CLND, only positions contained within the filter measure and their rollup positions are visible to the application. When CLND hierarchy is filtered, only a consecutive range is considered. The CLND filter measure only needs to be loaded with the beginning and ending position, for example, the beginning and ending days.

- For CLND hierarchy, all positions between the beginning and ending points will be visible. Therefore, user can only load the beginning and the ending positions to have the range between them to be visible to the application.
- For all NON-CLND hierarchies, positions are filtered one by one. Only the positions that are set to true in the Filtering measure will be visible.

Visibility of a rollup position is automatically computed and it is visible if one of its base positions is visible.

Note:

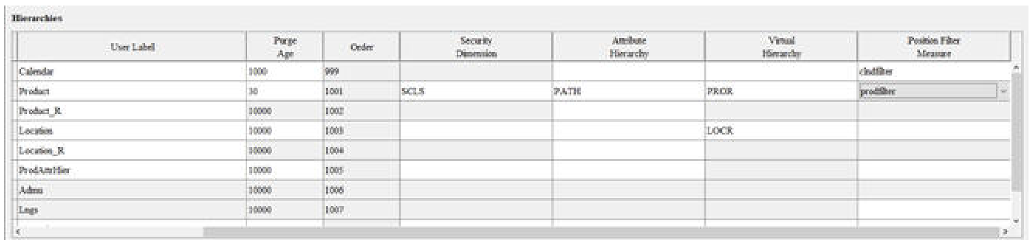
Position Visibility Filter cannot be used to filter DPM positions. It cannot be used to “hide” a DPM position from the creating application. It cannot be used to force a DPM position to be visible to an application that is not the position’s creating application.

Note:

Position Visibility Filter cannot be used to circumvent position security setting in Hierarchy property.

Positions in virtual hierarchy inherits the same visibility filter from the source hierarchy.

Figure 3-17 Position Filter Measure



Hierarchies

User Label	Purge Age	Order	Security Dimension	Attribute Hierarchy	Virtual Hierarchy	Position Filter Measure
Calendar	1000	999				clndfilter
Product	30	1001	SCLS	PATH	PROR	prodfilter
Product_R	10000	1002				
Location	10000	1003			LOCR	
Location_R	10000	1004				
ProdAttrHier	10000	1005				
Admu	10000	1006				
Lngs	10000	1007				

The Position Filter Measure must meet the following criteria:

- A Boolean 1-dimensional measure with the base dimension at the root of the filtering hierarchy (ex, SKU, STOR).

- Must have storage database, have Base and Agg state as READ and Boolean OR as the default ag type.
- Measure default NAVALUE is false.
- The measure is updated from the data source only, i.e., either from flat file or from integration interface table. It should not be included in any workbook or batch rules that can update its value.
- The position filter measure should not be shared on DPM-enabled hierarchy.

When users click the Position Filter Measure field, a drop-down list will be pre-populated with candidate measures and users can then select the right filter measure.

Positions in virtual hierarchy is automatically filtered the same way as the source (i.e. original) hierarchy.

Position security is applied after the visibility filter.

Visibility filter(s) are excluded from the Measure Analysis workbook.

Delete a Hierarchy

Complete the following steps to delete a hierarchy.

Navigate: In the Configuration Components pane, select **Project** and then **Hierarchies**. The Figure 3-15 opens in the workspace.

1. Select the hierarchy to delete.
2. Choose one of the following methods:
   - Click **Delete**. The hierarchy is removed.
   - Press **Delete** on your keyboard.
   - Use the right-click menu to select **Remove Selected Item**.

> **Note:**
>
> The CLND, PROD, LOC, ADMU, and LNGS hierarchies cannot be deleted from the configuration.

Copy (Clone) Hierarchies

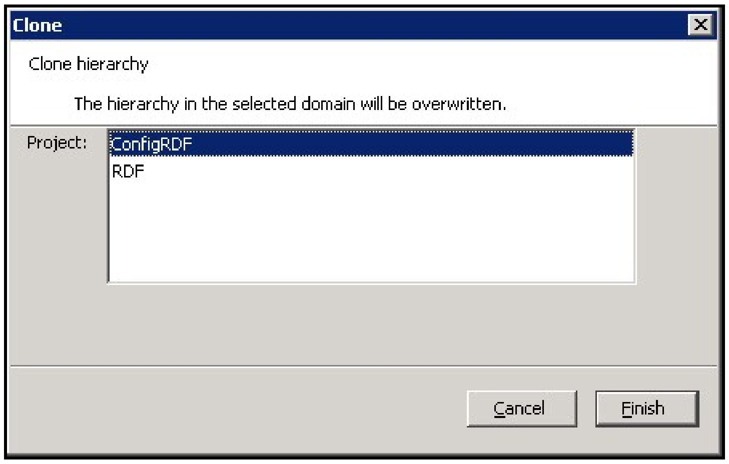
The RPASCE Configuration Tools allows for the hierarchies of an existing project to be copied into a new or existing project.

Complete the following steps to copy (clone) hierarchies.

Navigate: In the Configuration Components pane, select **Project** and then **Hierarchies**. The Figure 3-15 opens in the workspace.

1. Right-click **Hierarchies** in the Configuration Components pane and select **Copy**. The Figure 3-18 opens.

Figure 3-18 Clone Window



Clone

Clone hierarchy

The hierarchy in the selected domain will be overwritten.

Project: ConfigRDF
RDF

2. Select the destination project for the hierarchies to be copied.
3. Click **Finish**. The hierarchies in the selected project are overwritten.

> **Note:**
>
> Each project has a single set of hierarchies. Hierarchies can only be copied from one project to another, thus multiple projects must be open in the RPASCE Configuration Tools before the copy process is initiated.

Working with Position Formats

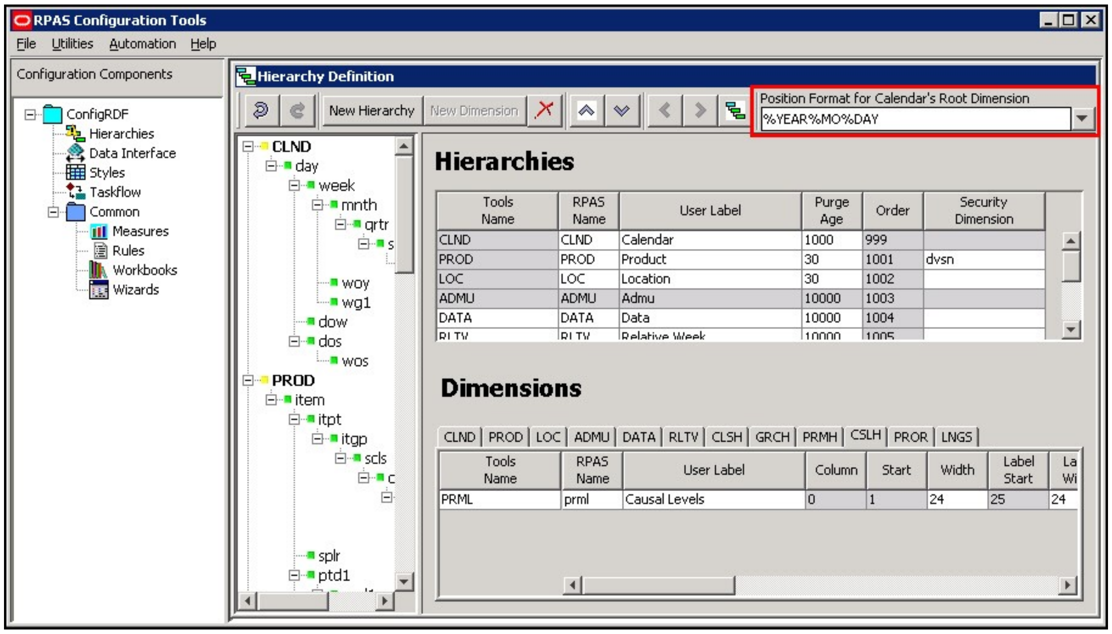
The Position Format is the date/time format used for the names of positions in the root dimension of the CLND (Calendar) hierarchy (typically *day*). Positions in the root dimension of the CLND hierarchy need names in a special format for RPAS CE to map abstract positions to actual dates and times in order to support time-aware calculations.

> **Note:**
>
> See Appendix – Calculation Engine User Guide and Appendix – Rules Function Reference Guide for more information.

Specifying the Position Format

Navigate: In the Configuration Components pane, select **Project** and then **Hierarchies**. The Figure 3-15 opens in the workspace.

Figure 3-19   Example of Position Format in Hierarchy Definition Window



Position Format for Calendar's Root Dimension: %YEAR%MO%DAY

Hierarchies

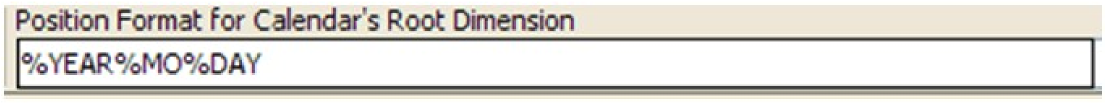
Tools Name	RPAS Name	User Label	Purge Age	Order	Security Dimension
CLND	CLND	Calendar	1000	999	
PROD	PROD	Product	30	1001	dvsn
LOC	LOC	Location	30	1002	
ADMU	ADMU	Admu	10000	1003	
DATA	DATA	Data	10000	1004	
RLTV	RLTV	Relative Week	10000	1005	

Dimensions

Tools Name	RPAS Name	User Label	Column	Start	Width	Label Start	La Wi
PRML	prml	Causal Levels	0	1	24	25	24

The Figure 3-20 is located in the upper right-hand side of the Hierarchy Definition toolbar.

Figure 3-20   Position Format Field



Position Format for Calendar's Root Dimension

%YEAR%MO%DAY

This is a combo box that is populated with some of the more commonly used formats. However, the configuration administrator may also type directly in the combo box if a different format is desired.

Specify the position format as a concatenated sequence of strings and arguments using the appropriate syntax. Refer to Position Format Syntax for more information.

Position Format Syntax

The Position Format field uses the following syntax conventions:

- %YEAR – Four-digit Gregorian calendar year.
- %YR – Two east significant digits of the year (for example, 15 is 2015).
- %MO – Two-digit representation of month (for example, 01 is January).
- %MON – Three-character abbreviation of the month name.
- %MONTH – Varying length full name of the month, displays up to nine characters.

Note:

Even when configuring a solution in another language, RPASCE expects the month names and abbreviations (%MONTH and %MON) used in the position names to be in English (for example, Jan, Feb, Mar, and so on).

- %DAY – Two-digit representation of the day of the month (for example, 01 is the first day of the month)
- %HR – Two-digit representation of the hour of the day (for example, 22 is 10 p.m.).
- %MIN – Two-digit representation of minutes past the hour.
- %SEC – Two-digit representation of seconds past the current minute.
- %MSEC – Three-digit milliseconds past the current second.

The Position Format is not case sensitive, so %YEAR is the same as %year.

The length of the position name must not exceed 24 characters. The Position Format field performs validation on the Position Format in order to enforce this limitation. For example, the Position Format %YEAR%MONTH%DAY evaluates to a total of 15 characters (4 for the year, 9 for month, and 2 for day).

Examples:

- Format: %YEAR%MO%DAY
- A position that represents the January 31, 2013 would have the name 20130131
- Format: %YR%MON%DAY
- A position that represents the January 31, 2013 would have the name 13Jan31

Working with Dimensions

Dimensions are the components within a hierarchy that define the structure and roll up within a hierarchy. For example, the dimensions for a calendar hierarchy can be day, week, month, and year, or they can be accounting periods.

Create a Dimension

Complete the following steps to create a dimension.

Navigate: In the Configuration Components pane, select **Project** and then **Hierarchies**. The Figure 3-15 opens in the workspace.

1. Select the hierarchy or dimension under which to create the new dimension.

> **Note:**
>
> When this guide uses terms like top and bottom level dimensions or over and under, these terms are to be interpreted visually. The bottom level is at the top of the hierarchy, and the top levels are at the end of the hierarchy branches. For example, the top dimension visually is the root dimension, which is the lowest dimension in the hierarchy. The highest dimensions in the hierarchy are at the bottom end of the hierarchy branches. For instance, Day is the bottom level of a Calendar hierarchy, but it falls directly beneath CLND.

2. Choose one of the following methods:
   - From the Hierarchy Definition right-click menu, select **New Dimension**.
   - Click **New Dimension** from the toolbar.

   Create the first dimension, which becomes the root dimension, for a new hierarchy when positioned on the hierarchy. Once the root dimension is defined, new dimensions cannot be defined directly under the hierarchy. New dimensions are added under other dimensions. For example, after "Day" is added to the CLND hierarchy, CLND cannot be selected again to add "Hour." However, "Week" can be added under the "Day" dimension. There can only be one root dimension created per hierarchy.

   > **Note:**
   >
   > Certain processes that support the purging of data or positions and the mapping of real dates/times to positions require a dimension in the CLND hierarchy that is named "day" and represents the day level. Such a dimension must be defined, although the user label can be changed from "day" if needed for localization purposes.
   >
   > There is no limit on the number of dimensions that may be created for a hierarchy.

3. Define the dimension as necessary. Refer to "Defining Dimension Properties" for more information.

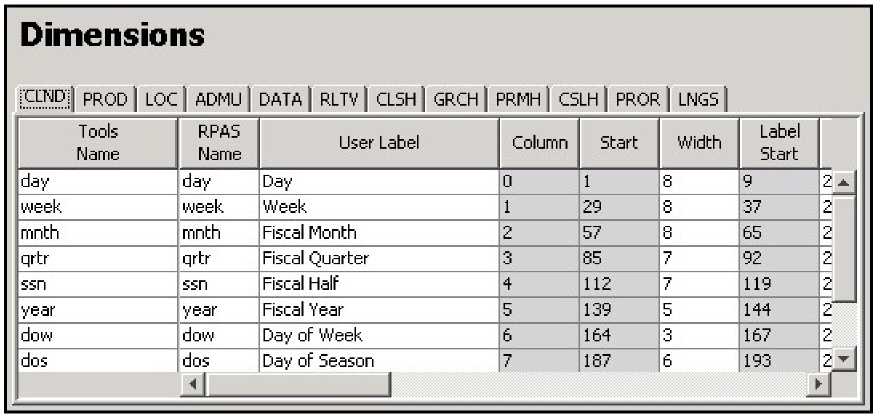
Defining Dimension Properties

Complete the following steps to define the dimension properties.

Navigate: In the Configuration Components pane, select **Project** and then **Hierarchies**.

1. Select a dimension using one of the following methods:
   - From the Hierarchy tree, select the dimension you want to modify.
   - From the Dimensions region of the Hierarchy Definition window, select the hierarchy tab that contains the dimension you want to define or modify.

Figure 3-21 First Half of the Dimensions Properties Window PROD Tab Selected

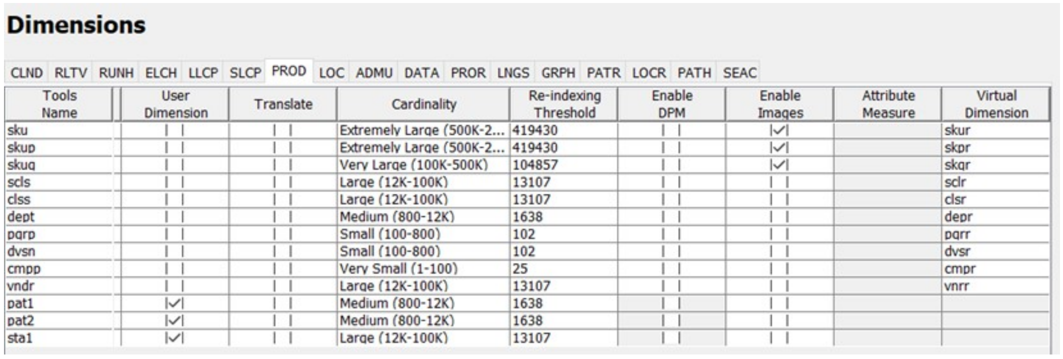


Dimensions

CLND | PROD | LOC | ADMU | DATA | RLTV | CLSH | GRCH | PRMH | CSLH | PROR | LNGS

Tools Name	RPAS Name	User Label	Column	Start	Width	Label Start	
day	day	Day	0	1	8	9	2
week	week	Week	1	29	8	37	2
mnth	mnth	Fiscal Month	2	57	8	65	2
qrtr	qrtr	Fiscal Quarter	3	85	7	92	2
ssn	ssn	Fiscal Half	4	112	7	119	2
year	year	Fiscal Year	5	139	5	144	2
dow	dow	Day of Week	6	164	3	167	2
dos	dos	Day of Season	7	187	6	193	2

Figure 3-22 Second Half of the Dimensions Properties Window PROD Tab Selected

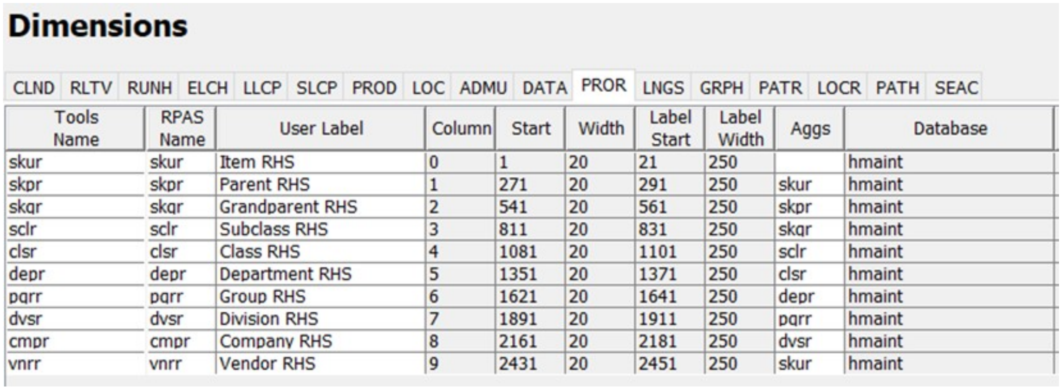


Dimensions

CLND | RLTV | RUNH | ELCH | LLCP | SLCP | PROD | LOC | ADMU | DATA | PROR | LNGS | GRPH | PATR | LOCR | PATH | SEAC

Tools Name	User Dimension	Translate	Cardinality	Re-indexing Threshold	Enable DPM	Enable Images	Attribute Measure	Virtual Dimension
sku	☐	☐	Extremely Large (500K-2...	419430	☐	✓		skur
skup	☐	☐	Extremely Large (500K-2...	419430	☐	✓		skpr
skug	☐	☐	Very Large (100K-500K)	104857	☐	✓		skgr
scls	☐	☐	Large (12K-100K)	13107	☐	☐		sclr
clss	☐	☐	Large (12K-100K)	13107	☐	☐		clsr
dept	☐	☐	Medium (800-12K)	1638	☐	☐		depr
pgrp	☐	☐	Small (100-800)	102	☐	☐		pgrr
dvsn	☐	☐	Small (100-800)	102	☐	☐		dvsr
cmpp	☐	☐	Very Small (1-100)	25	☐	☐		cmpr
vndr	☐	☐	Large (12K-100K)	13107	☐	☐		vnrr
pat1	✓	☐	Medium (800-12K)	1638	☐	☐		
pat2	✓	☐	Medium (800-12K)	1638	☐	☐		
sta1	✓	☐	Large (12K-100K)	13107	☐	☐		

Figure 3-23 First Half of the Dimensions Property Window PROR Tab Selected

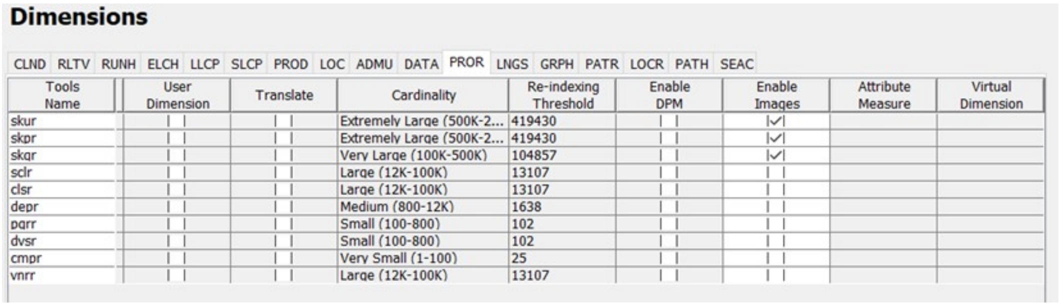


Dimensions

CLND | RLTV | RUNH | ELCH | LLCP | SLCP | PROD | LOC | ADMU | DATA | PROR | LNGS | GRPH | PATR | LOCR | PATH | SEAC

Tools Name	RPAS Name	User Label	Column	Start	Width	Label Start	Label Width	Aggs	Database
skur	skur	Item RHS	0	1	20	21	250		hmaint
skpr	skpr	Parent RHS	1	271	20	291	250	skur	hmaint
skgr	skgr	Grandparent RHS	2	541	20	561	250	skpr	hmaint
sclr	sclr	Subclass RHS	3	811	20	831	250	skgr	hmaint
clsr	clsr	Class RHS	4	1081	20	1101	250	sclr	hmaint
depr	depr	Department RHS	5	1351	20	1371	250	clsr	hmaint
pgrr	pgrr	Group RHS	6	1621	20	1641	250	depr	hmaint
dvsr	dvsr	Division RHS	7	1891	20	1911	250	pgrr	hmaint
cmpr	cmpr	Company RHS	8	2161	20	2181	250	dvsr	hmaint
vnrr	vnrr	Vendor RHS	9	2431	20	2451	250	skur	hmaint

Figure 3-24 Second Half of the Dimension Properties Window PROR Tab Selected



Dimensions

CLND | RLTV | RUNH | ELCH | LLCP | SLCP | PROD | LOC | ADMU | DATA | PROR | LNGS | GRPH | PATR | LOCR | PATH | SEAC

Tools Name	User Dimension	Translate	Cardinality	Re-indexing Threshold	Enable DPM	Enable Images	Attribute Measure	Virtual Dimension
skur	☐	☐	Extremely Large (500K-2...	419430	☐	✓		
skpr	☐	☐	Extremely Large (500K-2...	419430	☐	✓		
skgr	☐	☐	Very Large (100K-500K)	104857	☐	✓		
sclr	☐	☐	Large (12K-100K)	13107	☐	☐		
clsr	☐	☐	Large (12K-100K)	13107	☐	☐		
depr	☐	☐	Medium (800-12K)	1638	☐	☐		
pgrr	☐	☐	Small (100-800)	102	☐	☐		
dvsr	☐	☐	Small (100-800)	102	☐	☐		
cmpr	☐	☐	Very Small (1-100)	25	☐	☐		
vnrr	☐	☐	Large (12K-100K)	13107	☐	☐		

2. In the Dimensions properties region, select or double-click in the field to edit. Scroll to the right to see all of the fields. You can resize the columns by placing the cursor over the column until the double-sided arrow displays and then drag the column to the desired width.

   Other than the RPAS name, the columns can be reordered by dragging and dropping the headings. The column positions will return to the default order when the session is closed.

   You can specify the following dimension information:

> **Note:**
>
> Only four of the dimension properties are patchable in an existing PDS. Once the PDS is built, changes made to any other dimension properties are ignored.
>
> If a dimension's property is patchable, it stated in a note in its description listed in Dimension Properties Descriptions.

Dimension Properties Descriptions

Tools Name

The name of the dimension that is displayed within the RPASCE Configuration Tools. This field allows you to assign meaningful labels while working with a configuration in the Configuration Tools. For example, the Tools Name displays in Select Intersection window, making it easier for you to assign the appropriate intersections.

RPAS Name

The RPAS internal name of the dimension. This dimension name is used only by RPASCE (not the user) within the application.

User Label

The dimension description that is displayed to RPASCE users in the RPASCE Client.

> **Note:**
>
> Any alpha-numeric characters are allowed. Single or double quotes are not allowed.

Column

Identifies the order in which the dimension's positions fall in the meta-data load file. Use the left and right arrow buttons to change the order and the column value of the dimensions. Changing the column value will not affect the hierarchy structure or aggregation paths.

> **Note:**
>
> The up and down arrow buttons are used for defining the sequence of hierarchies.

Example:

Change the column values if the dimensions in the data load file are not in the same order as in the tree structure. Dimensions will be moved up or down in the table without impacting the aggregates.

Start

This is a read-only, calculated field. This field identifies the start position of the position names for this dimension in the hierarchy load file.

Width

This field identifies the width of position names for this dimension in the hierarchy load file.

Label Start

This is a read-only, calculated field. This field identifies the start position of the position label for this dimension in the hierarchy load file. The sum of the dimensions of the Start and Width fields determines the value of the Label Start.

Label Width

This field identifies the width of position labels for this dimension in the hierarchy load file.

> **Note:**
>
> When using comma separated value (CSV) files to load data into RPASCE, the following dimension are ignored: Column, Start, Width, Label Start, and Label Width. See the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide* for more information on Comma Separated Value (CSV) flat file format data load and export.

Aggs

This field establishes the relationship of the dimension to the other dimensions in the same hierarchy. Specifically, this field references the child dimension that aggregates up to this dimension (the parent dimension). It can be edited by using the drop-down list, or you can drag and drop dimensions in the left-hand side hierarchy pane.

User Dimension

When selected (checked), this field indicates that the dimension is user maintained. User-defined definitions (UDD) cannot have another dimension as a parent. Positions and position mappings (parent-child relationships) for user-defined dimensions are established in the RPASCE Administrative workbook template, “Hierarchy Maintenance." This meta-data cannot be loaded like regular (non-user-defined) dimensions in the hierarchy load process. A DPM-enabled dimension cannot be set to User Dimension and vice versa.

Translate

When selected, this field enables the position labels for the dimension to be translated into multiple languages. Positions are loaded into the PDS in the native language of the PDS via the standard hierarchy load process. Position labels for additional languages are loaded into special measures that are used in this multi-lingual PDS. With the proper setup, these translated position labels can be displayed in workbooks in the RPASCE Client instead of the loaded position labels. See the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide* for detailed instructions for enabling position label translation.

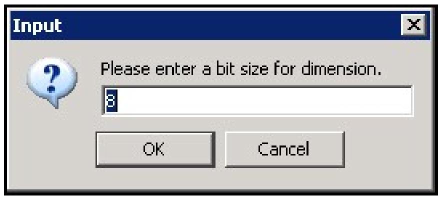
> **Note:**
>
> The PDS is always built as multi-lingual. Enabling the translation of a dimension is patchable and does not adversely affect the expected results if altered. However, disabling translation is not patchable and may result in failure.

Cardinality

Use this field to specify the approximate size of a dimension in terms of the number of positions the dimension is expected to contain. Based on the range you select, RPASCE allocates a number of bits within its internal representation of the dimension and all measures that contain that dimension in their intersection. This ability to configure the amount of space necessary to represent the positions of a dimension will result in smaller PDSs. The range options are:

- Very small: between 1 and 100 positions (8 bits)
- Small: between 100 and 800 positions (10 bits)
- Medium: between 800 and 12,000 positions (14 bits)
- Large: between 12,000 and 100,000 positions (17 bits)
- Very Large: between 100,000 and 500,000 positions (20 bits)
- Extremely Large: between 500,000 and 2,000,000 positions (22 bits)
- Ultra Large: between 2,000,000 and 4,000,000 positions (23 bits)
- Custom: selecting this option allows you to enter a specify bit number. An input window opens. Enter the bit number and click **OK**.

Figure 3-25   Input Window for Custom Cardinality Option


Input

Please enter a bit size for dimension.

8

Note:

The cardinality property of a dimension is patchable.

Re-indexing Threshold

Deprecated. This dimension property is no longer used. Use this field to specify the reindexing threshold. As a dimension exhausts its supply of unused indices, RPASCE recycles the deactivated positions and compacts the indices of a dimension. This operation is known as re-indexing. When reindexDomain is run, every dimension is analyzed to determine how many indices are still available to be assigned. If this amount falls below a certain number (the re-indexing threshold), that dimension is reindexed. By default, the threshold is set to 10% of the available positions in the dimension. This means that when more than 90% of a dimension's positions have been allocated, the dimension is reindexed. For more information about reindexing and the reindex utility, see the Hierarchy Management chapter of the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide*.

Enable DPM

Dynamic Position Maintenance (DPM) allows informal positions to be added to a dimension on-the-fly from the RPASCE Client. Select the Enable DPM option for the dimensions that will be enabled to support DPM. After Enable DPM is defined for the dimension, you must also specify workbooks and the dimensions in each workbook that will use DPM (see the Workbook Designer window for more details). A user defined dimension cannot be enabled to support DPM, and vice versa.

Note:

DPM can be enabled for all hierarchy dimensions except for CLND, ADMU, and LNGS.

When Enable DPM is selected for a specified dimension, it is also selected for all dimensions that roll up to it.

For more information on managing DPM positions using the Online Administration Tasks templates, see the *Oracle Retail Predictive Application Server Administration Guide*.

For more information on working with DPM positions in a workbook instance, see *Oracle Retail Predictive Application Server User Guide*.

Enabling DPM is patchable and will not adversely affect the expected results if altered. However, disabling DPM is not patchable. DPM may be unavailable for the templates, but not for the dimension.

DPM positions are application specific. Positions that are created in a specific application are visible *only* to the application. These positions will continue to be exclusive to the application until they are formalized. They are not controlled by the Position Visibility Filter and the application cannot hide these positions by using the PVF. Only after these positions are formalized, will their visibility to the application be determined by the Position Visibility Filter settings.

Enable Images

Select this option to enable the association of images (image paths) to positions along the specified dimension. To disable this feature, clear the option for the appropriate dimensions. This option is available for all hierarchy dimensions, except the calendar hierarchy. For the calendar hierarchy, the Enable Images column is unavailable or grayed. RPASCE supports GIF, BMP, and JPEG image formats. Once Enable Images is defined for a dimension, you must also specify the workbook that will use this feature (see the Workbook Designer window for more details). This is the legacy functionality for displaying images in the application. It is still supported for backward compatibility; however, the new preferred way for displaying images is using a dimension attribute measure of type string with the UI Type property set to media. For more information, see "Images and Contextual Help" in Oracle Retail Predictive Application Server Cloud Edition Administration Guide.

Note:

Enabling and disabling images is patchable and will not adversely affect the expected results if altered.

Attribute Measure

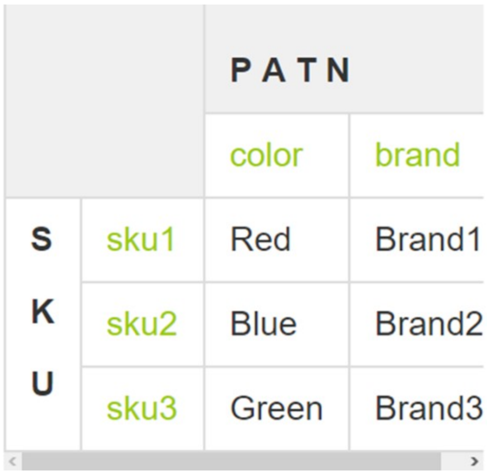
Attribute Measure is used to support dimension attributes and store dimension attribute values for the current dimension. See the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide* and *Oracle Retail Predictive Application Server Cloud Edition User Guide* for more information on Attribute Measure.

Attribute Measure is available for selection if the hierarchy of the current dimension has Attribute Hierarchy defined and qualified measures exist in the application. The candidate measures must meet the following criteria: String type, have storage database in the application, 2-D measure with the base intersection at current dimension and the attribute name dimension. The user can select, at most, one attribute measure from the drop-down list. The selected measure is displayed in the Attribute Measure field.

In the following example, the Prod hierarchy has the attribute hierarchy PATH configured with the attribute value dimension. PATV rolls up to the attribute name dimension PATN. Suppose the PATV dimension has position Blue, Red, Green, Brand1, Brand2, Brand,3 and the PATN dimension has the position color and brand. The PATV positions Blue, Red, and Green roll up to the PATN position color, and

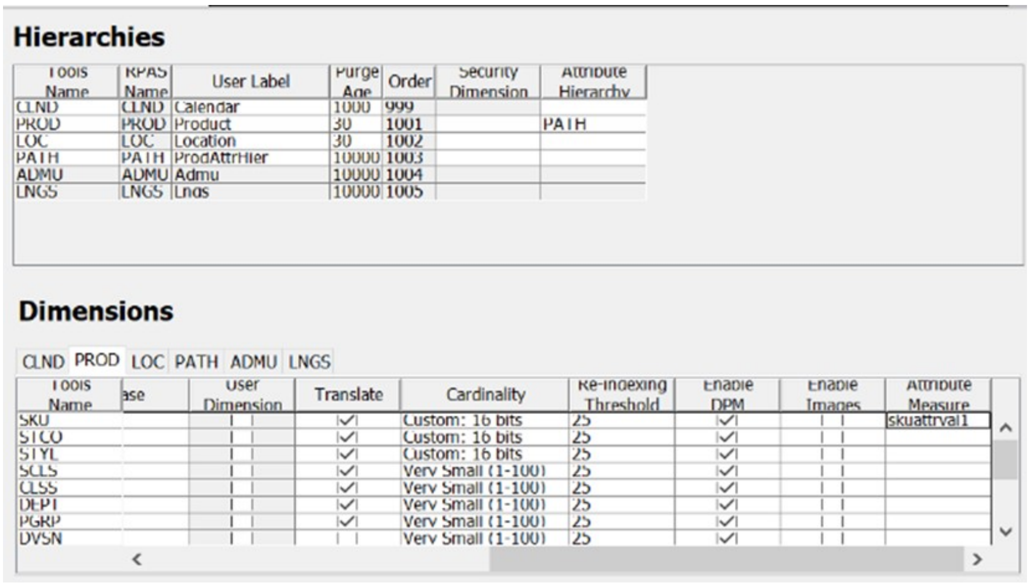
Brand1, Brand2, and Brand3 roll up to the PATN position brand. Then the Sku dimension has the qualified measure skuattrval1 available for selection.

Figure 3-26 The Example Content of the Attribute Measure skuattrval1



		PATN	
		color	brand
SKU	sku1	Red	Brand1
	sku2	Blue	Brand2
	sku3	Green	Brand3

Figure 3-27 The Dimensions Properties Table Showing Attribute Measure Field



Hierarchies

Tools Name	RPAS Name	User Label	Purge Age	Order	Security Dimension	Attribute Hierarchy
CLND	CLND	Calendar	1000	999		
PROD	PROD	Product	30	1001		PATH
LOC	LOC	Location	30	1002		
PATH	PATH	ProdAttrHier	10000	1003		
ADMU	ADMU	Admu	10000	1004		
LNGS	LNGS	Lngs	10000	1005		

Dimensions

CLND PROD LOC PATH ADMU LNGS

Tools Name	ase	User Dimension	Translate	Cardinality	Re-indexing Threshold	Enable DPM	Enable Images	Attribute Measure
SKU		☐	✓	Custom: 16 bits	25	✓	☐	skuattrval1
STCO		☐	✓	Custom: 16 bits	25	✓	☐	
STYL		☐	✓	Custom: 16 bits	25	✓	☐	
SCLS		☐	✓	Very Small (1-100)	25	✓	☐	
CLSS		☐	✓	Very Small (1-100)	25	✓	☐	
DEPT		☐	✓	Very Small (1-100)	25	✓	☐	
PGRP		☐	✓	Very Small (1-100)	25	✓	☐	
DVSN		☐	☐	Very Small (1-100)	25	✓	☐	

The Dimensions Properties Table Showing Attribute Measure Field

> **Note:**
>
> If the attribute measure setting is changed between configurations, the existing PDS must be patched with the new configuration to sync up the RPASCE metadata.

Virtual Dimension

Select a dimension from the drop-down list to enable the association of the virtual dimension with its original dimension. The drop-down list is only enabled if a virtual hierarchy has been specified for this hierarchy in the Hierarchies table. Only then is the drop-down list pre-populated with all the dimensions from the virtual hierarchy. The user can select only one dimension from the drop-down list as the virtual dimension for this dimension. If this field is left blank, then this dimension has no virtual dimension.

The following dimension properties cannot be set for the virtual dimension since the virtual dimension will inherit them from its original dimension: Column, start, width, Label Start, Label Width, Database, Cardinality, Re-indexing Threshold, and Translate.

The following dimension properties cannot be set for the virtual dimension because they do not apply to virtual dimensions: User Dimension and Enable DPM, Virtual Dimension. Virtual dimensions cannot be user-defined dimensions and DPM-enabled.

> **Note:**
>
> The Virtual dimension can be changed and patched in the RPASCE application, but the user must be careful when changing them. The change to the virtual dimension must be consistent with the virtual hierarchy setting.
>
> If a hierarchy is set to virtual, then all its containing dimensions must be set to virtual dimensions and the topology (such as dimension roll up relationship and so on) of the virtual hierarchy must match the original hierarchy.

Delete a Dimension

Complete the following steps to delete a dimension.

Navigate: In the Configuration Components pane, select **Project** and then **Hierarchies**. The Hierarchy Definition window opens in the workspace.

1. From the Dimensions region or from the Hierarchy navigation tree, select the dimension to be deleted.
2. Complete one of the following options:
   - Click the Delete icon.
   - Click **Delete**.
   - Select **Delete** from the right-click menu in navigating in the Hierarchy navigation tree.

> **Note:**
>
> Deleting a dimension causes all of the dimensions that are structurally dependant on it (its parents, grandparents, and so on) to also be deleted.

> **Note:**
>
> The user dimension contained in the ADMU hierarchy cannot be deleted.

Edit a Dimension

Complete the following steps to edit a dimension.

Navigate: In the Configuration Manager, select **Project** and then **Hierarchies**. The Hierarchy Definition window opens in the workspace.

1. Select the Hierarchy in which a dimension is to be edited.
2. From the Dimensions region, select the dimension to edit.
3. Update the dimension property as necessary. Refer to Defining Dimension Properties for more information about updating the dimension property.
4. To change the order of the dimension, click the left or right buttons to change the order of the dimensions. Re-ordering dimensions only affects the file, not the parent-child relationship of data.

   > **Note:**
   >
   > The up and down arrows are used to change the order of hierarchies only.

Create a Branch in a Hierarchy

Complete the following steps to create a branch in a hierarchy.

Navigate: In the Configuration Components pane, select **Project** and then **Hierarchies**. The Hierarchy Definition window opens in the workspace.

1. From the Dimensions region or from the hierarchy navigation tree, select the dimension that will be the base of the branched hierarchy. The base of the branched hierarchy is the root dimension.

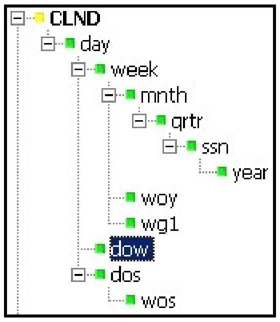
Note:

Ensure that the root dimension will have more than one parent (that is, where the branch starts), and create another parent dimension. Branches can never join together (for example, both style-subclass-class and style-supplier-class roll-ups in the same hierarchy are invalid).

2. Choose one of the following methods:
   - From the Hierarchy Definition right-click menu, select **New Dimension**.
   - Click **New Dimension** on the toolbar.
   - Press **Insert**.

   The new dimension is created underneath the selected dimension.

Figure 3-28 Example of Branch Hierarchy



Labeled Intersections

The Labeled Intersections window supports the addition, removal and modification of hierarchy intersections. A hierarchy intersection defines the dimensionality at which data is defined. An intersection may be defined as using no dimension (scalar), using a single dimension from a hierarchy, or multiple dimensions from different hierarchies.

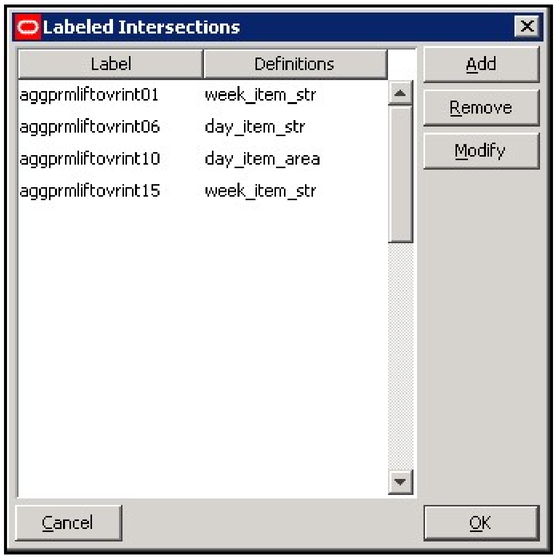
Note:

See the section Measure Definition and Base Intersections for more information on defining intersections for data.

Navigate: In the Configuration Components pane, select **Project** and then **Hierarchies**. The Hierarchy Definition window opens in the workspace.

From the Hierarchy Definition toolbar, select the Labeled Intersection icon. The Labeled Intersections window opens. This window allows you to add new intersections or remove or modify existing intersections.

Figure 3-29 Labeled Intersections Window



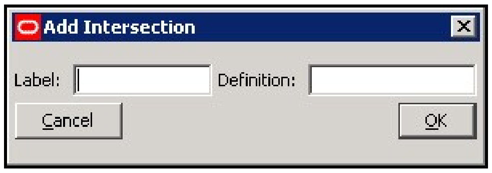
Label	Definitions
aggprmliftovrint01	week_item_str
aggprmliftovrint06	day_item_str
aggprmliftovrint10	day_item_area
aggprmliftovrint15	week_item_str

Adding a Labeled Intersection

Complete the following steps to add a labeled intersection.

1. Click **Add** from the Labeled Intersection window. The Add Intersection window opens.

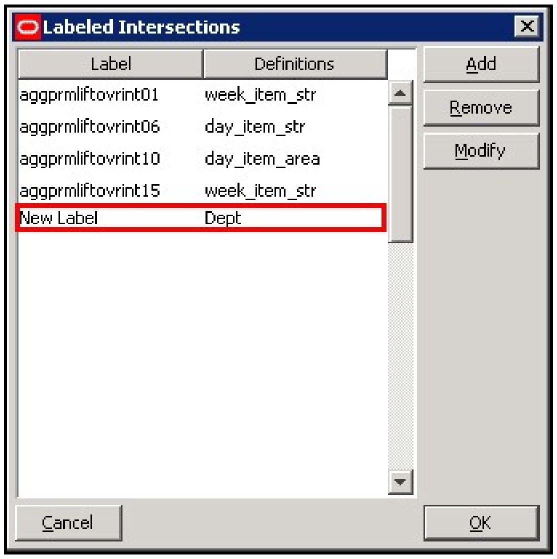
Figure 3-30 Add Intersections Window



2. Complete the following:
   a. In the Label field, enter a label for the Labeled Intersection.
   b. In the Definition field, enter the dimension names for the intersection.
   c. Click **OK**. The window closes, and the new entry displays in the Labeled Intersection window.

> **Note:**
>
> If the Definition field is left empty, the measure is assumed to be Scalar. If the level is non-scalar, dimension names are used to define the intersection. If multiple dimension names are to be specified, each dimension name must be separated by an underscore (_). The last dimension specified should not have an underscore following the dimension name. As well, there is no required order of dimensions.

Figure 3-31 Example of New Labeled Intersections



Label	Definitions
aggprmliftovrint01	week_item_str
aggprmliftovrint06	day_item_str
aggprmliftovrint10	day_item_area
aggprmliftovrint15	week_item_str
New Label	Dept

3. Click **OK**.
4. Once the labeled intersection is added, you can complete the following:
   - Define or update the Base Intersection of major or minor measure component using the labeled intersection.
   - Define or update the Load Intersection of a measure using the labeled intersection.
   - Define or update the Base Intersection of a worksheet using the labeled intersection.

> **Note:**
>
> RPASCE imposes a limit of five dimensions that can be defined in a measure or worksheet's base intersection.

Modifying a Labeled Intersection

Complete the following steps to modify a labeled intersection.

1. Select a labeled intersection.
2. Click **Modify** from the Labeled Intersection menu.
3. Only the Definition field can be modified.

   Click **OK**. If the change is undesired, click **Cancel**.

   When the Definition of an existing labeled intersection is modified, the base intersections of measures and worksheets, and load intersections of measures that are currently assigned the labeled intersection are automatically updated. No action is required.

Removing a Labeled Intersection

Complete the following steps to remove a labeled intersection.

1. Select a labeled intersection.

2. Click **Remove** from the Labeled Intersection window.
3. Click **OK**.

When an existing labeled intersection is removed, the base intersections of measures and worksheets, and load intersections of measures that are currently assigned the labeled intersection will be displayed as invalid (red). Warning messages also display in the Task List, indicating the intersections that must be updated. These intersections must be corrected prior to installing or patching a PDS instance.

Configure 2-Dimensional Dimension Attribute

Users can use the combination of Attribute Hierarchy and Attribute Measure to configure 2-dimensional dimension attributes. The 2-dimensional dimension attribute does not replace the static 1-dimensional dimension attribute using 1-dimensional measures. The dimension attributes defined using Attribute Hierarchy can only handle String type measures. The 2-dimensional dimension attribute is a complement, rather than a replacement, of the static 1-dimensional dimension attribute.

Define Attribute Hierarchy and Attribute Measure

To configure 2-dimensional attributes, first, users must configure the attribute hierarchy and attribute measure. See the previous Attribute Hierarchy and Attribute Measure sections for details. Using the examples there, users must configure PATH as the attribute hierarchy for PROD (Product) hierarchy and measure skuattrval1 as the attribute measure for SKU dimension.

The hierarchy PATH contains the PATN and PATV dimensions with PATV rolling up to PATN. New attributes such as Color and Brand are positions along the PATN dimension, and new attribute values such as red, blue, green, brand1, brand2, and brand3 are positions along the PATV dimension.

Add Dimension Attribute Positions

Second, the new attribute name and attribute value positions must be loaded into the application. Within Attribute Hierarchy, the new dimension attributes are simply the new positions on the attribute name dimension. In other words, the new attribute and all its valid positions and child positions must be provided in the attribute hierarchy load file. Usually, the loadHier call is used to add the attribute.

Enable Dimension Attribute

Third, after the dimension attribute positions are loaded into the PDS, the new attributes for the particular dimension must be enabled. As in the previous example, valid values have been loaded into the application's PATN and PATV dimensions. Now users must enable the Color and Brand attributes for the SKU dimension. This is achieved by running the following commands in the PDS's root directory:

```
dattrmgr -d . -register skuattrval1 -dattname color
```

```
dattrmgr -d . -register skuattrval1 -dattname brand
```

The measure name `skuattrval1` must match the one specified as the `attrmeas` property for the SKU dimension, and the dattname must be the attribute name in the PATN dimension.

Note:

The 2-dimensional dimension attributes do not support dynamic dimension attributes that are dimension attributes defined on a slice of the multi-dimension measure array. A dynamic dimension attribute requires its own infrastructure to store the slicing information and depends on the 1-dimensional dimension attribute infrastructure.

Data Interface Manager

Note:

The following section contains descriptions of features that are not supported in the current version of RPASCE such as Clear Intersection, Start Position, and Column Width properties. However, in order to assist in the process of migrating from a pre-V21 solution to the V21 PDS platform, the configuration components associated with these deprecated features remain present within the RPASCE Configuration Tools and therefore continue to be described within this guide. For additional information, see Appendix – Note on Deprecated Functionality.

The Data Interface Manager tool is used to specify information about how data will be loaded into the Planning Data Schema (PDS) using flat files. This includes properties of the file to be loaded and the intersection at which data is loaded into the PDS.

Data can only be loaded into stored, realized measures in the PDS. Therefore, only such measures can be used in the Data Interface Manager. One such measure may correspond to one fact within the PDS, or multiple measures from multiple applications can use the same fact to store the data. Refer to Shared Measure and Working with Measures section of this guide.

Shared Measure

In the PDS, one shared fact can be used to store data from multiple measures from multiple applications. For example, one measure from RDF and one measure from MFP both use the same fact in the PDS to store data. These measures are called shared measures. Shared measures of the same shared fact do not always have the same measure name and same measure base intersection. The base intersections of shared measures can be different because each application may use them at different levels.

Users configure shared measures inside the Data Interface Manager by specifying the Shared Fact Name and Shared Fact Base Intersection. Measures that do not have the Shared Fact Name configured are non-shared measures, that is, they do not share data with measures from other applications.

In this way, even when the PDS instance is initially generated with only one application, the fact table of shared measures are created at the expected intersection. For example, a PDS instance is initially created with MFPRCS, which has shared measure at SCLS base level. However, they are expected to be shared with measures defined at SKU base level in other applications. Users can explicitly set the shared fact intersection at SKU for shared measures, and the fact table will be created at SKU correctly. When a new application such as ASCS or RDF that uses the shared measure at SKU is added later, the fact table is valid and no data loss will occur.

Assumptions for Shared Measures

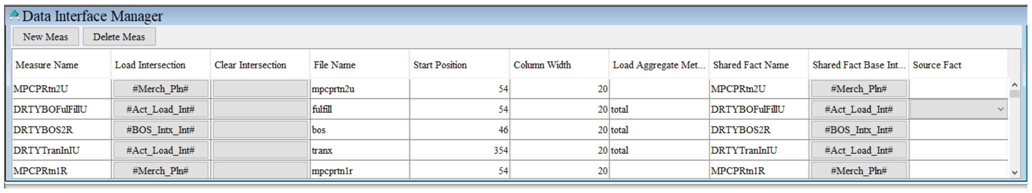
1. Shared Measures must have the common fact name provided by the configurator.
2. Shared Measures can have at most two different base intersections. Their base intersections must be at or higher than the fact intersection, but can only be higher at one base dimension. For example, if the shared measure's base intersection is week_scls_dstr while the fact intersection is week_sku_stor, this is an invalid configuration.
3. Shared measures must have the same measure navalue, defagg, defspread, and purgeage properties.
4. For the same shared fact, one application cannot have more than one measure mapped to it.
5. If no shared fact intersection is specified, then the shared measures must have the same base intersection. Their base intersection is used as the fact intersection.
6. For shared measures, labels are application-specific.
7. Shared measures must be loaded.

Specify the Data Interface for a Measure

Complete the following steps to specify the data interface for a measure.

Navigate: In the Configuration Components pane, select **Project** and then **Data Interface**. The Data Interface Manager window opens in the workspace.

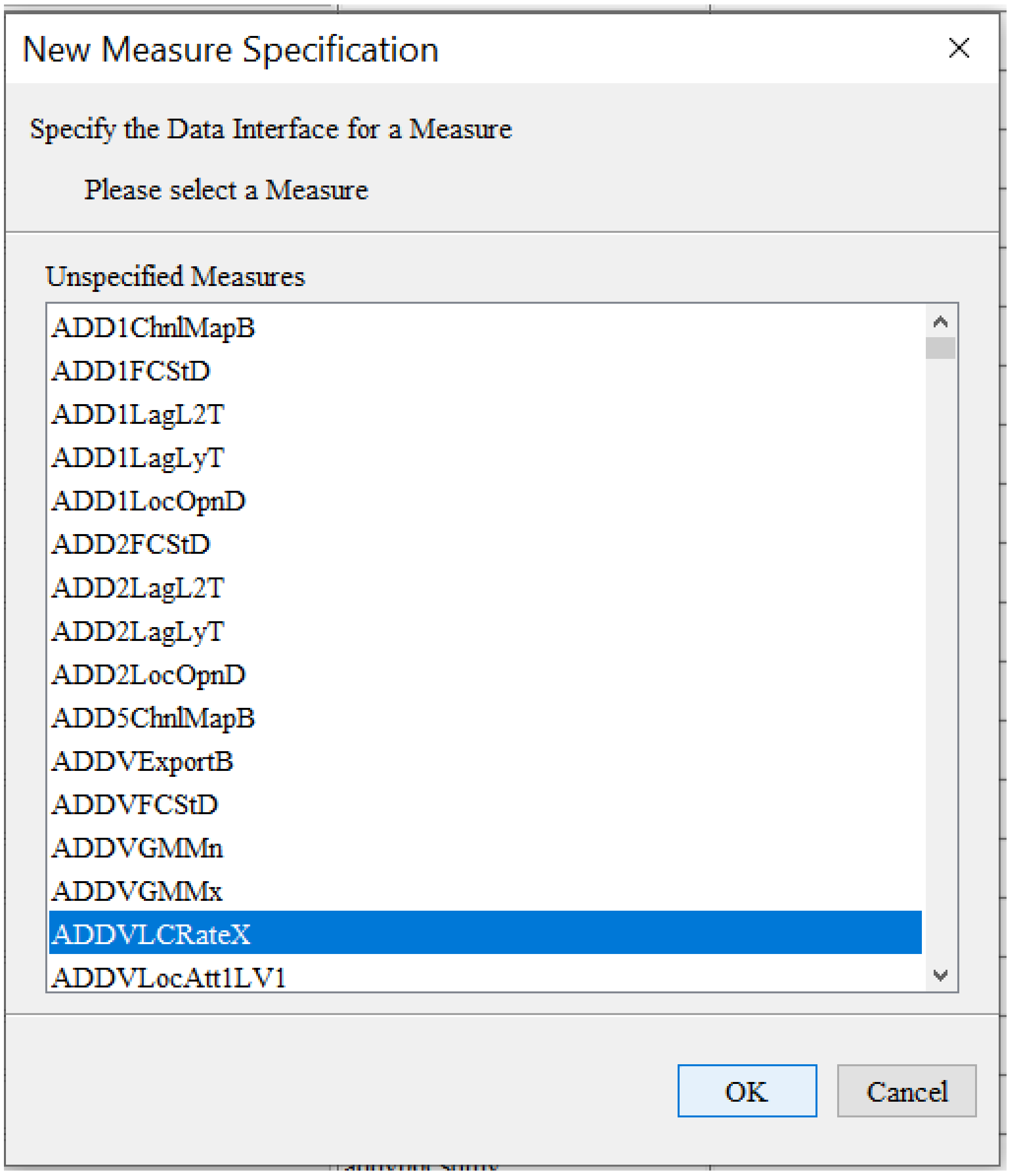
Figure 3-32 Example of Data Interface Manager Window



Measure Name	Load Intersection	Clear Intersection	File Name	Start Position	Column Width	Load Aggregate Met...	Shared Fact Name	Shared Fact Base Int...	Source Fact
MPCPRtn2U	#Merch_Pln#		mpcprtn2u	54	20		MPCPRtn2U	#Merch_Pln#	
DRTYBOFulFillU	#Act_Load_Int#		fulfill	54	20	total	DRTYBOFulFillU	#Act_Load_Int#	
DRTYBOS2R	#BOS_Intx_Int#		bos	46	20	total	DRTYBOS2R	#BOS_Intx_Int#	
DRTYTranInIU	#Act_Load_Int#		tranx	354	20	total	DRTYTranInIU	#Act_Load_Int#	
MPCPRtn1R	#Merch_Pln#		mpcprtn1r	54	20		MPCPRtn1R	#Merch_Pln#	

1. Click **New Meas**. The New Measure Specification window opens.

Figure 3-33 Example of New Measure Specification Window


New Measure Specification

Specify the Data Interface for a Measure

Please select a Measure

Unspecified Measures

ADD1ChnlMapB
ADD1FCStD
ADD1LagL2T
ADD1LagLyT
ADD1LocOpnD
ADD2FCStD
ADD2LagL2T
ADD2LagLyT
ADD2LocOpnD
ADD5ChnlMapB
ADDVExportB
ADDVFCStD
ADDVGMMn
ADDVGMMx
ADDVLCRateX
ADDVLocAtt1LV1

Note:

This is a filtered list of all possible measures in the configuration. Measures will be displayed in this list if they are realized, stored (has a defined database), and not already defined in the data interface tool.

2. Select the measure requiring a data interface definition.
3. Click **OK**. The measure displays in the Data Interface Manager window.

Add/Edit Data Interface Properties for a Measure

Complete the following steps to add and or edit data interface properties for a measure.

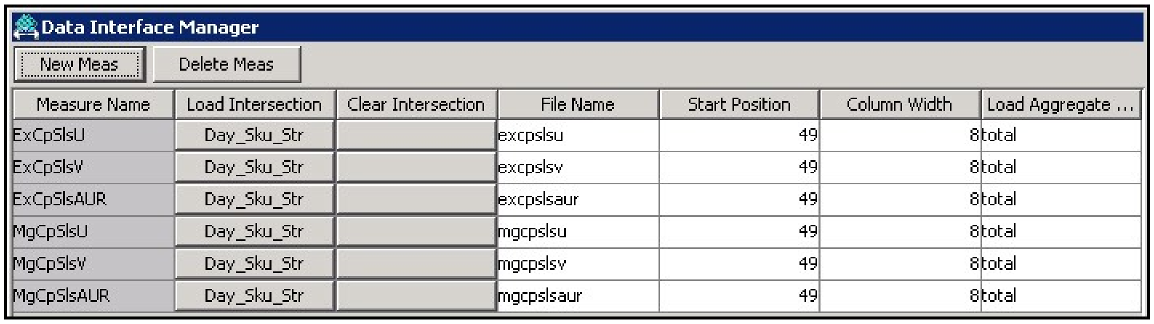
Navigate: In the Configuration Components pane, select **Project** and then **Data Interface**. The Data Interface Manager window opens in the workspace.

By default, the Load Intersection field is populated with the base intersection of the measure. If the data for a given measure is being loaded at a lower intersection than the base intersection of the measure, this value can be overridden to specify the intersection.

Note:

Data can be loaded at the same intersection or lower than the base intersection of a measure. Data cannot be loaded at a higher intersection than the base intersection.

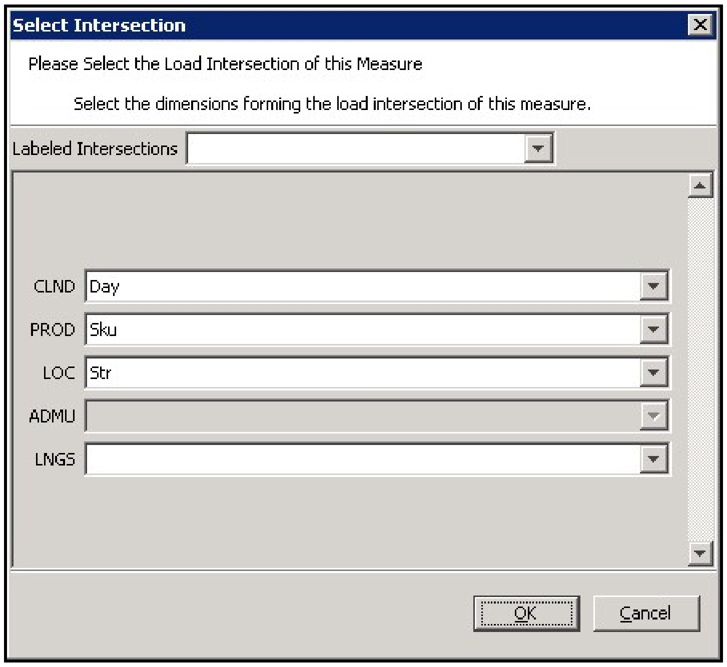
Figure 3-34 Data Interface Manager Window – Add/Edit



Measure Name	Load Intersection	Clear Intersection	File Name	Start Position	Column Width	Load Aggregate ...
ExCpSlsU	Day_Sku_Str		excpslsu	49	8	total
ExCpSlsV	Day_Sku_Str		excpslsv	49	8	total
ExCpSlsAUR	Day_Sku_Str		excpslsaur	49	8	total
MgCpSlsU	Day_Sku_Str		mgcpslsu	49	8	total
MgCpSlsV	Day_Sku_Str		mgcpslsv	49	8	total
MgCpSlsAUR	Day_Sku_Str		mgcpslsaur	49	8	total

1. Click the **Load Intersection** field to change its value. The Select Intersection window opens.

Figure 3-35 Select Intersection Window


Select Intersection

Please Select the Load Intersection of this Measure

Select the dimensions forming the load intersection of this measure.

Labeled Intersections

CLND	Day
PROD	Sku
LOC	Str
ADMU	
LNGS	

2. To specify the load intersection:
   a. Using the list options, select the appropriate dimensions or Labeled Intersection.

      Note:

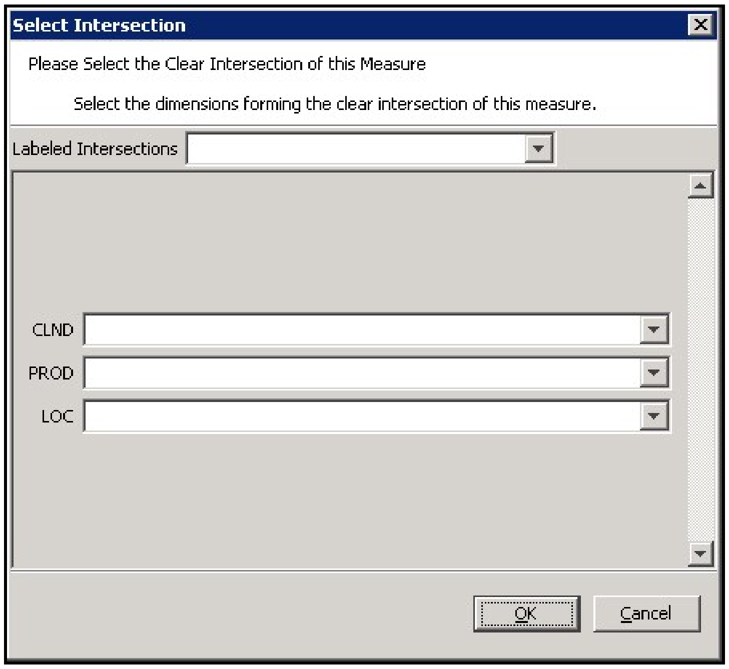
      Only those dimensions that are at the same level as the base intersection or lower is displayed for the load intersection.

   b. Click **OK** to save any changes and close the window.
3. Clear Intersection field is deprecated. Click the **Clear Intersection** field to change its value. The Select Intersection window opens.

   Note:

   The clear intersection allows support for the `.clr` measure data files. Using the `.clr` files with loadMeasure allows the clearing of selected portions of a measure's data arrays.

Figure 3-36 Select Intersection Window


Select Intersection
Please Select the Clear Intersection of this Measure
Select the dimensions forming the clear intersection of this measure.
Labeled Intersections
CLND
PROD
LOC
OK
Cancel

4. To specify the clear intersection:
   a. Using the list options, select the appropriate dimensions or Labeled Intersection.

      Note:

      Only those dimensions that are at the same level as the base intersection or higher are displayed for the clear intersection.

   b. Click **OK** to save any changes. Close the window.
5. In the File Name field, enter the file name from which data for the measure will be loaded. The File Name property is used in fact grouping algorithms. This is covered in detail in the Integration Tools chapter.
6. Start Position Field is deprecated. In the Start Position field, enter the character position in the file where the measure data starts. The start position defaults to the sum of all the dimension widths in the load intersection of the measure +1, and the value specified in the field must be that default or higher.
7. Column Width Field is deprecated. In the Column Width field, enter the number of characters in the file that will contain the measure data.

   Note:

   The Column Width defaults to 8, but it can be changed.

8. If the Load Intersection was overridden to specify that the data is to be loaded from an intersection lower than the measure's base intersection, the measure's

default aggregation method is used to aggregate the data unless the Load Aggregate field is populated to specify an alternate aggregation method. Click the **Load Aggregation Method** field and select the appropriate aggregation method from the option list.

> **Note:**
>
> Hybrid is not supported for the load aggregation for a measure.

9. If the measure is a shared measure, enter the shared fact name into the Shared Fact Name field and the shared fact base intersection into the Shared Fact Base Intersection field. The explicit specification of the shared fact name is mandatory for shared measures. Measures that do not have the shared fact name configured are considered non-shared measures.

   The shared fact name must not exceed 30 characters in length and must not contain spaces. It is recommended to follow measure naming conventions when configuring the Shared Fact field.

10. The specification of the shared fact intersection is optional; if not set, the measure's base intersection will be used as the fact intersection during integration configuration generation. Refer to the Integration Tools chapter.

11. The Source Fact property is used to indicate which measure, that is, the corresponding fact, can be updated via the ORDS Put (write) service. Only the measures that have the Source Fact set to TRUE can have their corresponding facts updated by the ORDS Put web service. For any new measures added into the Data Interface Manager, the Source Fact field defaults to True since measures inside the Data Interface Manager are load measures. Users can explicitly make this measure invisible by clicking this field and selecting False from the drop-down list. More details on Planning Data Schema Default Web Services (through ORDS) can be found in the *Oracle Retail Predictive Application Server Cloud Service Administration Guide*.

Delete Data Interface Information for a Measure

Complete the following steps to delete data interface information for a measure.

Navigate: In the Configuration Components pane, select **Project** and then **Data Interface**. The Data Interface Manager window opens in the workspace.

1. Select the measure you want to remove from the Data Interface Manager.
2. Click **Delete Meas**.
3. Click **Yes**. The measure is removed from the table.

Working with Styles

Note:

The following section contains descriptions of features that are not supported in the current version of RPASCE such as some style properties defined with the Style Manager. However, in order to assist in the process of migration of a pre-CE solution to the CE platform, the configuration components associated with these deprecated features remain present within the RPASCE Configuration Tools and therefore continue to be described within this guide. For additional information, see Appendix – Calculation Engine User Guide.

It is possible for the RPASCE Client user to modify the appearance of the data displayed for a given measure in a grid. Text font, size, and color may all be changed. Many attributes, such as precision (for decimal data types), alignment of the value in the cell, and the cell border may also change.

Using the Style Tool, it is possible to define styles that may be applied to measures. These predefined styles may specify any of a body of attributes that determine the appearance of the data within the client. It is then possible to specify a measure as using one of these pre-defined styles. The measure will then be displayed according to the specifications for that style.

Note:

The RPASCE Client is not aware of styles which are a configuration convenience. In the client, the individual properties are maintained individually. A style can therefore be thought of as a mechanism to easily set many individual properties.

The Style Definition Tool

The Style Definition Tool provides the following functionality:

- Allows the creation and management of named styles. New styles are generated as sub-styles of existing styles.
- Allows the specification of the attributes of named styles. Style attributes follow an inheritance scheme in which any unspecified attribute will inherit a value from its parent style if that style has a specification.
- Allows the specified styles to be visible to the Measure and Workbook Tools where measures are marked as using a style.

Style Attributes

A number of attributes may be specified for a style. These attributes will determine how the data for a measure that uses the style is displayed within the RPASCE Client. Style attributes follow an inheritance framework in which an attribute defined in one style is also defined for all of the children of that style unless a style is defined for a child. The attributes of a style that may be specified are as follows:

Name

The name of the style. This is used in the Measure and Workbook Tools to assign a style to a measure. Since styles are a configuration convenience, style names are not visible in the RPASCE Client.

Prefix

A cell value in the RPASCE Client will be prefixed with this string. For example, a prefix could be “$" to denote U.S. currency values. The prefix can be any character sequence but cannot exceed seven characters.

Suffix

A cell value in the RPASCE Client will be suffixed with this string. For example, a suffix could be “%" to denote that the value in the field is a percentage of something. The suffix can be any character sequence but cannot exceed seven characters.

Scale Factor

A cell value in the RPASCE Client could use a scale factor for display purposes. A value that is calculated as a fraction could be displayed as a percent by selecting the scale factor to be 0.01 (The UI divides by the scale factor).

For example, if the value in a cell is 0.5, the scale factor would have to be 0.01 for the cell to display 50.

The value entered in the field should be greater than zero.

Precision

Precision is the number of significant digits to be displayed in the cell of the RPASCE Client. If this number is set to 3, the client must always display 3 positions after the decimal. For example, the measure value is 1, with a Precision setting of 3; it will be displayed as 1.000. The value entered in the field should be greater than zero.

Separator

A cell value in the RPASCE Client could be formatted to have separators in the value. The separator and the format come from the regional settings on the computer. For example, when a separator is used, a value of 1000 would be displayed as 1,000 or 1.000. It can also be displayed in other formats depending on the regional settings.

Text Font

Deprecated.

Note:

Text font is not used by the RPASCE Client.

Text Style

Sets the display style of the text value in the RPASCE Client (Bold, Italic, and so on).

Text Size

Deprecated.

Note:

Text size is not used by the RPASCE Client.

Text Color

Sets the color of the cell values in the RPASCE Client.

Background

Sets the background color of the cells in the RPASCE Client.

Note:

In the RPASCE Client, the measure formatting background color for a measure takes priority over the ‘read/write' background color that can be set for the application. Therefore, the RPASCE Client ‘read/write' color will not be seen if styles are used through the Configuration Tools. If a specific read/write color is desired for all measures, set it as the background color of the default style, and do not override it for any other styles. On the other hand, the RPASCE Client ‘read only' background color takes priority over the measure formatting background color so that ‘protection processing' will be visible.

Alignment

Sets the alignment of values within the cells when viewed in the RPASCE Client (Left, Center, and Right

Border Style

Sets they style of border of cells. Border style determines the kind of borders (for example, single line, dotted line, and so on) and where the borders should be relative to the cell value (top, bottom, left, right, or any combination of these).

Note:

Border style is not used by the RPASCE Client.

Border Color

Sets the color of the border lines for cell values.

Note:

Border color is not used by the RPASCE Client.

Time Format

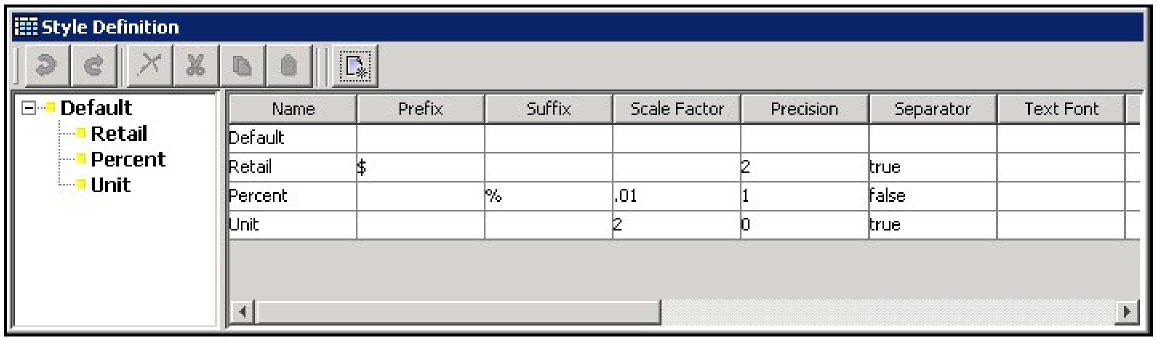
Sets the time format for styles. Options are No Time, Twenty-Four Hour, and Twelve Hour.

Create a Style

Complete the following steps to create a style.

Navigate: In the Configuration Components pane, select **Project** and then **Styles**. The Style Definition window opens in the workspace.

Figure 3-37 Style Definition Window



Name	Prefix	Suffix	Scale Factor	Precision	Separator	Text Font
Default						
Retail	$			2	true	
Percent		%	.01	1	false	
Unit			2	0	true	

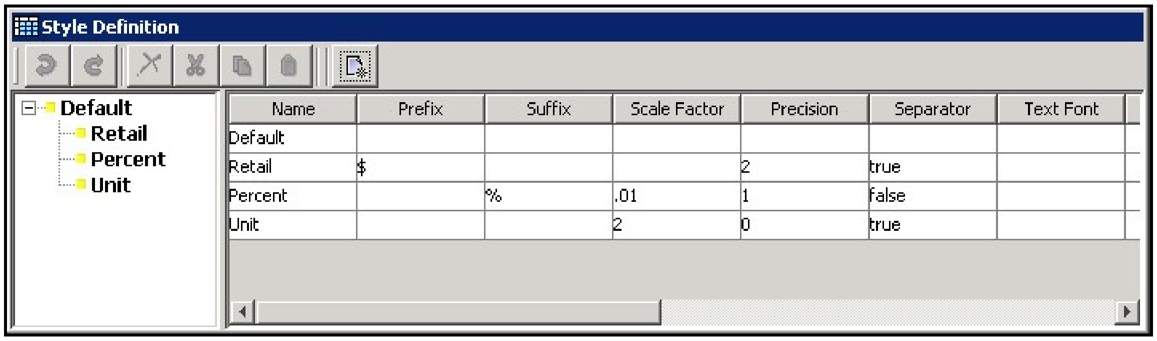
1. Select or create the Style that will be the parent of the new style. All styles must ultimately be descendents of the Default Style.
2. Choose one of the following methods:
   a. From the toolbar, click **Create a new style**.
   b. Select **New Style** from the right-click menu.
   c. Press **Insert**.

   If the Style Attributes for Default are populated, all of its descendants will inherit the same attributes unless you specify new attributes for the new styles. A new style will be created with inherited attribute values for all the properties set in its parent style.

The inherited style attribute values are displayed with lighter shade (gray) to differentiate from the un-inherited values (black). Notice that the style attributes for the style Default -are shown in black while the style attributes for the style Percent are in gray.

3. Change the values of any of the new style's attributes where a different value is required than that which has been inherited. For those attributes that have been overwritten from the “Default" value will be display in black while those that have not been changed will remain gray to indicate they are inherited.

Figure 3-38 Style Definition Window



Name	Prefix	Suffix	Scale Factor	Precision	Separator	Text Font
Default						
Retail	$			2	true	
Percent		%	.01	1	false	
Unit			2	0	true	

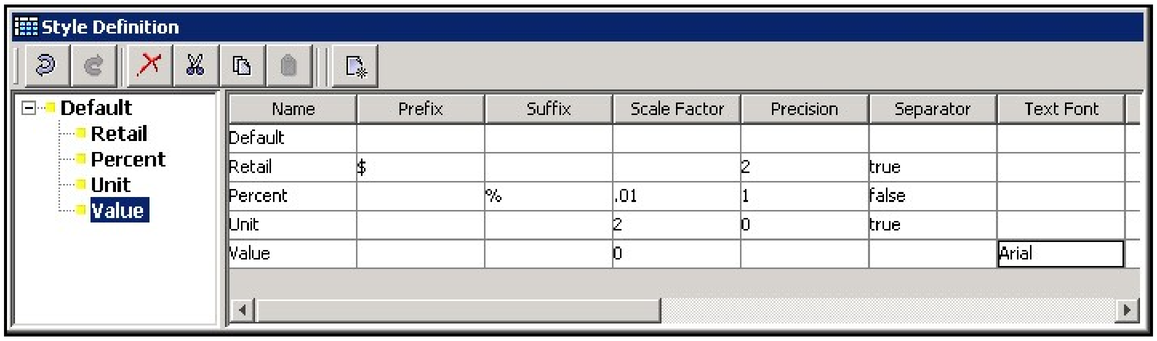
Remove a Style

Complete the following steps to remove a style.

Navigate: In the Configuration Components pane, select **Project** and then **Styles**. The Style Definition window opens in the workspace.

1. Select the style to be removed.

Figure 3-39 Style Definition Window

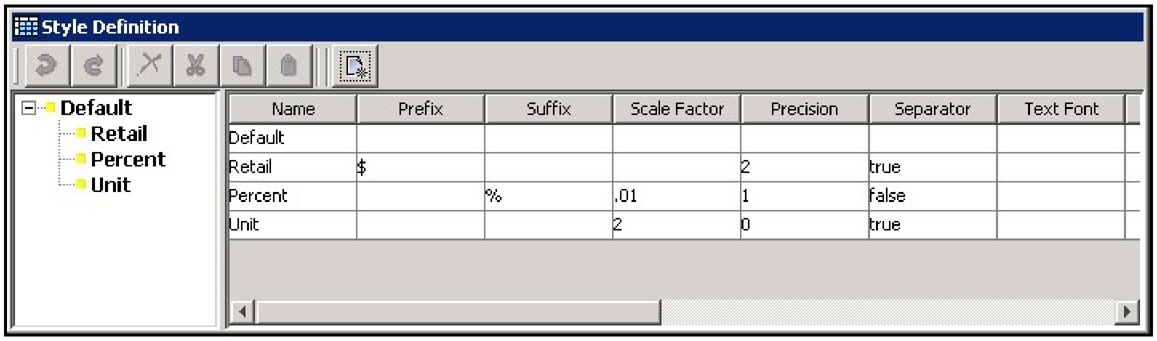


Name	Prefix	Suffix	Scale Factor	Precision	Separator	Text Font
Default						
Retail	$			2	true	
Percent		%	.01	1	false	
Unit			2	0	true	
Value			0			Arial

2. Choose one of the following methods:
   - From the toolbar, click **Delete Style**
   - Select **Remove** from the right-click menu.
   - Press **Delete**.

   The selected style and all of its child styles will be removed from the style description tool. Any measures using a deleted style will be displayed as invalid.

Figure 3-40 Style Definition Window



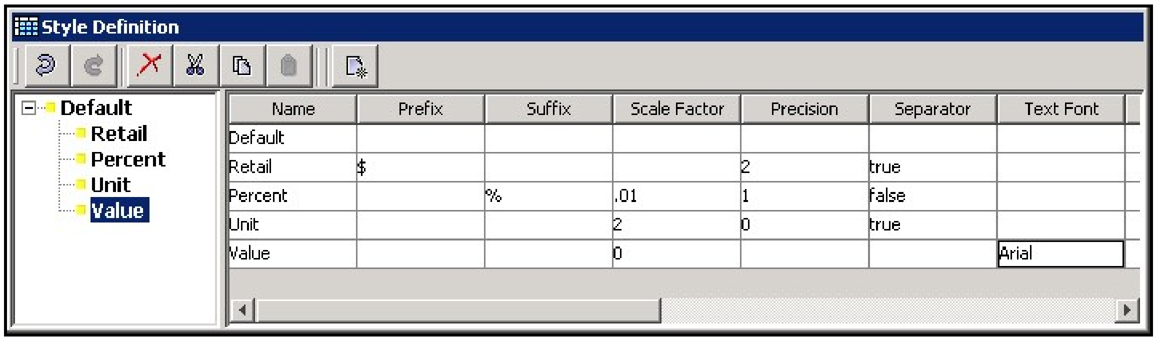
Name	Prefix	Suffix	Scale Factor	Precision	Separator	Text Font
Default						
Retail	$			2	true	
Percent		%	.01	1	false	
Unit			2	0	true	

Edit a Style

Complete the following steps to edit a style.

Navigate: In the Configuration Components pane, select **Project** and then **Styles**. The Style Definition window opens in the workspace.

Figure 3-41 Style Definition Window



Name	Prefix	Suffix	Scale Factor	Precision	Separator	Text Font
Default						
Retail	$			2	true	
Percent		%	.01	1	false	
Unit			2	0	true	
Value			0			Arial

1. Select the style to be edited.
2. Select the property of the style to be edited. Depending on the property selected, one of the following is displayed:
   - a color chooser (for all color selection properties like font color, background color, and so on)
   - a window (for Borders)
   - a list (for alignment, font, and text style)
   - a free flow text cursor (for all other properties)
3. Make the selection and enter the value for the property.

   If the edited value is changed to the same as its parent's value for an attribute, the value is automatically changed to inherit from the parent, and it is displayed as gray rather than black.
4. If the value is to be deleted (does not contain any value), select the property, and press **Delete**.

Working with Taskflows

Note:

The following section contains descriptions of features that are not supported in the current version of RPASCE. However, in order to assist in the process of migration of a pre-CE solution to the CE platform, the configuration components associated with these deprecated features remain present within the RPASCE Configuration Tools and therefore continue to be described within this guide. For additional information, see the Deprecated Features in the RPASCE Platform

The RPASCE Client provides a more flexible approach to user interaction with the workbooks configured for an RPASCE application. This flexibility allows users to focus less on the structural elements that make up the workbook configuration and more on the tasks that they use those structural elements to perform.

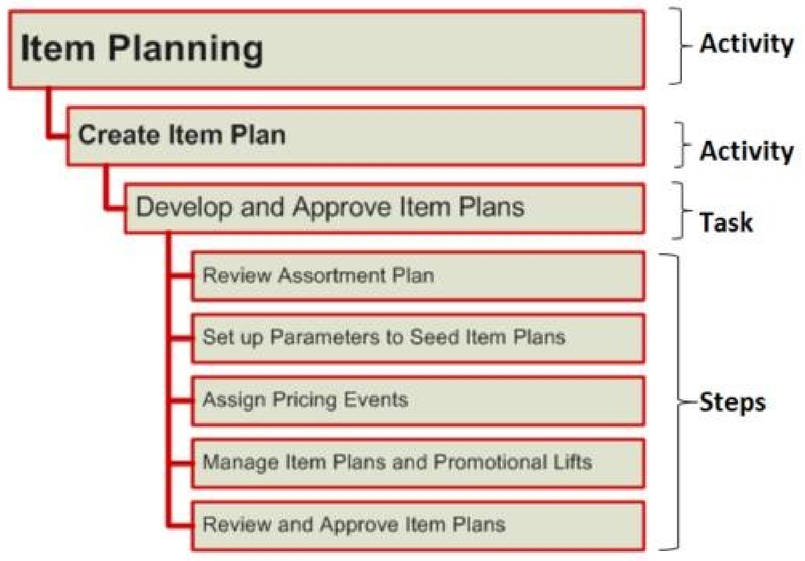
This flexibility is found in the RPASCE Client taskflow. The taskflow allows configurators to more closely describe and model their business practices within the client. Configurators can create taskflow elements (activity groups, activities, tasks, and steps) that can be associated with structural elements of the workbook configuration. These taskflow elements then provide a more intuitive and business practice-oriented view of the structural elements of the RPASCE application.

The activity group activity/task/step can be organized however the user wants. The task refers to a template and specific solution. Inside a Configuration Tools configuration, they all refer to the same solution. Names are all qualified by the Solution ID, so that they do not conflict later when they are manually combined. A single solution taskflow can have multiple activity groups, and a combined taskflow could conceivably have just one (for example, put all the activities from the various solutions under a single activity group).

When creating an activity, the activity group name, label, and description can be specified. When the sandbox is built, the activities are combined into an activity group based on the specified activity group properties. Each task is specific to a workbook template and therefore a specific solution.

Figure 3-42 shows the activity, task and steps for Item Planning.

Figure 3-42 Activity, Task and Steps for Item Planning


Item Planning
Create Item Plan
Develop and Approve Item Plans
Review Assortment Plan
Set up Parameters to Seed Item Plans
Assign Pricing Events
Manage Item Plans and Promotional Lifts
Review and Approve Item Plans
Activity
Activity
Task
Steps

Using the RPASCE Configuration Tool, you have the ability to create a customized activity taskflow for the RPASCE Client. This activity taskflow helps users of the RPASCE Client understand the tasks they must complete in order to meet their planning goals.

The RPASCE Configuration Tool works on a taskflow for a single solution (configuration). Everything within it will be qualified by Solution ID so as not to create conflicts with other solution's taskflows. For a multi-solution configuration, these need to be combined into a single multi-solution taskflow. This is a manual process of combining the taskflow XML and resource properties files.

Some RPASCE configured solutions are delivered with preconfigured taskflows. These preconfigured taskflows can be customized using the procedures in the following sections. Or, you can Create a Taskflow.

Here is an example of a taskflow in the RPASCE Client.

Figure 3-43 Taskflow within the RPASCE Client Task Module

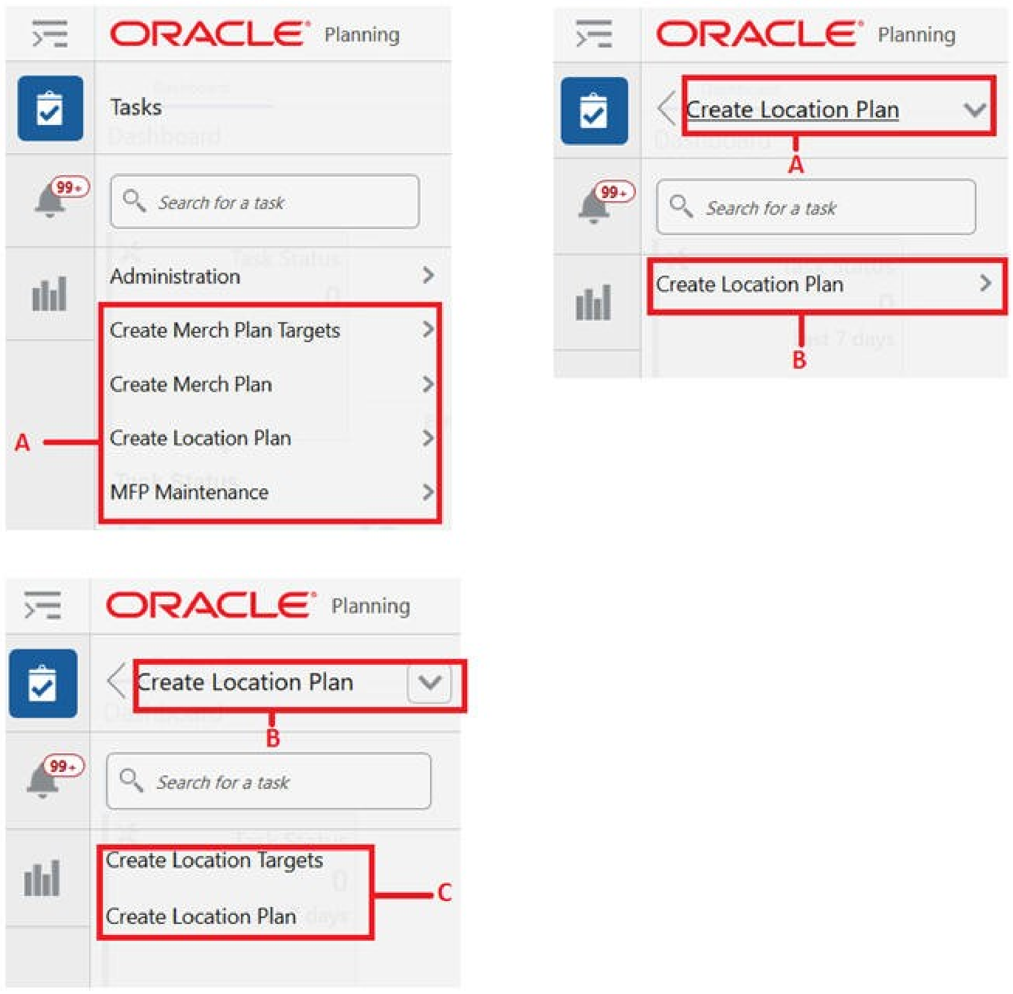


Figure 3-44 Taskflow Within the RPASCE Client Task Module

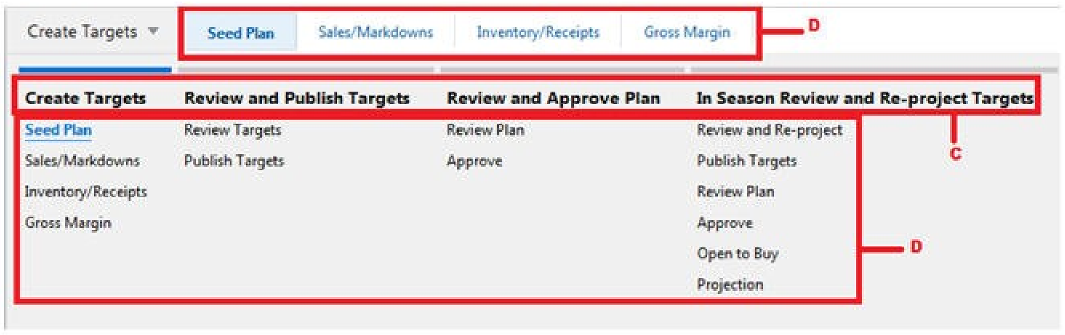


Table 3-2 describes the icons that appear with all the entries in the activity taskflow.

Table 3-2 Taskflow Icon Key for the Taskflow within the RPASCE Client Task Module

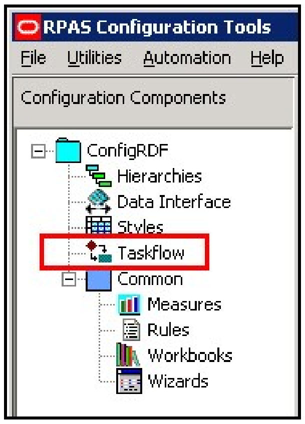
Legend	Icon Name	Description
A	Activities	These tabs represent the predefined activities of the application.
B	Tasks	These are the individual tasks within the selected activity.
C	Steps	These are the steps available in the selected task.
D	View Groups (Tabs)	These are the view groups defined within the selected step. If a step contains configured tabs, each tab is represented by a view group. If a step does not contain configured tabs, a single view group corresponding to the step itself is present.

Create a Taskflow

Complete the following steps to create a taskflow.

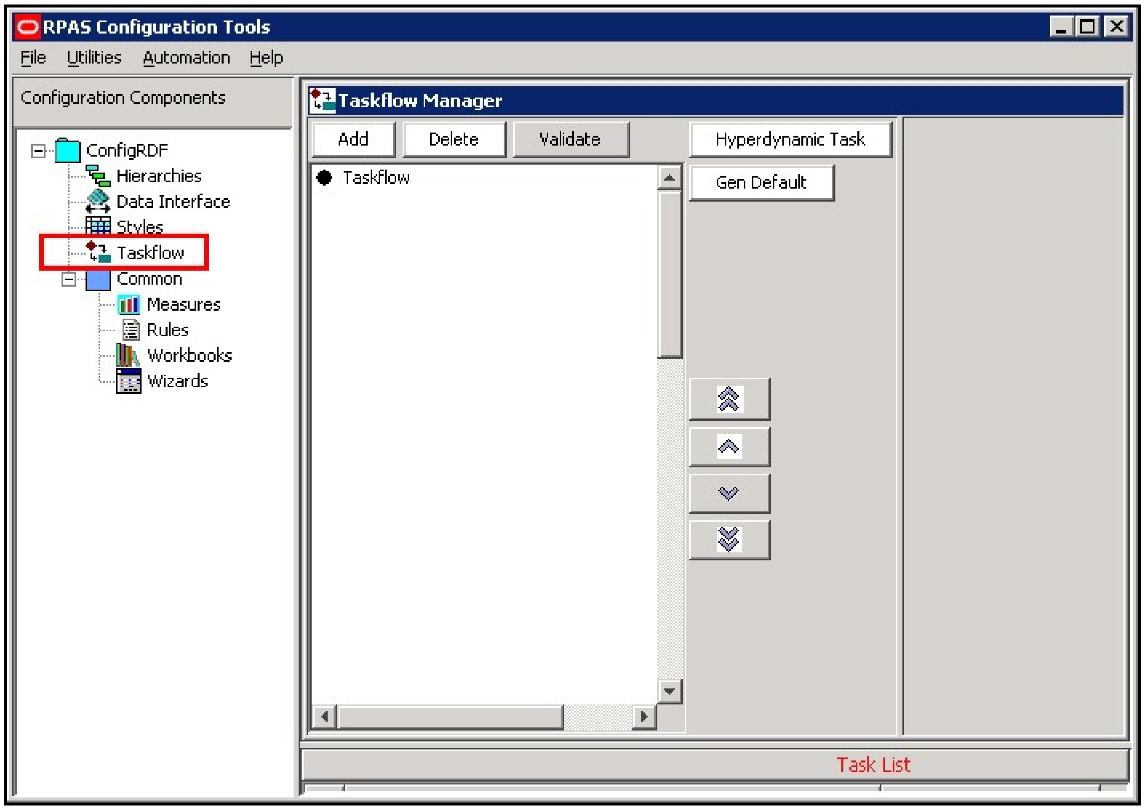
Navigate: In the Configuration Components pane, select **Taskflow** within the project for which you want to create a taskflow.

Figure 3-45 Taskflow Icon in the Configuration Components Pane



The Taskflow Manager window opens. The first time that you use the Taskflow Manager, it contains one bullet called Taskflow.

Figure 3-46 Taskflow Manager

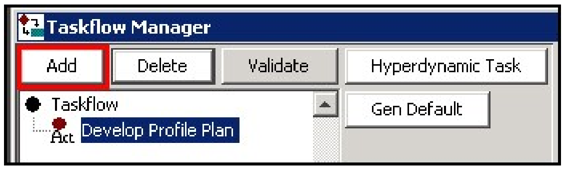


Adding an Activity to the Taskflow

Complete the following steps to add an activity to the taskflow.

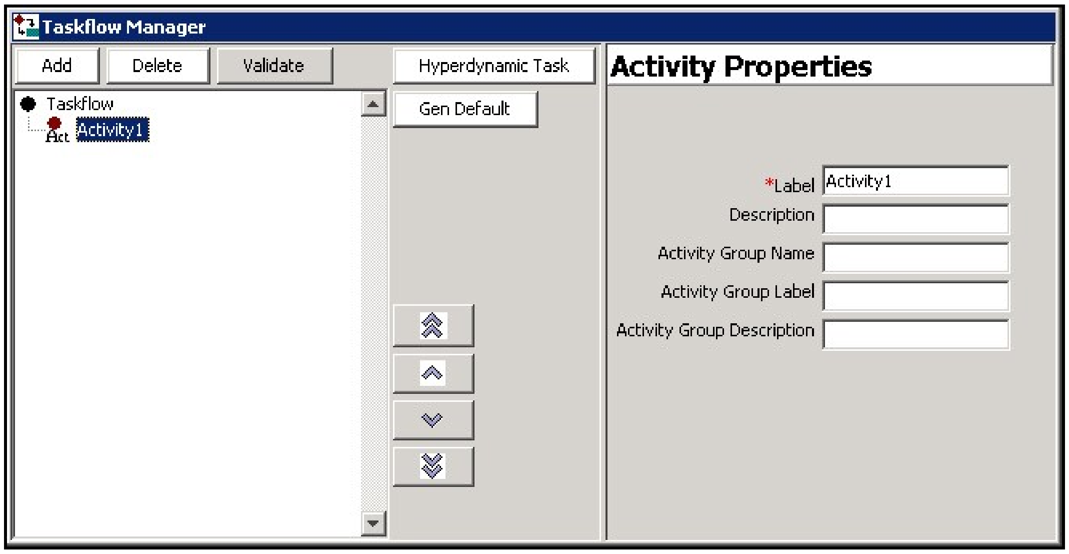
1. Select the Taskflow bullet inside the navigation pane of the Taskflow Manager. Click **Add**.

Figure 3-47 Adding an Activity to the Taskflow



2. An activity called Activity1 displays in the navigation pane. Select Activity1. The Activity Properties displays in the detail pane.

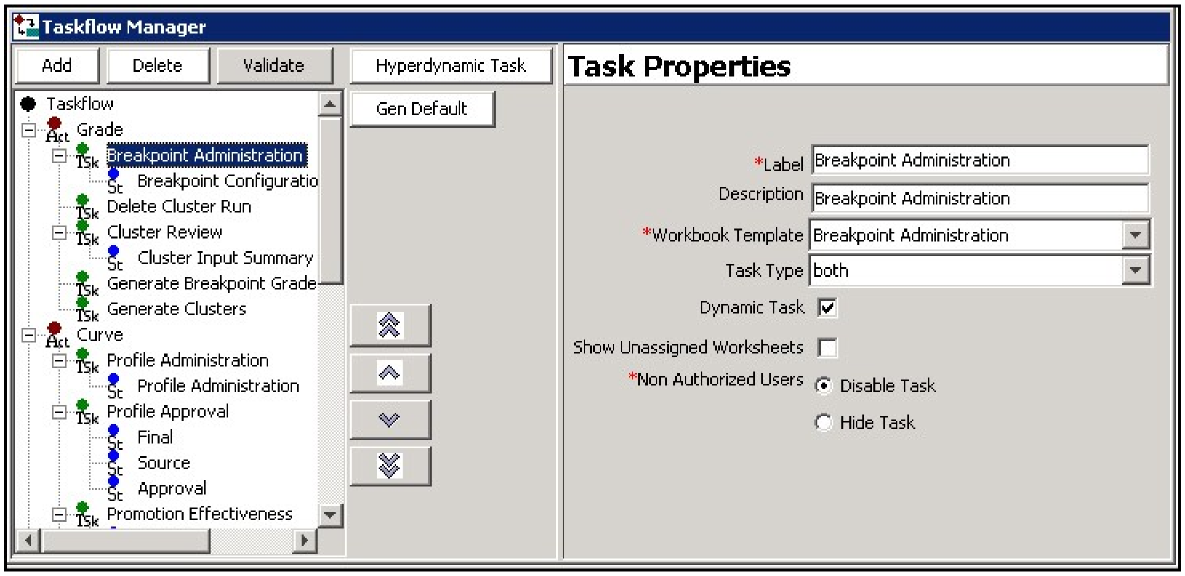
Figure 3-48   Activity1 in the Taskflow



*Label: Activity1
Description:
Activity Group Name:
Activity Group Label:
Activity Group Description:

3. Enter the name of the activity in the **Label** field. This is the name of the task as it displays in the taskflow of the RPASCE. The red asterisk denotes that this step is required. Note that as you type the name in the Label field, the name is updated in the navigation pane.

Figure 3-49   Activity Properties



*Label: Breakpoint Administration
Description: Breakpoint Administration
*Workbook Template: Breakpoint Administration
Task Type: both
Dynamic Task: ✓
Show Unassigned Worksheets:
*Non Authorized Users: [✓ Disable Task] Hide Task

4. [Optional] Enter a description for the activity in the **Description** field. The description displays when the user rolls the cursor over the activity name in the RPASCE Client.

5. Enter the activity group name in the **Activity Group Name** field, group label in the **Activity Group Label** field, and group description in the **Activity Group Description** field. This step is optional. Note that the Activity Group Name must not contain any spaces. Otherwise, when the user clicks **Validate**, error messages will show up in the pop-up validation dialog box.

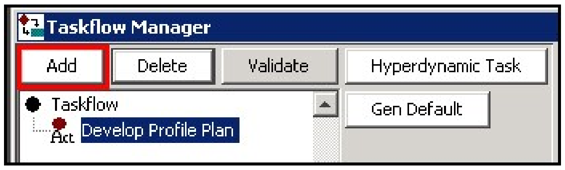
   After you have entered the activity properties, the new activity is created.

Adding a Task to the Taskflow

Complete the following steps to add a task to the taskflow.

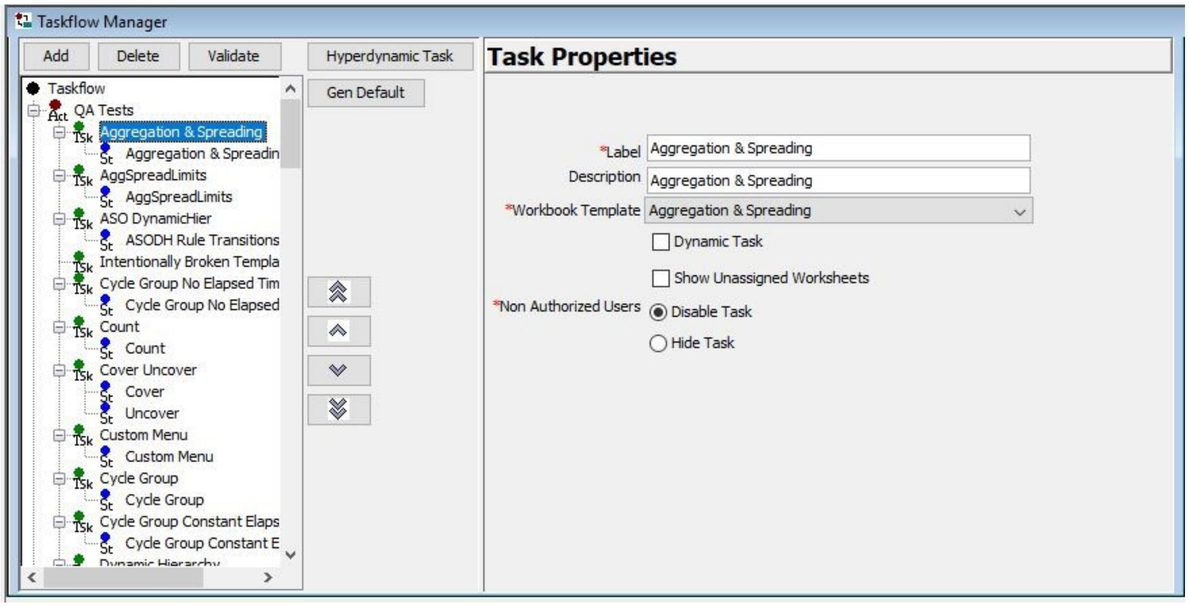
1. To create a task within an activity you created, select the activity in the navigation pane and click **Add**.

Figure 3-50 Adding a Task to an Activity



2. A task called Task1 displays in the navigation pane. Select Task1. The Task Properties display in the details pane.

Figure 3-51 Task Properties



*Label: Aggregation & Spreading
Description: Aggregation & Spreading
*Workbook Template: Aggregation & Spreading
☐ Dynamic Task
☐ Show Unassigned Worksheets
*Non Authorized Users: [✓ Disable Task] Hide Task

3. Enter the following information in the Task Properties.

Label

Enter the name of the task as you want it to appear in the RPASCE Client. This step is required.

Description

Enter the description of the task. This description displays when the user rolls the cursor over the task name in the RPASCE Client.

Workbook Template

Choose the workbook template that will be utilized in this task. This step is required.

Dynamic Task

Select this option if the steps for the task are dynamic based on the user selections made during the workbook wizard. When this option is selected, the user will not see any steps under a task in the RPASCE Client until a workbook is open.

> **Note:**
>
> Custom workbook code is still required to provide the ability to filter steps based on the wizard selections

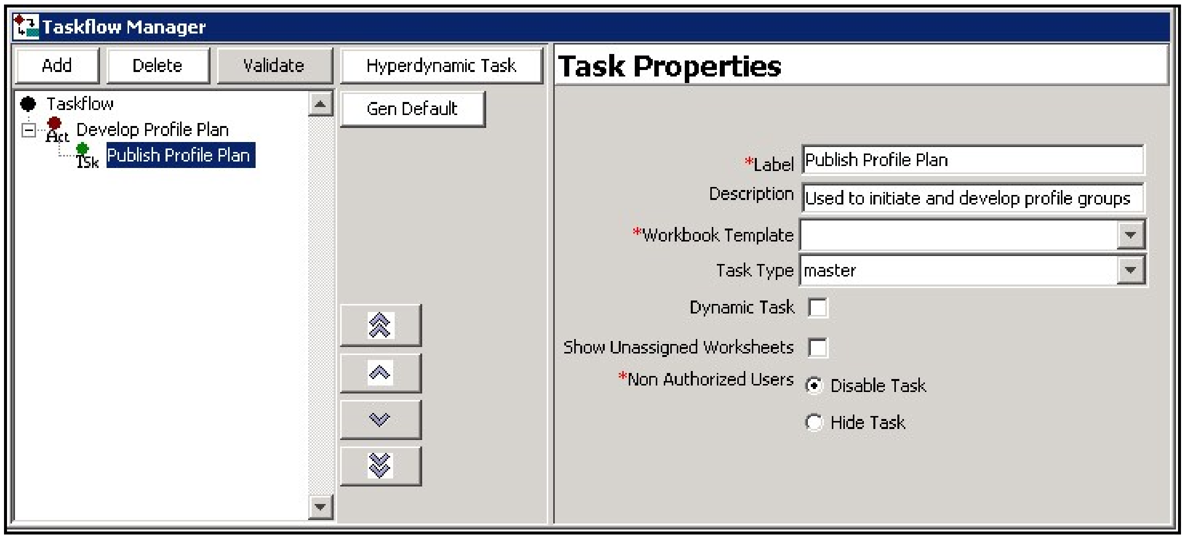
Show Unassigned Worksheets

By checking the unassigned flag, applications that include the potential for non-configured worksheets can allow those worksheets to be displayed within the RPASCE Client while continuing to maintain the ability to hide worksheets irrelevant to the task at hand for workbooks that do not contain non-configured worksheets.

Non-Authorized Users

Choose whether unauthorized users are able to see the task but are unable to edit it (Disable Task) or are not able to see it at all (Hide Task).

Figure 3-52 Task Properties



Task Properties

*Label: Publish Profile Plan
Description: Used to initiate and develop profile groups
*Workbook Template:
Task Type: master
Dynamic Task: ☐
Show Unassigned Worksheets: ☐
*Non Authorized Users: [✓ Disable Task] Hide Task

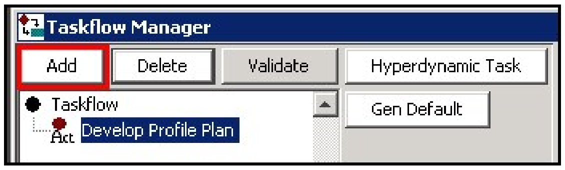
Once you've entered the task properties, the new task is created.

Add a Step to the Taskflow

Complete the following steps to add a step to the taskflow.

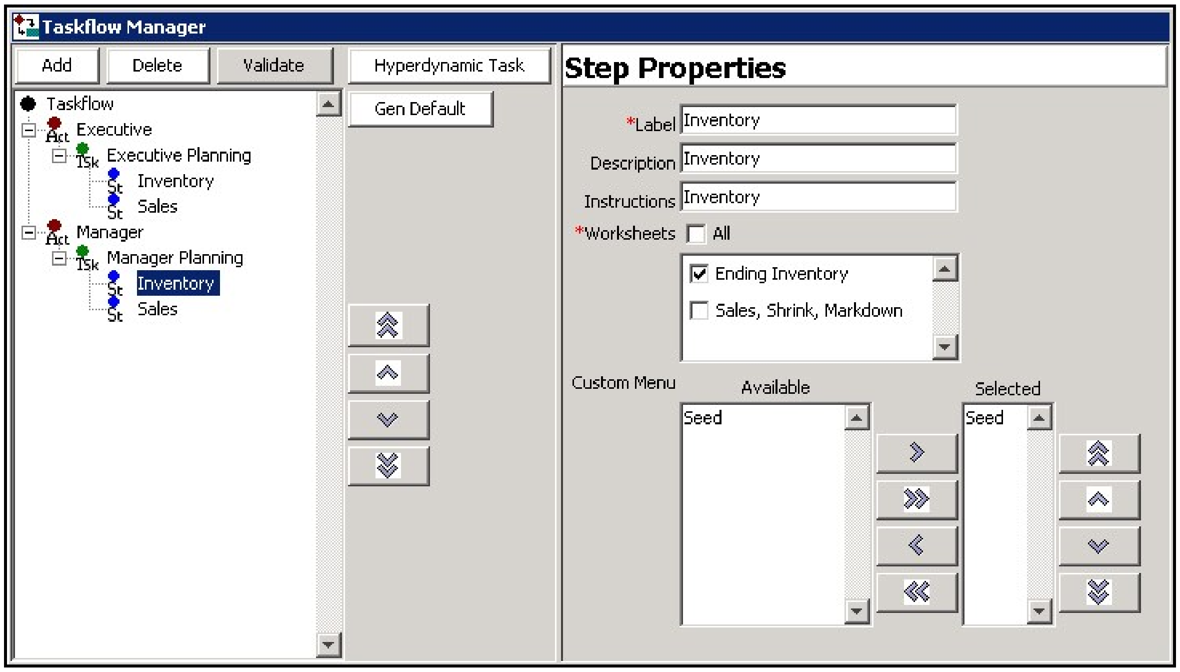
1. To add a step to a task you created, select the task and click **Add**.

Figure 3-53 Adding a Step to a Task



2. A step called Step1 displays in the navigation pane. Select Step1. The Step Properties display in the details pane.
3. Enter the following information in the details pane:

Figure 3-54 Step Properties



*Label: Inventory
Description: Inventory
Instructions: Inventory
*Worksheets: ☐ All
☑ Ending Inventory
☐ Sales, Shrink, Markdown

Label

Enter the name of the step as you want it to appear in the RPASCE Client. This is required.

Description

Enter the description of the step. This description displays when the user rolls the cursor over the step name in the RPASCE Client.

Worksheets

Select the worksheets that will be utilized in this step. You can select all worksheets to be included by checking the All check box. Alternatively, you can select a subset of worksheets from that workbook template. This is required.

Custom Menu

If you want to include a custom menu in this step, select it in the Available area and then use the arrow to move it to the Selected area. You can change the order of the custom steps by using the up and down arrows.

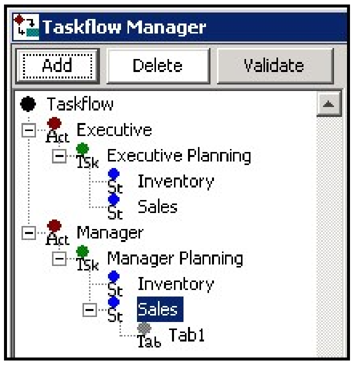
Once you have entered the step properties, the new step is created.

Add a Tab to the Taskflow

Complete the following steps to add a tab to the taskflow.

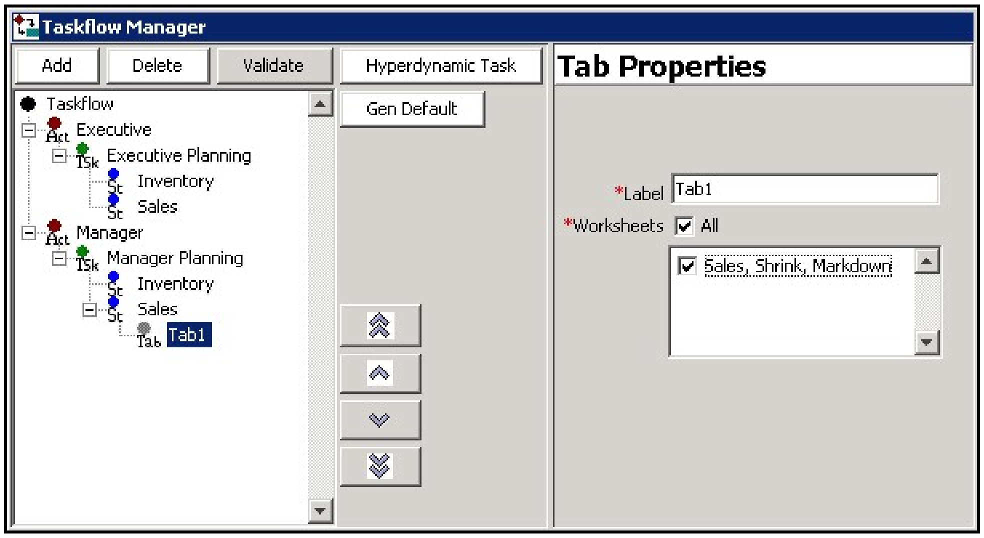
1. To add a tab to a step you created, select the step and click **Add**.

Figure 3-55 Adding a Tab to a Step



2. A tab called Tab1 displays in the navigation pane. Select **Tab1**. The Tab Properties displays in the details pane.

Figure 3-56 Tab Properties



Tab Properties

*Label Tab1

*Worksheets ☑ All

☑ Sales, Shrink, Markdown

3. Enter the following information for the tab:

Label

Enter the name of the tab as you want it to appear in the RPASCE Client. This is required.

Worksheets

Select the worksheets that will be utilized in this tab. You can select all worksheets to be included by checking the All check box. Alternatively, you can select a subset of worksheets. This is required.

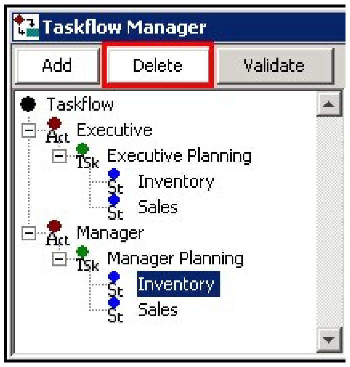
Once you have entered the tab properties, the new tab is created.

Delete Items from the Taskflow

Complete the following step to delete items from the taskflow.

1. To delete any activity, task, step, or tab in the taskflow, select it and click **Delete**.

Figure 3-57 Deleting Items in the Taskflow



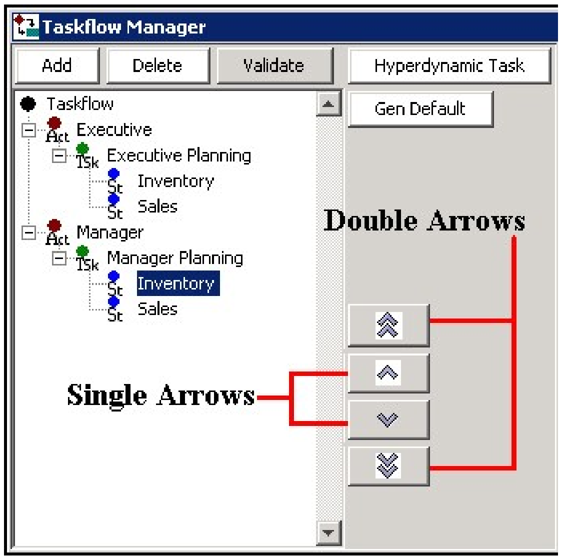
Edit Items from the Taskflow

To edit an activity, task, step, or tab that you have already created, select it in the navigation pane. Its properties display in the details pane. Then edit the properties you want.

Order Items in the Taskflow

To change the order of activities, tasks, steps, or tabs, select the item that you want to move and use the arrow buttons to the right of the navigation pane. Click a single arrow to move it one level in the list. Click the double arrow to move it to the top or bottom of the list.

Figure 3-58 Ordering Items in the Taskflow



Validate the Taskflow

As you create the taskflow, the Task List underneath the Taskflow Manager displays configuration elements that may cause validation errors. In addition, the Validation Results tool shows you if you have configured elements that are unavailable to users. For instance, if a configured taskflow does not have a task associated with a given workbook template, then the RPASCE Client user is not able to build or open any workbooks built from that template.

Note:

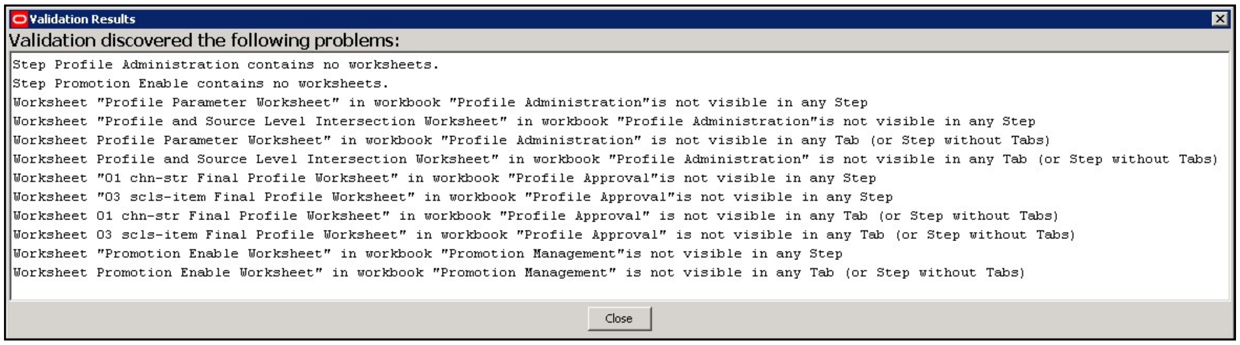
The presence of validation problems does not prevent you from building the PDSBase.

To validate your taskflow, click **Validate** at any time. The Validation Results window opens. The Validation Results tool checks for the following:

- Workbook templates that are not associated with a task
- Worksheets that are not a part of any step
- Custom menu items that are not available in any step
- Activity Group names that contain any spaces.

When conditions such as these exist, they display in the Validation Results window. Once you have reviewed the results, you can choose to resolve any issues if necessary.

Figure 3-59   Validation Results


Validation Results

Validation discovered the following problems:

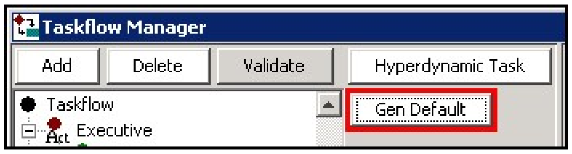
```
Step Profile Administration contains no worksheets.
Step Promotion Enable contains no worksheets.
Worksheet "Profile Parameter Worksheet" in workbook "Profile Administration"is not visible in any Step
Worksheet "Profile and Source Level Intersection Worksheet" in workbook "Profile Administration"is not visible in any Step
Worksheet Profile Parameter Worksheet" in workbook "Profile Administration" is not visible in any Tab (or Step without Tabs)
Worksheet Profile and Source Level Intersection Worksheet" in workbook "Profile Administration" is not visible in any Tab (or Step without Tabs)
Worksheet "01 chn-str Final Profile Worksheet" in workbook "Profile Approval"is not visible in any Step
Worksheet "03 scls-item Final Profile Worksheet" in workbook "Profile Approval"is not visible in any Step
Worksheet 01 chn-str Final Profile Worksheet" in workbook "Profile Approval" is not visible in any Tab (or Step without Tabs)
Worksheet 03 scls-item Final Profile Worksheet" in workbook "Profile Approval" is not visible in any Tab (or Step without Tabs)
Worksheet "Promotion Enable Worksheet" in workbook "Promotion Management"is not visible in any Step
Worksheet Promotion Enable Worksheet" in workbook "Promotion Management" is not visible in any Tab (or Step without Tabs)
```

The RPASCE Configuration Tools also performs validation within the Task List underneath the Taskflow Manager. As you edit within the Taskflow Manager, the Task List reports potential issues and errors within the configuration. For more information, see the Task List section.

Generate Default Mapping

Click **Gen Default** at any time to revert the taskflow to the default mapping.

Figure 3-60   Generate Default Mapping



The default mapping provides a basic structure to taskflow elements and can serve as either a simple configuration of workbook elements or as a starting point for a customized configuration of the taskflow elements.

The default mapping is set up as follows:

- There is one activity generated for every workbook group in the application. The label of the workbook group is used as the activity's label and description.
- Each activity contains a single task for each workbook present in the workbook group associated with that activity. The label of that workbook is used as the task's label and description. By default, workbooks are unavailable for unauthorized users in all tasks.
- Each task contains a single step for each workbook tab present in the workbook associated with that task. The label of the worksheet is used as the step's label, description, and instructions. Each step includes the worksheets contained within the workbook tab associated with that step and does not include worksheets contained within other workbook tabs present in the workbook. Each step contains all Custom Menus present in the workbook.
- The default mapping contains no taskflow tab elements.

Hyperdynamic Tasks, Steps, and Tabs

Some RPASCE solutions use custom workbook template libraries to extend the workbook creation functionality of RPASCE. For some of these solutions, some or all of the content of a workbook is determined, not by the configuration, but by the custom template library at the time the workbook is built.

Because the worksheet and tab content of these workbooks is dynamic, it is not possible to configure the taskflow in the same manner that a taskflow is configured for standard workbooks. Instead, these custom workbooks use special taskflow objects named hyperdynamic tasks, step, and tabs. Hyperdynamic tasks, steps, and tabs differ from their standard counterparts in the following ways:

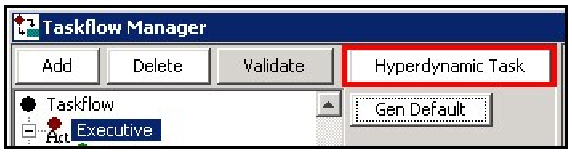
- A hyperdynamic task can contain only hyperdynamic steps.
- A hyperdynamic step can contain only hyperdynamic tabs.
- Hyperdynamic steps and tabs do not select visible worksheets based on the worksheets of the workbook assigned to the task. Instead, they are populated by providing the names of worksheets that may not exist within the configured workbook at the time the taskflow is configured.
- At the time of workbook creation, the RPASCE Client uses the configured list of worksheet names in coordination with the set of worksheets in the built workbook to determine which worksheets will be visible in the RPASCE Client and how they will be assigned to tabs.
- Because the worksheet membership of hyperdynamic tasks, steps, and tabs is not determined until workbook build time, they cannot be validated using the taskflow validation functionality.

Creating a Hyperdynamic Task

To create a hyperdynamic task to an activity, complete the following steps:

1. Select the activity and click **Hyperdynamic Task**.

Figure 3-61 Hyperdynamic Task Button



2. The new hyperdynamic task displays in the navigation pane. Select the new task. The new task properties is displayed in the detail panel.
3. Enter the following information in the Task Properties.

   Label

   Enter the name of the task as you want it to appear in the RPASCE Client. This step is required.

   Description

   Enter the description of the task. This description displays when the user rolls the cursor over the task name in the RPASCE Client.

Workbook Template

Choose the workbook template to be utilized in this task. This step is required.

Dynamic Task

Select this option if the steps for the task are dynamic based on the user selections made during the workbook wizard. When this option is selected, the user does not see any steps under a task in the RPASCE Client until a workbook is open.

> **Note:**
>
> Custom workbook code is still required to provide the ability to filter steps based on the wizard selections.

Show Unassigned Worksheets

By checking the unassigned flag, applications that include the potential for non-configured worksheets can allow those worksheets to be displayed within the RPASCE Client while continuing to maintain the ability to hide worksheets irrelevant to the task at hand for workbooks that do not contain non-configured worksheets.

Non-Authorized Users

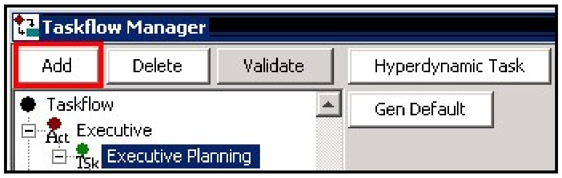
Choose **Disable Task** to allow unauthorized users to see the task but not access it. Choose **Hide Task** if you do not want unauthorized users to see the task at all.

Adding a Hyperdynamic Step to the Taskflow

To add a hyperdynamic step to a hyperdynamic task you have created, complete the following steps:

1. Select the hyperdynamic task and click **Add**.

Figure 3-62 Adding a Step to a Task



2. A step named Step1 displays in the navigation pane. Click **Step1**. The Step Properties display in the details pane.
3. Enter the following information in the Step Properties:

   Label

   Enter the name of the step as you want it to appear in the RPASCE Client. This is required.

   Description

   Enter the description of the step. This description displays when the user rolls the cursor over the step name in the RPASCE Client.

Instructions

Enter instructions that explain what to do in the step. These instructions display underneath the task pane.

Worksheets

Enter the names of the worksheets that can be visible within the step.

Custom Menu

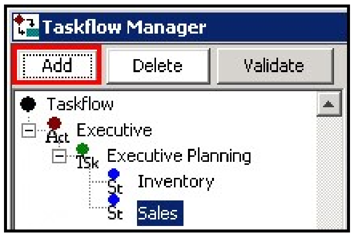
If you want to include a custom menu in this step, select it in the Available area and then use the arrow to move it to the Selected area. You can change the order of the custom steps by using the up and down arrows.

Adding a Hyperdynamic Tab to the Taskflow

To add a hyperdynamic tab to a hyperdynamic step you created, complete the following steps:

1. Select the hyperdynamic step and click **Add**.

Figure 3-63 Adding a Tab to a Step



2. Tab1 displays in the navigation pane. Click **Tab1**. The Tab Properties displays in the details pane.
3. Enter the following information for the tab:

   Label

   Enter the name of the tab as you want it to appear in the RPASCE Client. This is required.

   Worksheets

   Select the worksheets that will be utilized in this tab.

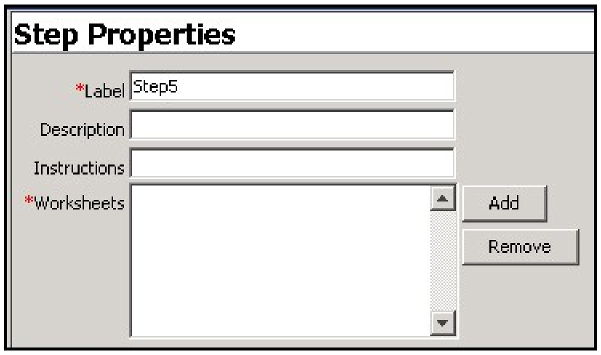
After you have entered the tab properties, the new tab is created.

Adding Worksheets to a Hyperdynamic Step or Tab

To add a worksheet to a hyperdynamic step or tab, complete the following steps:

1. Select the hyperdynamic step or tab. It displays in the detail pane.

Figure 3-64 Step Properties Detail Pane



Step Properties

*Label: Step5

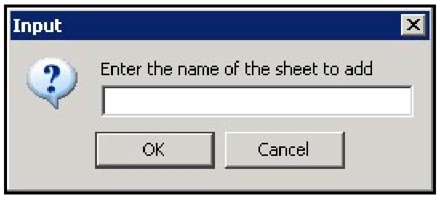
Description:

Instructions:

*Worksheets:

2. To add a worksheet, click **Add** and specify the name of the worksheet in the pop-up window.

Figure 3-65 Add a Worksheet Name



Input

Enter the name of the sheet to add

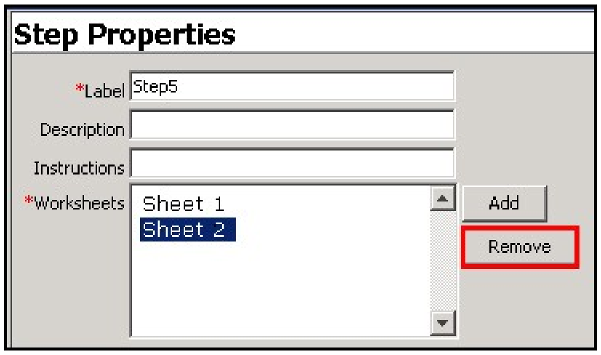
3. When finished, click **OK**.

Removing a Worksheet from a Hyperdynamic Step or Tab

To remove a worksheet from a hyperdynamic step or tab, complete the following steps:

1. Select the step or tab that contains the worksheet you want to remove. The details of the step or tab display in the detail pane.
2. Select the worksheet to remove and click **Remove**.

Figure 3-66 Removing a Worksheet from a Hyperdynamic Step or Tab



Step Properties

*Label: Step5

Description:

Instructions:

*Worksheets: Sheet 1, Sheet 2

The selected worksheet is removed from the step or task.

Creating the Taskflow for a Taskflow Created in a Release Earlier than 13.3.1

If you have a taskflow created in a release earlier than 13.3.1 that you want to use in 14.0 or later (without running through ConfigTools for some reason), at minimum you need to do the following:

Add a single `<activity_group>` around the existing `<activity>` tags to group them.

Add a `<solution>` tag with the solution ID to each task in the taskflow.

Add resources to the property file for the new activity group, for example:

```
solution1.ActivityGroup1=sample activity group label
```

```
solution1.ActivityGroup1.Desc=sample activity group description
```

Add a resource to the property file for the solution's label:

```
solution1.Solution.label=sample solution label
```

You need to qualify all `<name>` and `<description>` IDs by putting the `solutionId.` in front of them. Then do the same in the properties file so they match up.

Upgrading retailers may have multiple taskflows on the same server, where the users who switch between them use profiles. In this case, you probably want to construct a multi-solution taskflow that includes each solution. The simplest way to do this is to combine taskflows from Configuration Tools or construct as previously noted by putting each `<activity_group>` in the multi-solution taskflow xml, one after another, and then concatenating all the `.properties` resource files.

Note:

The name of the activity group (for example, the ActivityGroup1 in above) must not contain spaces. Otherwise, the taskflow will not display correctly in the RPASCE UI.

4

Solutions

This chapter describes information for:

Working with Solutions

A solution corresponds to an application configuration (for example, Merchandise Financial Planning or Item Planning). Each project can contain one or more solutions. For each solution, measures, rules, workbooks, and wizards are defined. Once a solution is created, it can be moved from one project to another.

Note:

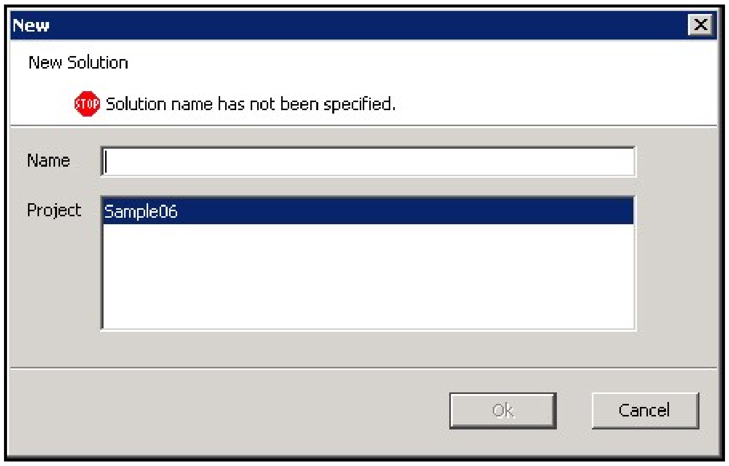
Some solutions, such as Curve, Grade, and RDF, have configuration steps that are specific to those solutions. For more information, see the corresponding configuration guide for the solution.

Create a Solution

Complete the following steps to create a solution.

Navigate: From the File menu, select **New** and then **Solution**. The Figure 4-1 opens.

Figure 4-1 New Window



Solution name has not been specified.

Sample06

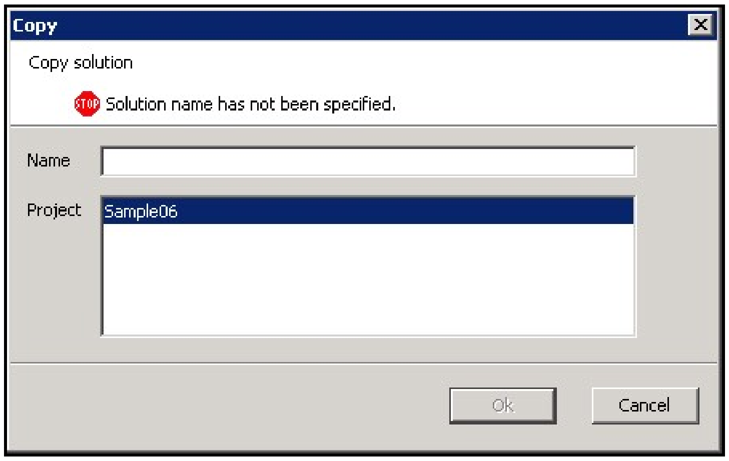
1. In the **Name** field, enter the name of the solution.
2. In the **Project** field, select the project in which the solution is to belong. Multiple projects will be listed if multiple projects are currently open.
3. Click **OK** to save any changes and close the window.

Copy a Solution

Complete the following steps to copy a solution.

1. Select the solution to be copied.
2. Right-click in the Configuration Components pane and click **Copy**. The Copy solution window opens.

Figure 4-2 Copy Solution Window



Solution name has not been specified.

Sample06

3. Type the new name for the solution in the text box.

Note:

This is the name that the solution is called after it has been copied.

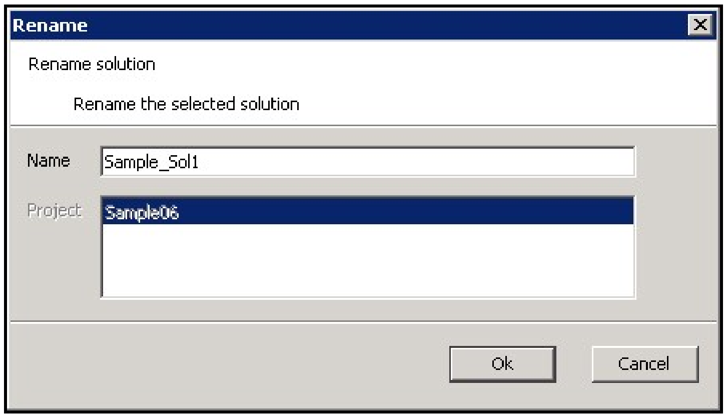
4. Select the project where the solution is to be copied.
5. Click **Finish** to save the copied solution in the specified project.

Rename a Solution

Complete the following steps to rename a solution.

1. Select the solution to be renamed.
2. Right-click in the Configuration Components pane and select **Rename**. The Figure 4-3 opens.

Figure 4-3 Rename Window


Rename
Rename solution
Rename the selected solution
Name: Sample_Sol1
Project: Sample06

3. Delete the old name from the resulting field and type the new name.
4. Click **OK** to save the new name.

Move a Solution

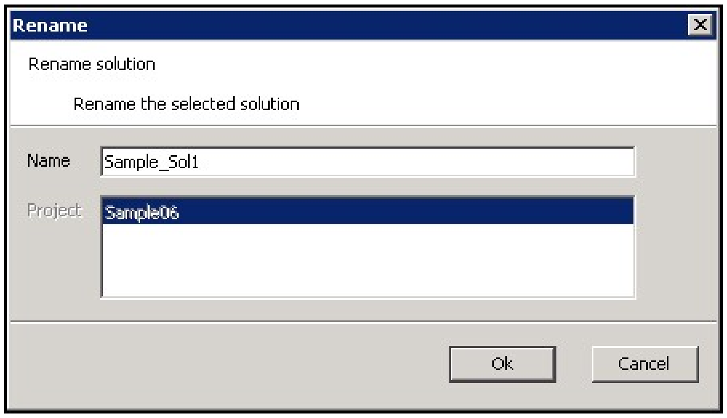
Complete the following steps to move a solution.

Note:

The Move a Solution operation is only available when multiple projects are open.

1. Select the solution to be moved.
2. Right-click in the Configuration Components pane and select **Move**. The Figure 4-4 opens.

Figure 4-4 Move Window


Rename

Rename solution

Rename the selected solution

Name: Sample_Sol1

Project: Sample06

3. Type the name of the solution in the text box.

> **Note:**
>
> This is the name that the solution is called after it has been moved.

4. Select the destination project for the solution from the resulting Project list.
5. Click **OK** to move the solution to the specified project.

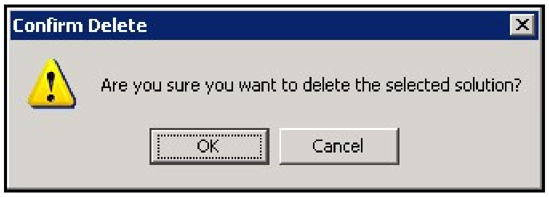
Delete a Solution

Complete the following steps to delete a solution.

Select the solution to be deleted.

Right-click in the Configuration Components pane. The Figure 4-5 opens.

Figure 4-5 Confirm Delete Window


Confirm Delete

Are you sure you want to delete the selected solution?

Click **OK** to complete the deletion.

Measures and Components

This section describes:

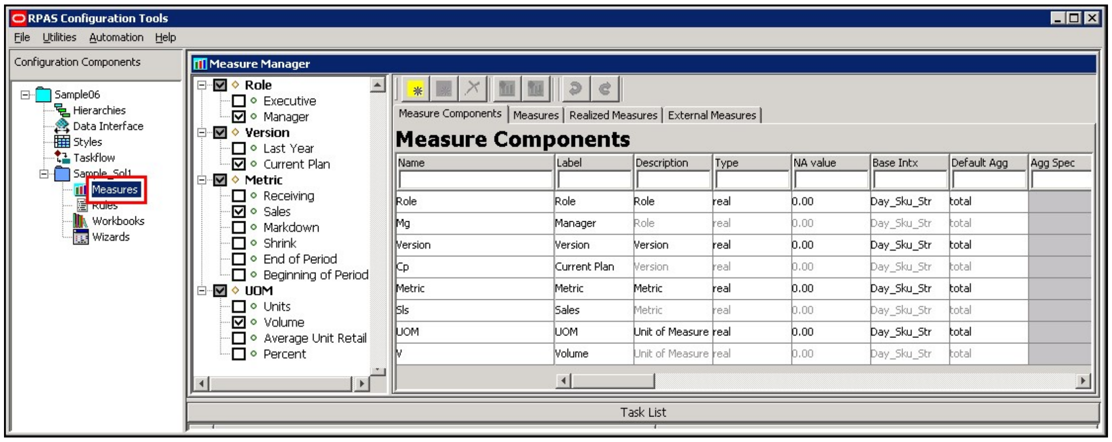
- Measure Manager
- Measure Component Design

- Create a Major Component
- Create a Minor Component
- Defining Measure Component Properties

Measure Manager

The Measure Manager window allows you to define major and minor components of measures and to specify properties for each component. Once the component structure is defined, the Measure Manager generates measures by combining the components that are selected. You may then select (realize) the valid measures and further update the properties for individual measures.

Figure 4-6 Measure Manager Window



Measure Components

Name	Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spec
Role	Role	Role	real	0.00	Day_Sku_Str	total	
Mg	Manager	Role	real	0.00	Day_Sku_Str	total	
Version	Version	Version	real	0.00	Day_Sku_Str	total	
Cp	Current Plan	Version	real	0.00	Day_Sku_Str	total	
Metric	Metric	Metric	real	0.00	Day_Sku_Str	total	
Sls	Sales	Metric	real	0.00	Day_Sku_Str	total	
UOM	UOM	Unit of Measure	real	0.00	Day_Sku_Str	total	
V	Volume	Unit of Measure	real	0.00	Day_Sku_Str	total	

Measure Properties

This section describes the fields displayed in the Measures tab.

Inheritance

Measure properties are inherited at the component level. The properties defined for a component are inherited by the minor components that belong to that component and to the measures that are associated with that component unless it is overridden at a lower level.

When a measure can inherit a property from more than one of the components that construct it, the measure inherits from the component that belongs to the highest major component in the component tree. For many properties, it is a good practice to set the properties for just one major component or for minor components in just one major component branch.

Overriding

Measure property inheritance can be overridden at the minor component or at the measure level. Once a property is set at a lower level, changes made to that property at a higher level will no longer be inherited at that lower level.

Measure Components

A major component is the highest level in the component inheritance hierarchy. Properties defined at this level are inherited by all minor components that are created under the major component.

Within each major component, you can create one or more minor components. You can also create a minor component under a minor component and also modify properties at the minor component level.

Once major and minor components are defined, the Measure Manager generates measures that are based on the combination of selected components. These measures cannot be used elsewhere in the configuration tools until the valid prototype measures are realized.

Note:

Components have no structural impact on the built solutions, and they are not exposed to end users. Components are intended to be a convenience to aid the configuration administrator to easily group measures together and to set measure properties at higher levels.

Component Process

The following steps describe the component process.

1. Create major components, from which measures are composed.
2. Create minor components, which are sub-groupings or specific items in a major component.
3. Define measure properties at the major component level. The minor components will inherit the properties associated with the major component they belong within.
4. If necessary, modify the measure properties at the minor component level.

Measure Naming Conventions

All components used in RPASCE configurations must adhere to the following naming convention:

Characters allowed:

- Capital and lowercase letters (A, b…Z)
- Numerals (1, 2, 3…)
- Underscore (_)

With the exception of underscores, no non-alphanumeric characters are allowed. Measure component names must start with a letter. Spaces are not allowed.

Note:

Since the names of realized measures are limited to 30 characters, and those names are constructed by concatenating the names of the components from which the measure is built, it is usually good practice to abbreviate the names of components where necessary. However, there is no limit on the number of characters for measure labels.

Measure Component Design

The following two basic principles must be kept in mind to make the Measure Manager as powerful as possible.

- Major and minor components should be designed with the idea of maximizing the inheritance of properties and minimizing the amount of property overriding.
- Use minor components to make measure definition manageable.

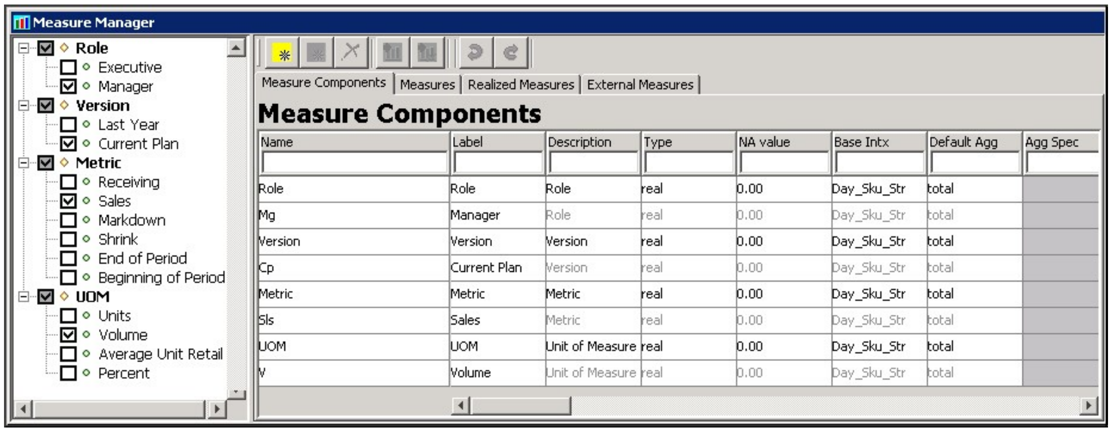
For example, consider a configuration that has 2000 measures and 1500 of the measures is of data type real. Avoid grouping all 1500 measures into a single minor component because smaller subgroups of 1500 measures cannot be easily edited. Minor components can also have minor components, so within the 1500 measures, the configuration administrator may break them out further. This could be based on the aggregation method, such as total, max, recalc, and base intersections. Ideally, filtering by the checking of a lowest level minor component should allow the configuration administrator to easily view and manage every resulting measure for that minor component.

Create a Major Component

Complete the following steps to create a major component.

Navigate: In the Configuration Components pane, select Project then Solution then, Measures and then the Measure Components tab. The Measure Manager window opens in the workspace.

Figure 4-7   Measure Manager Window - Measure Components Tab



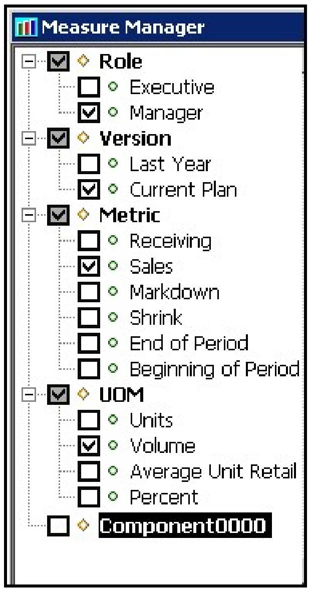
- [✓] Role
  - [ ] Executive
  - [✓] Manager
- [✓] Version
  - [ ] Last Year
  - [✓] Current Plan
- [✓] Metric
  - [ ] Receiving
  - [✓] Sales
  - [ ] Markdown
  - [ ] Shrink
  - [ ] End of Period
  - [ ] Beginning of Period
- [✓] UOM
  - [ ] Units
  - [✓] Volume
  - [ ] Average Unit Retail
  - [ ] Percent

Measure Components

Name	Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spec
Role	Role	Role	real	0.00	Day_Sku_Str	total	
Mg	Manager	Role	real	0.00	Day_Sku_Str	total	
Version	Version	Version	real	0.00	Day_Sku_Str	total	
Cp	Current Plan	Version	real	0.00	Day_Sku_Str	total	
Metric	Metric	Metric	real	0.00	Day_Sku_Str	total	
Sls	Sales	Metric	real	0.00	Day_Sku_Str	total	
UOM	UOM	Unit of Measure	real	0.00	Day_Sku_Str	total	
V	Volume	Unit of Measure	real	0.00	Day_Sku_Str	total	

1. Right-click in the left-hand pane of the Measure Manager window and select **Add Major Component** or click **Add Major Component**. The major component displays in the Measure Manager navigation tree with a default name.

Figure 4-8 Measure Manager Navigation Tree



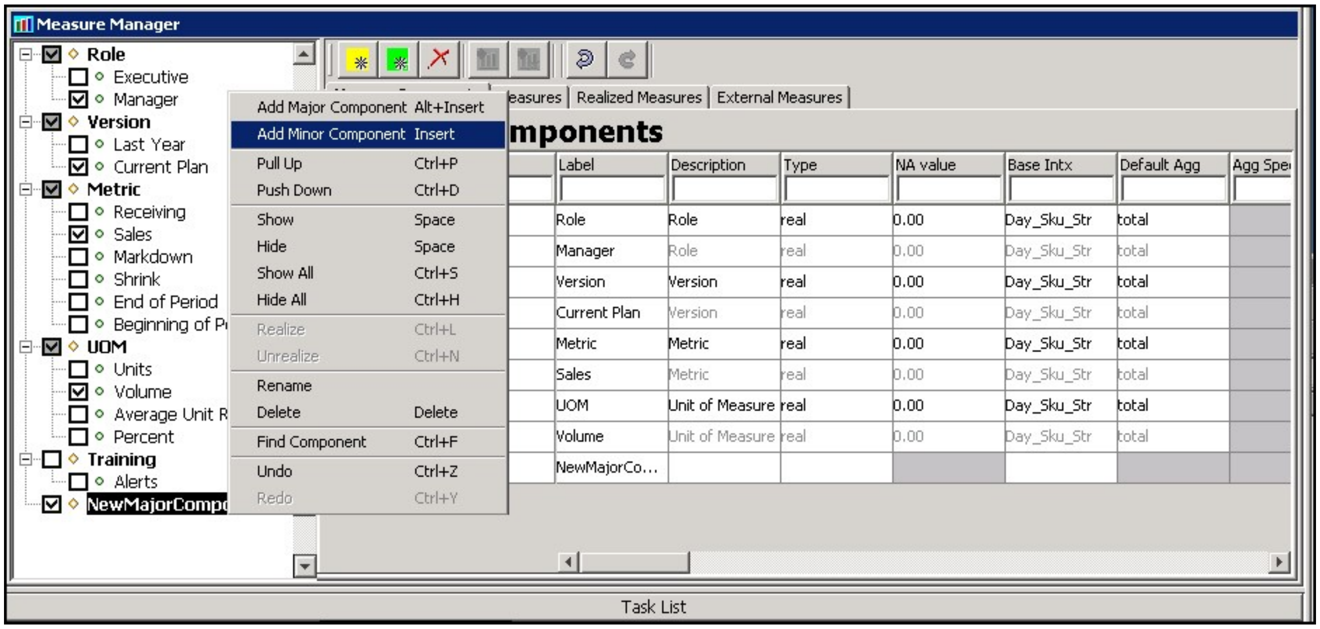
2. To change the component name, right-click on the component and select **Rename**, or select the component from the navigation tree and then modify its **Label** field from the **Measure Components** tab.

Create a Minor Component

Complete the following steps to create a minor component.

Navigate: In the Configuration Components pane, select Project then Solution then, Measures and then the Measure Components tab. The Measure Manager window opens in the workspace.

Figure 4-9   Major Measure Component Selected

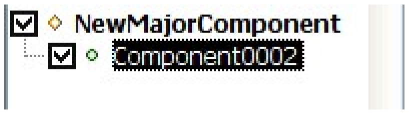


Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spe
Role	Role	real	0.00	Day_Sku_Str	total	
Manager	Role	real	0.00	Day_Sku_Str	total	
Version	Version	real	0.00	Day_Sku_Str	total	
Current Plan	Version	real	0.00	Day_Sku_Str	total	
Metric	Metric	real	0.00	Day_Sku_Str	total	
Sales	Metric	real	0.00	Day_Sku_Str	total	
UOM	Unit of Measure	real	0.00	Day_Sku_Str	total	
Volume	Unit of Measure	real	0.00	Day_Sku_Str	total	
NewMajorCo...						

1. Select the major or minor component that the new minor component is to be added beneath. In Figure 4-9, **NewMajorComponent** is selected.
2. Choose one of the following methods:
   - Right-click the Measure Definition navigation tree and select **Add Minor Component**.
   - Click **Add Minor Component**.
   - Press **Insert**.

   The minor component displays in the Measure Manager navigation tree with a default name.

   Figure 4-10   New Minor Component

3. To change the name, right-click on the component from the navigation tree and select **Rename.** You can also select the component from the navigation tree and modify the **Label** field from the Measure Components tab.

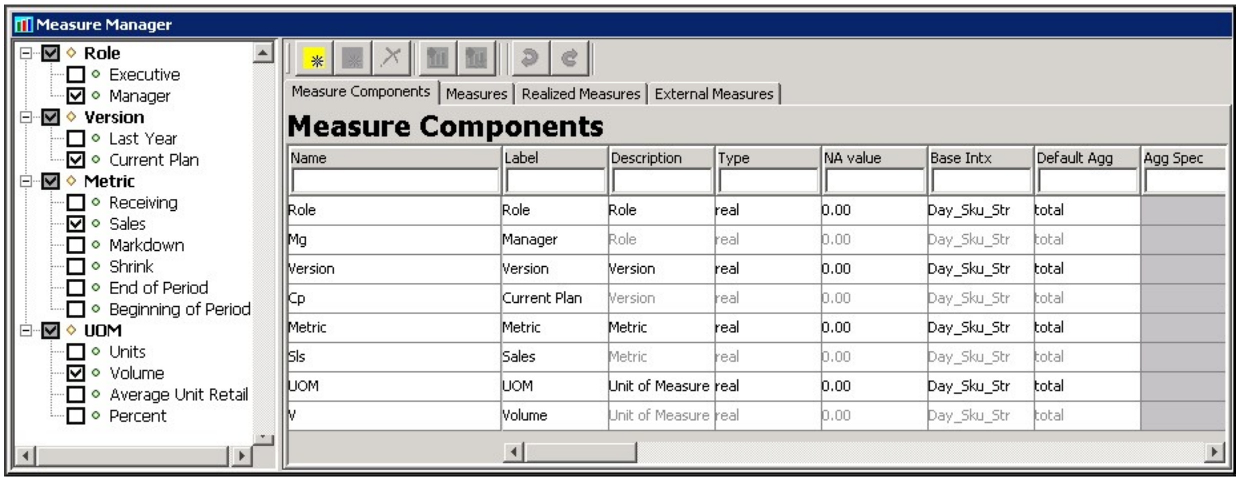
Defining Measure Component Properties

Complete the following steps to define the Measure Component properties.

Once components have been created, open the Measure Manager and select the Measure Components tab.

Select the major or minor components you want to view from the Measure Manager navigation tree. The selected components display in the Measure Components tab.

Figure 4-11   Measure Components Tab



- ✓ Role
  - ☐ Executive
  - ✓ Manager
- ✓ Version
  - ☐ Last Year
  - ✓ Current Plan
- ✓ Metric
  - ☐ Receiving
  - ✓ Sales
  - ☐ Markdown
  - ☐ Shrink
  - ☐ End of Period
  - ☐ Beginning of Period
- ✓ UOM
  - ☐ Units
  - ✓ Volume
  - ☐ Average Unit Retail
  - ☐ Percent

Measure Components

Name	Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spec
Role	Role	Role	real	0.00	Day_Sku_Str	total	
Mg	Manager	Role	real	0.00	Day_Sku_Str	total	
Version	Version	Version	real	0.00	Day_Sku_Str	total	
Cp	Current Plan	Version	real	0.00	Day_Sku_Str	total	
Metric	Metric	Metric	real	0.00	Day_Sku_Str	total	
Sls	Sales	Metric	real	0.00	Day_Sku_Str	total	
UOM	UOM	Unit of Measure	real	0.00	Day_Sku_Str	total	
V	Volume	Unit of Measure	real	0.00	Day_Sku_Str	total	

Specify the information for the component properties (for example, Name, Label, Description, and so on), which will apply to the measures that are inheriting property values from the component.

Note:

The values that are entered for major components are inherited by the child minor components and the auto-generated measures. Not all properties need to be entered for all components. Properties that are grayed out in the component properties table cannot have a value in the current context. Typically, this is based on the data type of the measure. For example, only components of Boolean data type can have an alert category or an alert expression.

Measure Component Properties

This section describes the fields displayed in the Measure Components tab.

Name

The name (identifier) of a component is used to identify the component within the Configuration Tools. Measure names are built by concatenating the names of the components from which the measure is built. They are concatenated in the order (from top to bottom) of the sequence that the components display in the list of components. You can override the measure name by manually entering a new name in the Name field. Measure names can be up to 30 characters in length. They can include letters and numbers but must start with a letter.

Label

The label of the component is used to generate measure labels in a similar way that measure names are generated. Labels are displayed to RPASCE end users. There is no maximum size limit but keep the grid display limitations in mind when creating a measure label.

Note:

Even though measure/fact labels are not required to be unique, RPASCE strongly suggests against duplicate labels, because they are used as identifiers within the user interface of RPASCE applications. The labels of Report Category measures are used in OAS UI and have additional restrictions. See the Create Reporting Views in PDS section of *Oracle Retail Predictive Application Server Cloud Edition Administration User Guide* for fact label restrictions.

Description

A description of the component is used to generate measure descriptions in the same way that measure labels are generated. The configuration administrator can enter any text to provide more information beyond the measure label to the end user. The description can be viewed by the end user in the RPASCE user interface.

Type

Select one of the following data types:

- Real – Floating point numeric values. Most measures are of this type.
- Int – Numeric integer values. There are no special spreading algorithms for integer measures, which should normally be used only for measures that are calculated bottoms up. Formatting can be used to display real measures as integer value in the RPAS CE client.
- Boolean – True or false values, which are typically used for flags and indicators.
- Date – Date and time. This can easily be converted to position names using standard RPASCE functions.
- String – Variable length strings, which are typically used for notes and names.

NA Value

This is a value (typically zero for numeric measures) that is not physically stored but is inferred. It is used to help with storage and calculation efficiency, and it may be changed by RPASCE (in full-evaluation mode) if better efficiencies can be obtained with a different value. See Appendix – Calculation Engine User Guide and Appendix – Rules Function Reference Guide for more information.

Base Intx

The Base Intersection. The lowest level at which data is used for a measure. In the PDS, the measure is stored as a fact. For non-shared measures, they are stored at the measure’s base intersection. For shared measures, the measure’s base intersection may be higher than the fact intersection. Inside a workbook (for performance reasons), values for the measure may be stored higher than the base intersection. Nevertheless, whether stored or not, values for aggregated levels may be viewed in a workbook and used in calculations in workbooks or applications. Double-click this field to open the Select Intersection window. The hierarchies that were defined using the Hierarchy tool are displayed. One dimension from each hierarchy can be selected, but a dimension is not required for each hierarchy. Alternatively, a measure may be marked as scalar. A scalar measure has only one value at any combination in the positions of dimensions of the application. A Labeled Intersection may also be selected as the base intersection of a measure. The Labeled Intersection field is populated based on the Labeled Intersections defined through the Labeled Intersection window accessed in the Hierarchy Definition manager.

Note:

RPASCE imposes a limit of five dimensions that can be defined in a measure's base intersection.

Multi-Level Display

Deprecated. This property indicates whether the measure supports the display of its data at intersection levels lower than its base intersection. Possible values are true and false. See the *Oracle Retail Predictive Application Server User Guide* for more details about multi level display and about editing a measure.

Default Agg

The default aggregation method should be selected from the valid aggregation methods for the component. The valid aggregation method depends on the data Type selected for the component. See Appendix – Calculation Engine User Guide and Appendix – Rules Function Reference Guide for more information on aggregation and spread methods.

Note:

Only measures with an aggregation type of ambig, pst, or pet can be aggregated from lower than the partition levels to higher than the partition levels in a global application.

Agg Spec

This hybrid aggregation mechanism is designed to allow the configuration administrator to specify a complex method to aggregate the values of a measure. It allows a different aggregation method to be specified for each hierarchy in the measure's intersection. When a measure with the hybrid agg type needs to be aggregated, this is accomplished by separately aggregating each hierarchy of the intersection according to the agg method for that hierarchy.

Example:

Measure XYZ is defined at day_sku_str and has a hybrid aggregation type. The specifics for the aggregation are as follows:

- Calendar should be aggregated by the first method.
- Location should be aggregated by the total method.
- Product should be aggregate by the total method. -stopped

Suppose that XYZ must be aggregated to the level of mnthclssrgn_. The process of generating this new value is accomplished by three successive aggregations:

- day_sku_str_ to day_clssstr_ by total (product)
- day_clssstr_ to day_clssrgn_ by total (location)
- day_clssrgn_ to mnthclssrgn_ by first (calendar)

In this example, the user is allowed visibility to and control over the mechanism by which pst is performed.

A brief description of the user interface functionality/constraints is as follows:

- The hybrid aggregation method now displays in the deff agg drop-down selector.
- When a measure is specified for hybrid agg, the agg spec (aggregation specification) field becomes editable.
- An agg spec can be typed in or built through a window (double-click the agg spec editor or select **Ctrl-Space** to launch it).
- This window looks very similar to the standard wizard for workbooks. On the right, the ordering of hierarchies in the intersection of the measure is set by dragging the hierarchies in the list. On the left, a separate aggregation type is selected for each hierarchy. For the most part, these are the aggregation types that are available for the measure based on its type.

Exceptions are as follows:

- Recalc or hybrid cannot be used within an agg spec.
- First and last can be used only on the Calendar hierarchy only.
- An aggregation type must be specified for each hierarchy in the intersection.
- If a value is to be typed into agg spec, the syntax and meaning is the same as the arguments used by the aggregate function of the rule engine.
- A hybrid aggregation measure must be read only in its agg state.
- A hybrid measure must have a spread type of none.

Note:

The hybrid aggregation type is not supported for extended measures or for the load aggregation method for a measure. Unlike Aggregate procedure, the recalc aggregation type is not supported for any hierarchy for am measure using the hybrid aggregation type. Measures that use the hybrid aggregation type cannot be aggregated from a local domain into the global application.

Default Spread

The default spread method should be selected from the valid spread methods for the component. The valid default spread method depends on the data Type selected for the component. See Appendix – Calculation Engine User Guide and Appendix – Rules Function Reference Guide for more information on aggregation and spread methods.

> **Note:**
>
> The spread method can be overridden on edit in the RPASCE User Interface. For all populated spread methods (ending with pop), the spread method is the same as the underlying method (for instance, prop_pop is like prop), except that only cells with a value that is different from the naval are used in the spreading, and cells with a value equal to the naval are ignored.

Base State

The ability of the measure at the base level to be modified. The available options are read or write.

Agg State

The editability of the measure at the aggregate level, which are all intersections higher than the base intersection (read or write). Set the Base State to write and the Agg State to read for those measures that need to be manipulable, but where there is no business requirement to manipulate them other than at their base intersection. Usually there is no sensible way to spread such measures. The manipulability of measures will change according to protection processing principles. Therefore, base state and agg state should only be used to override the result of protection processing (for example, to make a measure non-manipulable that protection processing would otherwise allow to be manipulated). See Appendix – Calculation Engine User Guide and Appendix – Rules Function Reference Guide for more information on the Agg State of measures.

Database

The physical location in the file system of the database that stores the data for this measure. Those measures that contain data that persists beyond the lifetime of a given workbook store their information within a database within the RPASCE application. This field is used to specify the path to the location of the database to use for the measure. All databases are contained within the data directory of the application. If the specification does not begin with the data directory, `data/` is attached to the beginning of the entry at the time of installation (for instance, the entry Sales is registered as `data/Sales`)

Note:

The presence of multiple measures within a single database can create unnecessary contention when the measures' data are being updated as part of a workbook commit or batch calculation. In order to avoid this, measures that are updated through batch calculations or workbook commits should be configured with unique values for the database property so that they each get a separate database. Measures that are only updated by data load and that are read-only for workbook and batch operations can be grouped together without causing contention.

View Type

The View Type field holds properties for two types of measures:

- Those that are calculated when viewed
- Those that are synchronized with other measures.

If the view type is none, the measure is of neither type.

If the View Type is view_only, the measure is not calculated during a normal calculate cycle, and it is calculated on-the-fly when required (for instance, for viewing). Such measures must have an aggregation type of recalc and must appear on the left-hand side of only one expression in a rule group. They may not appear on the right-hand side of any expressions. The measure must not have a database assigned. See Appendix – Calculation Engine User Guide of this guide for more information.

Synchronized measures are in effect, views of two or more other measures where changes and lock to those other measures are immediately reflected in the synchronized measure (and vice versa).

Example:

A closing stock (cs) measure may be synchronized with a season opening stock (sos) measure and an opening stock (os) measure so that a change to opening stock in week 3 will immediately cause the same change to be applied to closing stock in week 2 (since closing stock in week 2 and opening stock in week 3 are the same). Synchronized measures require a synchronization type in the View Type property, which must be one of `sync_first_lag`, `sync_lead_last`, `sync_first` or `sync_last`, and a list of measures to synchronize with in the Sync With property.

- none – The measure is calculated normally.
- view_only – The measure is calculated when viewed.
- sync_first_lag – Period 1 is from the first measure (no calendar). Periods 2...N are from the second measure 1...N-1 (lag) [for example, bop synchronized with os and eop].
- sync_lead_last – Periods 1...N-1 are from the first measure 2...N (lead). Period N is from the second measure (no calendar) [for example, eop synchronized with bop and cs].
- sync_first – Gets Period 1 from the measure (similar to pst along calendar dimension) [for example, os synchronized with bop].
- sync_last – Gets Period N from the measure (equivalent to pet along calendar dimension) [for example, cs synchronized with eop].

Sync With

A comma-separated list of measures used for synchronization. This depends on the View Type.

Insertable

This field indicates whether the measure can be inserted as an extra measure in workbooks built from templates that are not configured to contain the measure. Insertable measures can be added to a workbook during the wizard process on the Extra Measures wizard page before a workbook is built, or by inserting the measure in the Show/Hide window in the RPAS CE Client inside a built workbook. Measure security must also be defined for Insertable measures in the RPASCE Security Administration workbook template. Possible values are true and false. See the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide* for additional information about measure security.

> **Note:**
>
> The Extra Measures wizard is not available by default for every workbook; it must be configured as a custom wizard page.

Non-Translatable

The selected value of true will represent that a measure should be omitted from generated translation resources. The false or unspecified value will represent that a measure should be included in these resources and all legacy configurations will therefore treat all measures as translatable unless further configuration is performed.

UI Type

The UI Type property affects how the user interacts with a measure within the client. There are two supported values for UI Type; by default, UI Type is unspecified, meaning that the measure exhibits no special behavior within the client.

- Picklist – When a measure is given a UI Type of picklist, it will no longer accept edits within the client. Instead, the user will be provided with a drop-down box that contains a set of valid values. Selecting an item in this drop-down will set the cell of the measure to the appropriate value. For picklist measures, the contents of the drop-down box are determined by the range attribute.
- Media – When a measure is given a UI Type of media, the RPASCE Client will not display the values of the measure's cells. Instead, it will examine the contents of a cell to extract image location information, retrieve the appropriate image resource and display that image in the cell. For a measure to have a UI Type of media, that measure must be a string type measure.

Note:

For information regarding the values that should be loaded into a media UI Type measure; see the Media Measures section of the *Oracle Retail Predictive Application Cloud Edition Administration Guide*.

Range

Specify an allowed range for the measure at edit time. For numeric values, the syntax is specified as Lower Bound:Upper Bound. If the RPASCE UI user attempts to enter a value in the RPASCE Client outside of this range, the modification is rejected. For string measures, any entry in this field is ignored.

For date measures, the lower and upper bounds specify both the date and the time. The default time for the lower bound is 12:00:00 AM. The default time for the upper bound is 11:59:59 PM. The time portion of the bounds is included regardless of the style setting because users can change the style setting on the client side.

For numeric or string measures with a UI Type of picklist (numeric and string), values are comma separated value/label pairs, where the label is given in brackets, such as a(labela), b(labelb), and c(labelc). If a label is not specified (for example, a, b, c, d), the value is also used as the label. The value of the cell is used in calculations; however, labels (if specified) will be displayed in the user interface, both in the grid and in the picklist.

Note:

The symbols of **,** (comma) and **:** (colon) are reserved characters for picklist and range definitions of a measure with a UI Type of picklist. These characters cannot be used to define the Label part of a picklist Value/Label pair.

If a cell contains a value that is not valid for the picklist, the value is displayed in the grid. When the measure's range is specified in this manner, all cells in the RPASCE Client will display these same values as valid options for the picklist. The valid set of options for a picklist measure can also be defined in such a way that they are context sensitive, which means that they vary from position to position. For example, a picklist measure with a base intersection at the SKU dimension could have valid values that vary according to which class the SKU belongs. The configuration administrator sets this up by setting the range property of the picklist measure as measurerange = measS where measS is the name of a string measure that holds the valid picklist options (in the valid formats described earlier) in each of its cells. The measure that holds the valid picklist values (in this example, measS) can have a base intersection at any of the dimensions in the hierarchies and the values shown in the picklist measure for any intersection are effectively “looked up" using normal ‘nonconforming measure' handling.

The valid values for a picklist for a cell are referenced from a measure dynamically. If required, it is possible for the valid values of picklists to change during the life of the workbook as a result of calculations or end-user edits. The value used will always be that of the last calculate, so direct or indirect (through calculation) edits to the picklist value measure are ignored when a calculation is pending.

Purge Age

The number of days (without a load) before measure data is purged. See the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide* for details of how this property is used in the loadmeasure utility.

Lower Bound and Upper Bound

If the range of valid values for a numeric (real or integer) measure applies across the whole application, the range property can be used to specify valid values for data entry validation. If the range of valid numeric values varies according to positions in the hierarchies, the Lower Bound and Upper Bound must be used instead. If specified by the configuration administrator, the Lower Bound and Upper Bound properties must be a valid, realized measure name that provides the bounding values for this measure's data cells. These properties must contain measure names (not numeric values).

Note:

If one (but not the other) of Lower Bound and Upper Bound is specified, only one limit is checked. For example, if a Lower Bound is set, but an Upper Bound is not, valid values are greater than or equal to the value held in the Lower Bound measure, but with no upper limit. The Lower Bound and Upper Bound measures can be non-conforming with respect to the measure that has bounds, and the value to be used will be obtained by normal non-conforming processing (that is, replicated down from higher levels or aggregated up from lower levels).

Sp Value Type and Sp Value

These two properties specify the special value type and the special value value. They are used to define the way in which special values are handled by RPASCE. These two properties can be used together to specify how to display cell values in the User Interface (UI) that have a value equal to the naval of the measure. In particular, it supports solutions that want to interpret cells with the naval as meaning "no value" by displaying a null value to the end user.

The SP Value Type property specifies the type of value that will be shown. Valid values for this property are Null, Cell Value, and User Entered. The default behavior is that such cells will have their cell value shown, which is what the value of Cell Value in the SP Value Type property means. However, these properties can be used to override the default to either show null, which is defined later, or to display a specific value. To configure the values to display null, assign Null as the SP Value Type property. When Null is configured, the cell will be blank in the RPAS CE Client when the value equals the naval for a numeric, a date, or a string measure. For Boolean measures, it will be a grayed-out check box. When a specific value is required, you should select User Entered for the SP Value Type and enter the value to be displayed in the SP Value field.

For the Special Value field, the entry in this field must be of the same data type as the measure, and validation is enforced in this field.

- For a Boolean measure, when the Special Value Type field is set to User Entered, the only valid entry for the Special Value field is either true or false.
- For a Date measure, when the Special Value Type field is set to User Entered, a date in the format of YYYYMMDD can only be entered by the user.
- By default, each measure will be registered with Cell Value as the default Special Value behavior. For cases where a Special Value Typesetting other than User Entered is used, but a Special Value entry is provided, the Special Value entry will not be used. Instead, the measure will be registered with Cell Value as the default behavior.
- When Special Value Type is set to User Entered, but a Special Value entry is not entered, the Special Value Type will not be used. Instead, the measure will be registered with Cell Value as the default behavior.
- When a PDS has been built that includes a measure with a special value setting, and that special value setting for that measure is removed. When the PDS is patched the measure will get updated with Cell Value as the default special behavior.
- RPASCE allows for the special value measure property to be updated.

Dim Attr Type

This check box option is used to indicate that the measure must be registered as a 1-dimension attribute. Dimension attributes allow for additional information to be defined for the positions of a given dimension. This is commonly used to define and display an alternate label for a position (other than the loaded position label) or to display supplemental information about a position (such as the status of a given position).

The following requirements must be met for a measure to be eligible to be a dimension attribute:

- The measure must be realized.
- The measure must be 1-dimensional, which means that its base intersection must only have one dimension.
- The measure must be stored, which means that it must have a defined database.

Dim Attr Name (optional field)

This is only to be used if the measure is set to be a dimension attribute measure. When a dimension attribute is displayed within the RPASCE Client, the Dim Attr Name will be used in place of the measure name. If no Dim Attr Name is supplied, the RPASCE Client displays the measure name.

Dim Attr Label (optional field)

This is only to be used if the measure is set to be a dimension attribute measure. When a dimension attribute is displayed within the RPASCE Client, the Dim Attr Label will be used in place of the measure label. If no Dim Attr Label is supplied, the RPASCE Client displays the measure label.

Allowed Aggs

The set of the allowable aggregation methods for the measure based on the measure data type. You can add so called extended measures to RPASCE Client views that are normal measures, but with aggregations based on different aggregation methods. The aggregation methods that are available for selection are based on the Allowed Aggs of the base measure. The same base measure can have multiple extended measures based on different aggregation methods.

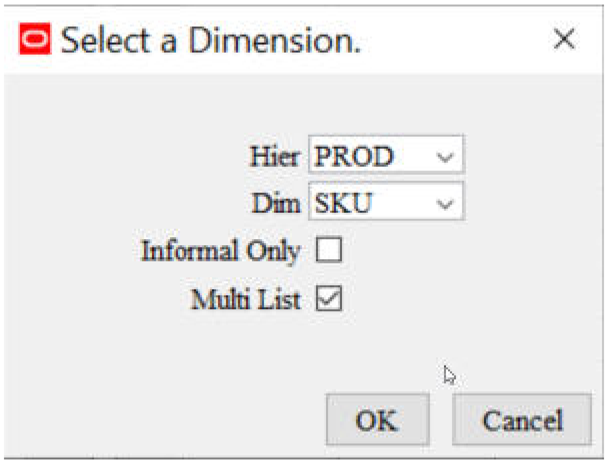
Style

One of the styles defined within the style tool may be specified as the default style to be used to display measures based on this component inside the User Interface. See the section on the Style Manager for details on the specification of style information.

Single Hier Select

This property is only valid for components that have a Type (data type) of string and non-picklist. It specifies that cell contents are to be entered by users using the Single Select Widget. This is a widget in the RPASCE Client that presents the end user with a view of the positions along the dimension set in the configuration. The user may then select any single position. When a user double clicks the Single Hier Select field, a Select a Dimension window pops up where the user must select the hierarchy and dimension whose positions will be displayed to the user in the RPASCE Client.

Figure 4-12 Select a Dimension Window

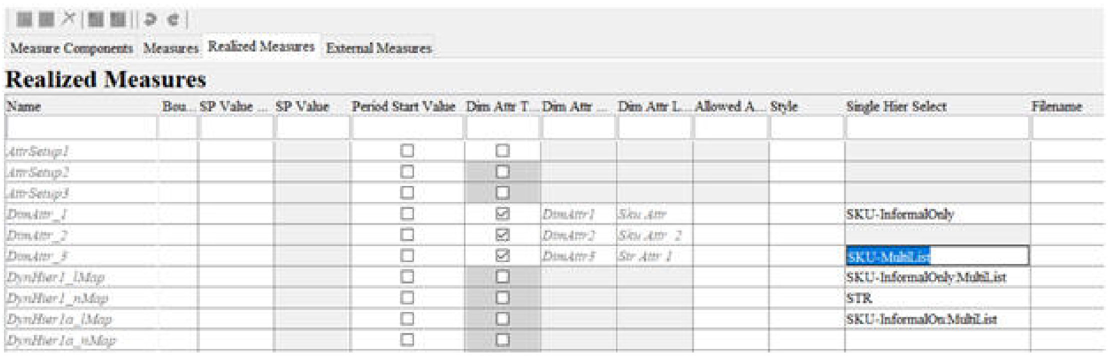


Select a Dimension.

Hier PROD
Dim SKU
Informal Only ☐
Multi List ☑

For the Single Hier Select measures that have both Agg State and Base State properties set to READ, the pop-up dialog box contains an additional Multi List check box. Users can select the Multi List check box to enable the Multi Item Display on RPASCE UI, so users can see a concatenated list of positions that meet different criteria for a measure (For example, alert or promotion) so that users can see which positions have alerts or offers and what details that apply to them.

Figure 4-13 Measure Definition Table



Measure Components | Measures | Realized Measures | External Measures

Realized Measures

Name	Bou...	SP Value ...	SP Value	Period Start Value	Dim Attr T...	Dim Attr ...	Dim Attr L...	Allowed A...	Style	Single Hier Select	Filename
AttrSetup1				☐	☐						
AttrSetup2				☐	☐						
AttrSetup3				☐	☐						
DimAttr_1				☐	☑	DimAttr1	Sku Attr			SKU-InformalOnly	
DimAttr_2				☐	☑	DimAttr2	Sku Attr 2				
DimAttr_3				☐	☑	DimAttr3	Str Attr 1			SKU-MultiList	
DynHier1_lMap				☐	☐					SKU-InformalOnly MultiList	
DynHier1_nMap				☐	☐					STR	
DynHier1a_lMap				☐	☐					SKU-InformalOn MultiList	
DynHier1a_nMap				☐	☐						

Filename (read only)

Measures included into the Data Interface Manager have a filename specified. This field displays the value, if one exists, for the filename of the measure. See the section on the Data Interface Tool for details on data interface properties. This property cannot be modified and is displayed for diagnostic purposes.

Load Intx (read only)

Measures included in the Data Interface Manager have a load intersection [Load Intx] specified. This field displays the value, if one exists, for the load intersection of the measure. See the section on the Data Interface Tool for details on data interface properties. This property cannot be modified and is displayed for diagnostic purposes.

Clear Intx (read only)

This field is deprecated. Measures included in the Data Interface Manager may have a clear intersection [Clear Intx] specified. This field displays the value, if one exists, for the clear intersection of the measure. See the section on the Data Interface Tool for details on data interface properties. This property cannot be modified and is displayed for diagnostic purposes.

Start (read only)

This field is deprecated. Measures included in the Data Interface Manager will have a start position specified. This field displays the value, if one exists, for the start position of the measure. See the section on the Data Interface Tool for details on data interface properties. This property cannot be modified and is displayed for diagnostic purposes.

Width (read only)

This field is deprecated. Measures included in the Data Interface Manager will have a width specified. This field displays the value, if one exists, for the width of the measure. See the section on the Data Interface Tool for details on data interface properties. This property cannot be modified and is displayed for diagnostic purposes.

Load Agg (read only)

Measures included in the Data Interface Manager will have a load aggregation method [Load Agg] specified. This field displays the value, if one exists, for the load aggregation method of the measure. See the section on the Data Interface Tool for details on data interface properties. This property cannot be modified and is displayed for diagnostic purposes.

Materialized (read only)

Certain measures may be registered as display only measures in order to improve performance within a workbook. This marking is done automatically at the time of installation. If the measure is marked as display only, that fact will be reflected in this field. This property cannot be modified and is displayed only for diagnostic purposes.

Creator (read only)

Certain extensions make use of plug-ins to the RPASCE Configuration Tools to automatically generate configuration content. This field displays the creator of the given measure. The value user represents content generated by a user of the Configuration Tools. A different value represents content generated by a plug-in. This property cannot be modified and is displayed for diagnostic purposes only.

Signature (read only)

The signature of a measure is used to resolve ambiguity that may result from overriding the name property of a measure. This field contains the value of the measures signature property. The contents of this field are created and maintained automatically by the Measure Manager. This property cannot be modified and is displayed only for diagnostic purposes.

Shared Fact Name (read only)

Measures included in the Data Interface Manager can have a Shared Fact Name specified if these measures are shared measures. This field displays the name of the common fact shared among measures from multiple applications. See the section on the Data Interface Tool for details on data interface properties. This property cannot be modified and is displayed for diagnostic purposes.

Shared Fact Base Intx (read only)

Measures included in the Data Interface Manager can have a Shared Fact Base Intx specified if these measures are shared measures. This field displays the base intersection of the fact shared among multiple measures. See the section on the Data Interface Tool for details on data interface properties. This property cannot be modified and is displayed for diagnostic purposes.

Report Category

The report category field is used for data visualization and reporting with Oracle Analytic Server. Users select measures that will be potentially reported on, with each measure assigned a reporting category. The report category can be set at the Measure

Component level, Measure, or Realized Measure level. If configured at the component levels, the values will be concatenated into the Report Category field of the final realized measure. The Report Category value of each realized measure can be overridden by directly editing in the Measure or Realized Measure field.

Requirements for Report Category

- The value of the Report Category field must follow the Measure Naming Conventions, such as always starting with a letter and no special characters allowed except "_". The value length must not exceed 21 characters because the value is used to construct the view name later during the OAT Task.
- The measure must have a storage database.
- Measures of the same report category must have the same measure base intersection; if the measures are shared measures, the shared fact base intersection must be the same.

If no storage database is specified while the report category is being configured, the report category will be overridden to null during the integration configuration generation to prevent later OAT task failure when creating schema objects for data visualization.

If the same category has different fact base intersection, a hard error will be generated during integration configuration generation to halt the process. Otherwise, the OAT task will fail during the creation of schema objects for data visualization.

Source Fact (read only)

Measures included in the Data Interface Manager can have a Source Fact flag set to True to indicate that the data for these measures can be updated by the ORDS Put service. See the section on the Data Interface Tool for details on data interface properties. This property cannot be modified and is displayed for diagnostic purposes.

Customer Managed

The measure property customer managed has a value of true or false. By default, this property has a value of false. Users can explicitly configure this property to true to indicate that this measure is writable in the PL/SQL procedures added by customers.

Customer managed measures are writeable in the PL/SQL procedures added by customers. All other measures in the configuration are read-only from the customer's PL/SQL.

The following are qualifying criteria for customer-managed measures:

- Must have the db field specified.
- Cannot be shared measures.
- Should not be used in cycle groups or the left-hand side of special expressions, as these measures must be in the same fact group. Making part of these measures as customer-managed measures/facts will split this fact group since customer-managed measures are assigned to a separate fact group.

Edit Components

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Measures**. The Measure Manager window opens in the workspace.

Move Components

1. From the Measure Manager navigation tree, select the component to be moved.
2. Drag the component to the new location and release it.

> **Note:**
>
> A minor component cannot be moved to a different major component.
>
> The configuration administrator cannot move a component so that it would be a descendent of another component that is used in the specification of a realized measure.

Push Components Down

1. From the Measure Manager navigation tree, select the component to be pushed down.
2. Right-click in the Measure Definition menu and select **Push Down**.

> **Note:**
>
> The component is pushed down one level in the component hierarchy, and a new component is created to take the place of the pushed down component.

> **Note:**
>
> A major component cannot be pushed down.

Pull Components Up

1. From the Measure Manager navigation tree, select the component to be pulled up.
2. Right-click and select **Pull Up**. The component is pulled up one level in the component hierarchy.

> **Note:**
>
> A minor component cannot be pulled up to become a major component, nor can a major component be pulled up.

Display or Hide Components

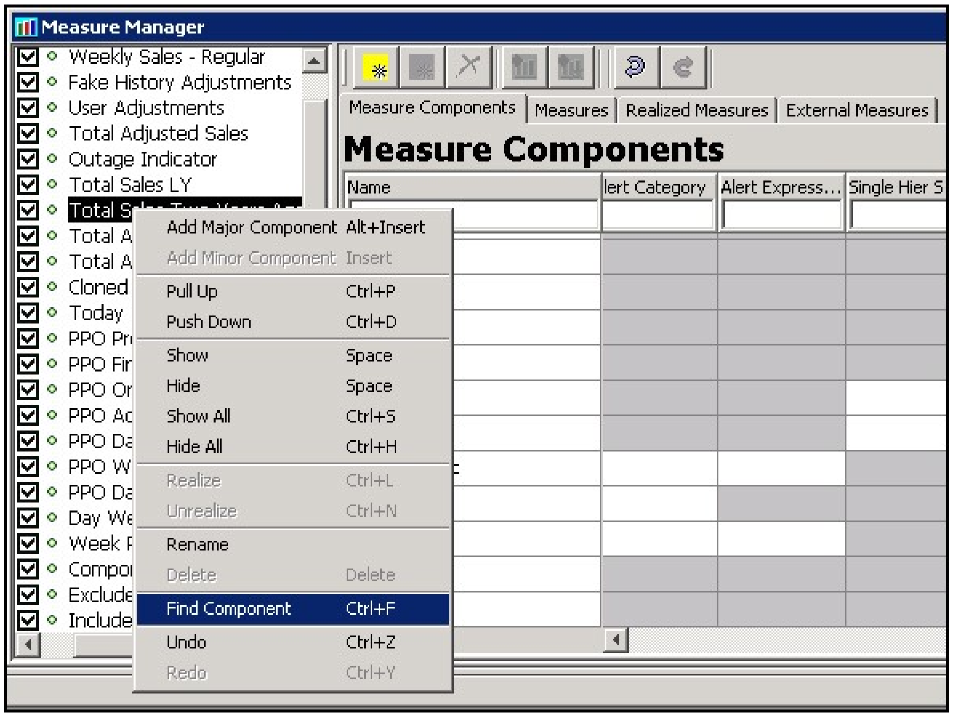
- To display information about a component, select the check box next to the component name.
- To hide information about a component, clear the check box next to the component name.

> **Note:**
>
> Selecting or clearing a check box for a major or minor component causes the check boxes for all minor components underneath it to be selected or cleared. This check box is also used to enable a component so it becomes active when measures are generated.

Find a Component

1. Right-click in the Measure Manager navigation tree and select **Find Component**. The Input window opens.

Figure 4-14 Find Component Menu Option



2. Type in the name of the desired component and click **OK**. The tree will scroll to bring the specified component into view.
3. Select the desired component.

Note:

The full name (case sensitive) of the component is required in order to find it. If there is no component with the exact name entered, the tree will not scroll.

Rename a Component

Navigate: From the Measure Manager navigation tree, select the component to be renamed.

Choose one of the following methods:

- Right-click in the Measure Manager navigation tree and select **Rename**. Type the new name for the component.
- Double-click the component and type the new name.
- Select and change the name of the component in the Measure Component tab.

Note:

Changing the name of a component results in a change in the name of any measure that inherits from the component unless the measure has overridden the name property.

Remove Components

Navigate: From the Measure Manager navigation tree, select the component to be removed.

Choose one of the following methods:

- Click **Remove Component**.
- Press **Delete**.
- Right-click in the Measure Manager navigation tree and select **Delete**.
- The component is removed from the solution.

Note:

It is not possible to remove a component that is used (or that has a descendent component that is used) in the specification of a measure.

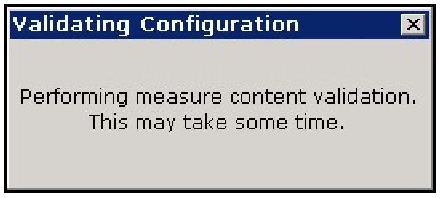
Measure Validation within the Measure Manager

The Measure Manager performs large amounts of validation on the properties of the measures that are created within it. Much of this validation involves dependencies of one property of a measure upon another property of the measure. Some validation involves dependencies of a property of a measure upon the hierarchies that are defined in the Hierarchy Tool. These forms of validation are performed automatically as edits are made in the Measure Manager.

In addition, the validity of rules, rule groups, and workbooks depends on the properties of the measures that they contain. Validation of the measure content of the rules, rule groups, and workbooks of a large solution can take a significant amount of time.

To facilitate the configuration process, this second form of validation does not occur as edits are made in the Measure Manager. Instead, this validation is deferred until Measure Manager is closed by selecting a different tool/option from the Configuration Components pane. Then, a window briefly displays to indicate the validation process is running.

Figure 4-15 Validation Configuration Window



Performing measure content validation.
This may take some time.

When the full suite of solution level measure content validations is complete, the window is no longer displayed and the selected tool is activated.

Note:

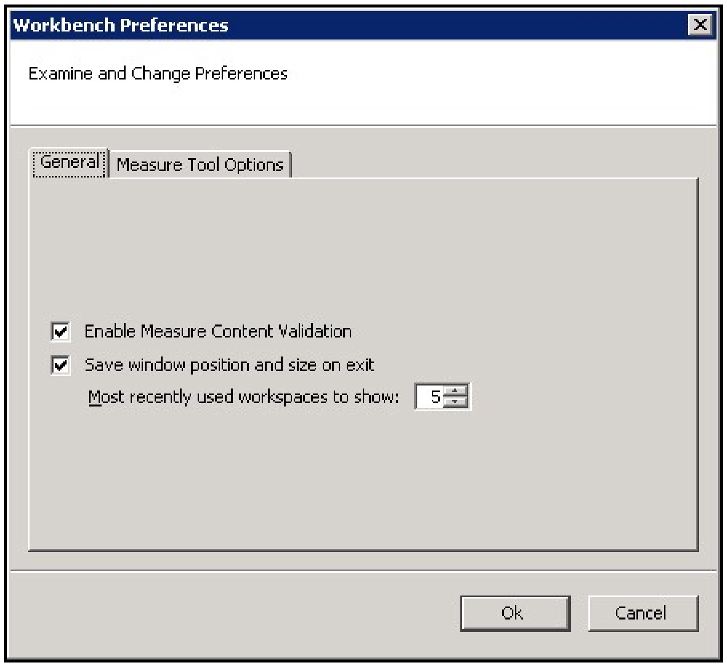
For fast performing validations, a window may only display for a few seconds. If the validation process is lengthy, the window displays for a considerably longer time.

Disabling Measure Content Validation

For some cases you may disable the full validation (for instance, when performing a number of changes to measure properties that require switching back and forth between multiple tools). Validation can then be manually initiated from the Rule Definition window by selecting **Perform measure content validation** from the Rule Definition toolbar. Complete the following procedure to turn off real-time validation.

1. Select **File** and then **Tools Preferences**. The Workbench Preferences window opens.

Figure 4-16 Workbench Preferences – General Tab



☑ Enable Measure Content Validation
☑ Save window position and size on exit
Most recently used workspaces to show: 5

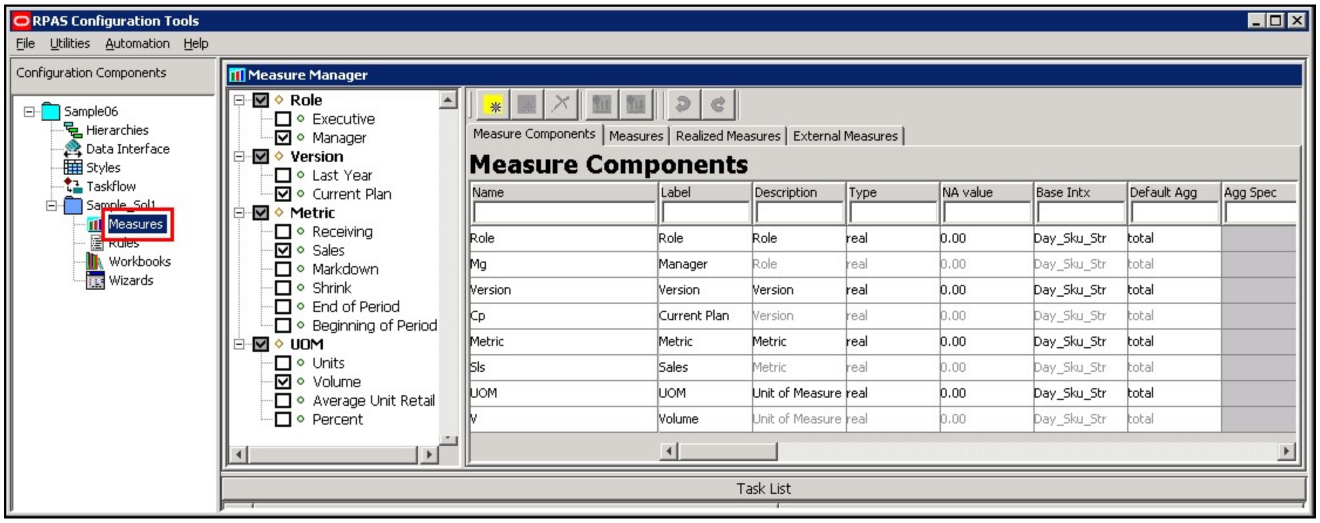
2. Clear the **Enable Measure Content Validation** check box.
3. Click **OK**.

Working with Measures

Measure Manager Overview

The Measure Manager allows you to create and name measures by selecting major and minor components that are already defined. By default, the measures inherit the properties that are defined for the components. To create a measure, select the components that will be used to construct measures. The Measure Manager will generate measures for all of the combinations of selected components. This saves you from the tedious task of manually creating all required measures.

Figure 4-17   Measure Manager Components



- ☑ Role
  - ☐ Executive
  - ☑ Manager
- ☑ Version
  - ☐ Last Year
  - ☑ Current Plan
- ☑ Metric
  - ☐ Receiving
  - ☑ Sales
  - ☐ Markdown
  - ☐ Shrink
  - ☐ End of Period
  - ☐ Beginning of Period
- ☑ UOM
  - ☐ Units
  - ☑ Volume
  - ☐ Average Unit Retail
  - ☐ Percent

Measure Components

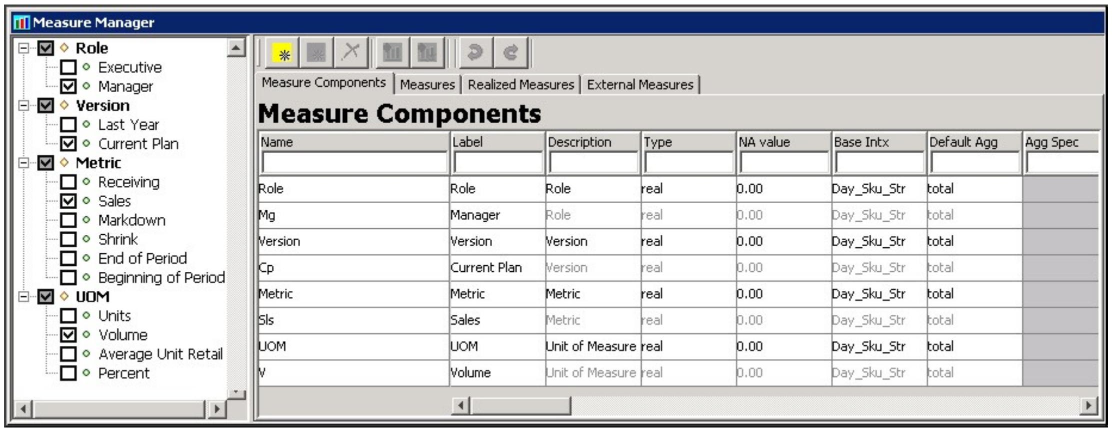
Name	Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spec
Role	Role	Role	real	0.00	Day_Sku_Str	total	
Mg	Manager	Role	real	0.00	Day_Sku_Str	total	
Version	Version	Version	real	0.00	Day_Sku_Str	total	
Cp	Current Plan	Version	real	0.00	Day_Sku_Str	total	
Metric	Metric	Metric	real	0.00	Day_Sku_Str	total	
Sls	Sales	Metric	real	0.00	Day_Sku_Str	total	
UOM	UOM	Unit of Measure	real	0.00	Day_Sku_Str	total	
V	Volume	Unit of Measure	real	0.00	Day_Sku_Str	total	

You may then override the properties for individual measures by entering them the same way as on the Measure Components tab. Once properties are overridden at the measure level, changes made at the component level will no longer spread down to that measure as it will retain the overridden value. An overridden value can also be restored to its inherited value or you may override an inherited value to be unspecified.

When a measure is auto-generated by the Measure Manager, it cannot be edited or used in any other configuration component such as rules and workbooks until it is realized. A measure does not need to be realized if it is not going to be used.

The **Measures** tab displays all auto-generated measures for the selected components.

Figure 4-18   Measure Manager Measures Tab



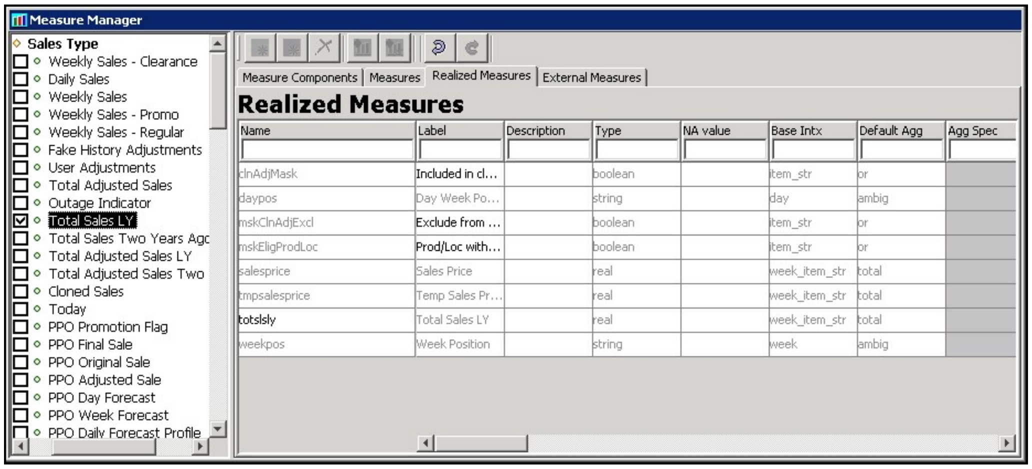
- ☑ Role
  - ☐ Executive
  - ☑ Manager
- ☑ Version
  - ☐ Last Year
  - ☑ Current Plan
- ☑ Metric
  - ☐ Receiving
  - ☑ Sales
  - ☐ Markdown
  - ☐ Shrink
  - ☐ End of Period
  - ☐ Beginning of Period
- ☑ UOM
  - ☐ Units
  - ☑ Volume
  - ☐ Average Unit Retail
  - ☐ Percent

Measure Components

Name	Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spec
Role	Role	Role	real	0.00	Day_Sku_Str	total	
Mg	Manager	Role	real	0.00	Day_Sku_Str	total	
Version	Version	Version	real	0.00	Day_Sku_Str	total	
Cp	Current Plan	Version	real	0.00	Day_Sku_Str	total	
Metric	Metric	Metric	real	0.00	Day_Sku_Str	total	
Sls	Sales	Metric	real	0.00	Day_Sku_Str	total	
UOM	UOM	Unit of Measure	real	0.00	Day_Sku_Str	total	
V	Volume	Unit of Measure	real	0.00	Day_Sku_Str	total	

The **Realized Measures** tab only displays those measures that are realized.

Figure 4-19   Measure Manager Realized Measures Tab



Measure Manager

- Sales Type
- [ ] Weekly Sales - Clearance
- [ ] Daily Sales
- [ ] Weekly Sales
- [ ] Weekly Sales - Promo
- [ ] Weekly Sales - Regular
- [ ] Fake History Adjustments
- [ ] User Adjustments
- [ ] Total Adjusted Sales
- [ ] Outage Indicator
- [x] Total Sales LY
- [ ] Total Sales Two Years Agc
- [ ] Total Adjusted Sales LY
- [ ] Total Adjusted Sales Two
- [ ] Cloned Sales
- [ ] Today
- [ ] PPO Promotion Flag
- [ ] PPO Final Sale
- [ ] PPO Original Sale
- [ ] PPO Adjusted Sale
- [ ] PPO Day Forecast
- [ ] PPO Week Forecast
- [ ] PPO Daily Forecast Profile

Measure Components | Measures | Realized Measures | External Measures

Realized Measures

Name	Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spec
clnAdjMask	Included in cl...		boolean		item_str	or	
daypos	Day Week Po...		string		day	ambig	
mskClnAdjExcl	Exclude from ...		boolean		item_str	or	
mskEligProdLoc	Prod/Loc with...		boolean		item_str	or	
salesprice	Sales Price		real		week_item_str	total	
tmpsalesprice	Temp Sales Pr...		real		week_item_str	total	
totslsly	Total Sales LY		real		week_item_str	total	
weekpos	Week Position		string		week	ambig	

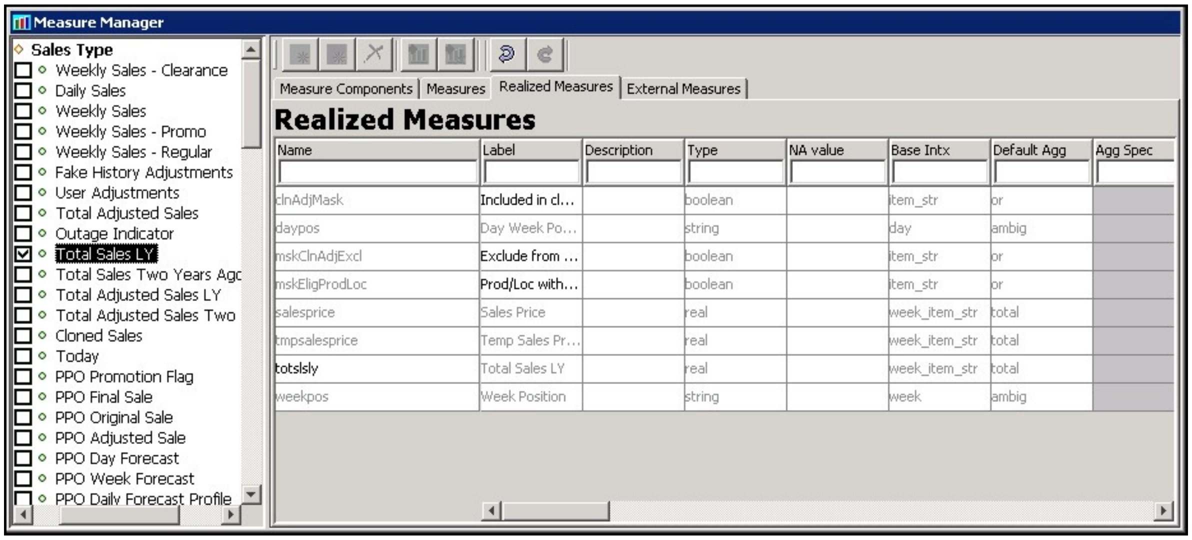
Realized and Unrealized Measures

Complete the following steps to access the Measure Manager window - Measures Tab.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Measures**. The Measure Manager window opens in the workspace.

1. Select the Measures tab.

Figure 4-20   Measure Manager Measures Tab



Measure Manager

- Sales Type
- [ ] Weekly Sales - Clearance
- [ ] Daily Sales
- [ ] Weekly Sales
- [ ] Weekly Sales - Promo
- [ ] Weekly Sales - Regular
- [ ] Fake History Adjustments
- [ ] User Adjustments
- [ ] Total Adjusted Sales
- [ ] Outage Indicator
- [x] Total Sales LY
- [ ] Total Sales Two Years Agc
- [ ] Total Adjusted Sales LY
- [ ] Total Adjusted Sales Two
- [ ] Cloned Sales
- [ ] Today
- [ ] PPO Promotion Flag
- [ ] PPO Final Sale
- [ ] PPO Original Sale
- [ ] PPO Adjusted Sale
- [ ] PPO Day Forecast
- [ ] PPO Week Forecast
- [ ] PPO Daily Forecast Profile

Measure Components | Measures | Realized Measures | External Measures

Realized Measures

Name	Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spec
clnAdjMask	Included in cl...		boolean		item_str	or	
daypos	Day Week Po...		string		day	ambig	
mskClnAdjExcl	Exclude from ...		boolean		item_str	or	
mskEligProdLoc	Prod/Loc with...		boolean		item_str	or	
salesprice	Sales Price		real		week_item_str	total	
tmpsalesprice	Temp Sales Pr...		real		week_item_str	total	
totslsly	Total Sales LY		real		week_item_str	total	
weekpos	Week Position		string		week	ambig	

2. In the components tree, select the check boxes for the components that will be filtered. The Measure Manager will show measures using all the combinations of selected components. This process filters the list of prototype measures that are shown from all combinations of components to the combinations of components that have been selected. It uses those components to determine which prototype measures to show.

Realizing a Measure

1. Select the components that are used for the measures to be realized.
2. Select the check box in the Realized column for each auto-generated measure to be realized.

Unrealizing a Measure

1. Select the components that are used for the measures to be unrealized.
2. Clear the check box in the Realized column for each auto-generated measure to be unrealized.

Rename a Measure

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Measures**. The Measure Manager window opens in the workspace.

1. Select the components that are used in the measures to be renamed.
2. In the **Measures** tab, or the **Realized Measures** tab, click the name of the measure that is to be renamed.
3. Type the new name for the measure.

> **Note:**
>
> The measure must be realized before it can be renamed.

Show All Measures

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Measures**. The Measure Manager window opens in the workspace.

Right-click in the Measure Manager window and select **Show All**.

> **Note:**
>
> Due to memory constraints when working with very large numbers of components, all auto-generated measures may not be displayed. In this case, the configuration administrator will receive an error message indicating that some measure components should be cleared.

Hide Measures by Component

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Measures**. The Measure Manager window opens in the workspace.

1. In the Measure Manager window, select the Measures tab.

2. Choose one of the following methods:
   - Clear the check boxes next to the components to hide.
   - Select the component used in the measures to hide.
3. Right-click in the Measure Manager navigation tree and select **Hide** or press the spacebar.

Hide All Measures

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Measures**. The Measure Manager window opens in the workspace.

1. In the Measure Manager window, select the **Measures** tab.
2. Right-click the Measure Manager navigation tree, and select **Hide All.**

Sort Measures by Property Value

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Measures**. The Measure Manager window opens in the workspace.

In the Realized Measures tab, measures can be sorted by property value.

1. Select the Realized Measures tab.
2. Hold down the control key (Ctrl) and click in the filter field at the top of the table for the property of the measures to be sorted.

   The measures are sorted in alphabetical order according to the value of the property.

Filter Measures by Property Value

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Measures**. The Measure Manager window opens in the workspace.

In the Realized Measure tabs, the configuration administrator can filter measures by property value.

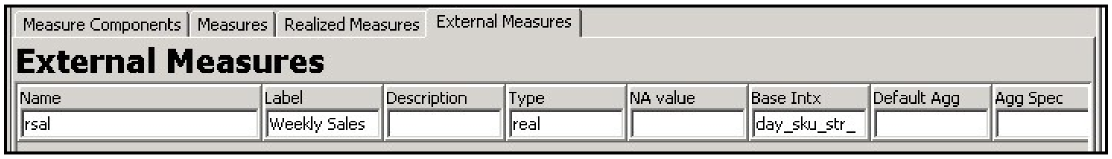
1. In the Realized Measures tab, click in the filter field at the top of the table for the property of the measures to be filtered.
2. Enter the value on which ti filter the measures.

> **Note:**
>
> This field is case sensitive.

External Measures

Under most circumstances, measures exist only within the solution in which they are defined. When working with a project with multiple solutions, it is sometimes desirable to make use of a measure defined in a different solution. The Measure Manager allows the import of a measure defined in a different solution (but not a different project) into the current solution. These measures (called external measures) then become visible in the External Measures tab of the Measure Manager. Within this tab, it is possible to modify certain measure properties so that the use of the measure in the Solution into which it has been imported will differ from its use in the Solution in which it was originally defined. For example, the configuration administrator may want to modify a writable measure so that it is read-only in the solution into which it is imported.

Figure 4-21 Measure Manager External Measures Tab



External Measures

Name	Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spec
rsal	Weekly Sales		real		day_sku_str_		

Within the Measure Selectors present in the Rule and Workbook tools and the Expression Builder, there is a check box named Include External Measures. When this check box is selected, the Measure Selector will include those measures that were imported into the solution in addition to those that are present due to component selections in the Measure Selector.

The following properties may be overridden for an external measure:

- Label
- Description
- Base State
- Agg State
- UI Type
- Range

Import a Measure

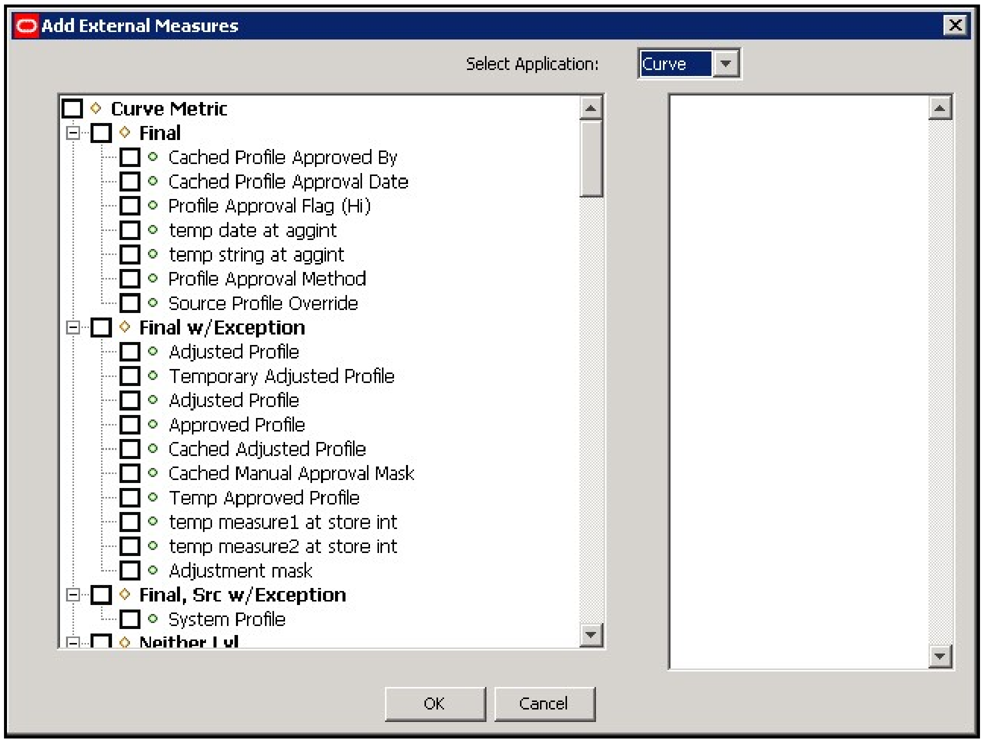
Note:

You may only import a measure into a solution for a project that has at least one other solution.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Measures**. The Measure Manager window opens in the workspace.

1. Select the External Measures tab, and click **Import Measure** on the toolbar. The Figure 4-22 opens.

Figure 4-22 Add External Measures Window



Add External Measures

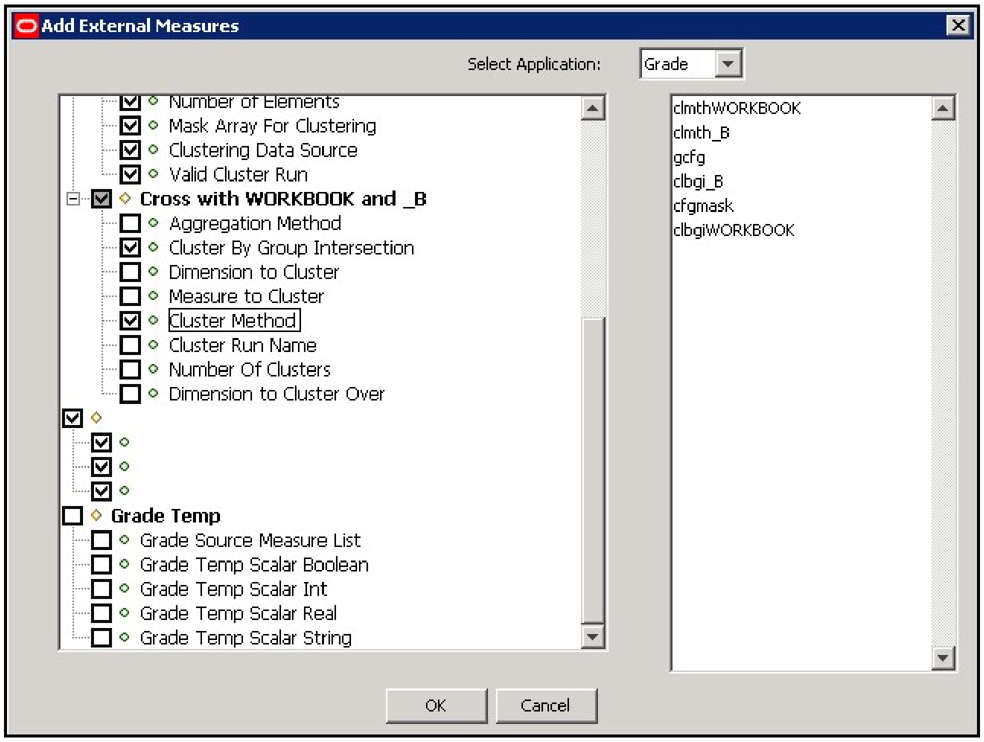
Select Application: Curve

- ☐ Curve Metric
  - ☐ Final
    - ☐ Cached Profile Approved By
    - ☐ Cached Profile Approval Date
    - ☐ Profile Approval Flag (Hi)
    - ☐ temp date at aggint
    - ☐ temp string at aggint
    - ☐ Profile Approval Method
    - ☐ Source Profile Override
  - ☐ Final w/Exception
    - ☐ Adjusted Profile
    - ☐ Temporary Adjusted Profile
    - ☐ Adjusted Profile
    - ☐ Approved Profile
    - ☐ Cached Adjusted Profile
    - ☐ Cached Manual Approval Mask
    - ☐ Temp Approved Profile
    - ☐ temp measure1 at store int
    - ☐ temp measure2 at store int
    - ☐ Adjustment mask
  - ☐ Final, Src w/Exception
    - ☐ System Profile
  - ☐ Neither Lvl

OK Cancel

2. In the Add External Measures window, select the solution in which the desired measure is defined.
3. Use the measure selector (left-hand side) to select the measures to import.

Figure 4-23 Selecting Measures to Be Imported



Add External Measures

Select Application: Grade

- ☑ Number of Elements
- ☑ Mask Array For Clustering
- ☑ Clustering Data Source
- ☑ Valid Cluster Run
- ☑ Cross with WORKBOOK and _B
  - ☐ Aggregation Method
  - ☑ Cluster By Group Intersection
  - ☐ Dimension to Cluster
  - ☐ Measure to Cluster
  - ☑ Cluster Method
  - ☐ Cluster Run Name
  - ☐ Number Of Clusters
  - ☐ Dimension to Cluster Over
- ☑
  - ☑
  - ☑
  - ☑
- ☐ Grade Temp
  - ☐ Grade Source Measure List
  - ☐ Grade Temp Scalar Boolean
  - ☐ Grade Temp Scalar Int
  - ☐ Grade Temp Scalar Real
  - ☐ Grade Temp Scalar String

clmthWORKBOOK
clmth_B
gcfg
clbgi_B
cfgmask
clbgiWORKBOOK

OK Cancel

External measures that are already imported display in bold.

4. From the right-hand list, select the measure to import and click **OK**. The selected measures display in the External Measure tab.

Remove an Imported Measure from a Solution

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Measures**. The Measure Manager window opens in the workspace.

1. Click the **External Measures** tab, and select the measure to remove.
2. Click **Remove Imported Measure** from the Measure Manager toolbar, or right-click and select **Remove Import**. The selected imported measure is removed from the External Measures tabs.

Rule Sets

A rule set is a collection of rule groups. It is used as a placeholder for containing rule groups, which makes the visual display of rules easier. A workbook uses the rule groups specified in one rule set.

Note:

A rule set is a tools concept only. It does not appear in the configured solution.

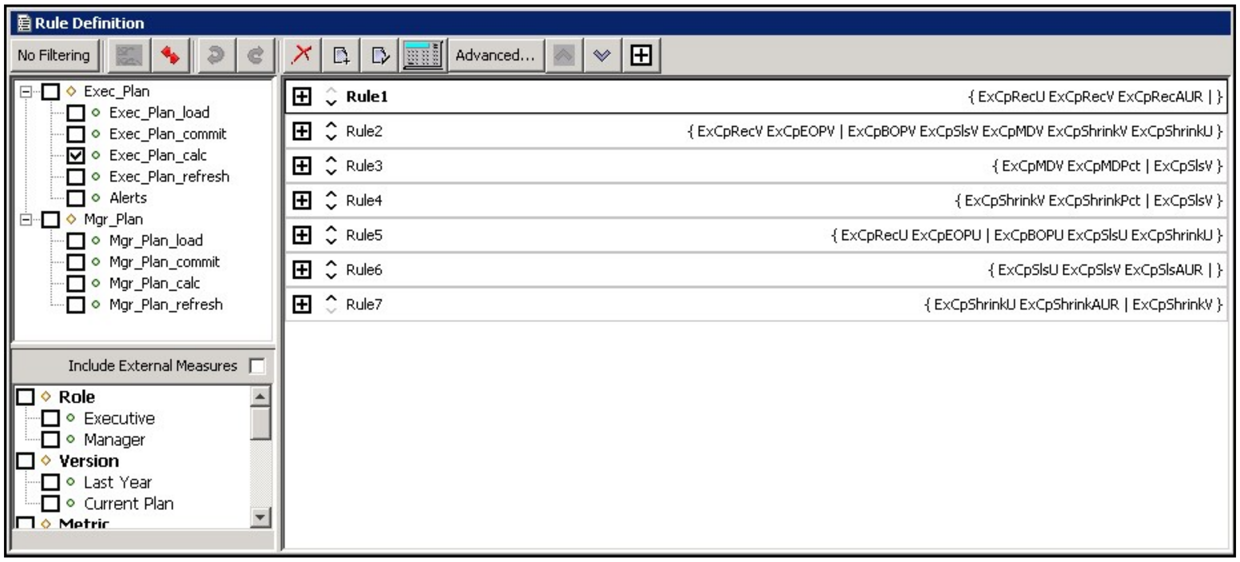
A rule set is created along with the following default rule groups: load, commit, calc, and refresh.

The rule group names are prefixed with the name of the rule set followed by an underscore. After the default rule groups are created, the configuration administrator can create additional rule groups as necessary. The configuration administrator can also rename the rule groups that were automatically generated, but these rule groups cannot be deleted.

Create a Rule Set

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Rule Definition window opens in the workspace.

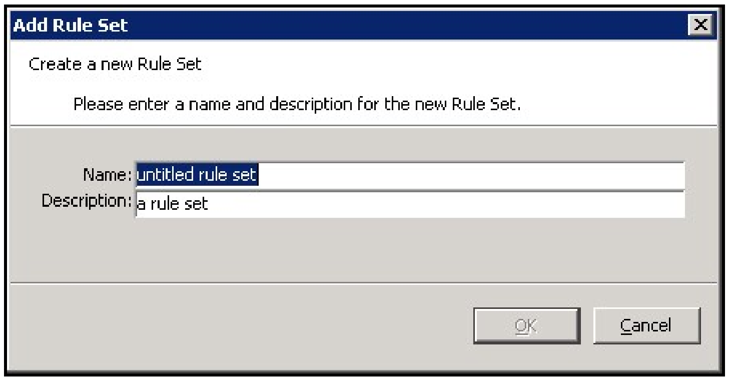
Figure 4-24 Rule Definition Window



Rule1	{ ExCpRecU ExCpRecV ExCpRecAUR | }
Rule2	{ ExCpRecV ExCpEOPV | ExCpBOPV ExCpSlsV ExCpMDV ExCpShrinkV ExCpShrinkU }
Rule3	{ ExCpMDV ExCpMDPct | ExCpSlsV }
Rule4	{ ExCpShrinkV ExCpShrinkPct | ExCpSlsV }
Rule5	{ ExCpRecU ExCpEOPU | ExCpBOPU ExCpSlsU ExCpShrinkU }
Rule6	{ ExCpSlsU ExCpSlsV ExCpSlsAUR | }
Rule7	{ ExCpShrinkU ExCpShrinkAUR | ExCpShrinkV }

1. From the toolbar, click **New** and select **Rule Set**, or select **Create/Rule Set** from the right-click menu. The Figure 4-25 opens.

Figure 4-25 Add Rule Set Window



Create a new Rule Set

Please enter a name and description for the new Rule Set.

Name: untitled rule set
Description: a rule set

2. Complete the following:
   a. In the **Name** field, enter the name of the rule set.

   > **Note:**
   >
   > A rule set name can be a maximum of ten alphanumeric or underscore characters. It must not have a name that is the same as any other rule set that exists in the project.

   b. In the **Description** field, enter a description of the rule set.
   c. Click **OK** to save any changes and close the window. The rule set displays in the Rules navigation tree.

Delete a Rule Set

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Rule Definition window opens in the workspace.

1. In the Rule Definition window, select the rule set to be deleted.
2. Choose one of the following methods:
   a. From the toolbar, click **Delete**, and select **Delete Rule Set**.
   b. Right-click in the Rule Definition window select **Delete/Remove**, and select **Delete Rule Set** from the list.

Note:

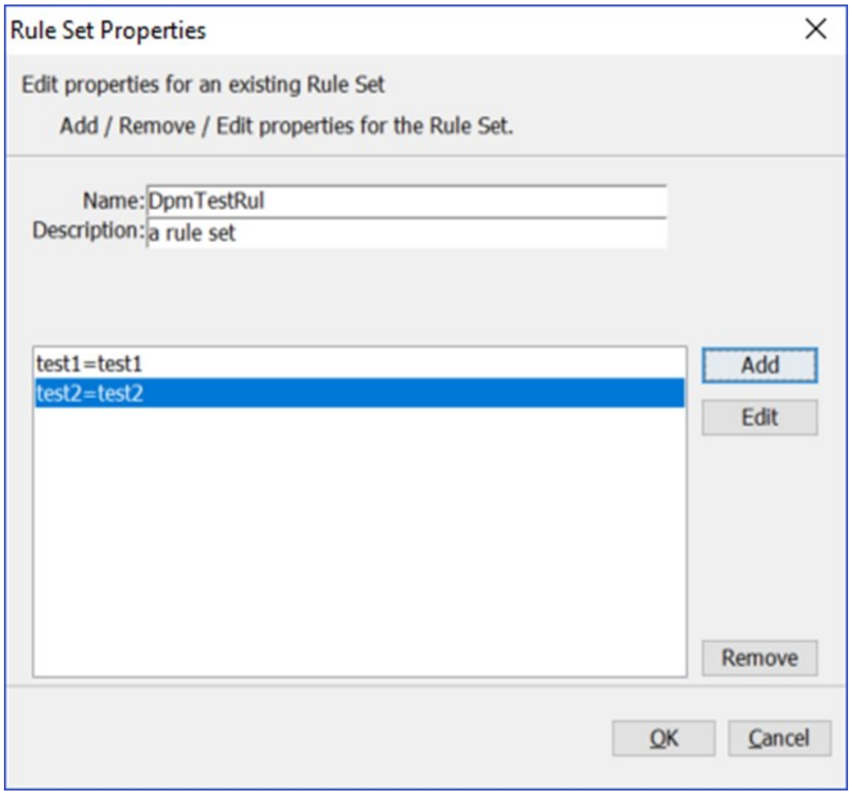
When a rule set is deleted, any rules that were used in rule groups in that rule set will still be in the rule pool, but they will be unused. The rules will be permanently lost when the project is closed unless they are used in another rule group.

Edit Rule Set Properties

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Rule Definition window opens in the workspace.

1. In the Rule Definition window, select the rule set, right click, select E**dit Properties**, then select **Rule Set**. The Rule Set Properties Window opens in the workspace.

Figure 4-26 Rule Set Properties Window



Rule Set Properties

Edit properties for an existing Rule Set

Add / Remove / Edit properties for the Rule Set.

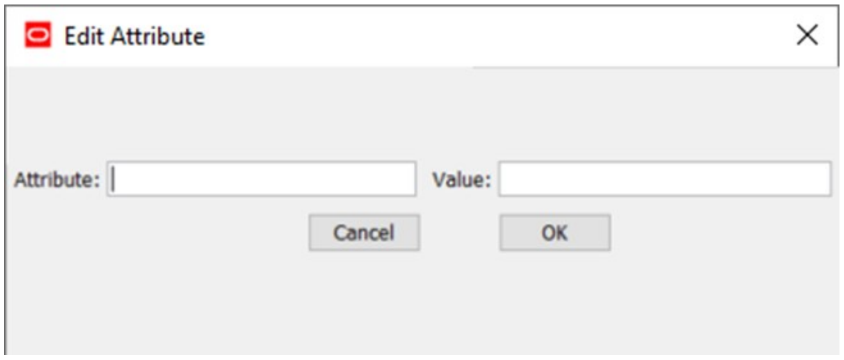
Name: DpmTestRul

Description: a rule set

test1=test1
test2=test2

2. In the Rule Set Properties window, Click **Add** to open the Edit Attribute window in order to add a new rule set attribute.

Figure 4-27 Rule Set Edit Attribute Window



3. To edit an existing rule set attribute, select the attribute/value pair and then click **Edit.** The Edit Attribute window opens in the workspace. The window is pre-populated with the selected rule set attribute and value that the users can edit.
4. To remove an existing rule set attribute, select the attribute/value pair and then click **Remove**.

Rule Groups

In RPASCE, a rule group is an ordered collection of rules that are treated as a unit by the calculation engine with the integrity of all the rules in the rule group being maintained together.

Rules within a rule group are given a priority. The calculation engine uses this to select a calculation path that follows business priorities. It does this by using rule priorities to determine which rule to enforce when there is a choice to be made. Although there may only be one active rule group at any time, multiple rule groups can be defined to satisfy different calculation requirements.

Types of Rule Groups

Rule groups may be one of four different types:

- **Load** – The RPASCE application automatically uses the load rule group when loading data into a workbook from the PDS.
- **Calculate** – The RPASCE application uses a calculate rule group to apply the effects of user changes to cells. RPASCE supports multiple calculation rule groups. Menu options may be configured to allow for the transition through different calculation rule groups in order to support special processes, such as authorizations. RPASCE ensures a smooth transition from one rule group to another.
- **Refresh** – The RPASCE application uses a refresh rule group to refresh the data from the PDS (for example, to update actuals). Multiple refresh rule groups can be specified and selected by the user.
- **Commit** – The RPASCE application automatically uses the commit rule group when committing data from the workbook to the PDS.

A measure that does not have data in the PDS may be loaded into a workbook by using a rule in the load rule group to calculate it based on other measures that are loaded. Similarly, a measure that exists in a PDS, but not a workbook, may be committed by using a rule in the commit rule group that calculates it from other measures that are in the workbook.

Rule Group Validation

Within a solution, there may be many rules defined, and each rule is validated individually. Rules within a rule group are also validated in the context of all the other rules in that rule group. While a rule may be perfectly valid syntactically, it may not be valid within the context of a particular rule group. In Rule group validation, each rule in a rule group must represent a completely different measure relationship, which means that the following restrictions apply:

- No two rules in a rule group may use exactly the same collection of measures. If such a condition were allowed, the calculation engine would be unable to calculate either of the rules, because they would be dependent upon each other, so neither could be calculated first. This is explicitly validated in the rule tool.
- For similar reasons, a rule normally does not use a collection of measures that is a subset of the collection of measures in another rule. If the measures that are only in the larger rule were all changed, the situation would be equivalent to the previous one. There are circumstances where this technique is valuable. For example, in a load rule group where there may be a rule to load a measure from the PDS, but other rules that include that measure. In this case, this condition is not explicitly validated.

- There must be one (and only one) expression that calculates a recalc measure used in a rule group.
- Other than in the special circumstance of a rule constructed from multiple result functions or procedures, a measure may only be on the left-hand side of one expression in a rule.

> **Note:**
>
> When performing validation on rules in RPASCE Configuration Tools, a proxy sandbox is used that only contains a small portion of information that is present in a the PDS. This proxy sandbox does not contain any information about the positions that exist in various hierarchies and dimensions. Therefore, when performing rules validation, any rule that contains a reference to a position will not be properly validated within Configuration Tools and will be marked as invalid, despite the fact that it will execute in the final PDS.

Multiple Refresh Rule Groups

The Refresh Rule Group updates (supplies new values for) data, which can generally be thought of as being external to the workbook. An example would be actuals:

- If a workbook is built in week 5, the user would have actuals for weeks 1-4.
- In week 6, the user can refresh the actuals to get the actuals for week 5.

The Configuration Tools and RPASCE support the use of multiple refresh rule groups. Within the rule tool, there is the ability to create multiple refresh rule groups within a rule set. These multiple refresh rule groups can then be assigned to a workbook template using the Workbook Designer, and they will be available for selection within the RPASCE Client.

- A workbook contains all of the rule groups in a single rule set, so if multiple refresh rule groups are required in a workbook, they must all be in the same rule set.
- You may consider naming rule groups so the usage of the refresh rule groups are reflected in their names, such as `refresh_all`, `refresh_actuals`, and `refresh_manager`.

Rule Group Transitions

Although only a single rule group may be active at any time, RPASCE supports the transition from one rule group to another, so the active rule group may be changed. The calculation engine ensures the integrity of measure relationships at all times, so this transition process is not merely a case of switching from one rule group to another, because there are no guarantees that the integrity of the rules in the rule group being transitioned into would have been maintained. There are different forms of rule group transitions. When designing rule groups, you must consider the impact of anticipated rule group transitions.

Automatic rule group transitions occur under the following circumstances:

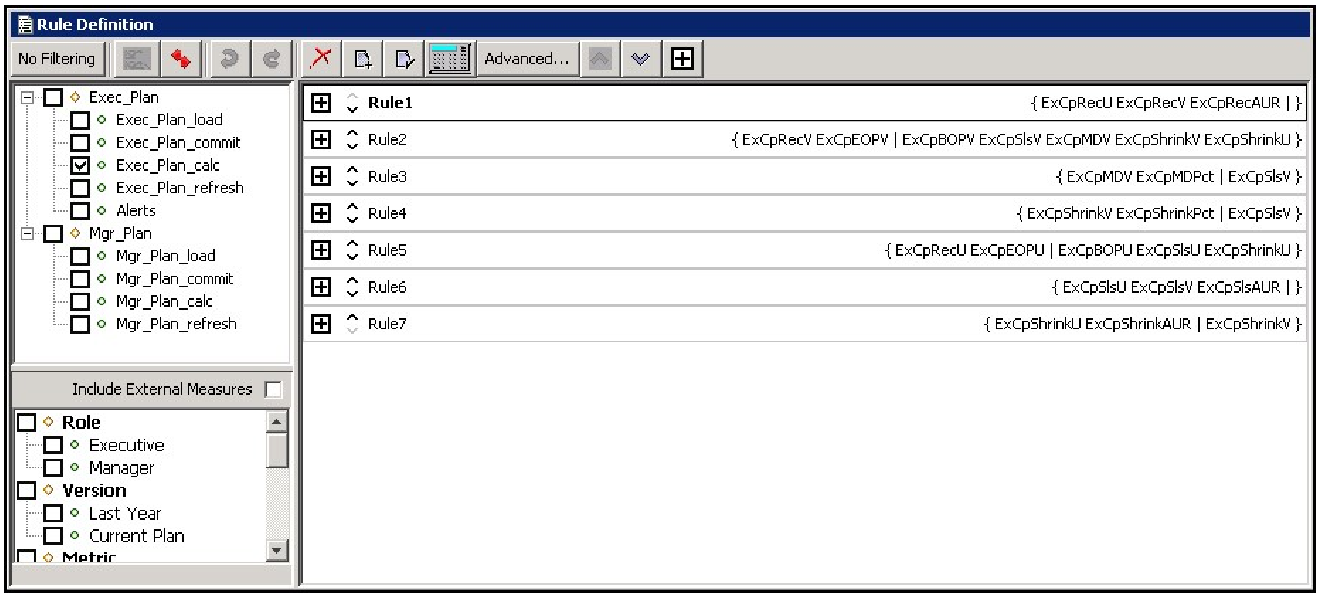
- **On workbook building** – Data is loaded using the load rule group. This typically loads measures by calculating them from the data values held on the PDS using the master modifier but may also calculate other measures that are not explicitly loaded. When the load is complete, the system automatically runs a full transition to the calculate rule group.

- **On data refreshing** – Data refreshing causes some measures to be updated from values held in the PDS. The measures that are affected by the refreshed measures are treated as affected in the calculate rule group, and a normal calculation of that rule group follows. Effectively, data refreshing causes a calculation using the calculate rule group as if the cells that were refreshed were directly changed by the user.
- **On data committing** – There is a full transition from the current calculate rule group to the commit rule group. This typically commits measures by calculating them on the PDS by using the master modifier. There is then a null transition back to the calculation rule group (that is, no transition process is executed since the assumption is that nothing in the workbook has changed), so no transition is required.
- **On executing custom menus** – There is a full transition between each rule group in the custom menu, which is followed by a full transition back to the default calculate rule group. An optimal custom menu transition type is implemented to handle the rule group transition between the custom menu rule groups and the calc rule group of the workbook. The goal is to skip all rules in the calc rule group that are not impacted by the rules in the custom menu action, directly or indirectly, when transition back to the calc rule group from custom menu rule groups.

Create a Rule Group

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

Figure 4-28 Rule Definition Window



Exec_Plan
- Exec_Plan_load
- Exec_Plan_commit
- Exec_Plan_calc
- Exec_Plan_refresh
- Alerts

Mgr_Plan
- Mgr_Plan_load
- Mgr_Plan_commit
- Mgr_Plan_calc
- Mgr_Plan_refresh

Role
- Executive
- Manager

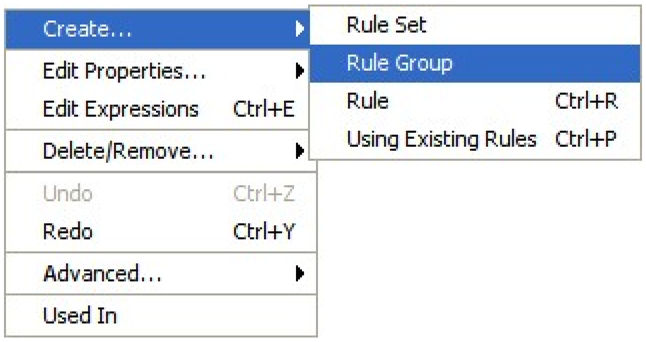
Version
- Last Year
- Current Plan

Metric

Rule1	{ ExCpRecU ExCpRecV ExCpRecAUR | }
Rule2	{ ExCpRecV ExCpEOPV | ExCpBOPV ExCpSlsV ExCpMDV ExCpShrinkV ExCpShrinkU }
Rule3	{ ExCpMDV ExCpMDPct | ExCpSlsV }
Rule4	{ ExCpShrinkV ExCpShrinkPct | ExCpSlsV }
Rule5	{ ExCpRecU ExCpEOPU | ExCpBOPU ExCpSlsU ExCpShrinkU }
Rule6	{ ExCpSlsU ExCpSlsV ExCpSlsAUR | }
Rule7	{ ExCpShrinkU ExCpShrinkAUR | ExCpShrinkV }

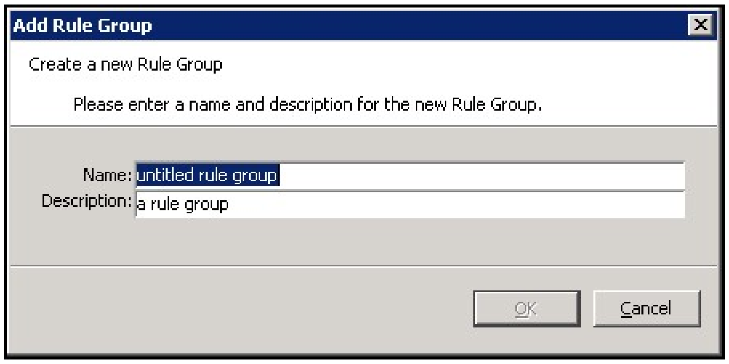
1. In the Rule Definition window, select the rule set to be used to create a new rule group.
2. From the toolbar, click **New** and select **Rule Group**, or select **Create – Rule Group** from the right-click menu.

Figure 4-29 Rule Group Menu Option



The Figure 4-30 opens.

Figure 4-30 Add Rule Group Window


Add Rule Group

Create a new Rule Group

Please enter a name and description for the new Rule Group.

Name: untitled rule group

Description: a rule group

3. Complete the following:
   a. In the Name field, enter the name of the rule group.

   > **Note:**
   >
   > A rule group name can be a maximum of 16 alphanumeric or underscore characters. It must not have a name that is the same as any other rule group that exists in the project.

   b. In the Description field, enter a description of the rule group.
   c. Click **OK** to save any changes and close the window. The new rule group displays in the Rule Definition tree window.

Delete a Rule Group

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select the rule group to be deleted.

> **Note:**
>
> Only user-created rule groups within a rule set can be deleted. You cannot delete the default load, commit, calc, and refresh rule groups. If Delete is selected for one of the default rule groups, it will not be deleted, but all of the rules will be removed from the rule group. In either case, the rules will still exist within the rule pool, but they will be lost when the project is closed if they are not used in another rule group.

2. From the toolbar, click **Delete**, and select **Rule Group**. The group is removed from the Rule navigation tree.

Copy a Rule Group

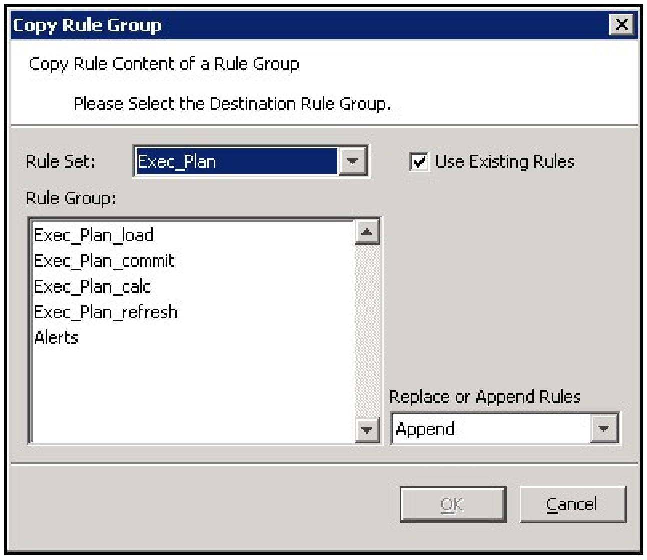
If two rule groups are similar, it may be beneficial to copy one rule group into the other to prevent having to create a rule group from scratch. When copying the rules of a rule group into another rule group, it is possible to specify whether existing rules will be used or copies of the rules will be created. The Use Existing Rules check box defaults to using any existing rules in the rule pool. If this check box is selected, the copy rule group operation will use the same rules that the source rule group has. If this check box is unchecked, the copy rule group operation will create copies of the rules and use those copies for appending to or replacing rules in the destination rule group.

> **Note:**
>
> The rule group to be copied into must already exist. A new rule group cannot be created through this process.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select the rule group to be copied.
2. From the toolbar, click **Advanced**, and select **Copy Rule Group**. The Copy Rule Group window opens.

Figure 4-31 Copy Rule Group Window


Copy Rule Group

Copy Rule Content of a Rule Group

Please Select the Destination Rule Group.

Rule Set: Exec_Plan

[✓] Use Existing Rules

Rule Group:

Exec_Plan_load
Exec_Plan_commit
Exec_Plan_calc
Exec_Plan_refresh
Alerts

Replace or Append Rules: Append

3. Complete the following:
   a. Using the **Rule Set** list, select the destination rule set.
   b. From the **Rule Group** area, select the desired destination rule group.
   c. Using the Replace or Append Rules list, select either:

      Replace – To overwrite all rules that already exist in the destination rule group.

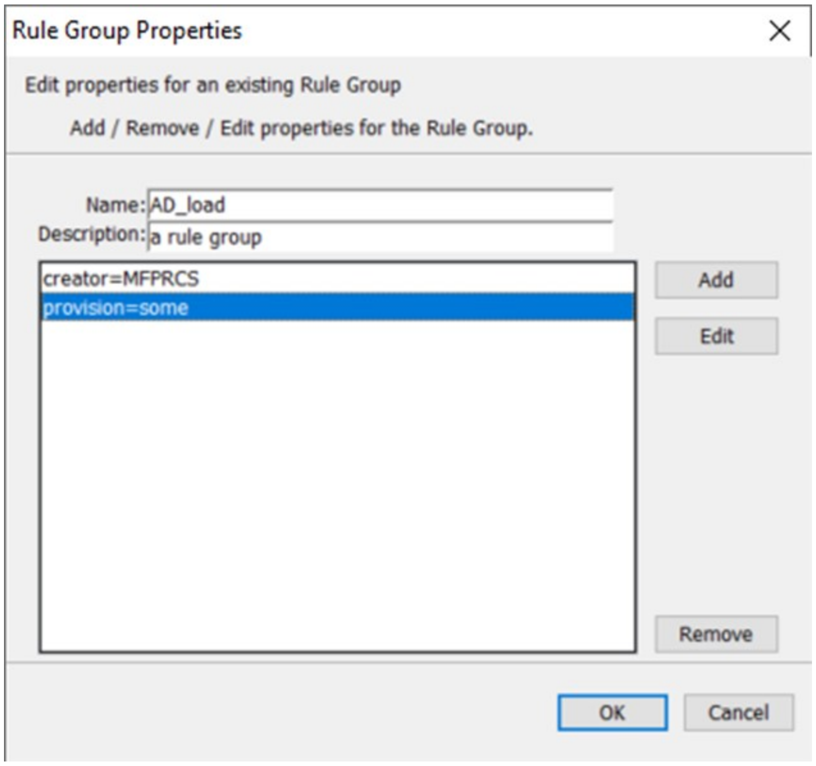
      Append – To add to the rules already in the destination rule group.
   d. Click **OK**. The Confirm Operation window opens.
4. Click **OK**. The rules are copied to the selected rule group.

Edit Rule Group Properties

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, find the rule set containing the rule group to be modified. Select the rule group, right click, select **Edit Properties**, then select **Rule Group**. The Figure 4-32 opens in the workspace.

Figure 4-32 Rule Group Properties Window

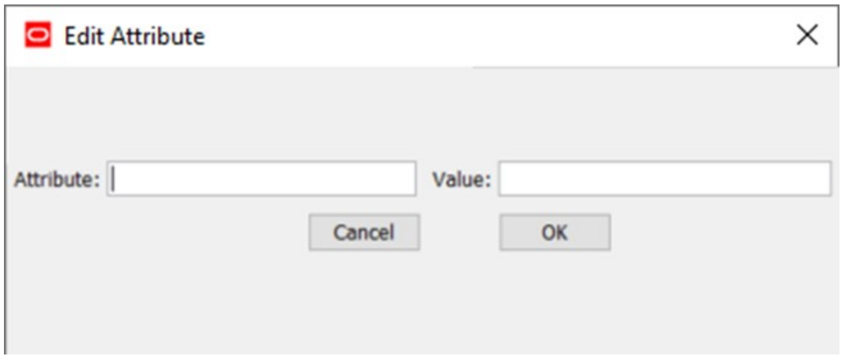

Rule Group Properties
Edit properties for an existing Rule Group
Add / Remove / Edit properties for the Rule Group.
Add
Edit
Remove
OK
Cancel

Name: AD_load
Description: a rule group

creator=MFPRCS
provision=some

2. In the Rule Group Properties window, click **Add** to open the Edit Attribute window in order to add a new rule group attribute.

Figure 4-33 Rule Group Edit Attribute Window



3. To edit an existing rule group attribute, select the attribute/value pair and then click **Edit**. The Edit Attribute window opens in the workspace. The window is pre-populated with the selected rule group attribute and value that the user can edit.
4. To remove an existing rule group attribute, select the attribute/value pair and then click **Remove**.

Measure Validation in the Rule Definition Window

If Enable Measure Content Validation is not selected in the Workbench Preferences window, measure content validation is turned off in the Measure Manager. Measure validation must be manually initiated in the rule tool. See Measure Validation within the Measure Manager.

From the Rule Definition window, click **Perform Measure Content Validation** on the Rule Definition toolbar.

Rule Definition

The Rule Definition tool allows the configuration administrator to define, organize, and manage rule sets, rule groups, and rules. It also allows for the creation of expressions and addition of expressions to rules.

Rules are groups of expressions that describe the relationship between measures.

When a rule has multiple expressions, those expressions are given a priority sequence to help the calculation engine select a calculation path that follows business priorities. When given a choice, the calculation engine will always select the highest priority expression in the rule that is available to be selected. Considerable care should be taken in the design of rules to ensure that appropriate expression priorities are established. The business priority may vary from implementation to implementation, and it may vary from one type of plan to another in the same implementation.

Rules are also given a priority sequence within a rule group to help the calculation engine select a calculation path that follows business priorities. When given a choice, the calculation engine will always select the highest priority rule that needs to be calculated.

Those who are configuring the calculation requirements of a solution are expected to fully understand the operation of the RPASCE calculation engine.

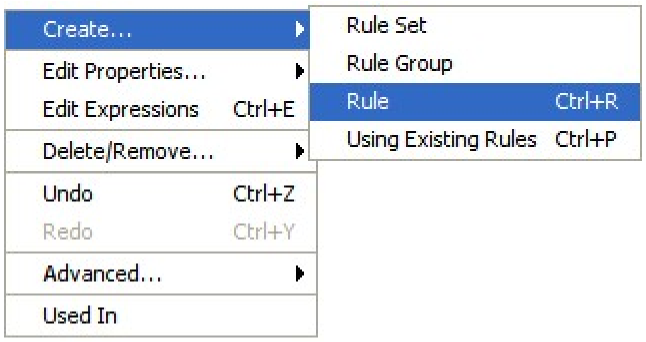
Create a Rule and Add It to a Rule Group

Complete the following steps to create a rule and add it to a Rule Group.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

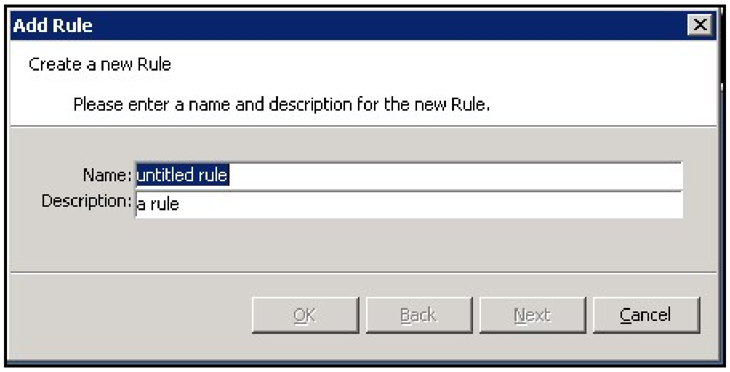
1. In the Rule Definition window, select the rule group in which you want to create a rule. If you want the rule to be created in a particular position in the sequence of rules in the rule group, select the rule before which you want the new rule to be placed. The new rule will be created higher than the selected rule.
2. Choose one of the following methods:
   - From the toolbar, click **New** and select **Rule**.
   - Click **Create...**, then **Rule** from the right-click menu.
   - Press **Ctrl+R**.

Figure 4-34   Rule Menu Option



The Figure 4-35 opens.

Figure 4-35   Add Rule Window


Add Rule

Create a new Rule

Please enter a name and description for the new Rule.

Name: untitled rule

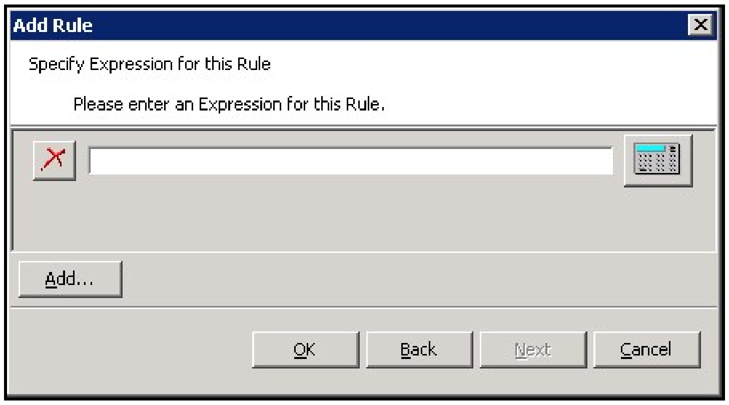
Description: a rule

3. In the Name field, enter the name of the rule.

> **Note:**
>
> A rule name can be a maximum of 24 alphanumeric or underscore characters. It must not have a name that is the same as any other rule that exists in the project. Rule names may start with a letter or an underscore but may not start with the letter ***r*** or ***R*** followed by a number.

4. In the Description field, enter a description of the rule.
5. Click **Next**. The Expression Builder window of the Add Rule window opens.

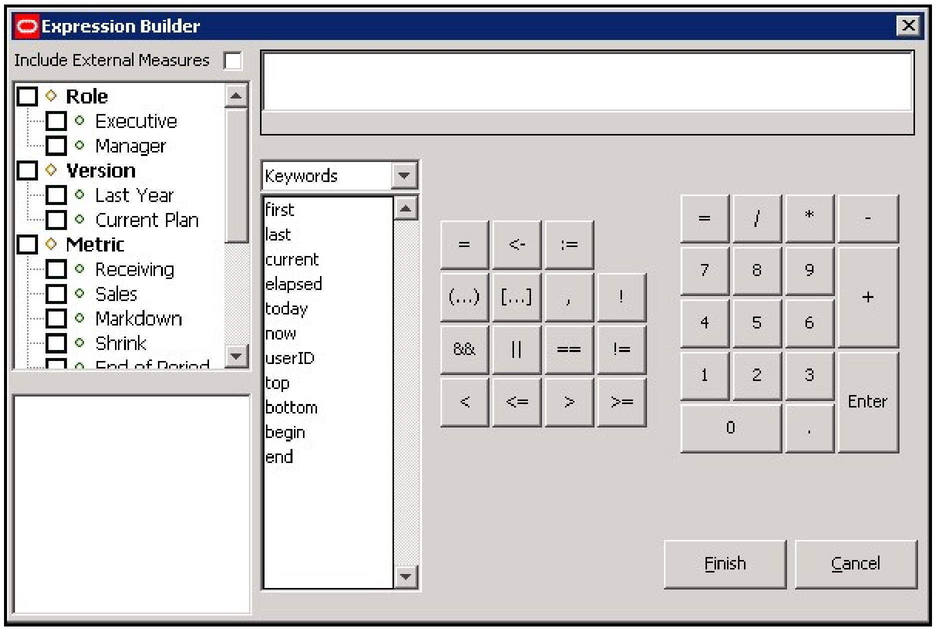
Figure 4-36   Specify Expression for Add Rule


Add Rule
Specify Expression for this Rule
Please enter an Expression for this Rule.
Add...
OK
Back
Next
Cancel

6. Type the expression in the input box or complete the following to use the Expression Builder.

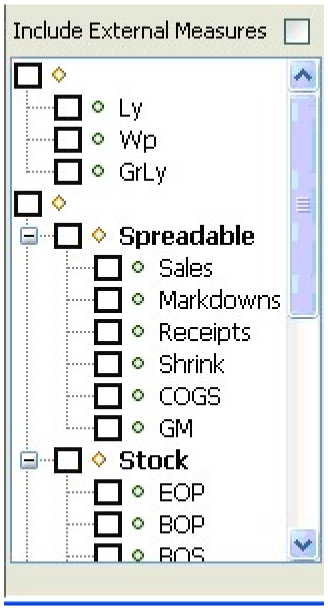
   a. Click **Expression Builder** (located to the right of the text box). The Expression Builder window opens.

   Figure 4-37   Expression Builder Window

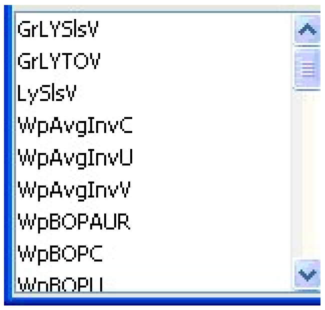
   b. In the upper left-hand pane, select the measure components to filter measures to be visible for pasting into the expression. If External Measures are required in the expression, check the **Include External Measures** check box. Realized measures meeting the filtering conditions are displayed in the lower left-hand pane.

Figure 4-38 Include External Measures Check Box



c. In the lower left-hand pane, double-click any measures to be used in the expression to get the measure name pasted at the insertion point in the expression.

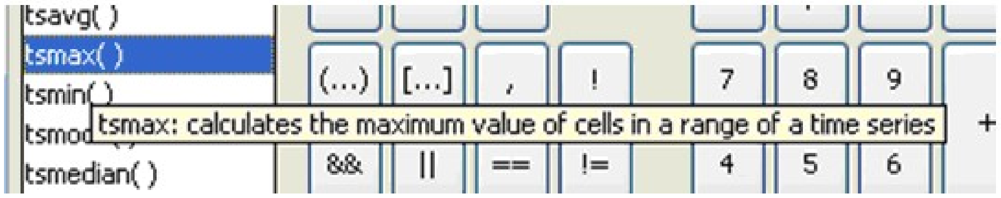
Figure 4-39 Measure Names



Note:

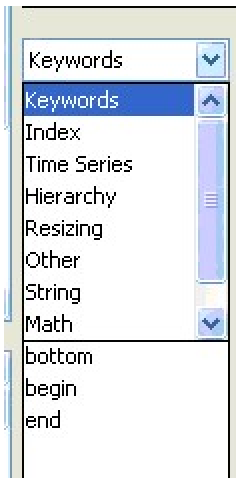
Place the mouse over the name of a function, procedure, keyword, or modifier to see a tooltip that explains its function.

Figure 4-40 Tooltip


tsmax: calculates the maximum value of cells in a range of a time series

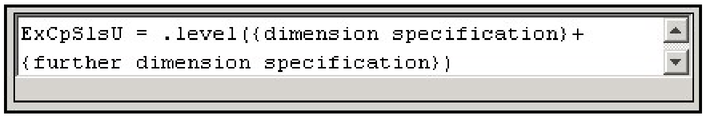
d. From the drop-down list, select a category of functions, procedures, keywords, or modifiers. Double-click on a specific function, procedure, keyword, or modifier to be used in the expression, and (if appropriate) outline syntax pasted at the insertion point in the expression.

Figure 4-41 Category Options List



The outline syntax of a function, procedure, keyword, or modifier is pasted with components of the syntax separated with braces **{}**. When the insertion point is in a component of the outline syntax in the expression, you will see a description of the component at the bottom of the expression window. When the insertion point is in a component of the outline syntax, anything that is entered or pasted replaces the whole component.

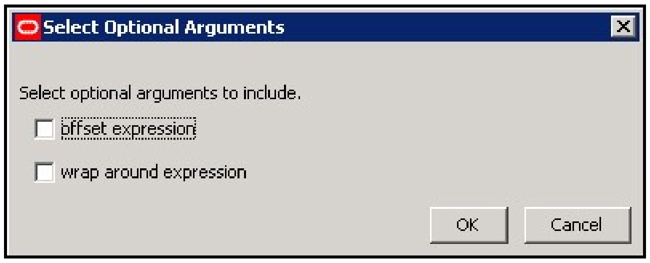
Figure 4-42 Expression Window



```
ExCpSlsU = .level({dimension specification}+
{further dimension specification})
```

If the function, procedure, keyword, or modifier that is pasted has optional arguments (for example, the cover function has optional arguments for an offset expression and a wrap-around expression), you will be presented with the following window to select which of the optional arguments to use in the expression. Note that all arguments are positional, so if a later argument is selected, all earlier arguments will be automatically selected. If an earlier argument is cleared, then all later arguments are automatically cleared.

Figure 4-43 Select Optional Arguments Window



Select optional arguments to include.

☐ offset expression

☐ wrap around expression

If the function, procedure, keyword, or modifier that is pasted has repeating arguments (for example the min function finds the minimum of a variable number of expressions), a window opens to select how many of the repeating arguments to use in the expression.

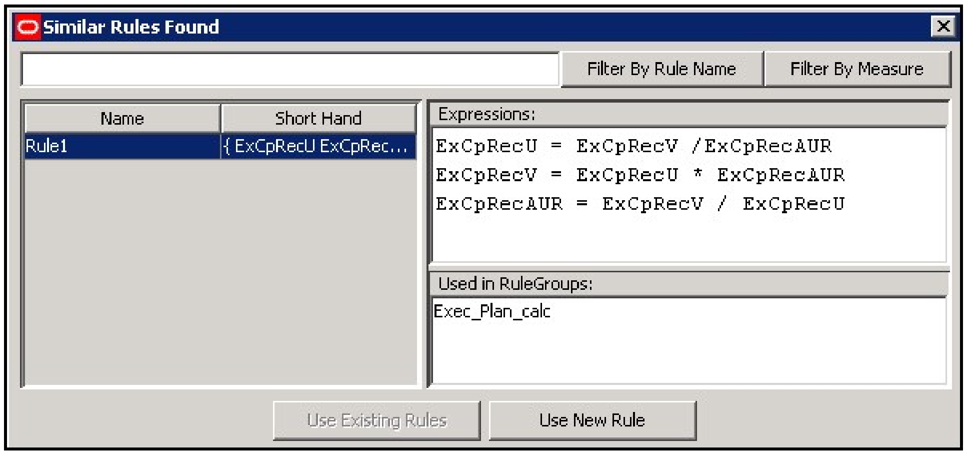
e. Use the keyboard or buttons to add the appropriate mathematical operators and constants to construct the expression.

f. When you have defined the expression as needed, click **Finish** and close the window.

> **Note:**
>
> If an invalid expression is created, a warning message is displayed.

7. To add further expressions to the rule, click **Add**, and repeat the process of entering an expression.
8. Click **OK** to save any changes and close the window.
9. If the newly defined rule's expressions use exactly the same measures as another rule that already exists in the Rule Pool, the Figure 4-44 opens. The window shows all rules that use exactly the same measures as the newly defined rule. This provides you with an opportunity to use an existing rule from the Rule Pool or to continue with the new rule. The Similar Rules Found window allows you to view rules, associated expressions, and rule groups that contain the rules. The rule table may be filtered based on the rule name or by measure. Click **Use Existing Rule** or **Use New Rule** to save changes and close the window.

Figure 4-44 Similar Rules Found Window



Name	Short Hand
Rule1	{ ExCpRecU ExCpRec...

Expressions:

```
ExCpRecU = ExCpRecV /ExCpRecAUR
ExCpRecV = ExCpRecU * ExCpRecAUR
ExCpRecAUR = ExCpRecV / ExCpRecU
```

Used in RuleGroups:

Exec_Plan_calc

The new rule is placed preceding the rule that was selected at the start of the process in the sequence of rules in the rule group or at the end of the rule group if no rule was selected.

Note:

If **Use Existing Rule** or **Use New Rule** is not selected and the window is closed manually, a new rule is created.

Add an Existing Rule to a Rule Group

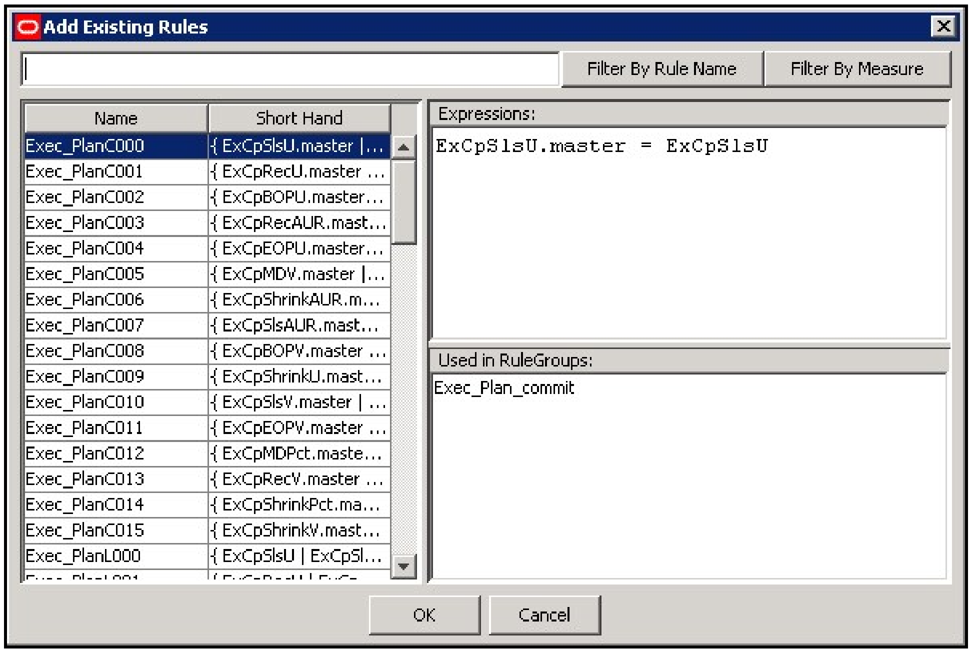
Using the Add Existing Rules window, you may select multiple rules to add to a rule group. If at least one of the selected rules is already in the rule group, the OK button will gray out disallowing the operation until that rule is cleared. When multiple rules are selected, the expression and rule group displays will go blank. However, if there is only one selected rule, the rule's expressions and the list of rule groups that use the rule will be displayed.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select the rule group to be used to add an existing rule. If the rule is to be placed in a particular position in the sequence of rules in the rule group, select the rule in which the new rule is to precede.
2. Choose one of the following methods:
   - From the toolbar, click **New** and select **Using Existing Rule**.
   - Select **Create** – Using Existing Rule from the right-click menu.
   - Press **Ctrl+P**.

   The Figure 4-45 opens.

Figure 4-45   Add Existing Rule Window



Add Existing Rules

Filter By Rule Name | Filter By Measure

Name	Short Hand
Exec_PlanC000	{ ExCpSlsU.master |...
Exec_PlanC001	{ ExCpRecU.master ...
Exec_PlanC002	{ ExCpBOPU.master...
Exec_PlanC003	{ ExCpRecAUR.mast...
Exec_PlanC004	{ ExCpEOPU.master...
Exec_PlanC005	{ ExCpMDV.master |...
Exec_PlanC006	{ ExCpShrinkAUR.m...
Exec_PlanC007	{ ExCpSlsAUR.mast...
Exec_PlanC008	{ ExCpBOPV.master ...
Exec_PlanC009	{ ExCpShrinkU.mast...
Exec_PlanC010	{ ExCpSlsV.master | ...
Exec_PlanC011	{ ExCpEOPV.master ...
Exec_PlanC012	{ ExCpMDPct.maste...
Exec_PlanC013	{ ExCpRecV.master ...
Exec_PlanC014	{ ExCpShrinkPct.ma...
Exec_PlanC015	{ ExCpShrinkV.mast...
Exec_PlanL000	{ ExCpSlsU | ExCpSl...

Expressions:

```
ExCpSlsU.master = ExCpSlsU
```

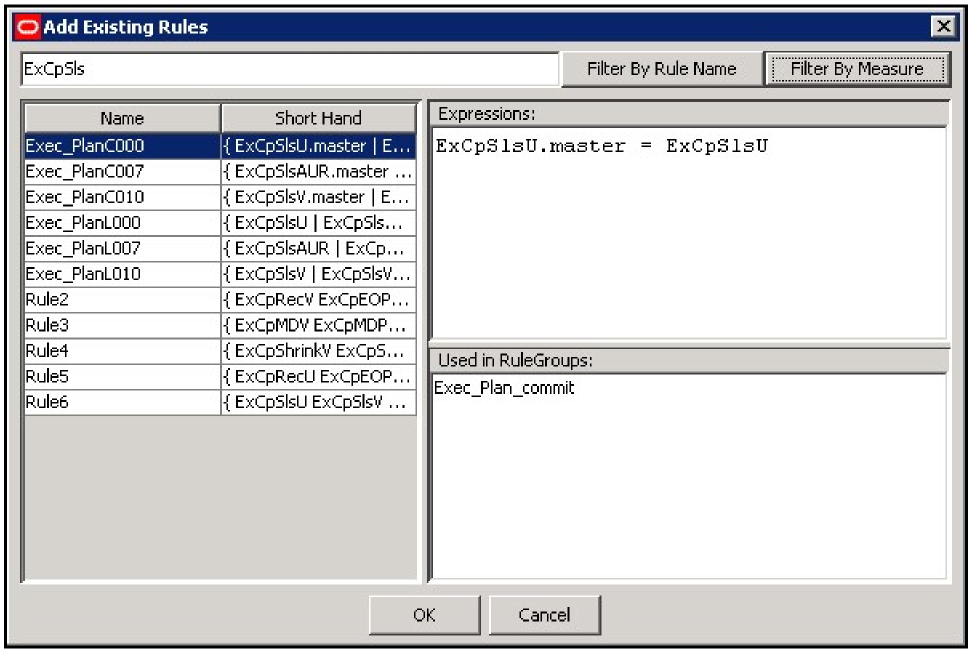
Used in RuleGroups:

Exec_Plan_commit

OK | Cancel

3. Filter by Rule Name or by Measure and view the associated expressions to find the correct rule. Filtering means that all data are compared, but only matching data is allowed to pass through. In order to filter by Rule Name or by Measure, type a filter string (for instance, ExCpSls in the input box directly under the Title Bar. Click **Filter by Measure**, and only those measures with ExCpSls display.

Figure 4-46   Filter on ExCpSls



Add Existing Rules

ExCpSls

Filter By Rule Name | Filter By Measure

Name	Short Hand
Exec_PlanC000	{ ExCpSlsU.master | E...
Exec_PlanC007	{ ExCpSlsAUR.master ...
Exec_PlanC010	{ ExCpSlsV.master | E...
Exec_PlanL000	{ ExCpSlsU | ExCpSls...
Exec_PlanL007	{ ExCpSlsAUR | ExCp...
Exec_PlanL010	{ ExCpSlsV | ExCpSlsV...
Rule2	{ ExCpRecV ExCpEOP...
Rule3	{ ExCpMDV ExCpMDP...
Rule4	{ ExCpShrinkV ExCpS...
Rule5	{ ExCpRecU ExCpEOP...
Rule6	{ ExCpSlsU ExCpSlsV ...

Expressions:

```
ExCpSlsU.master = ExCpSlsU
```

Used in RuleGroups:

Exec_Plan_commit

OK | Cancel

4. Click the rule in the table to select it.
5. Click **OK** to save changes and close the window.

The rule is placed higher than the rule that was selected at the start of the process in the sequence of rules in the rule group or at the end of the rule group if no rule was selected.

Apply a Rule Pattern to Create New Rules or to Update Existing Rules

The Apply Rule Pattern functionality allows you to create new rules or update existing rules according to a pattern established by a selected base or template rule. The rule tool recognizes inherent similarities or patterns in measure components used in some rules when compared to the base or template rule. Based on these similarities, the rule tool allows for the creation of new rules or update of existing rules to fit the pattern set by the base or template rule.

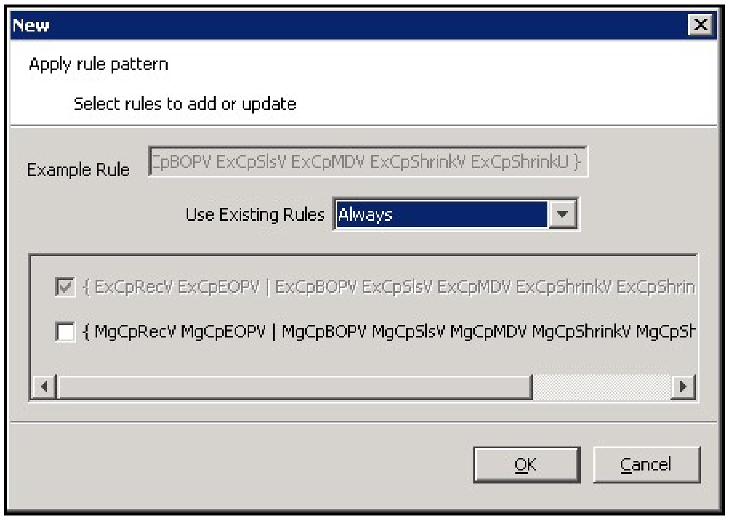
When the Apply Pattern capability is enabled, there is a possibility that some of the New or Updated rules will have the same expressions as a rule that already exists in the rule pool. If this happens, the Similar Rules Found window opens and provides the option of using the existing rule or actually creating a new one.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select any rule group that contains the rule to use as the pattern basis.
2. Select the rule whose pattern is to be used as a basis for creating or updating rules.
3. From the toolbar, click **Advanced** and then select **Apply Pattern**, or right-click and select **Advanced** and then, **Apply Pattern**.

   A Figure 4-47 opens containing a set of rules that are presented for selection. This set is composed of rules whose measures follow the selected rule's pattern in terms of the individual measure components used. The set of rules are composed of potential new rules or potential updated rules.

Figure 4-47 New Apply Rule Pattern Window


New

Apply rule pattern

Select rules to add or update

Example Rule: EpBOPV ExCpSlsV ExCpMDV ExCpShrinkV ExCpShrinkU }

Use Existing Rules: Always

☑ { ExCpRecV ExCpEOPV | ExCpBOPV ExCpSlsV ExCpMDV ExCpShrinkV ExCpShrin

☐ { MgCpRecV MgCpEOPV | MgCpBOPV MgCpSlsV MgCpMDV MgCpShrinkV MgCpSh

Example:

Consider a configuration with three components in the measure naming scheme:

- a version (such as Wp)
- a metric (such as Shrink)
- a unit of measure (such as V)

If the selected base rule is:

{ WpShrinkU WpShrinkAUR | WpShrinkV }

...then the rules:

Rule1: { WpSlsU WpSlsV WpSlsAUR | }

...and

Rule2: { WpRecU WpRecV WpRecC | }

...would fit the pattern and be included in the list.

Rule1 fits the pattern because its measures use the same measure components with the exception of the metric component. For the Metric component, the base rule uses Shrink, and Rule1 uses Sls consistently. In this case, the tool will present the rule:

{ WpSlsU WpSlsAUR | WpSlsV }

...as a possible update for Rule1.

The update results in the conversion of Rule1 to a rule that uses the same measures as the original Rule1, but it has the expression pattern of the base rule.

Rule2 fits the pattern because it uses the same Version as the base rule. For the Metric component, the base rule uses Shrink, and Rule2 uses Rec consistently. Unlike Rule1, Rule2 uses C as the Unit of Measure in one of its measures. This is not an exact fit like Rule1. In this case, the tool will present the rule { WpRecU WpRecAUR | WpRecV } as a possible new rule. Notice that this rule is forced to be an exact fit as Rule1 was.

4. Select the rule replacement policy in the Use Existing Rules drop-down menu. This allows the user to select, on a rule-by-rule basis, which rules to reuse and which to re-create. By default, this is set to **Always**.
   - When **Always** is selected, rules that already exist elsewhere in the configuration will always be used in place of the newly generated rules.
   - When **Never** is selected, new rules will always be created, even if an identical rule already exists elsewhere in the configuration.
   - When **Prompt** is selected, the Rule Locator window will be presented for each rule whose expressions already exist elsewhere in the configuration.
5. From the list of possible new and updated rules, select those to be updated or added to the rule group. In either case, if rule reuse is specified in Step 4, a new rule will not be created, but the rule existing elsewhere in the configuration will be used in its place.
6. Click **OK**.

   A selected rule that is labeled New Rule will be added to the end of the rule group. The new rule's name will default to the template rule's name suffixed with a number to keep the name unique. A selected rule that is labeled Update Rule is already in the rule group and will be replaced. This means that the old rule will be removed and replaced with a new rule whose expressions follow the base rule's expression pattern. In both cases, the rule will follow the pattern of the base rule's expression.

Delete a Rule from All Rule Groups

This procedure deletes the rule from all rule groups that contain this rule as well as from the Rule Pool. If the desired action is to remove the rule from a rule group, but to retain it in other rule groups, follow Remove a Rule from a Rule Group.

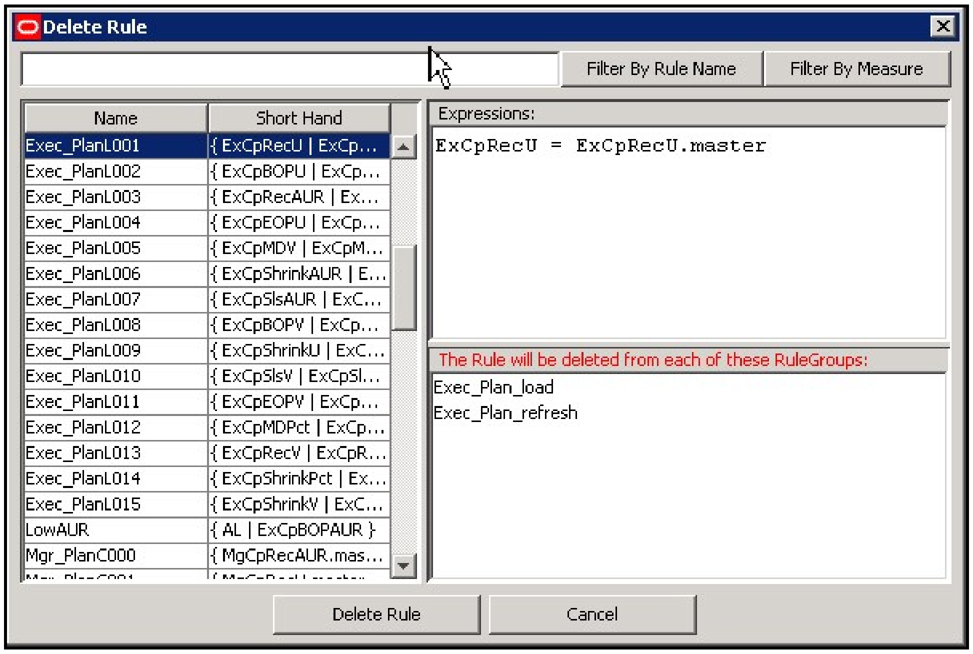
Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select any rule group that contains the rule to be deleted.
2. Select the rule to be deleted.

> **Note:**
>
> If multiple rules are selected, only the last selected rule is deleted.

3. From the toolbar, click **Delete** and select **Delete Rule**, or select **Delete**, then **Remove**, and then **Delete Rule** from the right-click menu. The Figure 4-48 opens.
4. Verify that the rule selected in the table is the rule to be deleted or click to select a different rule in the table.

Figure 4-48 Delete Rule Window



Name	Short Hand
Exec_PlanL001	{ ExCpRecU | ExCp...
Exec_PlanL002	{ ExCpBOPU | ExCp...
Exec_PlanL003	{ ExCpRecAUR | Ex...
Exec_PlanL004	{ ExCpEOPU | ExCp...
Exec_PlanL005	{ ExCpMDV | ExCpM...
Exec_PlanL006	{ ExCpShrinkAUR | E...
Exec_PlanL007	{ ExCpSlsAUR | ExC...
Exec_PlanL008	{ ExCpBOPV | ExCp...
Exec_PlanL009	{ ExCpShrinkU | ExC...
Exec_PlanL010	{ ExCpSlsV | ExCpSl...
Exec_PlanL011	{ ExCpEOPV | ExCp...
Exec_PlanL012	{ ExCpMDPct | ExCp...
Exec_PlanL013	{ ExCpRecV | ExCpR...
Exec_PlanL014	{ ExCpShrinkPct | Ex...
Exec_PlanL015	{ ExCpShrinkV | ExC...
LowAUR	{ AL | ExCpBOPAUR }
Mgr_PlanC000	{ MgCpRecAUR.mas...

Expressions:

```
ExCpRecU = ExCpRecU.master
```

The Rule will be deleted from each of these RuleGroups:

Exec_Plan_load
Exec_Plan_refresh

5. Click **Delete Rule** to delete the rule and close the window.

Remove a Rule from a Rule Group

This procedure removes the rules only from the currently selected rule group. If the desired action is to delete the rules from all rule groups and the rule pool, see Delete a Rule from All Rule Groups.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select the rule group that contains the rules to be removed.
2. Select the rules to remove.

   Note:

   Multiple rules can be selected by holding the Control (Ctrl) key as the individual rules are selected, or by clicking one rule and holding Shift key as another rule is selected, which selects all rules between the two that were clicked. A selected rule is indicated by a bold rule name

3. From the toolbar, click **Delete** and then select **Remove Rules**, or select **Delete**, then **Remove…**, and then **Remove Rules** from the right-click menu.

   The rules are removed from the rule group but are still in the rule pool. The rules are permanently lost when the project is closed, unless they are used in another rule group.

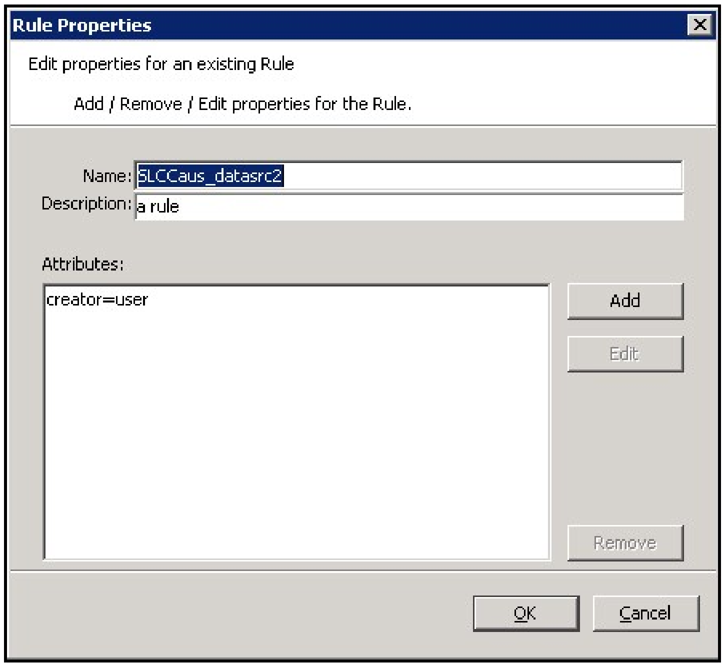
Edit Properties of a Rule

Complete the following steps to edit the properties of a rule.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select any rule group that contains the rule to edit.
2. From the rule group, select the rule to edit.
3. From the toolbar, click **Edit** and then **Select Rule**, or select **Edit Properties...**, then **Rule** from the right-click menu. The Figure 4-49 opens.

Figure 4-49 Rule Properties Window



Edit properties for an existing Rule

Add / Remove / Edit properties for the Rule.

Name: SLCCaus_datasrc2
Description: a rule

Attributes:

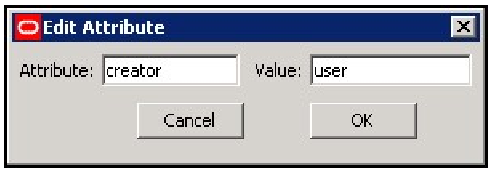
creator=user

4. To edit the name of the rule, enter a new name in the Name field.

> **Note:**
>
> A rule name can be a maximum of 24 alphanumeric or underscore characters. It must not have a name that is the same as any other rule that exists in the project. Rule names may start with a letter or an underscore but may not start with the letter ***r*** or ***R*** followed by a number.

5. To edit the description of the rule, enter a new description in the Description field.
6. To edit attributes for a rule:
   a. Select the attribute to edit.
   b. Click **Edit**. The Figure 4-50 opens.

   Figure 4-50 Edit Attribute Window

   Attribute: creator Value: user

   c. Update the information as necessary.
   d. Click **OK** to save any changes and close the window.
7. To remove attributes from a rule:
   a. Select the attribute to delete.

   b. Click **Remove**. The attribute is removed from the display box.

8. Click **OK** to save any changes and close the window.

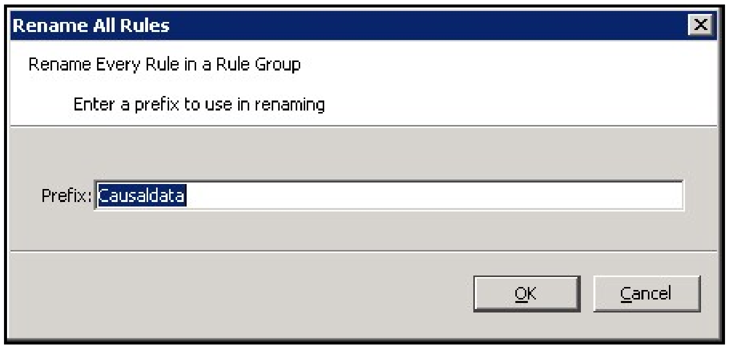
Rename All Rules in a Rule Group

Complete the following steps to rename all rules in a Rule Group.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select the rule group that contains the rules to rename.
2. From the toolbar, click **Advanced** and select **Rename All Rules**, or select **Advanced – Rename All Rules** from the right-click menu. The Figure 4-51 window opens.

Figure 4-51 Rename All Rules


Rename All Rules

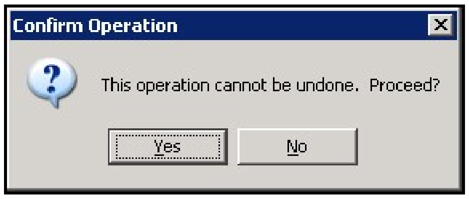
Rename Every Rule in a Rule Group

Enter a prefix to use in renaming

Prefix: Causaldata

3. In the Prefix field, enter a prefix up to ten characters in length, which is the start of all rule names.
4. Click **OK**. The Figure 4-52 window opens.

Figure 4-52 Confirm Operations


Confirm Operation

This operation cannot be undone. Proceed?

5. Click **Yes**.

> **Note:**
>
> All of the rules in the rule group are renamed with the prefix followed by a 4-digit numeric identifier generated by the rule tool. The rule tool will maintain the order that the rules were in before they were renamed, and it uses that order in generating the numeric identifier.

Note:

Since rules may appear in more than one rule group, use of this feature may generate rule names that look out of place in other rule groups, especially if the prefix implies the rule group.

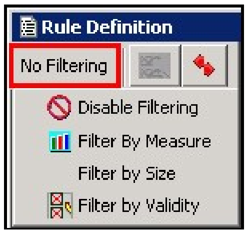
Filter Rules in a Rule Group

Complete the following steps to filter rules in a Rule Group.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select a rule group.
2. In the Rule Definition window, click **Rule Filtering**.

Figure 4-53 No Filtering Button in Rule Definition Window



Note:

This is a dynamic button, and the text changes depending on the current filter mode.

3. Select one of the following options:
   - **Disable Filtering** – All rules display.
   - **Filter by Measure** – Works in conjunction with the measure components box in the bottom left corner of the screen. Rules are filtered to show those whose measures conform to the selected component scheme.
   - **Filter by Size** – Rules are filtered to show those with more than one expression.
   - **Filter by Validity** – Only invalid rules display.

Note:

When rule filtering is active, the buttons used to reorder rules in the rule group are unavailable.

Reordering Rules in a Rule Group

Complete the following steps to reorder rules in a Rule Group.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select the rule group that contains the rules to reorder.
2. Select the rule to reorder.
3. Complete the following as needed:
   - Use the Up or Down arrows on the Rule Definition toolbar to move the rule up or down the list.
   - Click the up or down arrows to the left of the rule name to move the rule up or down the list.

Auto Generate Load and Commit Rules

Complete the following steps to auto generate Load and Commit Rules.

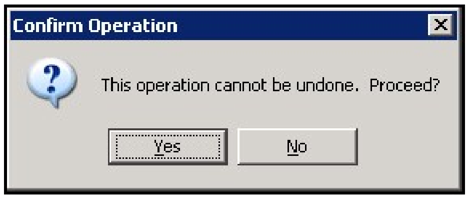
Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

Rules for the load and commit rule groups in a rule set can be auto-generated based on the calc rule group. The measures referenced in the calc rule group are assumed to be all of the measures in a workbook (if there are others, manually add their load and/or commit rules). A load or commit rule is generated for all of those measures that have a database allocated (those that are physically stored).

1. Select the rule set for which load or commit rules are to be auto generated or select any rule group in that rule set.
2. Complete one of the following methods:
   - From the toolbar, click **Advanced** and select **Generate Load Rules** or **Generate Calc Rules**
   - Select **Advanced** then **Generate Load Rules** or use the right-click menu to select **Advanced** then **Generate Calc Rules**.

   The Figure 4-54 window opens to inform you that this process cannot be undone.

   Figure 4-54   Confirm Operation

   Confirm Operation

   This operation cannot be undone. Proceed?

3. Click **Yes**.

   Rules are automatically generated and named for the load or commit rule group. There is one rule with a single expression that is generated for each measure used in the calc rule group for the rule set that has a database assigned.

In the load rule group, the rules are named `<rulesetname>Lnnn` where *nnn* is a 3-digit order number. The rules in a commit rule group are similarly named `<rulesetname>Cnnn`. The expression in a generated rule in a load rule group is of the form:

```
<measurename> = <measurename>.master
```

and in the generated commit rule group are of the form:

```
<measurename>.master = <measurename>
```

Copy Selected Rules to Another Rule Group

Complete the following steps to copy selected rules to another Rule Group.

When copying selected rules of a rule group into another rule group, it is possible to specify whether existing rules will be used or copies of the rules will be created. The Use Existing Rules check box defaults to using any existing rules in the rule pool. If this check box is selected, the copy selected rules operation will use the same rules that the source rule group has. If this check box is not selected, the copy selected rules operation will create copies of the rules and use those copies for appending to or replacing rules in the destination rule group.

When the user uses the Find/Replace feature, it is possible that a changed rule will have the same expressions as a rule that already exists in the rule pool. If the Use Existing Rules check box is selected, the Similar Rules Found window opens. You have the option of using the existing rule or creating a new one.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

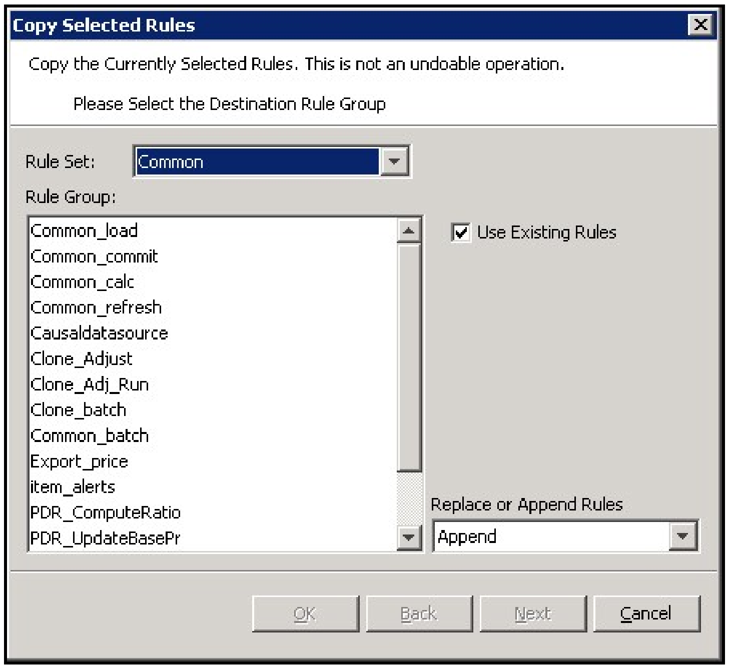
1. Select the Rule Group that contains the rules to copy, and select the individual rules to be copied.

   > **Note:**
   >
   > To select multiple rules, press the Ctrl key and click the rules to select, or click one rule and hold Shift key as selecting another rule, which selects all rules between the two that have been selected. A selected rule is indicated by a bold rule name.

2. Complete one of the following methods:
   - From the toolbar, click **Advanced**, and select **Copy Selected Rules**.
   - Select **Advanced – Copy Selected Rules** from the right-click menu.
   - Press **Ctrl+C**.

   The first Figure 4-55 opens.

Figure 4-55 Copy Selected Rules Window (1)


Copy Selected Rules

Copy the Currently Selected Rules. This is not an undoable operation.

Please Select the Destination Rule Group

Rule Set: Common

Rule Group:

Common_load
Common_commit
Common_calc
Common_refresh
Causaldatasource
Clone_Adjust
Clone_Adj_Run
Clone_batch
Common_batch
Export_price
item_alerts
PDR_ComputeRatio
PDR_UpdateBasePr

[✓] Use Existing Rules

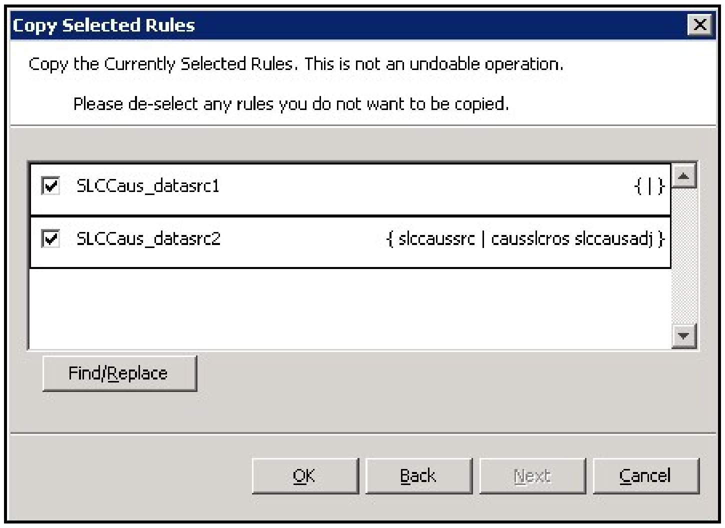
Replace or Append Rules: Append

3. In the Rule Set field, select the copy's destination rule set.
4. In the Rule Group area, select the desired copy's destination Rule Group.
5. Check the **Use Existing Rules** check box if the rules should be added to the destination Rule Group. If the User Existing Rules check box is clear, copies of the selected rules are created for the destination Rule Group instead.

> **Note:**
>
> If the Find or Replace functionality is used, copies are created for affected rules even if the **Use Existing Rules** check box is checked.

6. In the Replace or Append Rules field, select:
   - **Replace** – To remove all rules that already exist in the destination rule group before the copy.
   - **Append** – To add to the rules already in the destination rule group.
7. Click **Next**. The second Figure 4-56 opens. This window allows you to select or clear rules from those originally selected to copy when the check box beside the rule name is selected.

Figure 4-56 Copy Selected Rules Window (2)

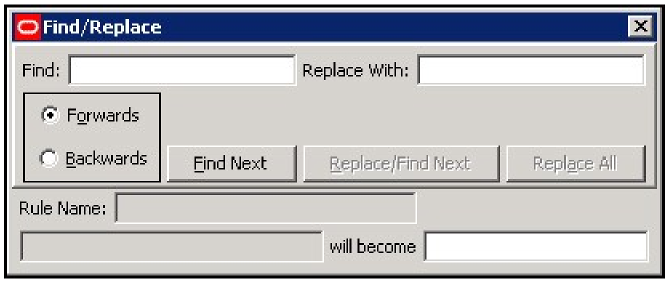

Copy Selected Rules
Copy the Currently Selected Rules. This is not an undoable operation.
Please de-select any rules you do not want to be copied.
SLCCaus_datasrc1 { | }
SLCCaus_datasrc2 { slccaussrc | causslcros slccausadj }
Find/Replace
OK
Back
Next
Cancel

Find and Replace Measures in the Copied Rules

The ability to find or replace in the copy rules is a very powerful and useful feature. This feature can be used to build a collection of rules and clone them to a very similar collection of rules. For example, a collection of rules that calculate a series of variances with one version can be cloned to produce rules that calculate a series of variances with another version.

1. Click **Find/Replace**. The Figure 4-57 opens.

Figure 4-57 Find/Replace Window



2. In the Find field, enter the portion of the measure to replace.

> **Note:**
>
> The Find function is case sensitive.

3. In the Replace With field, enter the string to replace the portion of the measure name.
4. Complete one of the following:
   - Select **Forwards** to search the rules in order

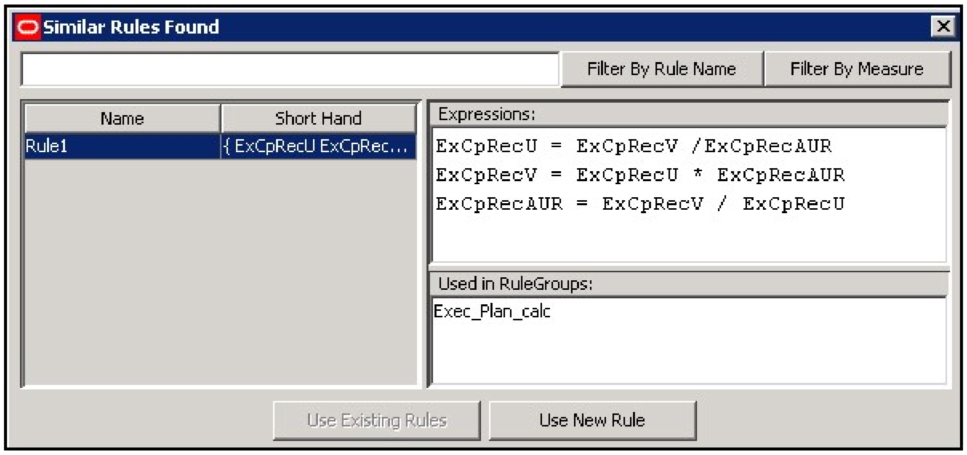
- Select **Backwards** to search the rules in reverse order.

> **Note:**
>
> Searching Forwards proceeds from left to right starting with the first measure of the first expression of the first selected Rule and will go through all expressions in all selected Rules. Similarly, searching Backwards flows in the reverse direction starting with the right most measure of the last expression in the last Rule selected.

5. Click **Find Next**. The first candidate measure to be replaced displays in the bottom left field.
6. Complete the following as needed:
   - Click **Replace/Find Next** to replace the current candidate measure and display the next candidate measure.
   - Click **Replace All** to replace all instances in all the selected Rules of the current candidate measure. For example, if a search for Wp finds WpRecV, click **Replace All** to complete a replace on all instances of WpRecV in all the selected Rules.
   - Click **Find Next** to skip over that occurrence of the portion of the measure name and go onto the next one.

   The Figure 4-58 opens.

Figure 4-58 Similar Rules Found Window



Name	Short Hand
Rule1	{ ExCpRecU ExCpRec...

Expressions:

```
ExCpRecU = ExCpRecV /ExCpRecAUR
ExCpRecV = ExCpRecU * ExCpRecAUR
ExCpRecAUR = ExCpRecV / ExCpRecU
```

Used in RuleGroups:

Exec_Plan_calc

7. Replace the rules.

   Using the Figure 4-58, you can replace a portion of the rules (for instance a prefix) either for all instances of the rules or only for the new instances (where the old instances are not affected).

   Example:

   Suppose you create a series of rules for the Executive (Ex) Calc Rule Group and you want to use these as a model for the Manager (Mg) Calc Rule Group. Each of these original rules contains expressions with the *Ex* prefix. Each such instance needs to be replaced with *Mg*.

   a. Click **Use Existing Rules** to replace every instance of *Ex*.
   b. Click **Use New Rule** to replace the new instances of *Ex* without affecting the previous rules that contain *Ex*.
8. Click **X** to close the Figure 4-57.
9. Click **OK**.
10. Click **Yes** to confirm.

    The copies of the rules are placed in the target rule group. These copies have names that start with as many characters as possible from the name of the original rule and end with an underscore and number.

Expressions and Rules

An expression describes and solves the relationship between measures in a way that causes a measure to be calculated through the expression. They form the basis for all calculations of the relationships between measures, and they are evaluated by the calculation engine during a calculation. In some cases, there may be business reasons for wanting more than one of the measures in a relationship to be calculable or solvable through that relationship. Expressions are written in a syntax that allows for the calculation of a single measure from other measures, constants, and parameters by using standard arithmetical functions and a rich set of mathematical, technical, and business functions. Expressions have multiple results.

Example:

Expression 1: ReceiptUnits = ReceiptsValue / ReceiptsPrice

This expression specifies the way ReceiptUnits are calculated. ReceiptUnits are calculated by dividing ReceiptsValue by ReceiptsPrice.

Note:

Measures are not only calculated based on expressions. They are also calculated based on spreading and aggregating.

Reorder an Expression in a Rule

Complete the following steps to reorder an expression in a Rule.

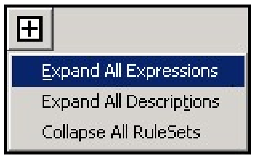
Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Figure 4-28, select any rule group that contains the rule whose expression is to be reordered.
2. Choose one of the following methods:

   Expand the rule to view the expressions associated with the rule. Click **Toggle** from the Rules toolbar and select **Expand All Expressions**.

   Click **Plus** for the rule.

Figure 4-59   Expand All Expressions Options



3. Use the up and down arrows to move the expression up or down the list.

Edit an Expression in a Rule

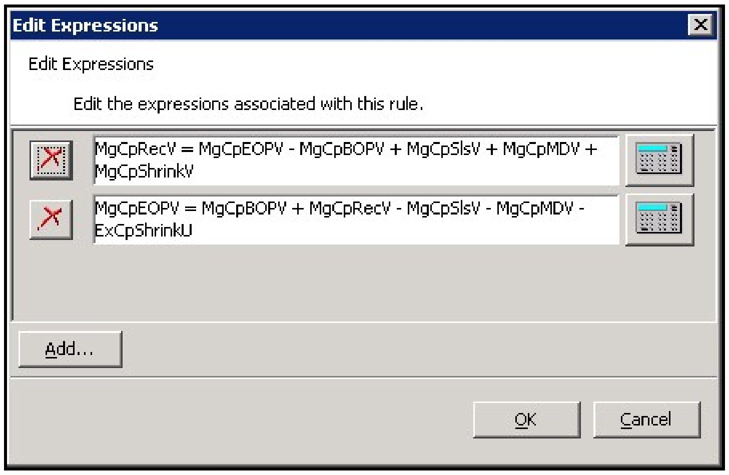
Complete the following steps to edit an expression in a Rule.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Figure 4-28, select the rule whose expression is to be edited.
2. Choose one of the following methods:
   - From the toolbar, click **Expression Builder**.
   - Select **Edit Expressions** from the right-click menu.
   - Press **Ctrl+E**.

   The Figure 4-60 opens.

Figure 4-60   Edit Expressions Window



MgCpRecV = MgCpEOPV - MgCpBOPV + MgCpSlsV + MgCpMDV + MgCpShrinkV

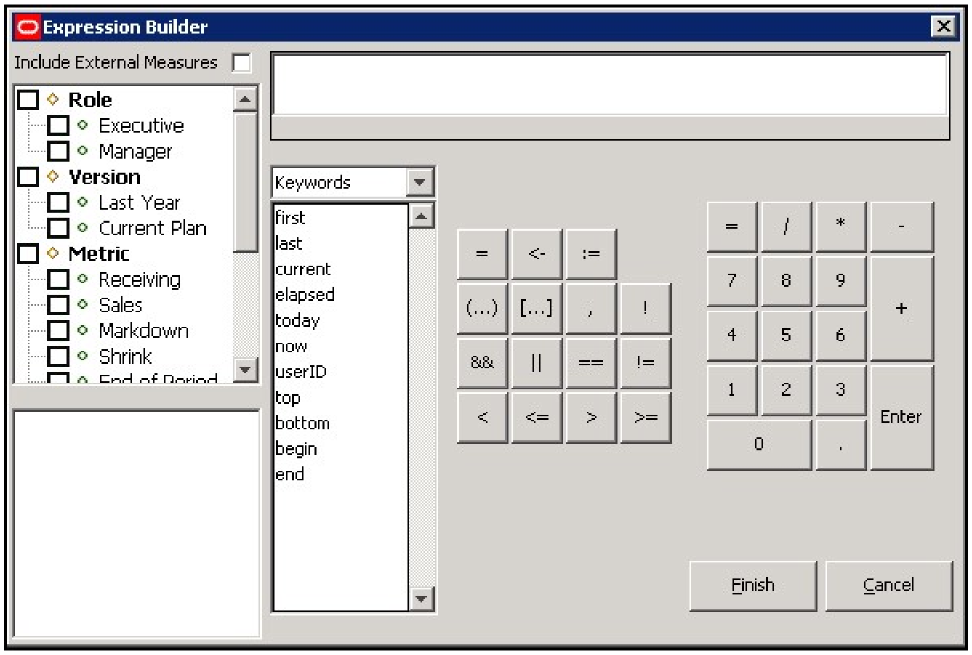
MgCpEOPV = MgCpBOPV + MgCpRecV - MgCpSlsV - MgCpMDV - ExCpShrinkU

3. Choose one of the following methods to edit an expression:

   Edit the expression in its text box.

   Click **Expression Builder** for the expression to edit. The Figure 4-61 opens. Use the Expression Builder to make necessary changes and click **Finish** when complete.

Figure 4-61 Expression Builder Window



4. Click **OK** to save the changes and close the window.

Delete an Expression from a Rule

Complete the following steps to delete an expression from a Rule.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select the rule whose expression is to be deleted.
2. Choose one of the following methods:
   - From the toolbar, click **Expression Builder**.
   - Select **Edit Expressions** from the right-click menu.
   - Press **Ctrl+E**.

   The Figure 4-60 opens.
3. Click **Delete** (located to the left of the expression in the Edit Expressions window.
4. Click **OK** to delete the expression. The expression is permanently deleted from the rule.

Note:

If the only expression in the rule is deleted, the rule will be flagged as being invalid, because it has no expressions.

Add an Expression to a Rule

Complete the following steps to add an expression to a Rule.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Rules**. The Figure 4-28 opens in the workspace.

1. In the Rule Definition window, select the rule that will have an expression added.
2. Choose one of the following methods:
   - From the toolbar, click **Expression Builder**.
   - Select **Edit Expressions** from the right-click menu.
   - Press **Ctrl+E**.

   The Figure 4-60 opens.
3. Click **Add** to add a new expression in the Edit Expressions window.
4. Choose one of the following methods:
   - Enter the expression in its text box.
   - Click **Expression Builder** for the expression to edit. The Figure 4-61 opens.

     Use the Expression Builder to define the rule and click **Finish** when complete.
5. Click **OK** to add the expression.

Named Rule Pattern

The named rule pattern functionality allows user to simplify rule generation by defining rule patterns explicitly and then generate large quantity of similar rules from them. Two steps are required: First, define rule patterns using tokens and expressions, and give a unique name for each pattern; second, provide token values and generate rules from the selected pattern.

The named rule patterns are saved into a separate rulepattern.xml file. When user opens the application configuration in ConfigTools, the saved rule patterns are loaded and can be used to generate new rules.

When applying the pattern, users need select which pattern, which realized measures or measure components to replace tokens in the pattern definition.

Note: Although example below show sophisticated rule generation, currently RPASCE limits support to only Load and Commit rules, while other types might be supported in future releases.

Mandate Requirements

Here is the list of the mandate requirements:

- The rule pattern name must be unique across the application. The name can be a maximum of 16 alphanumeric or underscore characters.
- Rule pattern name may start with a letter or an underscore, but may not start with the letter r or R, and must be immediately followed by a number.
- The token used in a rule pattern is wrapped between $$ and $$. For example, for rule pattern rp1, the pattern definition for a load rule is $$M1$$=$$M1$$.master.

- One rule pattern must contain only one expression.
- The realized measure name token must start with M, for example, $$M1$$.
- The measure component token must start with C, for example, $$C1$$$$C2$$$$C3$$$$C4$$ can be used for the measure ADDVMkdMn, ADDVMkdMx etc, while C1 for measure component AD, C2 for DV, C3 for Mkd or Nsls, and C4 for Mn.
- The generated rules have the naming convention of rulePatternName +"_" + ID (left padded with 0 to help sorting), for example., rp1_00001, rp1_00002 etc with rp1 as the rule pattern name.

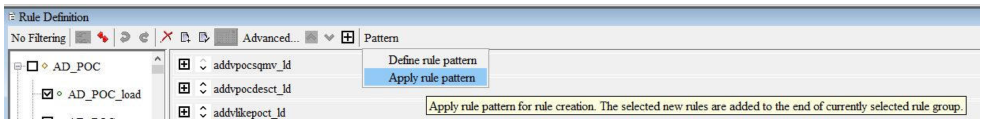
Navigate

In the Configuration Components Pane, select Project, then Solution, and then Rules.

In the Rule Definition Window, right click the New Pattern menu to display two menu items. The menu item Define rule pattern is always enabled, while the menu item Apply rule pattern is only enabled if a rule group is pre-selected.

1. Open the Pattern menu

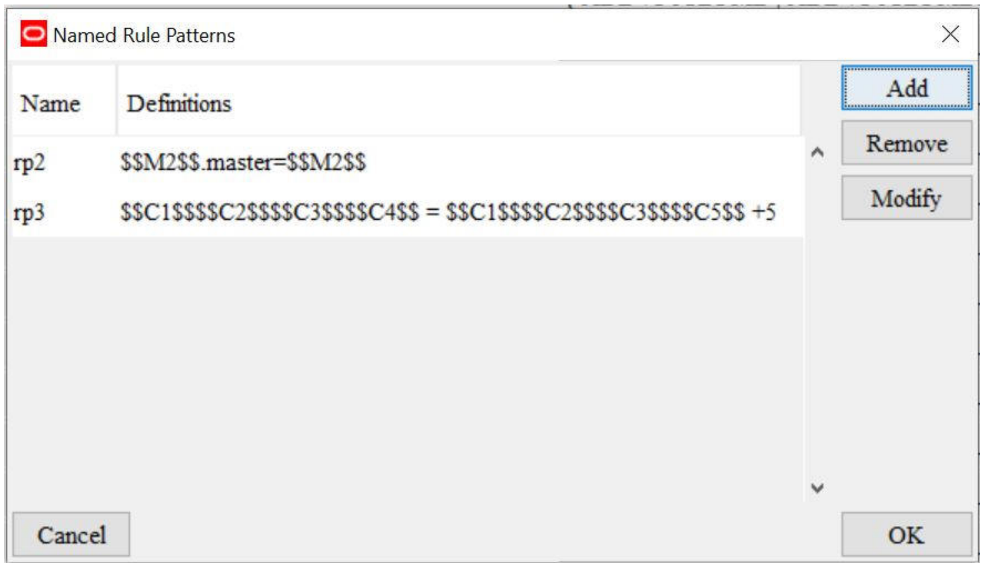
Figure 4-62 New Rule Pattern Menu



Apply rule pattern for rule creation. The selected new rules are added to the end of currently selected rule group.

2. Select Define rule pattern. Use the pop-up that is displayed to add, modify or remove rule patterns.

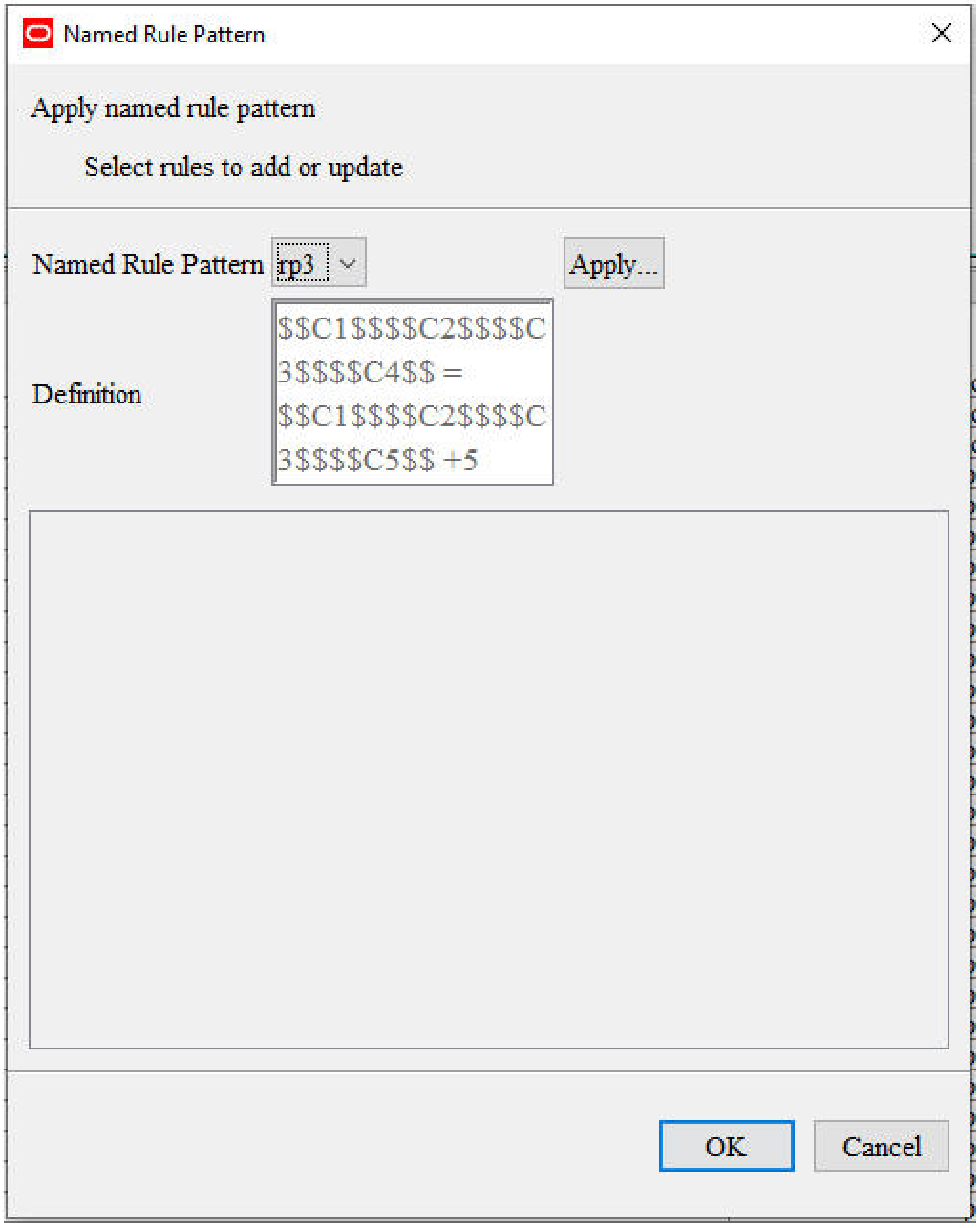
Figure 4-63 Named Rule Patterns



Name	Definitions
rp2	$$M2$$.master=$$M2$$
rp3	$$C1$$$$C2$$$$C3$$$$C4$$ = $$C1$$$$C2$$$$C3$$$$C5$$ +5

3. Pre-select a rule group where the rules are to be added or modified. When you select the Apply Rule Patterns menu item, the Named Rule Pattern dialog box is displayed.

Figure 4-64 Named Rule Pattern Dialog Box



Named Rule Pattern

Apply named rule pattern

Select rules to add or update

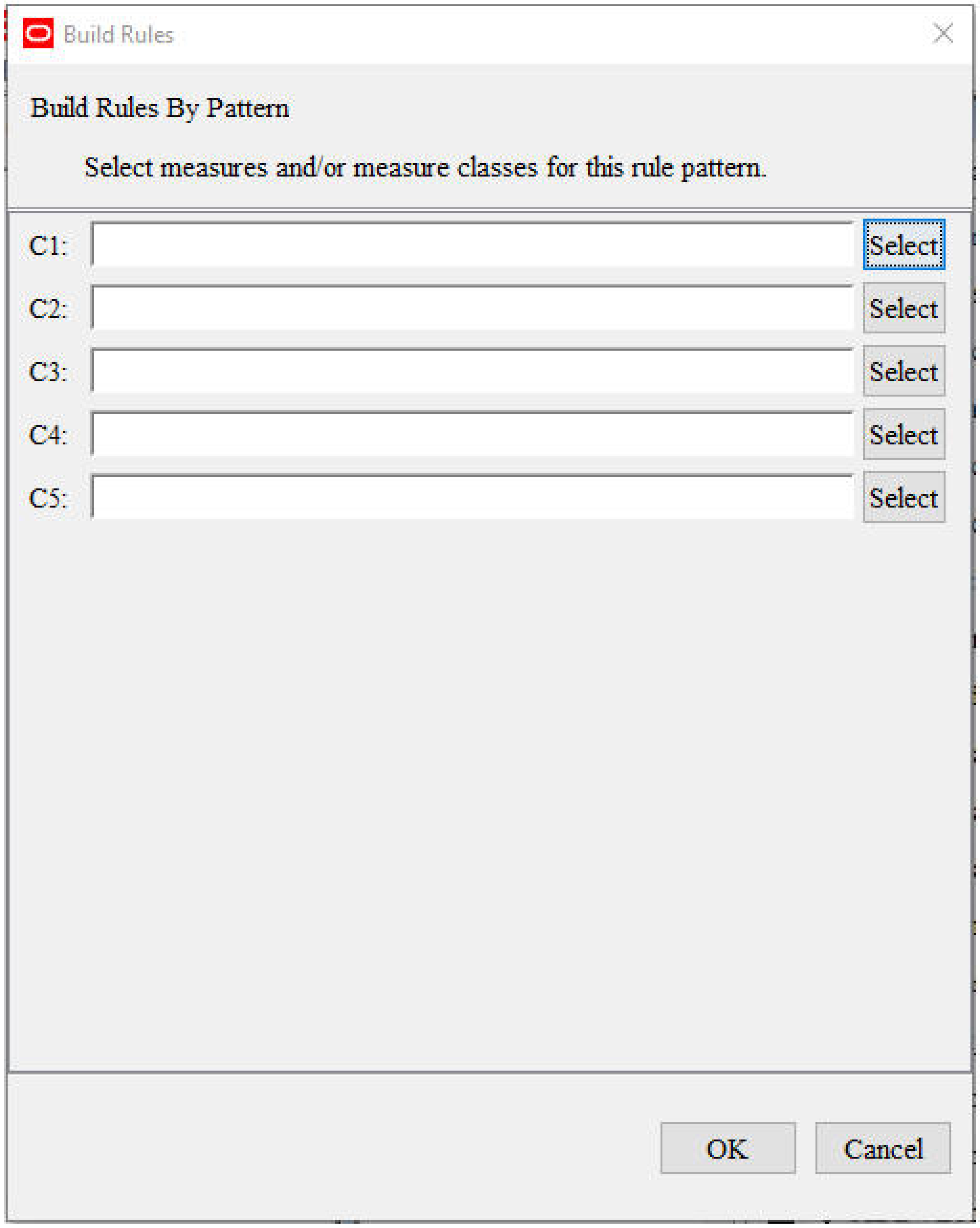
Named Rule Pattern: rp3

Definition: $$C1$$$$C2$$$$C3$$$$C4$$ = $$C1$$$$C2$$$$C3$$$$C5$$ +5

Select a named rule pattern rp3 from the drop-down list. The pattern definition is shown in the Read-Only text area below.

4. Click the Apply button to display the Build Rules dialog box.

Figure 4-65   Build Rules Dialog Box



Build Rules By Pattern

Select measures and/or measure classes for this rule pattern.

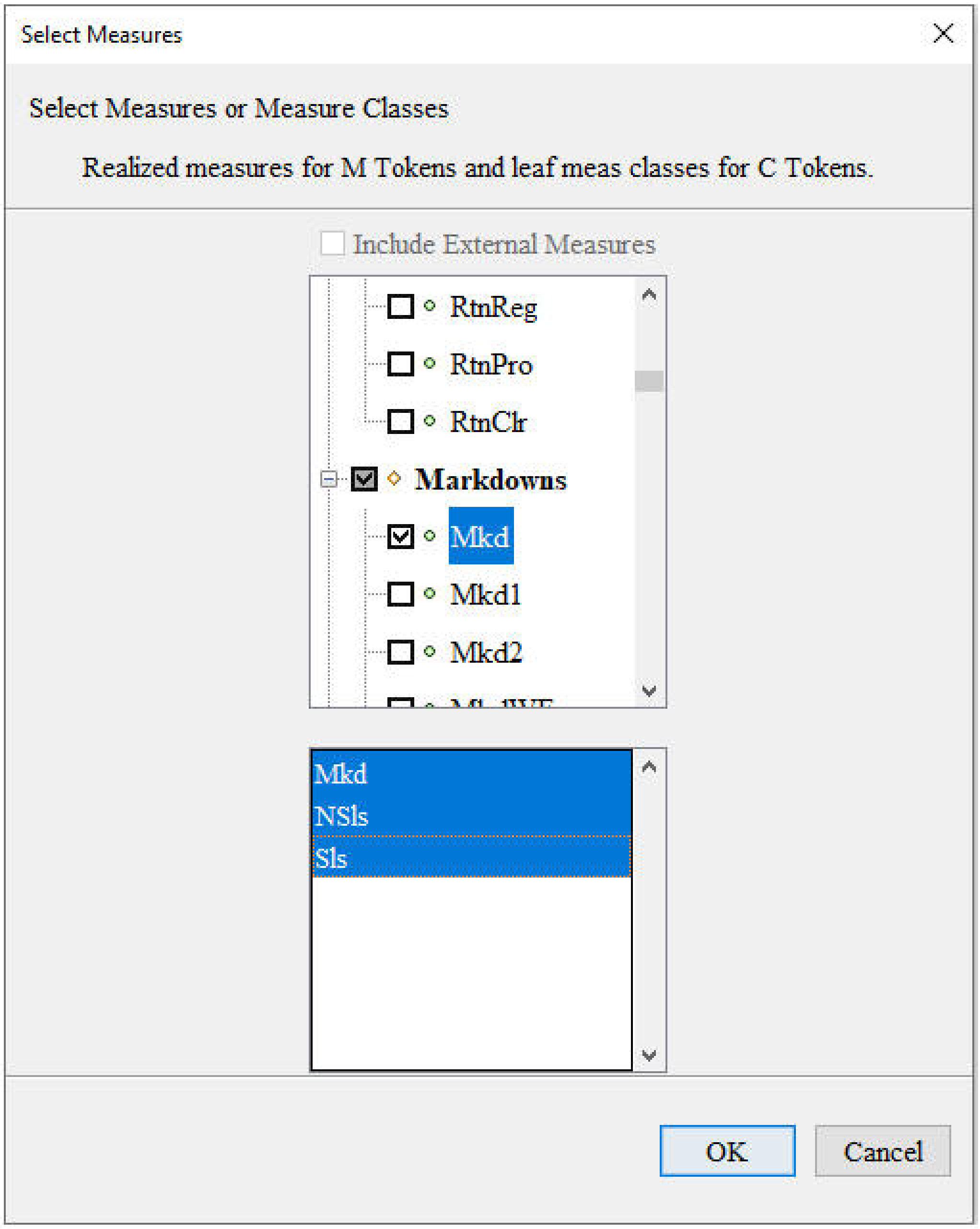
C1: Select
C2: Select
C3: Select
C4: Select
C5: Select

.

5. In the above dialog box, the user can either type token values directly into the blank text field, or click **Select** to display the Select Measures dialog box..

   The check box for Include External Measures is enabled for M tokens, that is, realized measure tokens. The user must explicitly check it to include external measures for this solution into selection.

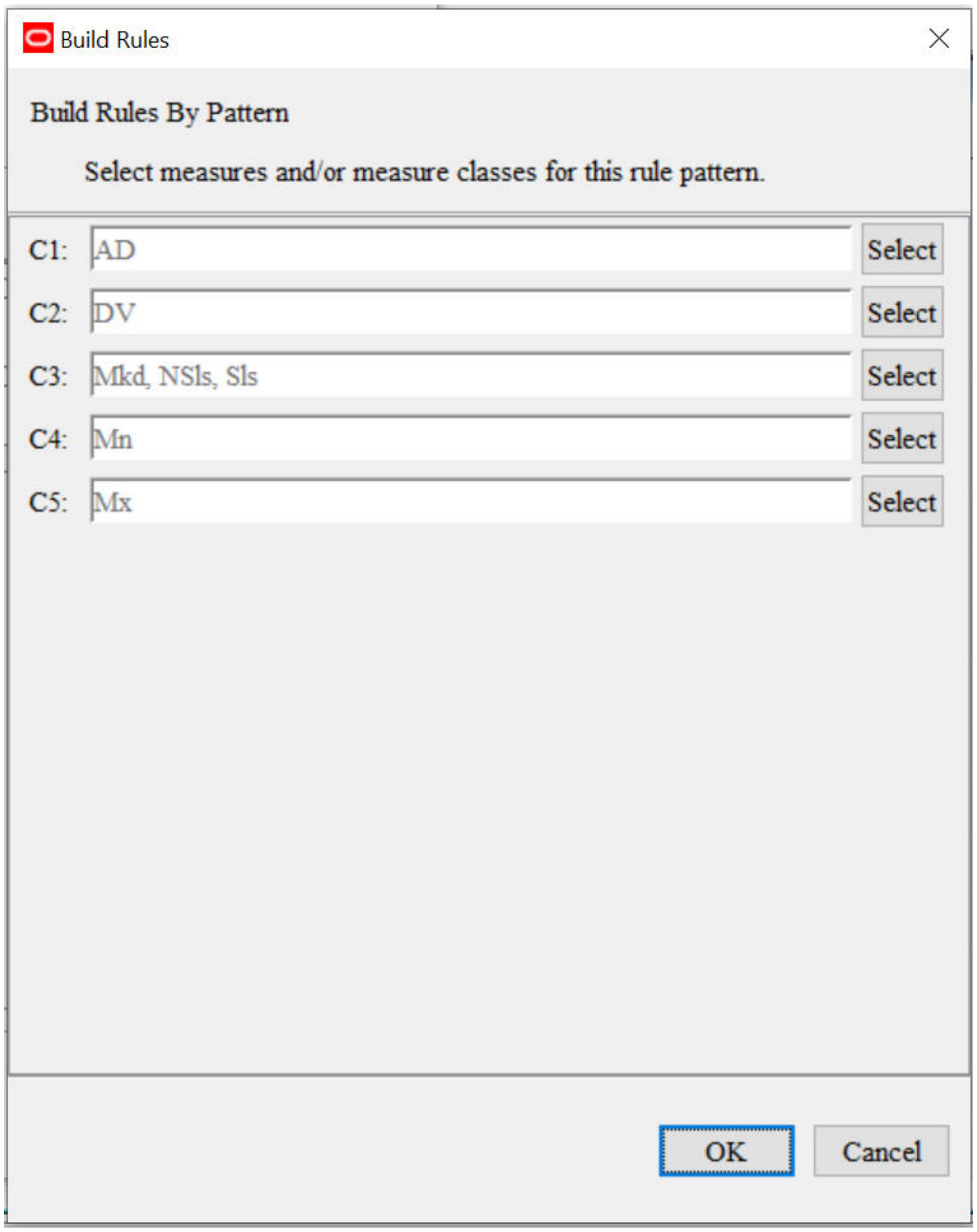
Figure 4-66 Select Measures Dialog Box


Select Measures
Select Measures or Measure Classes
Realized measures for M Tokens and leaf meas classes for C Tokens.
Include External Measures
RtnReg
RtnPro
RtnClr
Markdowns
Mkd
Mkd1
Mkd2
Mkd
NSls
Sls
OK
Cancel

Use the top tree pane to select measure components. The bottom pane displays the selected realized measures for M tokens or leaf measure classes for C tokens. For C tokens, all leaf measure classes of the selected measure components are displayed in the bottom pane. As shown, multiple leaf measure classes Mkd and Nsls are selected for token C3.

6. Once values for all tokens are selected, the user must click **OK** to accept them. Then, the Build Rules dialog box displays these values.

Figure 4-67 Build Rules Dialog Box



Build Rules

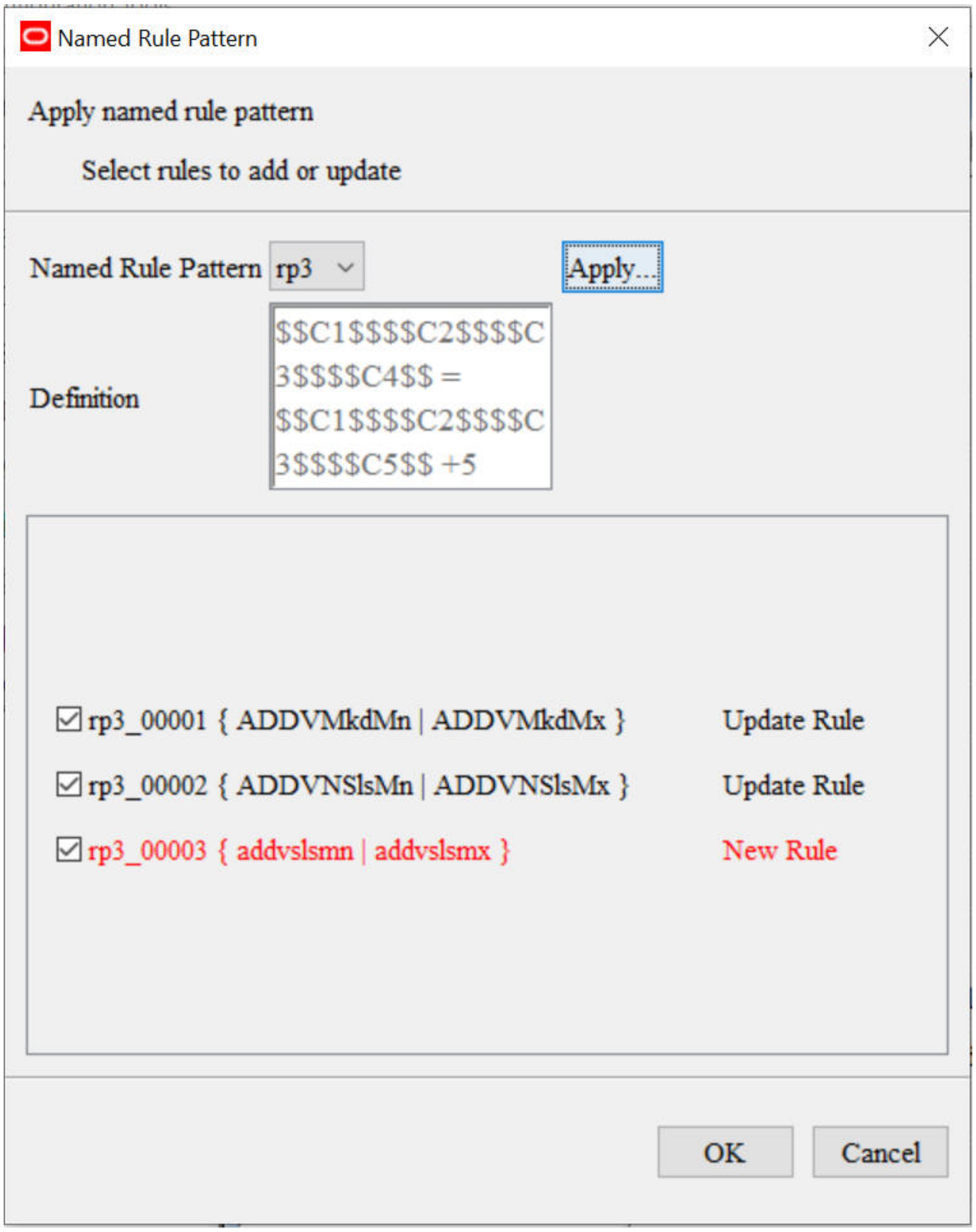
Build Rules By Pattern

Select measures and/or measure classes for this rule pattern.

C1:	AD	Select
C2:	DV	Select
C3:	Mkd, NSls, Sls	Select
C4:	Mn	Select
C5:	Mx	Select

7. Click **OK** to accept the token values. You see the Named Rule Pattern dialog box with the candidate rules generated.

Figure 4-68 Generated Candidate Rules



Named Rule Pattern

Apply named rule pattern

Select rules to add or update

Named Rule Pattern: rp3

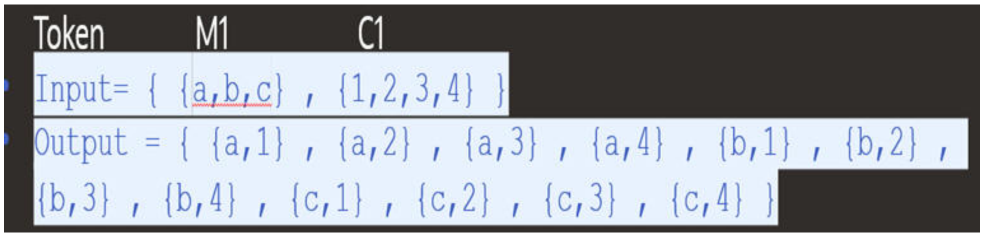
Definition:

```
$$C1$$$$C2$$$$C3$$$$C4$$ = $$C1$$$$C2$$$$C3$$$$C5$$ +5
```

☑ rp3_00001 { ADDVMkdMn | ADDVMkdMx }	Update Rule
☑ rp3_00002 { ADDVNSlsMn | ADDVNSlsMx }	Update Rule
☑ rp3_00003 { addvslsmn | addvslsmx }	New Rule

The candidate rules are generated based on the permutation of all selected token values.

Figure 4-69 Permutation of Selected Token Values



```
Token        M1          C1
Input= { {a,b,c} , {1,2,3,4} }
Output = { {a,1} , {a,2} , {a,3} , {a,4} , {b,1} , {b,2} ,
{b,3} , {b,4} , {c,1} , {c,2} , {c,3} , {c,4} }
```

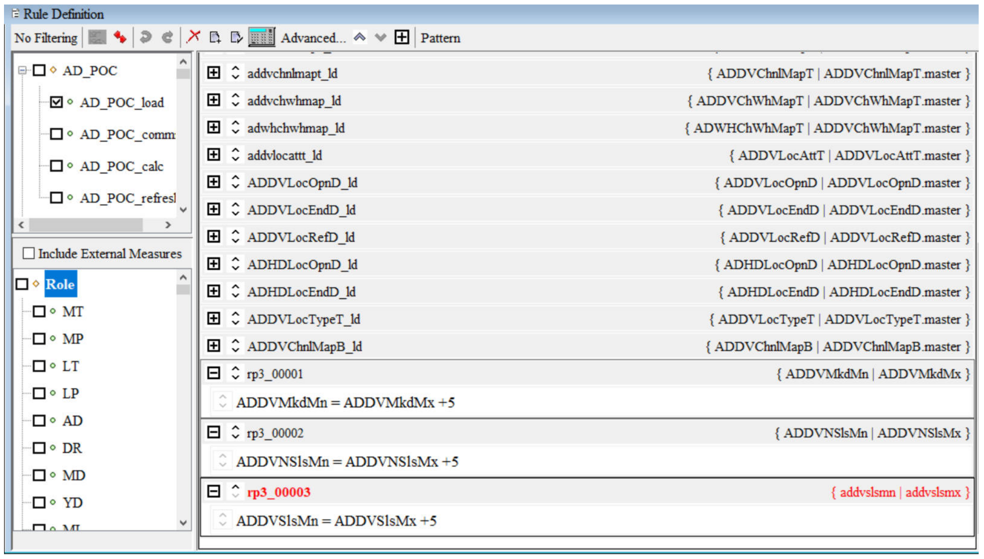
All generated candidate rules are displayed in the bottom Rule list pane, with the check box in front of each rule. By default, the check box in front of each rule is selected. The invalid rules are displayed in RED. The users can inspect and deselect candidate rules that they do not want to include in the pre-selected rule group.

If the pre-selected rule group already contains a rule having same rule signature (the same left hand side measures and the same right hand side measures) as the candidate rule, then this candidate rule has status of Update Rule instead of New Rule. Select this kind of candidate rule to update the existing rule in the rule group with the new rule definition.

The generated candidate rules rp3_00001 and rp3_00002 shown above have the status of Update Rule as they have the same rule signature as in existing rules rp4_00001 and rp4_00002 in the rule group AD_POC_LOAD.

8. Once the candidate rules are selected, click **OK** to confirm it. Then, the pre-selected rule group will contain the new and updated rules.

Figure 4-70 New and Updated Rules



```
addvchnlmapt_Id        { ADDVChnlMapT | ADDVChnlMapT.master }
addvchwhmap_Id         { ADDVChWhMapT | ADDVChWhMapT.master }
adwhchwhmap_Id         { ADWHChWhMapT | ADDVChWhMapT.master }
addvlocattt_Id         { ADDVLocAttT | ADDVLocAttT.master }
ADDVLocOpnD_Id         { ADDVLocOpnD | ADDVLocOpnD.master }
ADDVLocEndD_Id         { ADDVLocEndD | ADDVLocEndD.master }
ADDVLocRefD_Id         { ADDVLocRefD | ADDVLocRefD.master }
ADHDLocOpnD_Id         { ADHDLocOpnD | ADHDLocOpnD.master }
ADHDLocEndD_Id         { ADHDLocEndD | ADHDLocEndD.master }
ADDVLocTypeT_Id        { ADDVLocTypeT | ADDVLocTypeT.master }
ADDVChnlMapB_Id        { ADDVChnlMapB | ADDVChnlMapB.master }
rp3_00001              { ADDVMkdMn | ADDVMkdMx }
  ADDVMkdMn = ADDVMkdMx +5
rp3_00002              { ADDVNSlsMn | ADDVNSlsMx }
  ADDVNSlsMn = ADDVNSlsMx +5
rp3_00003              { addvslsmn | addvslsmx }
  ADDVSlsMn = ADDVSlsMx +5
```

The invalid rule rp3_00003 is highlighted in RED. Its validation error/warning message is displayedin the bottom Task List Pane so that the user can inspect and correct this rule.

RPASCE Functions, Procedures, Keywords, and Modifiers

RPASCE functions, procedures, keywords, and modifiers are mechanisms for completing operations within an expression that are controlled and executed by the calculation engine. There is a rich collection of available functions, procedures, keywords, and modifiers that can be further extended for an implementation if required.

See Appendix – Rules Function Reference Guide for details about RPASCE functions, procedures, keywords, and modifiers.

Workbooks

Note:

The following section contains descriptions of features that are not supported in the current version of RPASCE such as Workbook Transitions, use of the library and custom wizards, style overrides, Detail Popup or Tiled View worksheets, and Auto PQD and Lock PQD Dimensions properties. However, in order to assist in the process of migration of a pre-CE solution to the CE platform, the configuration components associated with these deprecated features remain present within the RPASCE Configuration Tools and therefore continue to be described within this guide. For additional information, see Appendix – Calculation Engine User Guide.

A workbook is an easily viewed, easily manipulated multidimensional framework that is used for interactive business functions in the configured solution. To present data, a workbook can contain any number of multidimensional spreadsheets, called worksheets, as well as graphical charts and related reports. All these components work together to allow you to view and analyze business functions.

The Workbook Designer allows for the creation selection, and integration of the various components of a workbook template, which is a pre-designed workbook that is formatted for RPAS CE users to view and manipulate data. It contains workbook tabs, worksheets, rule groups, wizards, and workflow processes.

Take the time to design a well-planned workbook. Workbooks must be laid out in a logical format and must be easy to navigate. When configuring a workbook, think about how the workbook will be used by the users in the RPASCE Client. Understand the business process flow and what end users will need to access most. Most likely, this information must be contained in the first workbook tab and worksheet.

The names of all of the workbook components must be intuitive to an end user.

Note:

The internal RPASCE names need to be unique across all workbook components in a project. This includes workbook, tab, worksheet, wizard, and custom menu names.

Workbook Components

The following sections describe components found in workbooks:

- Workbook Tabs
- Worksheets
- Workbook Designer

Workbook Tabs

A workbook tab is a major subdivision of a workbook. Each workbook contains at least one workbook tab by default, but additional tabs can be added for organizing workbooks to support business needs. The workbook designer allows you to define and name tabs and to specify their order in the workbook.

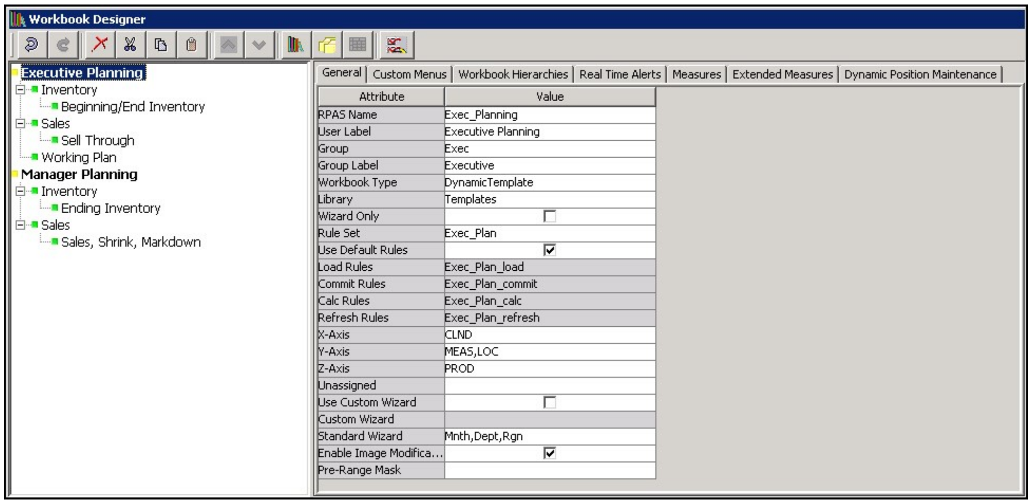
Worksheets

Worksheets are multidimensional spreadsheets that are used to display workbook-specific information. Workbooks can include one or many worksheets. Worksheets can present data in the form of numbers in a grid, or the numeric data values can be converted to a graphical chart.

Workbook Designer

The Workbook Designer provides a visual represent of your workbooks, workbook tabs, and worksheets.

Figure 4-71   Workbook Designer Window



Attribute	Value
RPAS Name	Exec_Planning
User Label	Executive Planning
Group	Exec
Group Label	Executive
Workbook Type	DynamicTemplate
Library	Templates
Wizard Only	☐
Rule Set	Exec_Plan
Use Default Rules	☑
Load Rules	Exec_Plan_load
Commit Rules	Exec_Plan_commit
Calc Rules	Exec_Plan_calc
Refresh Rules	Exec_Plan_refresh
X-Axis	CLND
Y-Axis	MEAS,LOC
Z-Axis	PROD
Unassigned	
Use Custom Wizard	☐
Custom Wizard	
Standard Wizard	Mnth,Dept,Rgn
Enable Image Modifica...	☑
Pre-Range Mask	

The Workbook Designer contains the following areas:

- **Workbook Tree -** The workbook tree provides a visual representation of the workbooks, workbook tabs, and worksheets. In the example provided, Executive Planning and Manager Planning are workbooks. Inventory and Sales are workbook tabs. Beginning/Ending Inventory and Sell Through are worksheets, which are contained in the Inventory and Sales workbook tabs.

- **Workbook Toolbar** - This toolbar is used to complete common tasks. The buttons available depend on the item selected in the Workbook Designer window.
- **Workbook Tabs** - The workbook tabs are used to define property at the workbook level. The tabs displayed depend on whether a workbook, workbook tab, or worksheet is selected from the Workbook tree. In Figure 4-71, Executive Planning (a workbook) is selected. The seven workbook tabs displayed are available to define specific properties for your workbook. For information on these tabs, refer to Workbook Tabs.

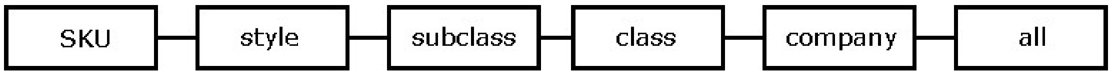
Participation Measures Overview

An extended measure or participation measure is a measure that contains the value of the current positions as a proportion of the value at a Parent level, for example, sales as a percent of the class sales. These measures can be viewed and edited, and they may be preconfigured through the RPASCE Configuration Tools or dynamically defined in the RPASCE Client in a worksheet.

Typical uses of this functionality are to define measures that are percentage participations of sales measures. Typically, these are either to a fixed level (such as class) so the participation of each item to the class can be viewed and manipulated, or they are to the next level up in the product hierarchy.

The following procedures use the sample product hierarchy structure as shown in Figure 4-72

Figure 4-72 Sample Product Hierarchy Structure



Note the following important points when using this feature:

- Changing the percentage of the extended measure will cause the values of the underlying measure to change to reflect the newly set percentage.
- Multiple extended measures can be defined for the same underlying measure; however, only one extended measure or the underlying measure can be edited before calculation occurs. All other versions will be protected.
- The value of an extended measure is a fraction between zero and one. You must format the measure to be displayed as a percentage if desired.

Create a Workbook

Complete the following steps to create a workbook.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Workbooks**. The Figure 4-71 opens in the workspace.

1. Choose one of the following methods:
   - From the Workbook Designer toolbar, click **New Workbook**.
   - Right-click in the Workbook tree area and select **New Workbook**.

- Select an existing workbook and press **Insert**.

  A new workbook is created.

2. Enter information for the tabs displayed across the top of the Workbook Designer window as necessary. Refer to Defining Workbook Properties for more information.

   Note:

   Double-click on the fields in the Value column to enter the information.

Patch Workbook with New Attributes

RPASCE can receive new positions and attributes from other systems via scheduled batch jobs and sometimes on demand. The PDS patching process incorporates this new information into existing PDSs, but not for existing workbooks. Existing workbooks must usually be re-built to pick up the new information, but if the existing workbooks have uncommitted data, that data will be lost.

One way to alleviate the problem is to patch the dimension attributes within existing workbooks from the RPASCE client. To do this, the user must pre-configure the workbook via the Dim Attr Setup Date Measures property in the General tab of the Workbook Design Window. For more information about this property, see the Defining Workbook Properties. Note that this approach only enables patching the defined two-dimensional attributes within the workbook.

Edit Workbook Properties

Complete the following steps to edit workbook properties.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Workbooks**. The Figure 4-71 opens in the workspace.

1. Select the workbook and click on the tab to edit. For information on these tabs, see Defining Workbook Properties.
2. Update the information as appropriate.
3. To remove information from any of the tables:
   a. Select the row.
   b. Right-click and select **Remove**.

Defining Workbook Properties

When a workbook is selected from the Workbook Designer window, the following tabs display in the workspace:

- General Tab
- Custom Menus Tab
- Workbook Hierarchies Tab
- Real Time Alerts Tab
- Workbook Transitions Tab

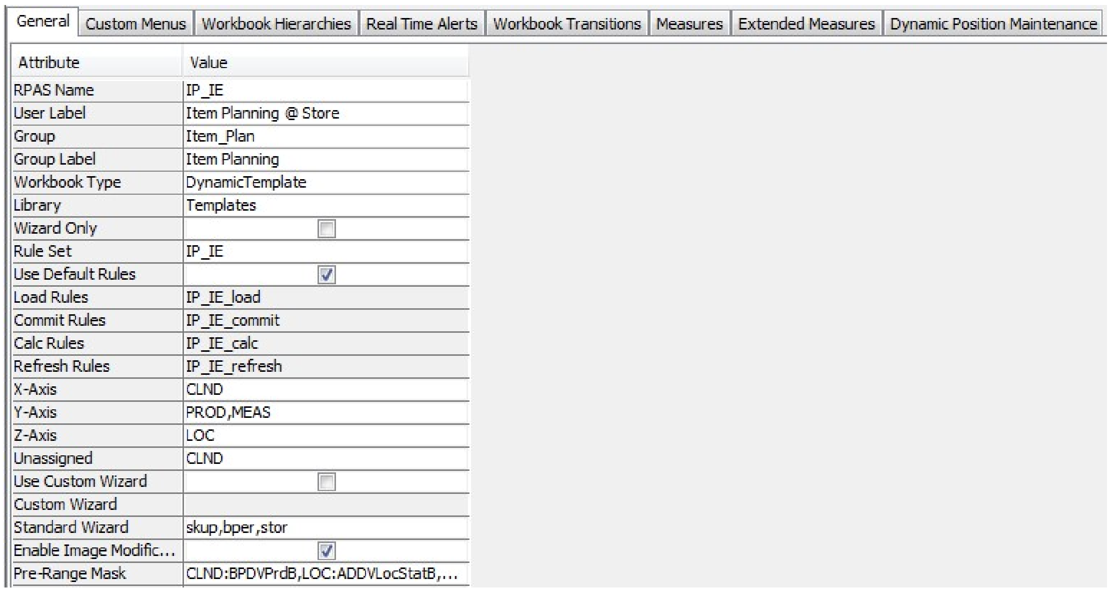
- Measures Tab
- Extended Measures Tab
- Dynamic Position Maintenance Tab

Refer to these topics for information on using these tabs to define the workbook properties.

General Tab

This section describes the default fields that display on the General tab.

Figure 4-73 General Tab



Attribute	Value
RPAS Name	IP_IE
User Label	Item Planning @ Store
Group	Item_Plan
Group Label	Item Planning
Workbook Type	DynamicTemplate
Library	Templates
Wizard Only	☐
Rule Set	IP_IE
Use Default Rules	☑
Load Rules	IP_IE_load
Commit Rules	IP_IE_commit
Calc Rules	IP_IE_calc
Refresh Rules	IP_IE_refresh
X-Axis	CLND
Y-Axis	PROD,MEAS
Z-Axis	LOC
Unassigned	CLND
Use Custom Wizard	☐
Custom Wizard	
Standard Wizard	skup,bper,stor
Enable Image Modific...	☑
Pre-Range Mask	CLND:BPDVPrdB,LOC:ADDVLocStatB,...

RPAS Name

This is the RPAS internal name of the workbook.

User Label

This is the label that the end user sees when selecting which workbook to build.

Group

In the RPASCE Client, workbooks are grouped together under tabs (workbook template groups) to make it easier for the end user to find and select the needed workbook when solutions have multiple workbooks. This is the internal RPAS name of the group that this workbook belongs to.

Group Label

In the RPASCE Client, workbooks are grouped together under tabs (workbook template groups) to make it easier for the end user to find and select the needed workbook when solutions have multiple workbooks. This is the label the end user sees of the group to which this workbook belongs to. If different labels are entered for the same workbook group against different workbooks, the workbook group label shown to the end user will effectively be arbitrary.

Workbook Type

This property is reserved for use when custom extensions are written. It enables the custom extension to determine the type of the template where the template type has a meaning defined by the custom extension writer. When there is no custom extension, this field is set to the value DynamicTemplate by default.

Library

When the Workbook Type is not DynamicTemplate, it needs to be associated with a relevant custom shared library. This field holds the name of that library. When there are no custom extensions, this field is set to Template by default. The name entered here needs to be consistent with the custom extension. For example, if a value of ABCTemplate is entered in this field, the custom library needs to be named ABCTemplateLib and the directory where the custom extension looks for configuration files in the application will be repos/ABCTemplates.

Wizard Only

The Wizard Only option is only used under circumstances when custom code is to be executed in a batch job at the end of the wizard process (which typically uses custom wizards) instead of building and opening a standard workbook. The selections made in the wizards are passed to the custom code. Therefore, a workbook with the Wizard Only option selected is not a workbook. However, the workbook infrastructure is used so that the process can have a name and label and be assigned to a workbook group. This allows end users to select a Wizard Only template using the same process as workbooks. Workbooks that have the Wizard Only option selected do not need tabs or worksheets defined, but they do need a name, label, and workbook group.

Rule Set

Selects the rule set to use with the workbook. The list of rule sets to select from includes all the rule sets in the same solution as the workbook template.

Use Default Rules

Select this option to use the default rules (Load, Commit, Calc, and Refresh) associated with the rule set. If this option is selected, the Load Rules, Commit Rules, Calc Rules, and Refresh Rules properties are unavailable. If the option is not selected, the Load Rules, Commit Rules, Calc Rules, and Refresh Rules properties are enabled.

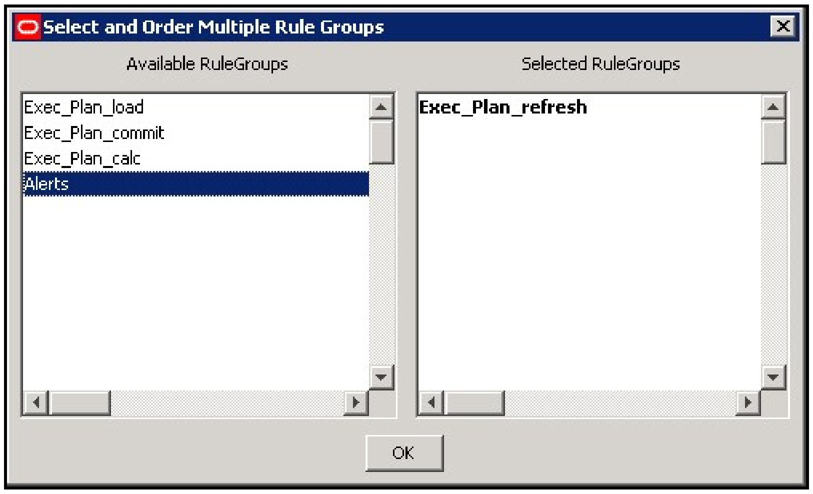
Load Rules, Commit Rules, and Calc Rules

Select the rule group to apply for each rule group type. Only rule groups from the selected rule set are offered.

Refresh Rules

When the Use Default Rules option is not checked, Refresh Rules is enabled. Select the rule groups to use as refresh rule groups. When enabled, click in the Refresh Rules field. The Figure 4-74 opens.

Figure 4-74 Select and Order Multiple Rule Groups Window



Exec_Plan_load
Exec_Plan_commit
Exec_Plan_calc
Alerts

Exec_Plan_refresh

Use this window to specify which groups are available and used to refresh the workbook.

Partial data in a workbook can be refreshed by refreshing with a rule group that only updates some of the measures in the workbook. The order that the rule groups appear is the order in which they display to the end user when presented with a choice of refresh rule groups within the workbook.

1. In the **Available Rule Groups** column, select the rule group to add, and drag it to the **Selected Rule Groups** column.
2. Click **OK** to save any changes and close the window.

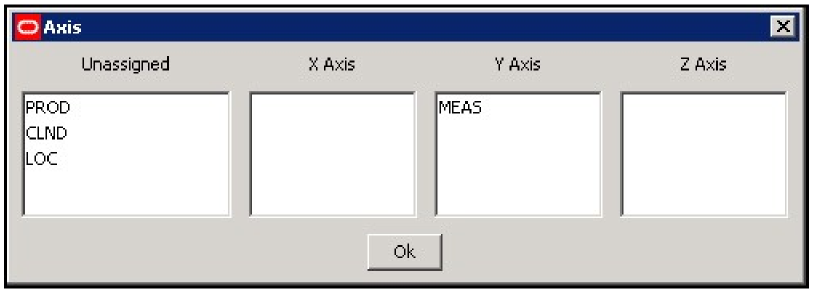
X Axis, Y Axis, Z Axis and Unassigned

These properties are used to define the default axis layout of the worksheets in the workbook (that is, which hierarchies appear in each axis). Before the hierarchies appear in the Axis window, you must make sure that the database and base intersection have been assigned from the Measure Manager.

The measures must also be made viewable.

1. To make measures viewable, right-click in **Default** and select **Add Matching**.
2. Click in the **X-Axis**, **Y-Axis**, **Z-Axis**, or **Unassigned** field. The Figure 4-75 opens.

Figure 4-75 Axis Window



3. Drag the hierarchies to the appropriate axis column.
4. Click **OK** to save any changes and close the window.

> **Note:**
>
> The hierarchies that appear in this process are the hierarchies used by measures placed on worksheets in the workbook. If no worksheets have yet been built, then no hierarchies appear in this process.

Use Custom Wizard

This determines the type of wizard to use for the workbook template. Select the check box to enable the Custom Wizard property and disable the Standard Wizard Property. Clear the check box to disable the Custom Wizard property and enable the Standard Wizard Property. See Wizards for more information on Custom Wizards.

Custom Wizard

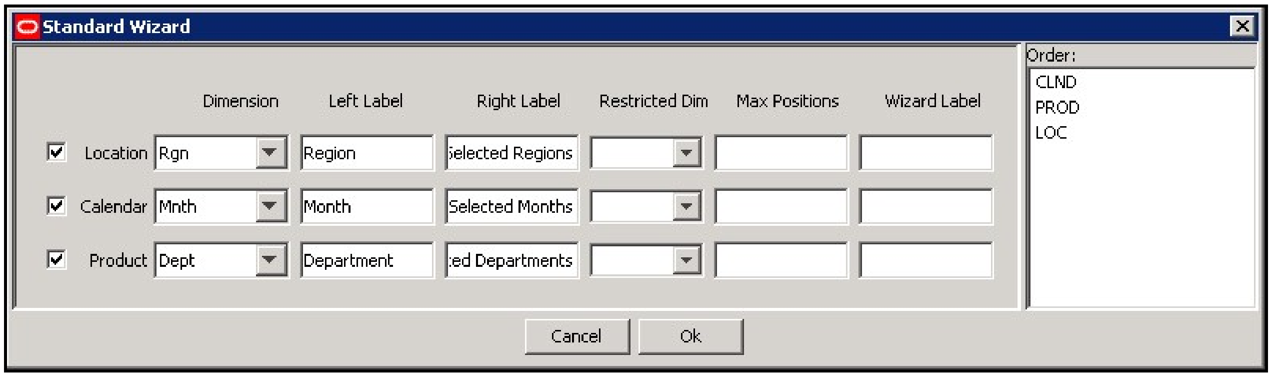
Select the custom wizard that to use to build the workbook. This field is only enabled when the Use Custom Wizard check box is selected. This field allows you to select a wizard from a list of wizards created in the Wizard Designer.

Standard Wizard

Select the dimensions to be selected by the end-user in the standard wizard, which presents a series of two tree selection panes to select the positions in the scope of the workbook to be built.

1. Click in the **Standard Wizard** field. The Figure 4-76 opens.

Figure 4-76 Standard Wizard Window



Standard Wizard

	Dimension	Left Label	Right Label	Restricted Dim	Max Positions	Wizard Label
☑ Location	Rgn	Region	;elected Regions			
☑ Calendar	Mnth	Month	Selected Months			
☑ Product	Dept	Department	:ed Departments			

Order:
CLND
PROD
LOC

Note:

Hierarchies used in the lowest base intersection for the measures used in the rule set assigned to the workbook will be displayed. For each hierarchy, there will be a choice of dimensions. The dimensions offered will be the lowest dimension in that hierarchy that is used in the base intersection of a measure in the rule set assigned to the workbook, plus all higher dimensions.

For a hierarchy in the workbook for which a wizard is not available, all the positions in the PDS will be pulled into the workbook. If position level security has been applied to the user, a subset of the position will be pulled in.

2. Select the check boxes next to the hierarchy names to enable the hierarchies for which that the end user in the RPASCE Client should select positions.
3. **Dimension:** Select the desired dimension from each of the enabled hierarchies.

Note:

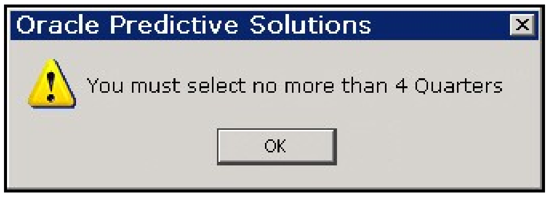
The dimension selected is the lowest dimension offered to the end-user in the scope selection wizard during the workbook build process. However, the workbook requires positions at the lowest dimension offered. Therefore, if the selected dimension is higher than the lowest dimension offered, the scope of the workbook includes all of the positions in that lowest dimension that are descended from the positions selected from that higher dimension.

4. **Left Label and Right Label:** (Optional) Enter the left and right labels for each hierarchy. These labels will be displayed on the left and right trees of the corresponding 2-tree wizards during the workbook build process. If these fields are left blank, no labels will be displayed over the hierarchy trees.
5. **Restricted Dim and Max Positions:** For applications such as pricing, a retailer might want to restrict the selection of SKUs in a planning workbook to only one category. A planner may be allowed to plan several categories within a department, but only one category per plan. Per the retailer's business process, a category would establish a coherent set of SKUs, the cross-item effects of which could be considered meaningful for a price optimization algorithm. Mixing in SKUs from two or more categories could be

considered as polluting the cross-item effects, and therefore an undesirable situation. What may also be required for this application is the ability to select SKUs or Classes that comprise a subset of possible SKUs or Classes within the Category, but not the whole category.

With this latter requirement, a single-select wizard at the category level would not allow the user to filter subsets of SKUs or Classes. What is required is the ability to make multiple selections at the SKU or Class levels in a standard RPASCE Two Tree selection wizard, while still ensuring that only one Category is used. Even though this can be achieved through the disciplined selection of SKUs and Classes in a Two Tree Wizard, RPASCE allows for setting up a hard constraint so that the wizard itself can keep the user from selecting subsets in more than one Category by displaying an error message and preventing the user from proceeding to the next wizard page until the constraint has been satisfied.

Figure 4-77 Error Message


Oracle Predictive Solutions

You must select no more than 4 Quarters

The constraint can be easily established within the Standard Wizard definition window in the Workbook Definition tool of the Configuration Tools. Two new fields are available for every Two Tree selection page in the standard wizard, one where the user selects the level from the hierarchy (Restricted Dim field), and another where the user enters the maximum number of selectable positions from that level (Max Positions field). These fields are optional, and if left empty, there is no limit on the number of positions that may be selected using the wizard. These fields are also available for Two Tree pages used in custom wizards.

Another possible business application of this feature is to constrain the length of the planning period. A retailer may want the planners to never plan more than 12 weeks at a time, and it may not matter whether these weeks belong to the same quarter or not. In such a case, the retailer will want to establish the constraint of 12 at the week level, the lowest level where the selection is made. The planner may select at the quarter level, thus automatically selecting all the weeks in the Quarter; however, RPASCE will ensure that whatever the definition of the Quarter is, it does not contain more than 12 weeks.

Apart from the functional ability to restrict selections to coherent set of positions provided by the Max Positions feature, this feature also allows system designers to constrain the size of workbooks by limiting the maximum number of positions that a user can add to a workbook. In the past, users have been known to add all positions to a workbook because such a selection is easy to make. They may only require 5-10% of those positions, but they still add them all because they can easily work with the desired subset in the workbook. System designers would like to prevent such abuse of the flexibility that workbooks provide, primarily because such abuse leads to wastage of disk space and because it slows down online performance due to the extra work that RPASCE has to do with unnecessary positions. System designers may therefore constrain the workbook to, for example, not include more than 500 SKUs at a time. The number 500 may not have any functional meaning but may be chosen because it does not constrain functionality in any way while still helping to constrain the size of the workbooks.

> **Note:**
>
> This is a patchable feature, for example, existing configurations can be enhanced to benefit from this feature.

6. **Wizard Label:** To attach a custom label to a wizard page, enter the desired label in this field. If this field is left blank, the wizard page will be given the default label.
7. **Order:** Adjust the order of the hierarchies as necessary by dragging them in the order pane. This will be the order that the position selection wizards are presented during the workbook build process.
8. Click **OK** to save any changes and close the window.

> **Note:**
>
> If any of the offered hierarchies are not enabled in the Standard Wizard window, the end user will not be presented with a position selection wizard to define the scope of the workbook being built for that hierarchy. All positions in the lowest dimension offered in that hierarchy that the end user has access rights to will be automatically selected.
>
> For a hierarchy in the workbook for which there is not a wizard, all the positions in the PDS will be pulled into the workbook. If position level security has been applied to the user, then a subset of the positions is pulled in.

Enable Image Modification

Select this option to allow users the ability to add, modify, or delete image paths for all image enabled dimensions in this workbook.

> **Note:**
>
> The commit rule group will receive additional rules when the Image Modification flag is selected for a workbook template. The load/refresh rule groups will receive additional rules if the workbook contains image enabled dimension levels, regardless of the Image Modification flag. If no refresh rule group is assigned to a workbook template, RPASCE will automatically create an internal rule group and assign that as the workbook's default refresh rule group.

Hierarchy Pre-Ranging

Hierarchy pre-ranging allows you to filter available positions for selection in two-tree wizards based on relationships established at a specific intersection between the positions of two or more hierarchies. For example, you can set pre-ranging up so that when users select the time period Fall 09 in the wizard, the subsequent SKU selection screen will only display Fall-specific products, such as sweaters and jackets.

Hierarchy pre-ranging can be enabled for both standard wizards and for two-tree wizards in custom wizards; however, in custom wizards the behavior is guaranteed only if two-tree wizards are used as is, that is, their code is not overridden by the implementation team.

Pre-ranging is achieved by setting up one or more Pre-range Mask measures for the workbook. Each hierarchy in the workbook template can be optionally assigned a pre-range mask measure. Each of the hierarchy-specific mask measures must contain a dimension that ties back to one or more hierarchies that display in the wizard, along with one or more dimensions along which we want to pre-select relevant positions. Multiple hierarchies can share a common mask measure. The masking function is applied in the order of the wizard pages such that the selections of a wizard page may affect the list of available positions on any subsequent wizard page, but the selections of wizard page do not affect the list of available positions on a previous page if you click **Back** to return.

This approach allows:

- One or more hierarchies within the workbook template not to be ranged, reducing the processing time and storage space.
- The mask measure only needs to contain the target hierarchy plus optional conditioning hierarchies at its base intersection, reducing the number of hierarchies per mask measure and thus storage space as well.
- The same mask measure can be used to range a common set of hierarchies in different workbook templates.

Example:

In a workbook template that contains PROD, LOC, and CLND hierarchies, two different mask measures are used: measure `prodmaskloc` (sku,str) to range down the positions that display in the LOC wizard, and measure `locmaskclnd` (str,week) to range down the positions that display in the CLND wizard. In the ConfigTools, “PROD:,LOC:`prodmaskloc`,CLND:`locmaskclnd` are specified in the Pre-Range Mask field under the General tab in the Workbook Designer tool.

Assuming the order of the wizard pages is PROD, LOC, and CLND, this is how the masks are applied:

- All positions in the PROD hierarchy are available for selection as PROD is not masked.
- Available positions for LOC hierarchy depend on the PROD hierarchy selections combining with the masking values of measure `prodmaskloc`.
- Available positions for CLND hierarchy will solely depend on the LOC hierarchy selections combining with the masking values of measure `locmaskclnd`. The PROD selections do not affect the CLND page directly.

The following sections describe additional components of Hierarchy pre-ranging:

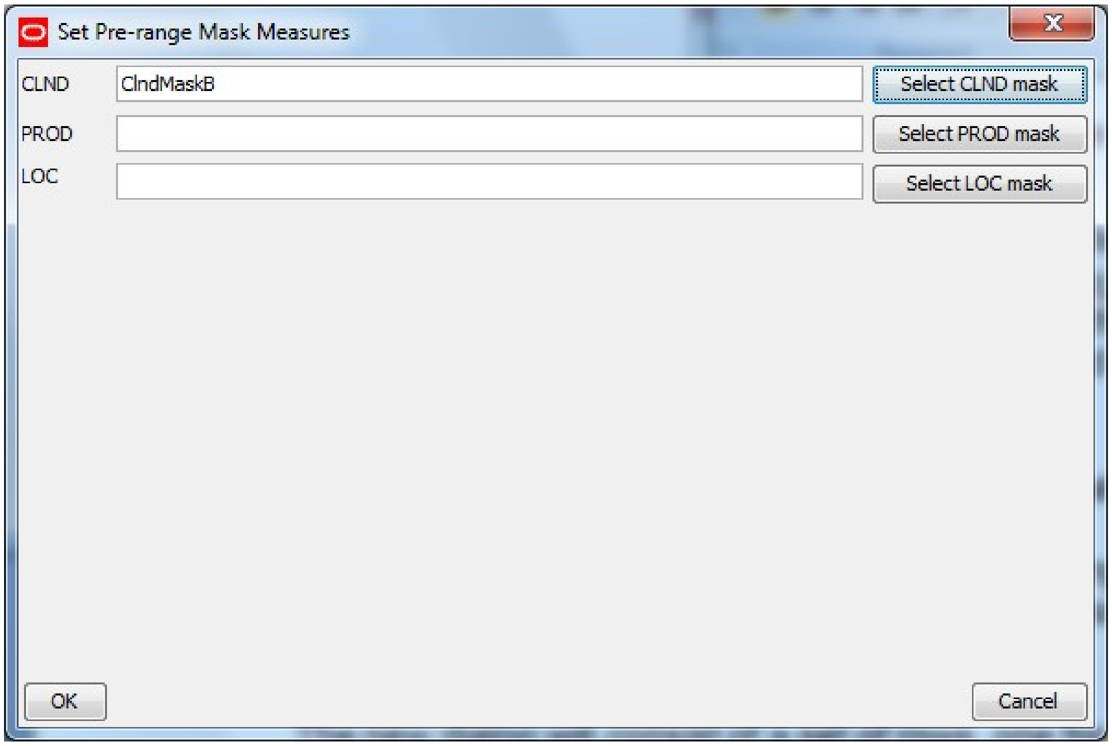
- Hierarchy Specific Pre-Range Mask Assignment
- Mask Measure's Type and Properties
- Driving Hierarchies Based on the Mask Measure's Base Intersection
- Evaluation of a Pre-Range Mask in the Wizard

- Hierarchy Without the Wizard
- Mask Measure Error
- Backward Compatibility with Existing Workbook Templates
- No Available Position after Pre-Ranging

Hierarchy Specific Pre-Range Mask Assignment

To assist in the configuration of the pre-range mask use the Figure 4-78 for the pre-range mask attribute. If you click in the field, the Figure 4-78 opens.

Figure 4-78 Set Pre-Range Mask Measures Window



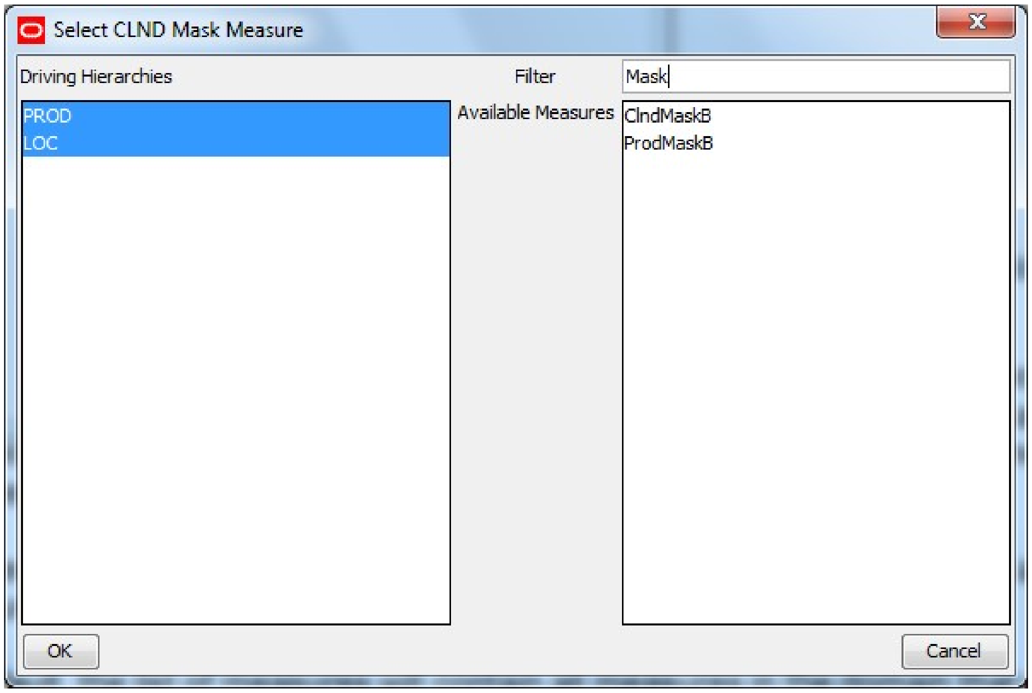
CLND	ClndMaskB
PROD	
LOC	

The use of this tab is optional. It is used to create a workbook specific, customized menu-driven process within

The Figure 4-78 consists of a set of rows, one for each hierarchy in the application, each row lists the name of the hierarchy associated with the row and provide a field in which a measure name can be entered.

Finally, each row has a button that allows the selection of a measure as opposed to typing in a measure name. Click **Select CLND mask**, **Select PROD mask**, or **Select LOC mask** to open a Mask Measure Selection window. Figure 4-79 shows the Select CLND Mask Measure window.

Figure 4-79   Mask Measure Selection Window



A Figure 4-79 has a list of hierarchies, a list of measures and a text field to accept a partial measure name as a filter.

By default, the list of measures will contain all Boolean measures in the application that meet validation criteria for the hierarchy being configured. Typing text in the filter text field will filter the measure list to remove those measures that do not meet the pattern. Additionally, selecting a hierarchy in the hierarchy list will remove all measures that do not also contain the selected hierarchy or hierarchies within its base intersection.

These controls are designed to handle the values present in an upgraded version of the configuration in which the older and simpler format for pre-range masking is present.

Mask Measure's Type and Properties

All pre-range masking measures must be of Boolean measure type, must have an aggregation method of ***or*** or ***and***, and cannot be a scalar measure. Measure intersection can be equal, higher than, or lower than the wizard intersection (intersection constructed from the base dimensions of the wizards). The RPASCE server uses normal non-conforming measure handling if the measure intersection is different from the wizard intersection. In other words, an aggregation is performed using the given agg type when the measure intersection is lower than the wizard intersection. On the other hand, if it is higher than the wizard intersection, then cells are mapped using the replicate method (the measure spread type is ignored). A measure whose intersection contains a mixture of dimensions higher than the wizard intersection and lower than the wizard intersection (for example, `sku_rgn_week` for measure intersection versus `sku_str_mnth` for wizard intersection) cannot be used as a Pre-Range Mask measure.

Driving Hierarchies Based on the Mask Measure's Base Intersection

Any additional hierarchies in the mask measure's base intersection are automatically assumed as conditioning hierarchies. For example, if a measure that has PROD, LOC, and CLND is assigned to range LOC wizard, then PROD and CLND are assumed to be conditioning hierarchies, and the positions selected in these hierarchies will determine the position available in LOC wizard.

Evaluation of a Pre-Range Mask in the Wizard

The masking intersection for the mask measure during evaluation is based on ordering of the wizards. It will only gather intersection and positions information from previous wizards. For example, if the wizard ordering is CLND, LOC, PROD, then at the LOC wizard, even if the user navigated to PROD wizard and then went back to LOC wizard, the masking evaluation only considers the selections made in the CLND wizard. Other hierarchies in the mask measure at the time of evaluation are considered all selected (unfiltered).

Hierarchy Without the Wizard

A workbook template can omit a wizard for a hierarchy where all the positions in that hierarchy will be included in the workbook by default. A mask measure can still be assigned for such a hierarchy and the positions to be included in the workbook will be determined by the masking evaluation. This operation is performed at the end of wizard process.

Mask Measure Error

Since the mask measure is set for each hierarchy, in the case when measure properties were changed from the backend, any error during the execution, like incorrect measure type, intersection, or agg type, will only disable the ranging for that hierarchy. A warning message will be logged but the workbook build operation will continue. The hierarchy where the error occurs will become un-masked.

Backward Compatibility with Existing Workbook Templates

When RPASCE has been upgraded but the PDS has not yet been patched, the field `prerangemask` in the template configuration file (`tmpl.cfg`) is still using the old format with only one single measure in it. The masking result is the same as before the RPASCE upgrade. It is essentially equivalent to assigning the single masking measure to all hierarchies in the workbook template.

No Available Position after Pre-Ranging

If there is no available position in a wizard during the workbook build process after masking evaluation, a CancelWizardException with an appropriate error message is thrown which can be displayed to the end user. The workbook building process ends.

DPM Like-Item Measure

RPASCE supports the use of one-dimensional string measures to store like-item information for newly added DPM positions. Such measures are called “DPM like-item measures" and must be configured in the workbook General Property tab. Once the like-item information, such as the like-item position names, is stored in the like-item measures, it can be used during workbook calculations or custom operations to populate other attribute measures.

> **Note:**
>
> Dynamic Position Maintenance is referred to as ***Placeholder Maintenance*** in the *Oracle Retail Predictive Application Server Cloud Edition User Guide* and several other documents. Both terms refer to the same functionality.

A DPM-like item measure must meet the following criteria:

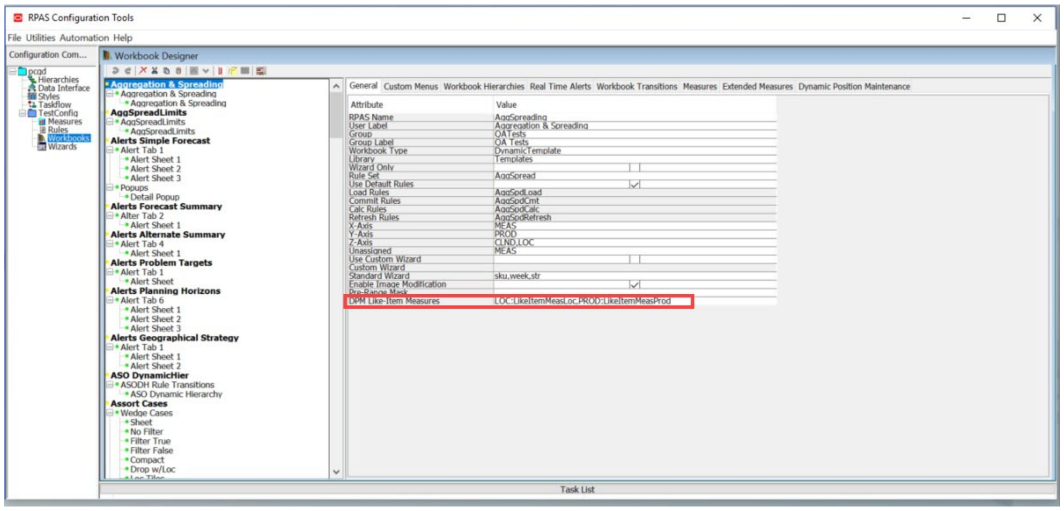
- It is a one-dimensional, string type measure.
- Its base dimension cannot be a workbook-only dimension since such a dimension does not exist in the PDS and cannot have the like-item measure configured.
- Its base dimension is one of the workbook's DPM-enabled dimensions.
- It can be a workbook-only measure and is used to satisfy workbook calculation or custom menu operations. However, this measure must be part of the workbook and must be in the workbook rule set.

The DPM like-items are configured in the workbook General Property table. A window is displayed once the user presses either the mouse or the keyboard in this field. In this window, the user can select one like-item measure for each DPM-enabled hierarchy in the workbook. The DPM-enabled hierarchy in the workbook is determined by the configuration in the Dynamic Position Maintenance tab.

Example:

In a workbook template, DPM is enabled for the `styl` (PROD Hierarchy) and `str` (LOC Hierarchy) dimensions in the Dynamic Position Maintenance tab. Two string measures, `LikeItemMeasProd` (`styl`) and `LikeItemMeasLoc` (`str`), are configured and added to the workbook load rules.

Figure 4-80 DPM Like-Item Measures



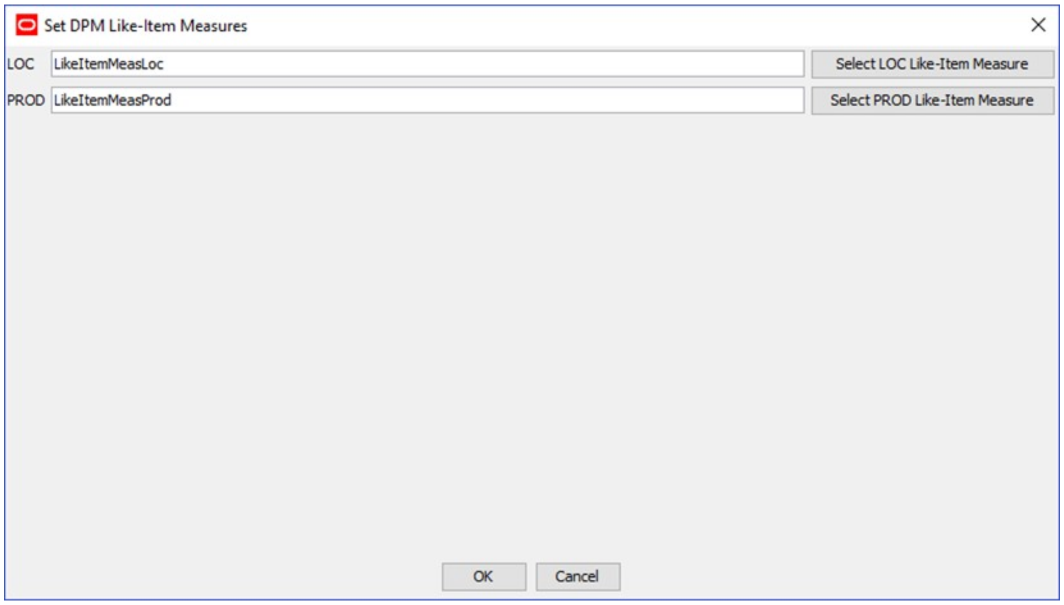
Attribute	Value
RPAS Name	AggSpreading
User Label	Aggregation & Spreading
Group	OATests
Group Label	OA Tests
Workbook Type	DynamicTemplate
Library	Templates
Wizard Only	☐
Rule Set	AggSpread
Use Default Rules	✓
Load Rules	AggSpdLoad
Commit Rules	AggSpdCmt
Calc Rules	AggSpdCalc
Refresh Rules	AggSpdRefresh
X-Axis	MEAS
Y-Axis	PROD
Z-Axis	CLND,LOC
Unassigned	MEAS
Use Custom Wizard	☐
Custom Wizard	
Standard Wizard	sku,week,str
Enable Image Modification	✓
Pre-Range Mask	
DPM Like-Item Measures	LOC:LikeItemMeasLoc,PROD:LikeItemMeasProd

In the ConfigTools, `LOC:LikeItemMeasLoc, PROD:LikeItemMeasProd` is specified in the **DPM Like-Item Measures**field under the General tab in the Workbook Designer tool.

Hierarchy Specific DPM Like-Item Measure Assignment

In order to assist in the configuration of the DPM like-item measure, use the Figure 4-81. If you click in the **DPM Like-Item Measures** field, the Figure 4-81 opens.

Figure 4-81 Set DPM Like-Item Measures Window


Set DPM Like-Item Measures
Select LOC Like-Item Measure
Select PROD Like-Item Measure
OK
Cancel

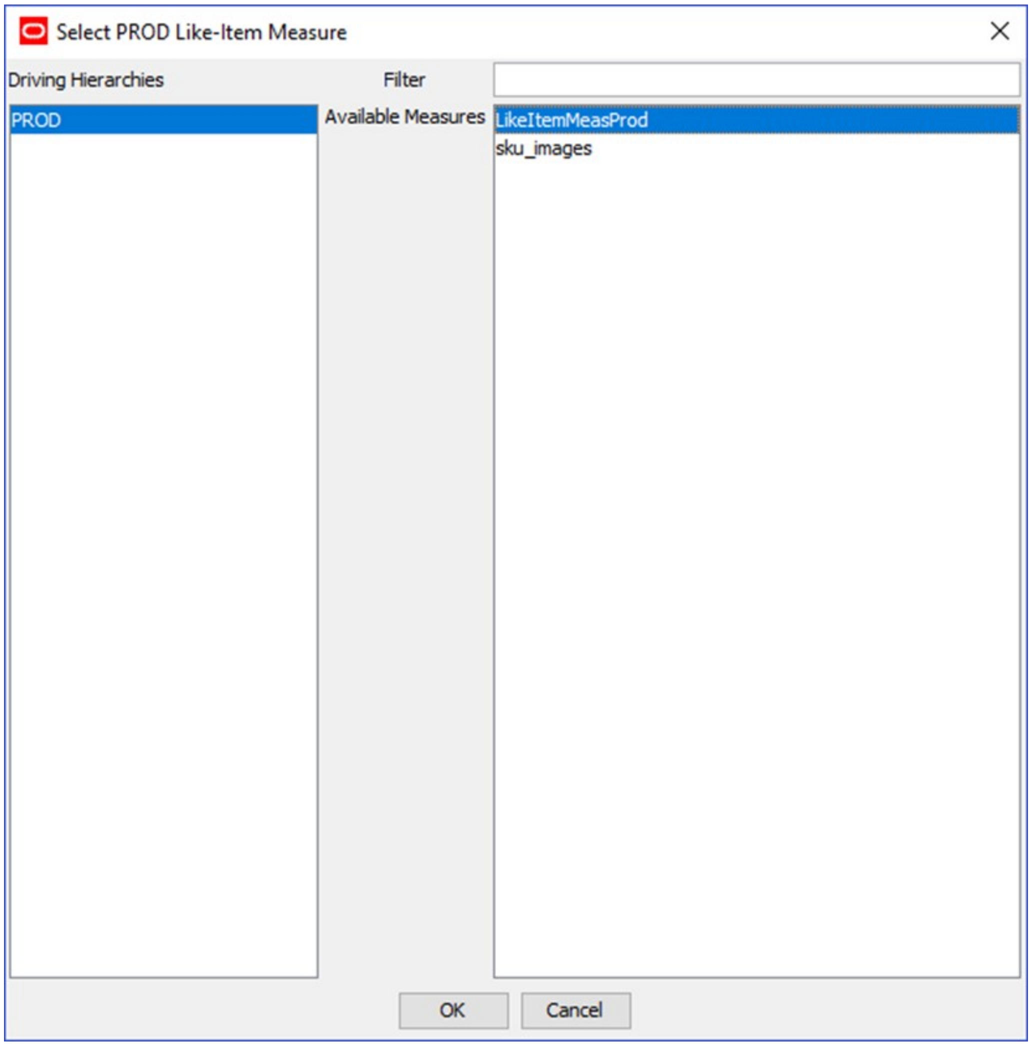
LOC LikeItemMeasLoc
PROD LikeItemMeasProd

This Figure 4-81 consists of a set of rows, one for each DPM-enabled hierarchy in the workbook. Each row lists the name of the hierarchy associated with the row and provides a field in which a measure name can be entered. If the user enters an invalid measure name and clicks **OK**, the invalid measure displays as red text in the General Property table.

If an existing DPM like-item measure contains a reference to a hierarchy that does not exist in the current workbook, when you bring up the Figure 4-82, this hierarchy reference is hidden.

As shown in Figure 4-81, each row has a button that can be used to select a measure instead of typing in a measure name. Click the button next to the measure to open a Figure 4-82 for that measure. Figure 4-82 shows the **Select PROD Like-Item Measure** window.

Figure 4-82 DPM Like-Item Measure Selection Window



The Figure 4-82 contains the hierarchy name, a list of measures, and a text field to accept a partial measure name as a filter.

By default, the list of measures contains all one string measures in the application that meet the validation criteria for the hierarchy being configured. Typing text into the filter text field filters the measure list to remove those measures that do not meet the pattern.

Patch and Upgrade Considerations

If a PDS instance is patched with a template for which a new like-item-measure has been configured, then the old workbooks built with the pre-patch template will no longer work. These workbooks must be rebuilt. On the other hand, if the like-item-measure configuration is removed from a template, the like-item-measure is not updated in the old workbook and no exception is thrown.

Dim Attr Setup Date Measures

The Dim Attr Setup Date Measure stores the dates of the last attribute update for the workbook hierarchy that has the Attribute Hierarchy defined in the application. This workbook hierarchy is referred as “Attribute-enabled hierarchy" here.

A Dim Attr Setup Date measure must meet the following criteria:

- It must exist in the application but is not required to be in the workbook rule set.
- It must be of Date type with storage database.
- It can be either one-dimensional or two-dimensional.
- If it is a one-dimensional, the measure base dimension must be the Attribute Name dimension of the Attribute Hierarchy of the Workbook Hierarchy.
- If it is a two-dimensional, the second base dimension of the measure must be from the workbook hierarchy that owns the attribute.

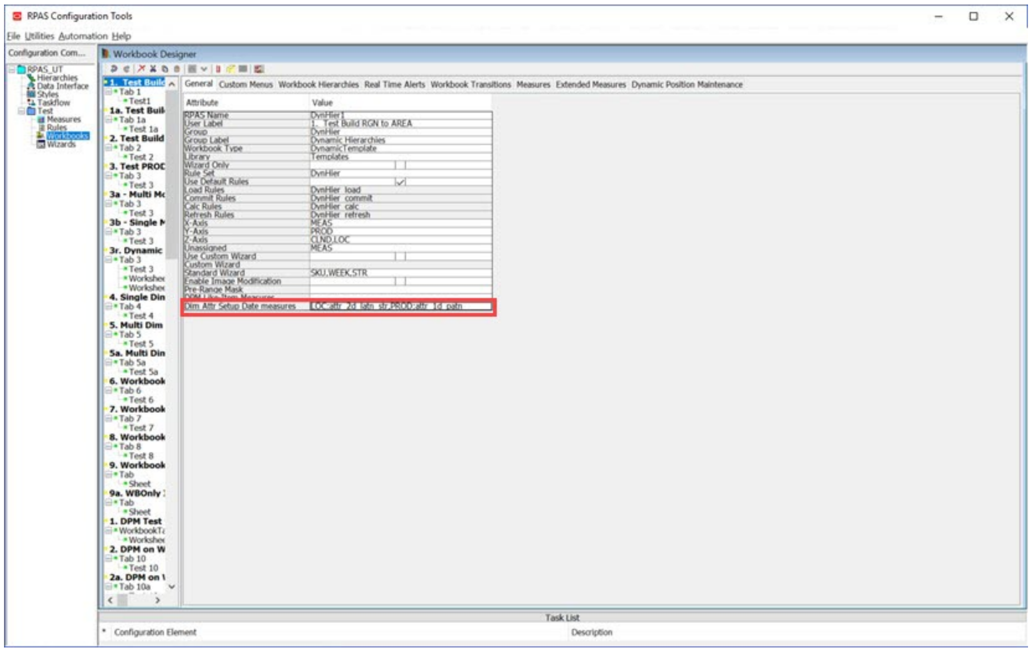
The Dim Attr Setup Date Measures are configured in the workbook General Property table. A window is displayed once the user presses either the mouse or the keyboard in this field. In this window, the user can select one dimension attribute setup date measure for each Attribute-enabled hierarchy in the workbook. The Attribute-enabled workbook hierarchy is determined by the configuration of the Attribute Hierarchy in the application Hierarchy Definition window.

Example:

Figure 4-83 shows an application and a workbook. In the application, the product hierarchy has the Attribute Hierarchy set to PATH, and PATN is the Attribute Name dimension of PATH. The workbook product hierarchy contains the dimensions DEPT and STYL. The workbook also contains LOC and CLND hierarchies, but they do not have the Attribute Hierarchy defined.

Users must configure the Dim Attr Setup Date measure for the workbook so that the workbook product dimension attributes are patchable. Application Measures having a base intersection of PATN, PATN/DEPT, or PATN/STYL are valid candidates. Application measures having a base intersection PATN/STR, PATN/WEEK, or PATN/DEPT/WEEK cannot be used as Dim Attr Setup Date measures in this example.

Figure 4-83   Dim Attr Setup Date Measures



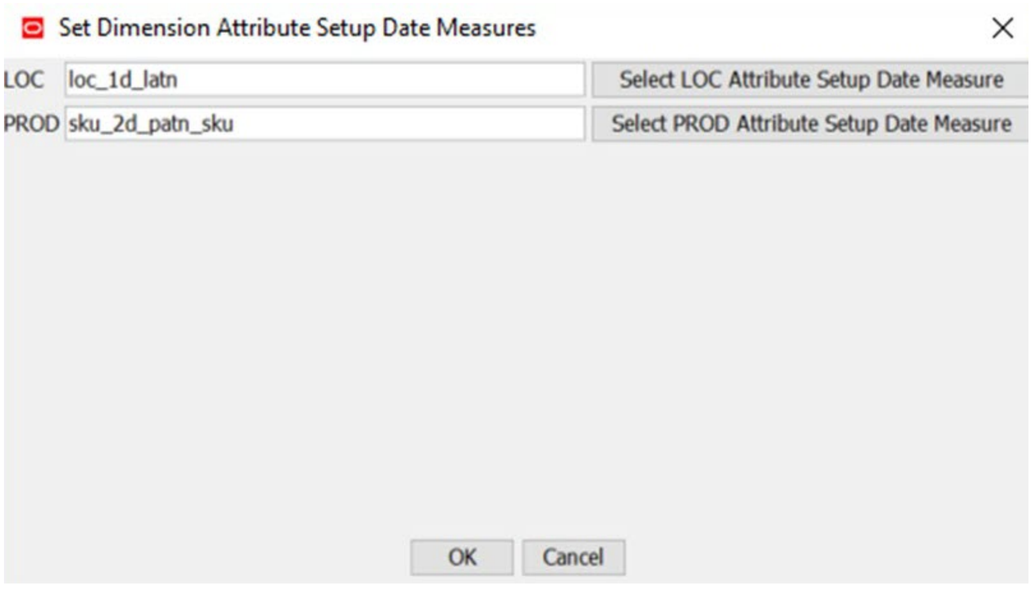
Attribute	Value
RPAS Name	DynHier1
User Label	1. Test Build RGN to AREA
Group	DynHier
Group Label	Dynamic Hierarchies
Workbook Type	DynamicTemplate
Library	Templates
Wizard Only	
Rule Set	DynHier
Use Default Rules	✓
Load Rules	DynHier_load
Commit Rules	DynHier_commit
Calc Rules	DynHier_calc
Refresh Rules	DynHier_refresh
X-Axis	MEAS
Y-Axis	PROD
Z-Axis	CLND,LOC
Unassigned	MEAS
Use Custom Wizard	
Custom Wizard	
Standard Wizard	SKU,WEEK,STR
Enable Image Modification	
Pre-Range Mask	
DPM Like-Item Measures	
Dim Attr Setup Date measures	LOC:attr_2d_latn_str,PROD:attr_1d_patn

In Figure 4-83, `LOC:loc_1d_latn, PROD:sku_2d_patn_sku` is specified in the **Dim Attr Setup Date measures** field under the General tab in the Workbook Designer window.

Hierarchy Specific Dim Attr Setup Date Measure Assignment

In order to assist in the configuration of the Dim Attr Setup Date measures, use the Workbook Designer window. Click in the **Dim Attr Setup Date measures** field, to open the Figure 4-84.

Figure 4-84   Dimension Attribute Setup Date Measures Configuration Window


Set Dimension Attribute Setup Date Measures
Select LOC Attribute Setup Date Measure
Select PROD Attribute Setup Date Measure
OK
Cancel

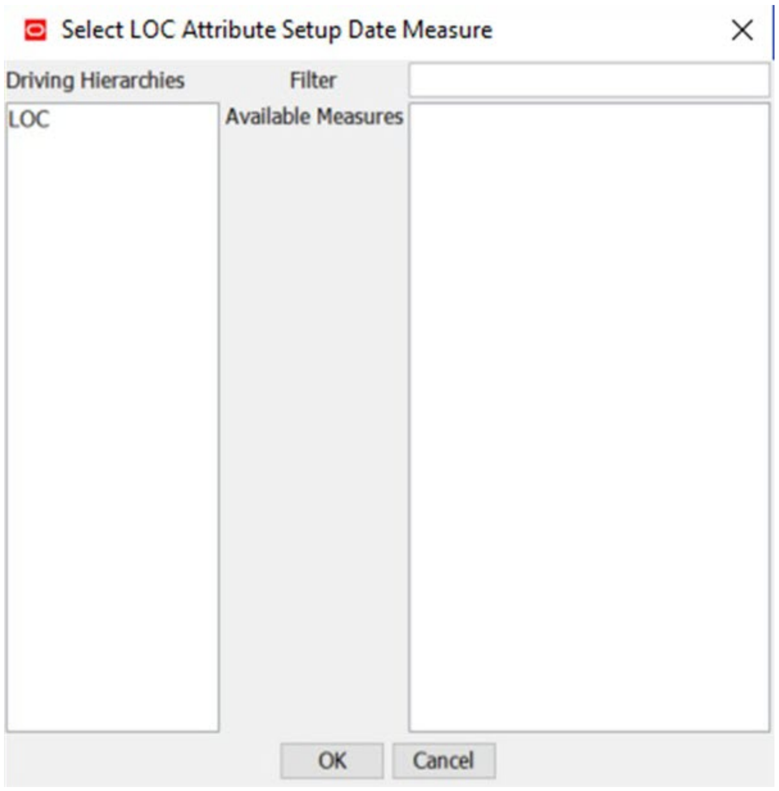
LOC	loc_1d_latn
PROD	sku_2d_patn_sku

The Figure 4-84 consists of a set of rows, one for each Attribute-enabled hierarchy in the workbook. Each row lists the name of the hierarchy associated with the row and provides a field in which a measure name can be entered. If you enter an invalid measure name and click **OK**, the invalid measure displays in red text in the General Property table.

If an existing Dim Attr Setup Date measure contains a reference to a hierarchy that does not exist in the current workbook, when you open the Figure 4-84, this hierarchy reference is hidden.

Each row has a button that can select a measure instead of typing in a measure name. Click the button next to the measure to open the Figure 4-85 for that measure.

Figure 4-85 Dimension Attribute Setup Date Measure Selection Window



The Figure 4-85 contains the hierarchy name, a list of measures, and a text field to accept a partial measure name as a filter.

By default, the list of measures contains all date measures in the application that meet the validation criteria for the hierarchy being configured. Typing text into the filter text field filters the measure list to remove those measures that do not meet the pattern.

Use Rolling Calendar

Some applications such as RDF prefer that the workbook calendar range automatically advances relative to the workbook build date instead of using a static set of calendar positions. The Rolling Calendar functionality can help them to achieve this goal.

First, the workbook template must be configured to use rolling calendar. The Use Rolling Calendar property determines if Rolling Calendar is allowed for the workbook template.

Select the check box to enable the Rolling Calendar Range Measures property. Clear the check box to disable the Rolling Calendar Range Measures property. See Rolling Calendar Range Measures for more information about Rolling Calendar.

Once the workbook template is configured with Rolling Calendar and the PDS build/patch is complete, users can then enable and select Rolling Calendar during the workbook wizard process. See the *Oracle Retail Predictive Application Server Cloud Edition User Guide* for details.

Rolling Calendar Range Measures

This field is only available when the **Use Rolling Calendar** check box is selected. This field allows you to select up to four rolling calendar range measures, which represent the time period threshold used within the Rolling Calendar wizard process. These time periods are the:

- Maximum periods into the future (MaxF)
- Minimum periods into the future (MinF)
- Maximum periods into the past (MaxP)
- Minimum periods into the past (MinP)

When no range measure is specified during the workbook template configuration, the RPASCE UI will default to 0 time periods in the case of MinF and MinP and the no-limit for MaxF and MaxP.

A qualified Rolling Calendar Range measure must meet the following criteria:

- It must exist in the application but is not required to be in the workbook rule set.
- It must be of Integer type with a valid storage database.
- It must be a scalar measure.

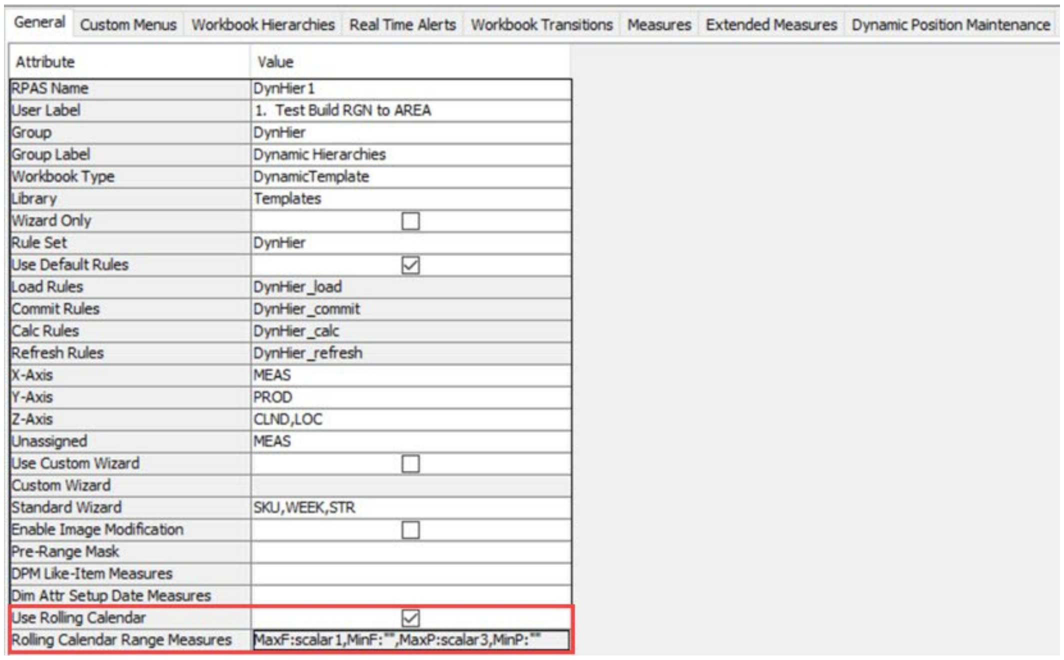
The Rolling Calendar Range measures are configured in the Workbook General Property table. A window is displayed once the user selects this field using either the mouse or the keyboard. In this window, the user can select up to one range measure for each Maximum Future Periods (MaxF), Minimum Future Periods (MinF), Maximum Past Periods (MaxP), and Minimum Past Periods (MinP) in the workbook.

Note that the rolling calendar settings in the workbook template configuration are patchable for the PDS, but not for existing workbooks, since the rolling calendar wizard process is not patchable.

Example:

Figure 4-86 shows an application and a workbook. In the application, there are four scalar, integer type measures with valid storage databases with measure names scalar1, scalar2, scalar3, and scalar4, respectively.

Figure 4-86 Use Rolling Calendar and Rolling Calendar Range Measures

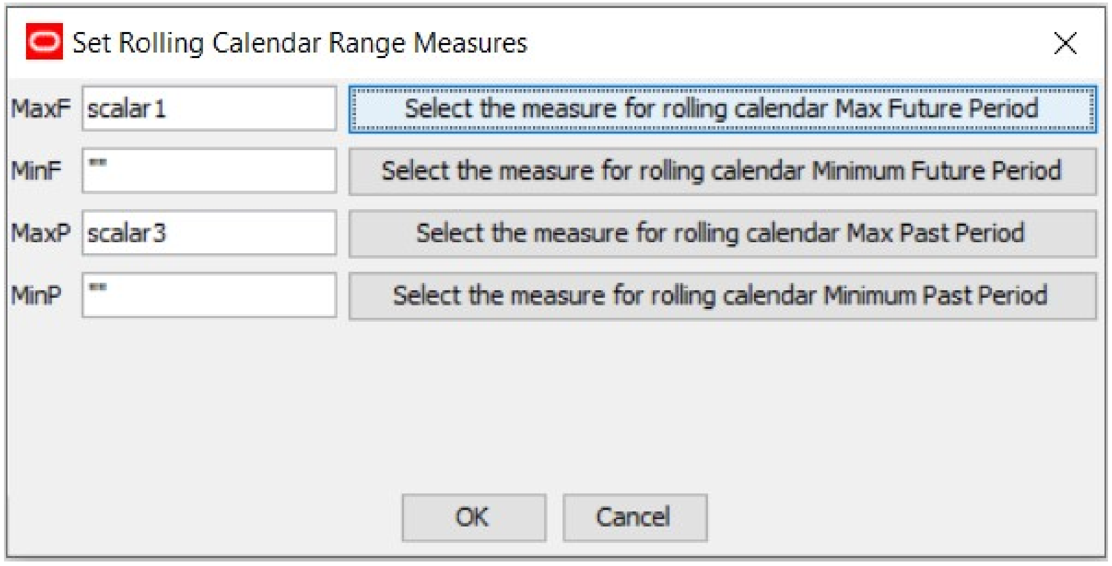


Attribute	Value
RPAS Name	DynHier1
User Label	1. Test Build RGN to AREA
Group	DynHier
Group Label	Dynamic Hierarchies
Workbook Type	DynamicTemplate
Library	Templates
Wizard Only	☐
Rule Set	DynHier
Use Default Rules	✓
Load Rules	DynHier_load
Commit Rules	DynHier_commit
Calc Rules	DynHier_calc
Refresh Rules	DynHier_refresh
X-Axis	MEAS
Y-Axis	PROD
Z-Axis	CLND,LOC
Unassigned	MEAS
Use Custom Wizard	☐
Custom Wizard	
Standard Wizard	SKU,WEEK,STR
Enable Image Modification	☐
Pre-Range Mask	
DPM Like-Item Measures	
Dim Attr Setup Date Measures	
Use Rolling Calendar	✓
Rolling Calendar Range Measures	MaxF:scalar1,MinF:"",MaxP:scalar3,MinP:""

Setting the Rolling Calendar Range Measures Assignment

In order to configure the Rolling Calendar Range measures, use Figure 4-87. From the General Properties Table, when you click in the **Rolling Calendar Range Measure** field, the Figure 4-87 opens.

Figure 4-87 Set Rolling Calendar Range Measures Window

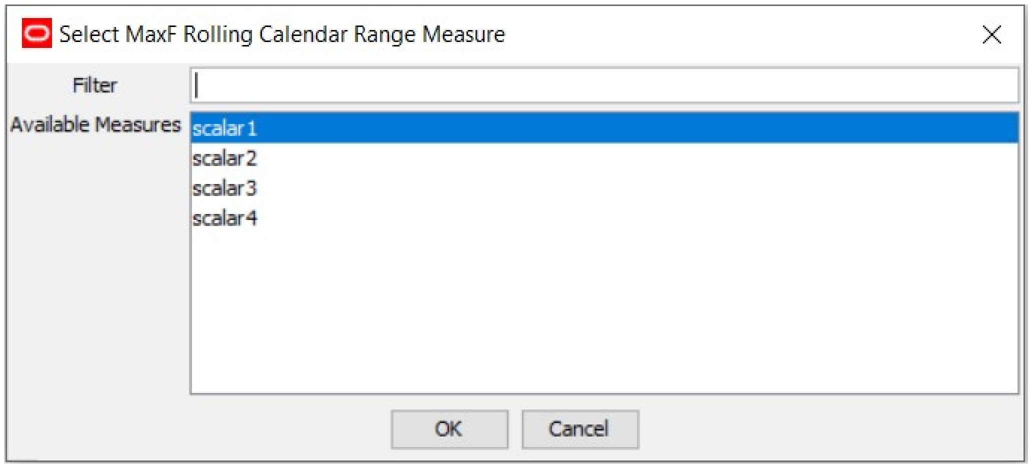


MaxF	scalar1	Select the measure for rolling calendar Max Future Period
MinF	""	Select the measure for rolling calendar Minimum Future Period
MaxP	scalar3	Select the measure for rolling calendar Max Past Period
MinP	""	Select the measure for rolling calendar Minimum Past Period

The Figure 4-87 consists of four rows, one for each Time Periods category. Each row lists the name of qualified range measures in the application and provides a field in which to enter a measure name. If you enter an invalid measure name and click **OK**, the invalid measure shows a validation error in the Task List Panel.

Each row has a button that you can use to select a measure instead of typing in a measure name. Click the button next to the measure to open the Figure 4-88 for the selected measure.

Figure 4-88 Rolling Calendar Range Measure Selection Window



The Figure 4-88 contains the Time Periods Category name such as MaxF in the top window label, a list of measures, and a text field to accept a partial measure name as a filter.

By default, the list of measures contains all scalar, integer type measures in the application. Typing text into the filter text field filters the measure list to remove those measures that do not meet the pattern.

Note:

If the value of the maximum Rolling Calendar Range Measure is less than the corresponding minimum measure, for example, if MaxP is 8 while MinP is 5, or MaxF is 6 while MaxF is 8, RPASCE will not flag it as a user error; instead, RPASCE resets the maximum range measure to the maximum available value in the application during the wizard process.

Login User Only

Some applications, such as A&IP FSL, prefer that the workbook USER dimension only contains the login user so that when users define a measure with the base intersection containing the USER dimension for storing the user preferred values, only the login user, instead of all users, has the data committed when users commit the data from the workbook to the PDS. For example, store the user selection for product attribute rollups in the workbook. The Login User Only property can help users to achieve this goal. When this property is set to true, the workbook template, if not ranged by wizard explicitly, should only include the login user within its USER dimension.

The row Login User Only contains a check box. By default, this check box is not selected. The user must select this check box to indicate that the User dimension of this workbook will only contains the login user and no other user will be included.

Dynamic Dimension Label Override Measures

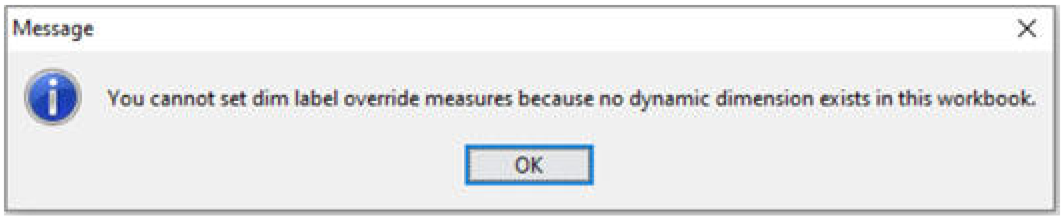
RPASCE supports the use of scalar, string measures to override the dimension labels for the dynamic dimensions defined within the Workbook Hierarchies tab. Such measures are called "Dynamic Dimension Label Override" measures and must be configured in the workbook General Property tab. The value of such scalar, string measures are calculated as part of the Custom Menu operation that invokes the RPASCE function dynHierRefresh, and the value of this override measure will be used as the dynamic dimension label afterwards.

A Dynamic Dimension Label Override measure must meet the following criteria:

- It is a scalar, String type measure.
- It can have a storage database or can be a workbook-only measure and is used to satisfy the custom menu operations. However, this measure must be part of the workbook and must be in the workbook rule set.

The Dynamic Dimension Label Override measures are configured in the workbook General Property table. A dialog box is displayed once the user clicks this field. In this dialog box, the user can select one label override measure for each dynamic dimension in the workbook. The dynamic dimension in the workbook is determined by the modified dim configuration in the Dynamic Hierarchies table within the Workbook Hierarchies tab. If the workbook does not contain any dynamic dimension, the warning message shown in Figure 4-89 is displayed when the field is clicked.

Figure 4-89 Warning Dialog Box Displayed When Workbook has no Dynamic Dimensions

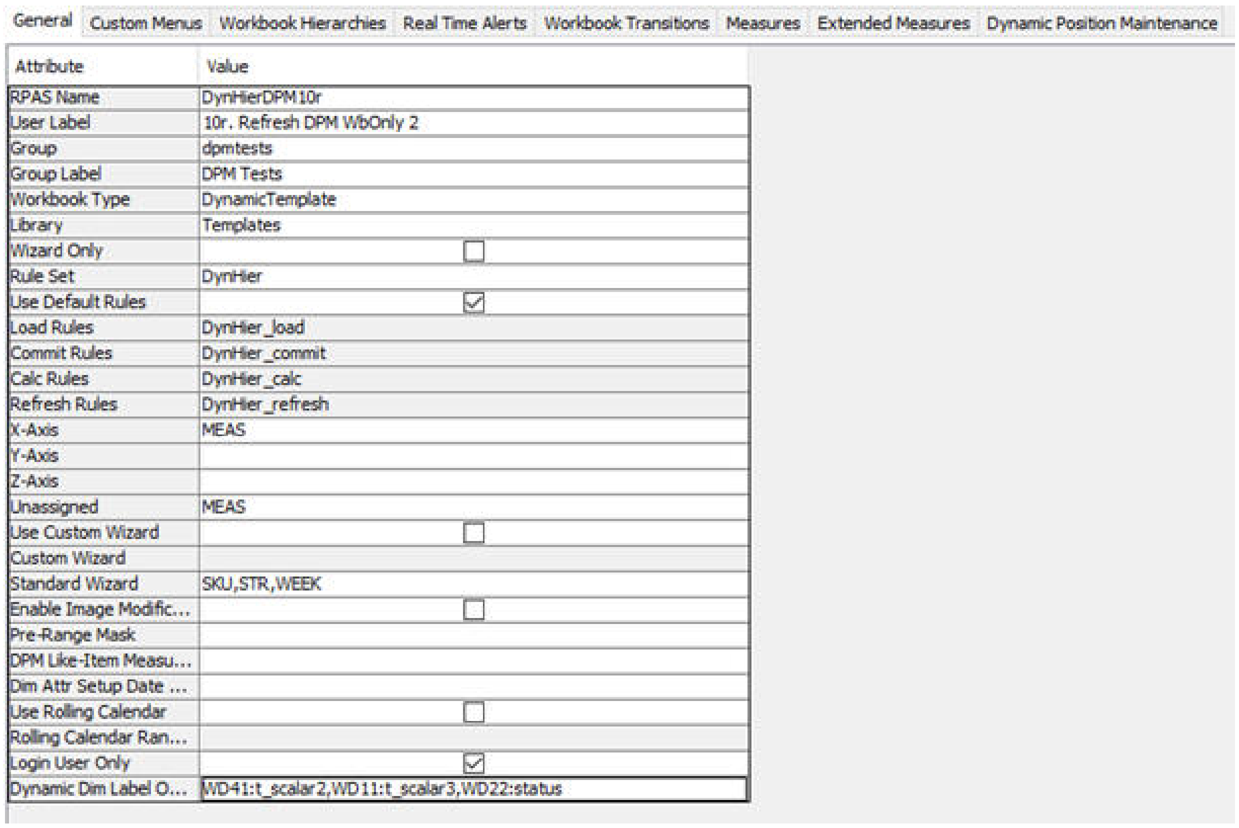

Message

You cannot set dim label override measures because no dynamic dimension exists in this workbook.

Example

In a workbook template, the dynamic dimensions are WD41, WD11 and WD22. Three string measures, t_scalar2, status and t_scalar3, are configured and added to the workbook rules, as shown in Figure 4-90.

Figure 4-90 Dynamic Dimension Label Override Measure



Attribute	Value
RPAS Name	DynHierDPM10r
User Label	10r. Refresh DPM WbOnly 2
Group	dpmtests
Group Label	DPM Tests
Workbook Type	DynamicTemplate
Library	Templates
Wizard Only	☐
Rule Set	DynHier
Use Default Rules	✓
Load Rules	DynHier_load
Commit Rules	DynHier_commit
Calc Rules	DynHier_calc
Refresh Rules	DynHier_refresh
X-Axis	MEAS
Y-Axis	
Z-Axis	
Unassigned	MEAS
Use Custom Wizard	☐
Custom Wizard	
Standard Wizard	SKU,STR,WEEK
Enable Image Modific...	☐
Pre-Range Mask	
DPM Like-Item Measu...	
Dim Attr Setup Date ...	
Use Rolling Calendar	☐
Rolling Calendar Ran...	
Login User Only	✓
Dynamic Dim Label O...	WD41:t_scalar2,WD11:t_scalar3,WD22:status

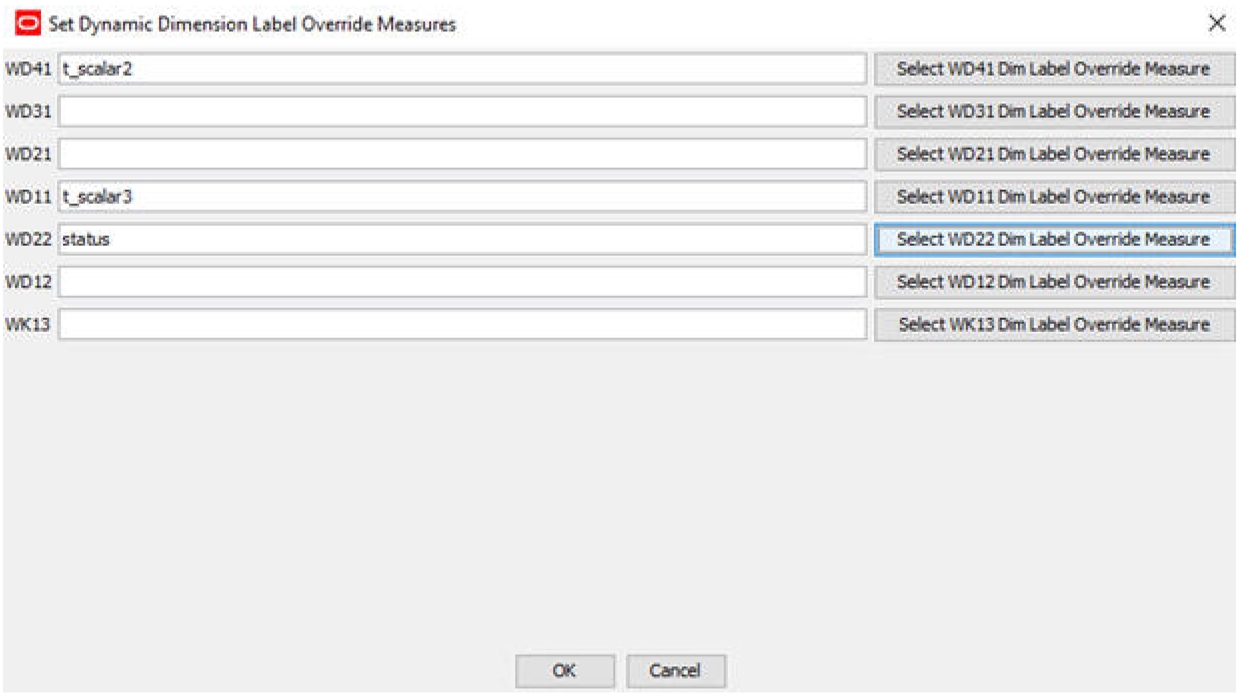
General Properties Table of the Workbook Showing Dynamic Dimension Label Override

In the Dynamic Dim Label Override field under the General tab in the Workbook Designer tool, the measure t_salar2 is configured to contain the override label for dynamic dimension WD41, the measure status for dimension WD11, and the measure t_scalar3 for dimension WD22.

Dynamic Dimension Label Override Measure Assignment

In order to assist in the assignment of these measures, a dialog box has been created. If the user clicks in the field, the dialog box shown in Figure 4-91 is displayed.

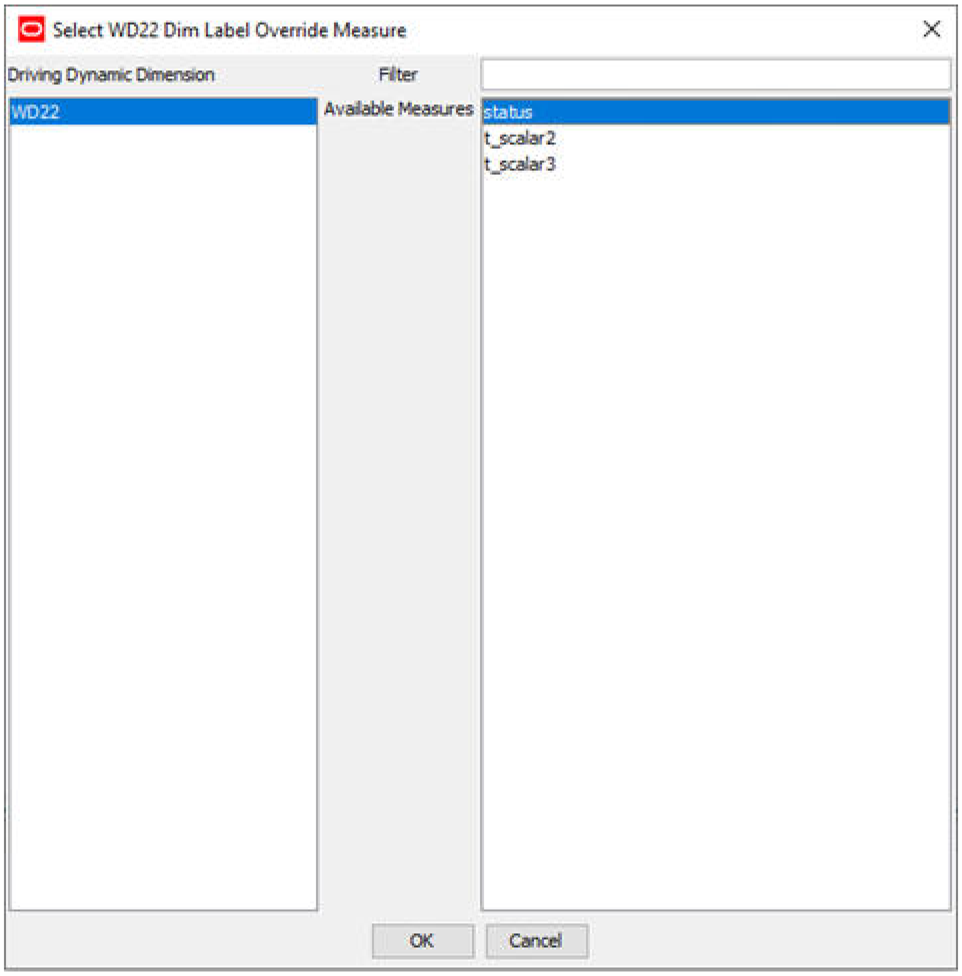
Figure 4-91 Dynamic Dimension Label Override Measure Configuration Window


Set Dynamic Dimension Label Override Measures
Select WD41 Dim Label Override Measure
Select WD31 Dim Label Override Measure
Select WD21 Dim Label Override Measure
Select WD11 Dim Label Override Measure
Select WD22 Dim Label Override Measure
Select WD12 Dim Label Override Measure
Select WK13 Dim Label Override Measure
OK
Cancel

WD41	t_scalar2
WD31	
WD21	
WD11	t_scalar3
WD22	status
WD12	
WK13	

This dialog box consists of a set of rows, one for each dynamic dimension in the workbook. Each row lists the dynamic dimension and provides a field in which a measure name can be entered. If the user enters an invalid measure name and clicks OK, the invalid measure will be shown in red in the General Property table. Each row has a button that can be used to select a measure from a qualified candidate list instead of typing in a measure name. Clicking this button displays a secondary dialog box shown in Figure 4-92.

Figure 4-92 Dynamic Dimension Label Override Measure Selection Window


Select WD22 Dim Label Override Measure
Driving Dynamic Dimension
Filter
Available Measures
WD22
status
t_scalar2
t_scalar3
OK
Cancel

This secondary dialog box contains the dynamic dimension name, a list of qualified measures, and a text field to accept a partial measure name as a filter.

By default, the list of measures contains all scalar, string measures in the workbook. Typing text into the filter text field filters the measure list to remove those measures that do not meet the pattern.

Patch and Upgrade Considerations

If a PDS instance is patched with a template for which a new dynamic dimension label override measure has been configured, then the old workbooks built with the pre-patch template will no longer work. These workbooks must be rebuilt. On the other hand, if the override-measure configuration is removed from a template, the dynamic dimension label override will not be updated in the old workbook and no exception will be thrown.

Sync DPM Positions to Virtual Dimension

Some applications, such as APCS, prefer to create the DPM positions for both original and virtual hierarchies in a single workbook template. The workbook property Sync DPM to Virtual Dimension, once configured to True, enables this. The DPM positions defined in the original dimension are automatically synchronized onto the corresponding virtual dimension at the end of each DPM process in a single workbook. It does not require rebuilding the workbook or building two workbooks.

The Sync DPM to Virtual Dim is configured in the workbook General Property table by selecting the check box. By default, this check box is not selected. Users must select the check box explicitly to set the property to True.

To synchronize DPM positions, the workbook must meet the following criteria:

- The workbook must have both the original and corresponding virtual dimensions.
- The workbook must have the DPM dimensions configured.
- DPM is enabled on the original dimension.
- The original dimension must not have any dynamic hierarchy defined on itself or its parent dimension(s), except for the dynamic hierarchy definition for the Workbook Only dimensions.

  For example, if the workbook has an LOC rollup relationship such as stor-regn-area, and stor-storecluster,

  where storecluster is a dynamic hierarchy defined in the workbook.

  Stor is the original dimension which has DPM enabled in the workbook.

  Stor has a corresponding virtual dimension StorR and the workbook contains StorR dimension.

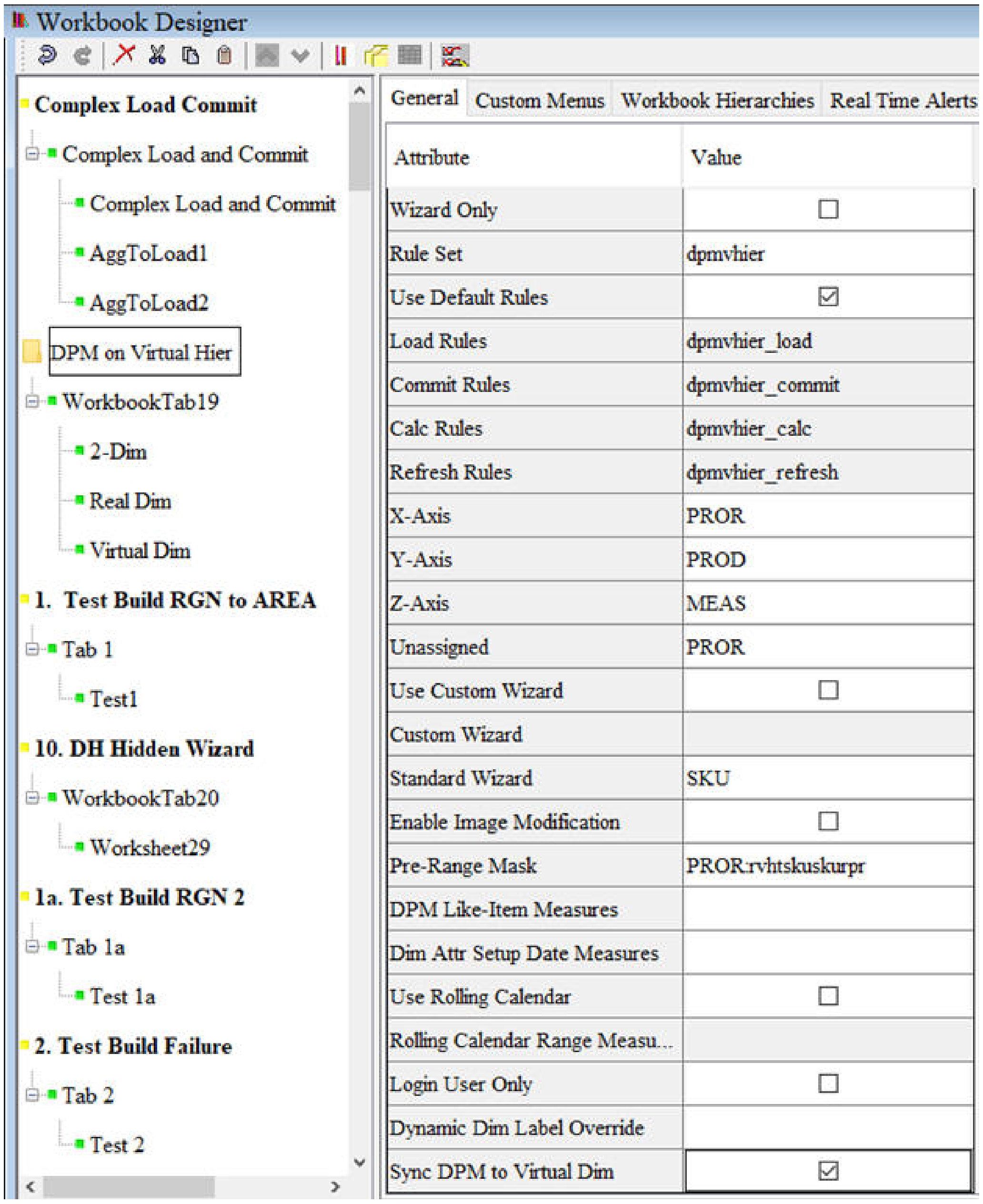
  If the dynamic hierarchy Store Cluster has *no* virtual dimension in the workbook, then the Sync DPM To Virtual Dim property can be set.

  If the dynamic hierarchy store cluster has a virtual dimension that is contained in the workbook, then this property cannot be set. Otherwise, the DPM process will fail.
- The position set for the original dimension must be the same as for the virtual dimension. Usually, users configure a Pre-Range Mask in the workbook property table to guarantee the same set of positions are selected for both dimensions. If the Pre-Range Mask is used, the mask may need to be recomputed before the next workbook build

  .

> **Note:**
>
> Users must only select the Sync DPM to Virtual Dim check box after completing all the other configurations for this workbook, such as Worksheet, DPM, Workbook Hierarchy configuration, and so on.

Figure 4-93 Configure the Workbook Sync DPM to Virtual Dim Property

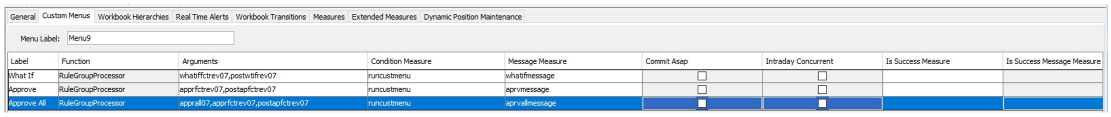


Attribute	Value
Wizard Only	☐
Rule Set	dpmvhier
Use Default Rules	☑
Load Rules	dpmvhier_load
Commit Rules	dpmvhier_commit
Calc Rules	dpmvhier_calc
Refresh Rules	dpmvhier_refresh
X-Axis	PROR
Y-Axis	PROD
Z-Axis	MEAS
Unassigned	PROR
Use Custom Wizard	☐
Custom Wizard	
Standard Wizard	SKU
Enable Image Modification	☐
Pre-Range Mask	PROR:rvhtskuskurpr
DPM Like-Item Measures	
Dim Attr Setup Date Measures	
Use Rolling Calendar	☐
Rolling Calendar Range Measu...	
Login User Only	☐
Dynamic Dim Label Override	
Sync DPM to Virtual Dim	☑

Custom Menus Tab

The use of this tab is optional. It is used to create a workbook specific, customized menu-driven process within the workbook where the defined menu options execute rule group transitions (which cause a series of calculations to be performed) and external scripts. Custom menus are typically used to define processes, such as an approval process.

Figure 4-94   Custom Menus Tab



General | Custom Menus | Workbook Hierarchies | Real Time Alerts | Workbook Transitions | Measures | Extended Measures | Dynamic Position Maintenance

Menu Label: Menu9

Label	Function	Arguments	Condition Measure	Message Measure	Commit Asap	Intraday Concurrent	Is Success Measure	Is Success Message Measure
What If	RuleGroupProcessor	whatiffctrev07,postwtifrev07	runcustmenu	whatifmessage	☐	☐		
Approve	RuleGroupProcessor	apprfctrev07,postapfctrev07	runcustmenu	aprvmessage	☐	☐		
Approve All	RuleGroupProcessor	apprall07,apprfctrev07,postapfctrev07	runcustmenu	aprvallmessage	☐	☐		

Create a Custom Menu

Complete the following steps to Create a Custom Menu.

1. Select the Custom Menus tab.
2. In the Menu Label field, enter the name of the menu that will be displayed in the RPASCE Client. This menu option displays as a top-level menu option, between the Window and Help menu options.
3. Right-click in the table area and select **Add**.
4. Enter the following information:
   - **Label:** The label that displays in the menu in the RPASCE Client. These labels display beneath the top-level menu option named in the Menu Label property, in the order that they are displayed in this window.

     Note:

     Duplicate menu names are not allowed.

   - **Function:** This field defaults to RuleGroupProcessor and cannot be changed.
   - **Arguments**: The processes that are to be executed by the menu option are specified in the argument's property. There may be several processes specified in the order they are to be executed and separated by commas. There should be no gap after the comma between any two adjacent arguments, with each string assumed to be the name of a rule group.

     When the end user selects the menu option in the RPASCE Client, RPASCE runs the processes from the argument's property in the specified sequence. RPASCE waits until each process has finished before executing the following process. After all the processes have been run, RPASCE runs a final transition using the full transition type back to the calc rule group for the workbook. This transition does not have to be explicitly specified in the argument's property.

     Rule groups are executed with a full transition from the previous rule group, and the calculations apply to the whole scope of the workbook (that is, they use full (batch) mode rather than incremental mode). These terms are explained in Appendix – Calculation Engine User Guide. The rule group transitions ensure that the integrity of all rules is enforced in the new rule group.
   - **Condition Measure:** This field is used to specify a scalar, Boolean condition measure in the workbook that will be checked by the custom menu to decide whether it must execute or not.

     For a measure to be a candidate for the condition measure of a custom menu item, the measure must meet the following criteria:

- In the workbook. The measure must exist within the rule groups for the workbook.

- Scalar

- Boolean

In the workbook. The measure must exist within the rule groups for the workbook.

If the value of the measure is True, the custom menu runs, but if the value is False, the menu does not execute and displays a message relating that the custom menu could not execute because the conditions were not met. (For more information on how this message can be customized, see the Message Measure bullet.)

If the condition measure is not specified, meaning the field is empty, the custom menu always runs. Table 4-1 specifies the behavior of custom menu execution and the display of custom messages based on whether a measure name has been entered (available) in the field and whether the measure's value has been set (True in case of the condition measure and a non-zero length string in case of the message measure).

Table 4-1 Behavior of the Custom Menu

Condition Measure	Return Message Measure	Behavior
Available & Set	Available & Set	The Custom Menu runs and displays the Custom Menu Response pop-up containing the value of the Return Message measure.
	Available but Not Set	The Custom Menu runs and displays the default message.
	Not Available	The Custom Menu runs and displays the default message.
Available but Not Set	Available & Set	The Custom menu does not execute but does display the Custom Menu Response pop-up containing the value of the Return Message measure.
	Available but Not Set	The Custom Menu does not execute and displays a Warning pop-up message reading Conditions for executing the *Custom Menu have not been met!*.
	Not Available	The Custom Menu does not execute and displays a Warning pop-up message reading Conditions for executing the *Custom Menu have not been met!*.
Not Available	Available & Set	The Custom menu runs and displays the Custom Menu Response pop-up containing the value of the Return Message measure.

Table 4-1 (Cont.) Behavior of the Custom Menu

Condition Measure	Return Message Measure	Behavior
	Available but Not Set	The Custom menu runs and displays the Custom Menu Response pop-up containing the value of the Return Message measure.
	Not Available	Custom Menu runs and displays the default message.

- **Message Measure:** This field is used to specify the scalar, String measure, the value of which displays by the custom menu in a window upon the menu's successful or failed execution. If the field is empty, RPASCE displays the default message that it historically displays. If a measure is specified, but the value is empty, RPASCE again displays the default message. If the value is a non-zero length string, then the value displays.

  For a measure to be a candidate for the return message measure of a custom menu item, the measure must meet the following criteria:

  - In the workbook. The measure must exist within the rule groups for the workbook.

  - Scalar

  - String

  To effectively use this feature, it is important to understand the execution of a custom menu. When you select a custom menu from the menu, RPASCE first checks if there is a condition measure available for controlling the execution. If there is none, it continues to run the rule groups or scripts in the argument of the custom menu. If a condition measure is available, RPASCE checks the value of the measure for controlling the execution of the custom menu. If the value is false, it checks the availability of the return message measure. If the message measure is unavailable, RPASCE displays a default message informing the user that menu could not be executed because the conditions were not met. However, if the measure is available, RPASCE examines its value. If the value is empty, it defaults to the same behavior as when the message measure was not available. If the value is not empty, it displays the custom message.

  If the custom menu can execute, either because the condition measure is unavailable or because it is set to true at the time the custom menu is invoked, RPASCE runs the rule groups in the argument of the custom menu specification. RPASCE will then look for the value of the message measure, and if the value is empty, it informs the user that the menu was successfully executed using the default message. If the value is not empty, it displays the value of the message measure. It then transitions to the Calc rule group and completes execution.

- **Commit ASAP:** It is possible to include commit rule groups (rule groups that include commit rules) in the list of rule groups that forms the arguments of a custom menu item. Normally, these commit rule groups execute synchronously, meaning that after submitting a commit request, the user must wait until the commit process ends. It is also possible to mark a custom menu item as Commit ASAP.

  When marked as Commit ASAP, a custom menu item runs the commit rule group asynchronously, returning the control to the user immediately. The commit ASAP process creates a temporary copy of the workbook and places a commit request in a queue. The commit rule group is executed as soon as the measure databases become available.

In this manner, it is possible to commit certain measures within the workbook without halting work on the workbook until the commit has been processed and without having to deal with contention issues associated with immediate commits.

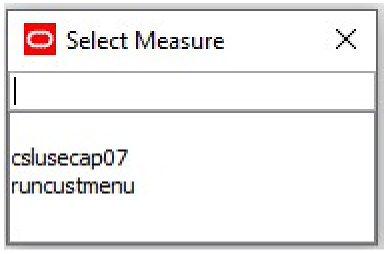
> **Note:**
>
> When a custom menu item is marked as Commit ASAP, that menu item may only have a single commit rule group in its list of arguments. Furthermore, it is recommended that the commit rule group be the last rule group in the custom menu item's list of arguments, as any subsequent rule groups that execute will not affect the values committed and any side effects of the commit rule group execution will not apply to the workbook.

> **Note:**
>
> The commit button in the Fusion UI always commits ASAP. This is independent of how the commit works in a custom menu.

- **Intraday Concurrent:** This field is used to flag the custom menu as one that can run concurrently with any intraday process running on the application. The default setting is false which means that the custom menu will not run when an intraday process is running. See the Oracle Retail Predictive Application Server Cloud Edition Administration Guide for more details on intraday processes.

  A custom menu that is configured to run intraday concurrent should be something that only accesses workbook data and/or runs commits via the commit ASAP functionality. Custom menus that update or read directly from the PDS should not be configured as intraday concurrent as this would conflict with the ride process.
- **Is Success Measure:** This field is used to specify a scalar, Boolean measure that exists within the rule groups of the workbook. User can select such a measure from the candidate measure list or leave it empty. This measure's value is set during the custom menu rule group execution and indicates whether or not the custom menu execution is successful. The rule groups defined for the custom menu action need include rules that populate the Is Success while processing the action.

Figure 4-95 Select Measure



cslusecap07
runcustmenu

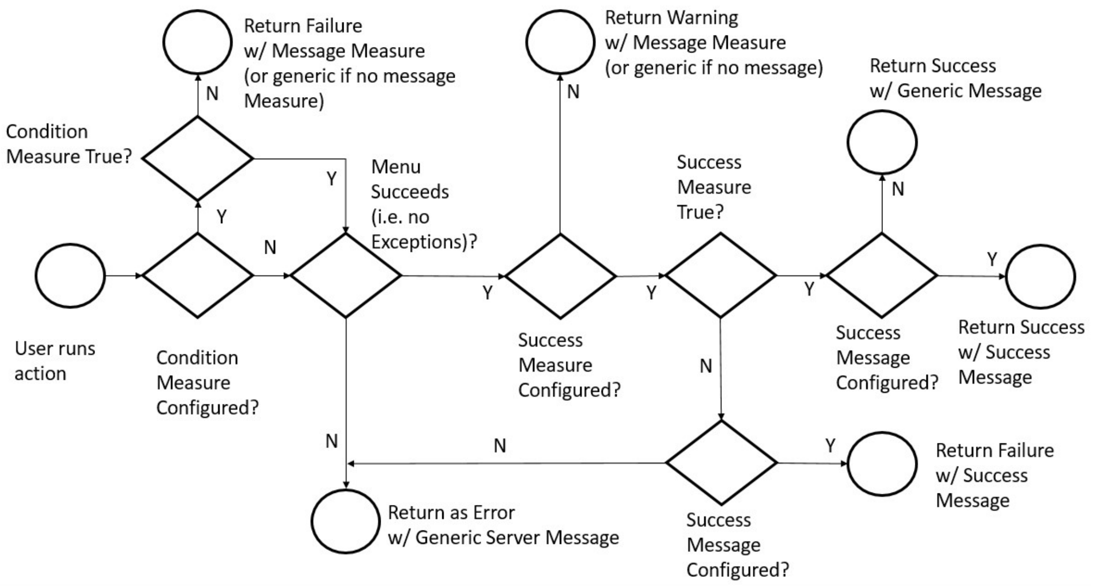
When a user initiates a custom action, the system performs that action through the execution of one or more rule groups defined in configuration for the workbook in which they are performing the task. This operation can require significant amount of time but cannot be performed asynchronously, as all data in the workspace could potentially be modified in the operation. Once the operation is complete, the user receives a notification that reports the completion of the custom action. Using Is Success Measure and the Is Success Message Measure (explained next), an action-specific response message can be used to alert the user whether the action completed successfully or whether it encountered problems and was not able to execute.

Depending on the Boolean value contained within the Is Success measure, the RPASCE client displays the Custom Menu execution response message accordingly. If the Is Success Measure is not set by users, then the RPASCE UI displays the default message. For details, see Figure 4-96.

Summary of Custom Menu Response Notification

The following figure summarizes the flow when either a generic message or customized message is displayed for custom menu execution.

Figure 4-96 Custom Menu Response Notification Diagram


Return Failure w/ Message Measure (or generic if no message Measure)
Return Warning w/ Message Measure (or generic if no message)
Return Success w/ Generic Message
Condition Measure True?
Menu Succeeds (i.e. no Exceptions)?
Success Measure True?
User runs action
Condition Measure Configured?
Success Measure Configured?
Success Message Configured?
Return Success w/ Success Message
Return Failure w/ Success Message
Return as Error w/ Generic Server Message
Success Message Configured?
Y
N

Workbook Hierarchies Tab

This tab is optional. Use this tab to configure hierarchical relationships whose parent-child relationships are not defined in the Hierarchy Definition Window or the loadHier process, but are data driven.

Note:

Virtual hierarchies and virtual dimensions cannot be modified inside the Workbook Hierarchies tab.

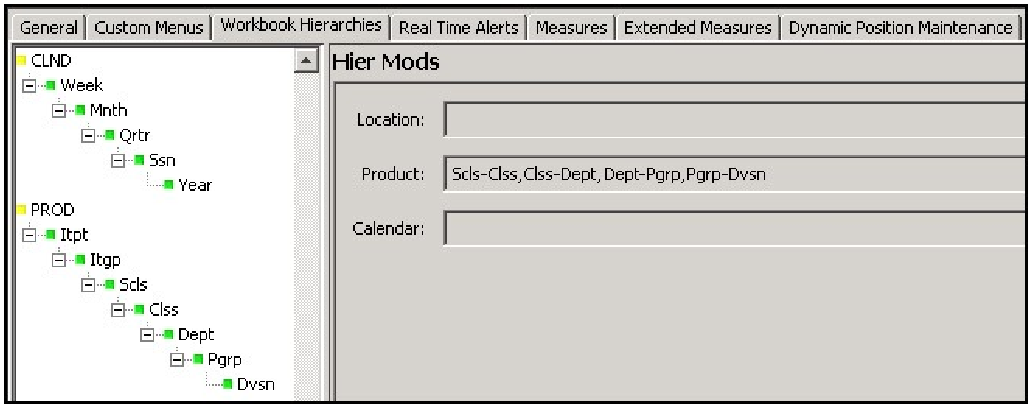
For instance, you can:

Hide individual dimensions or whole branches of a hierarchy by excluding them from the dimensions that are available to the workbook. (For more information, see Remove Dimension.)

Create mappings between dimensions that are not directly related in the hierarchy structures. (For more information, see Change Rollup.)

Create dynamic hierarchical relationships that are built using measure data during the workbook build process, and may vary each time a workbook is built, but the relationships within a workbook are constant. (For instructions on creating dynamic hierarchies, see Make Rollup Dynamic, Add Workbook Dimension, and Insert Workbook Dimension. For in-depth examples, see the Appendix – Dynamic Hierarchies.)

Figure 4-97   Workbook Hierarchies Tab

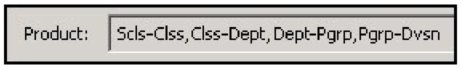


Location:
Product: Scls-Clss,Clss-Dept,Dept-Pgrp,Pgrp-Dvsn
Calendar:

Any hierarchies that do not have dimension relationships specified will use the full hierarchy specified in the Hierarchy Definition Window, with the lowest dimension in that hierarchy used as the base intersection of a measure used in the rule set for the workbook as its root.

Dimension relationships are specified as hyphenated child-parent pairs. In these pairs, the child is listed first (for example scls-clss). More than one child-parent pairs are separated by commas, as shown in Figure 4-98.

Figure 4-98   Child-Parent Pairs


Product: Scls-Clss,Clss-Dept,Dept-Pgrp,Pgrp-Dvsn

Note:

The hierarchy fields in the Hier Mods section are read-only. The child-parent pairs displayed cannot be edited there.

Every parent-child relationship in the workbook must be explicitly specified. Only dimensions that have been defined in The Hierarchy Definition Window can be used and must be specified by name. The Configuration Tools validates the dimension relationships to ensure that valid dimension names are used. It prohibits the use of the same dimension name in both the child and the parent dimensions. In addition, the Configuration Tools prohibits pairs where the child's Aggs attribute is the same as the parent's attibute. For example, Clss-Scls is not allowed, but Scls-Clss is.

Modify a Hierarchy in a Workbook

To modify a hierarchy in a workbook, complete the following steps:

1. Click the **Workbook Hierarchies** tab. In this tab you can see all hierarchies used in the base intersection of measures used in the rule set in the workbook.
2. Right-click a hierarchy or dimension and choose one of the following options:
   - Remove Dimension
   - Restore Dimension
   - Change Rollup
   - Make Rollup Dynamic
   - Remove Dynamic Rollup
   - Add Workbook Dimension
   - Insert Workbook Dimension

Important Note About Using Hier Mods Options

The use of mappings that are non-structural should be carefully managed to ensure they are only used where the non-structural mapping work. For example, consider a product hierarchy where a subclass may be supplied by multiple vendors (with a style always supplied by a single vendor), but in some parts of the business, subclasses are only supplied by a single vendor.

Example:

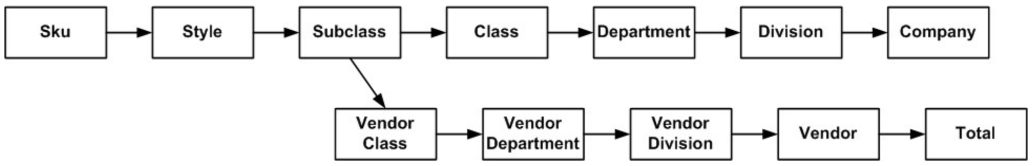
Figure 4-99 shows an example of a product hierarchy that includes a branch for vendor analysis is:

Sku-Style-Subclass-Class-Department-Division-Company

With a branch of:

Subclass-VendorClass-VendorDept-VendorDiv-Vendor-Total

Figure 4-99   Product Hierarchy

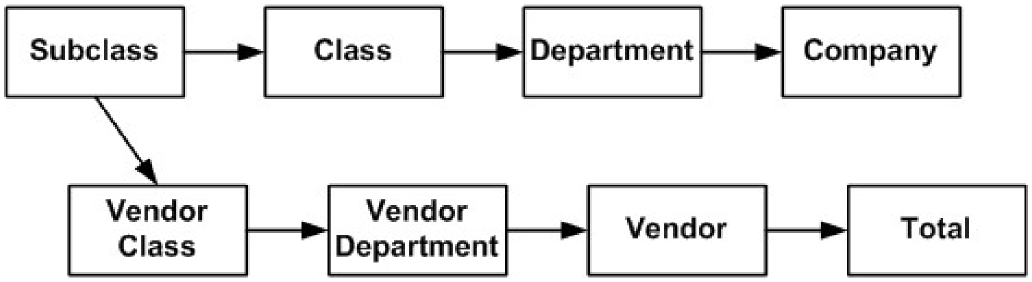


If you want a workbook that includes measures with a lowest base intersection of subclass that required the vendor branch but not the division dimension, you could specify the Hier Mods as follows:

scls-clss, clss-dept, dept-comp, scls-vcls, vcls-vdep, vdep-vend, vend-tot

The workbook would contain the subclass, class, department, and company dimensions, as well as a branch that contains the VendorClass, VendorDept, Vendor, and Total dimensions.

Figure 4-100   Workbook Dimensions



In Figure 4-100, if this workbook is accessed by a user in a part of the business where a subclass only includes styles from a single vendor, the hierarchy built in the workbook will work correctly. However, if it is used in a part of the business where a subclass includes styles with multiple vendors, RPASCE determines (by looking at the VendorClasses that the SKUs in the subclass belong to) that the scls-vcls relationship is ambiguous because the subclass should belong to multiple VendorClasses. In these circumstances, RPASCE builds the hierarchy using one of the valid scls-vcls relationships. As far as the end-user is concerned, the choice of VendorClass for the subclass is likely to be seen as arbitrary, and (in any case) the vendor branch is of little or no practical value in this case.

Remove Dimension

This option allows you to remove existing dimensions from a specific workbook. For instance, if you have an application that has a product hierarchy with the following dimensions:

subclass > class > department > division > company

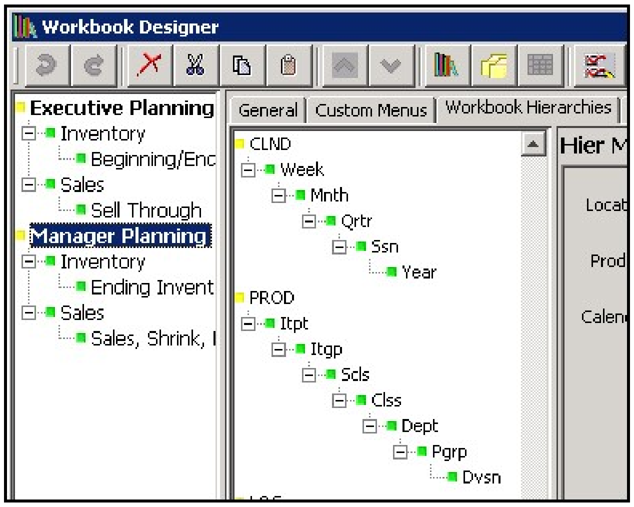
You want a specific workbook to stop at the department dimension and not include the division and company dimensions. You can make that workbook's product hierarchy end at department by completing the following steps:

Note:

You cannot remove the lowest child-parent relationship (the last two dimensions) of a hierarchy.

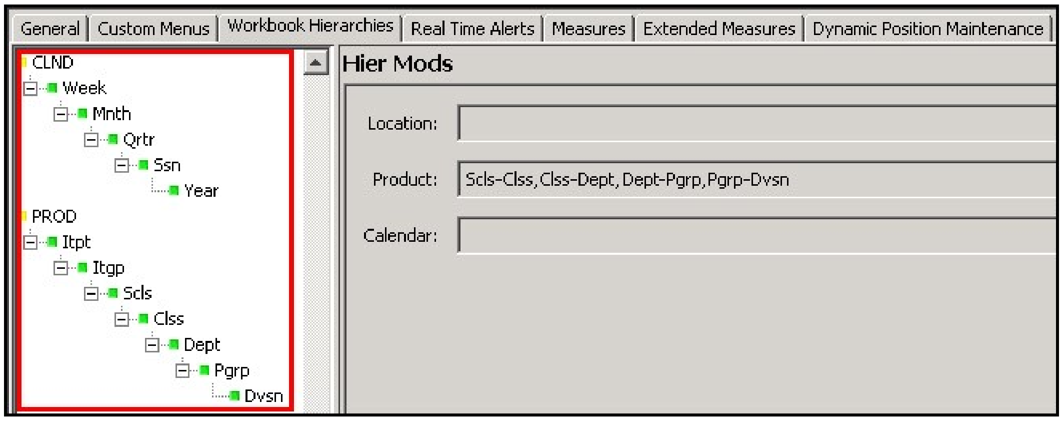
1. Select the workbook in the Workbook Designer window.

Figure 4-101 Workbook List in Workbook Designer



2. Navigate to the Workbook Hierarchies tab. The available hierarchies in the workbook are displayed.

Figure 4-102 Displayed Hierarchies and Dimensions in the Workbook Hierarchies Tab



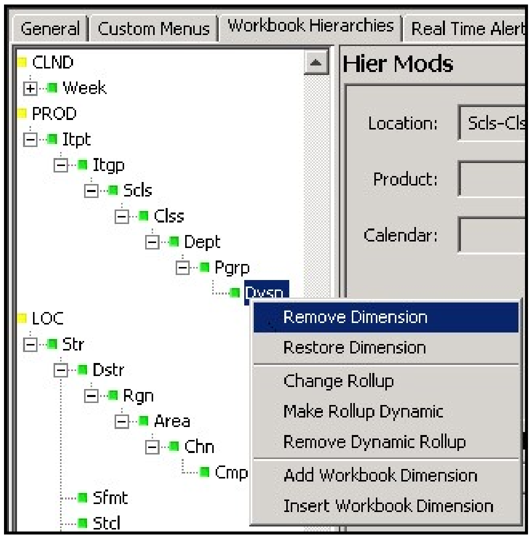
Location:

Product: Scls-Clss, Clss-Dept, Dept-Pgrp, Pgrp-Dvsn

Calendar:

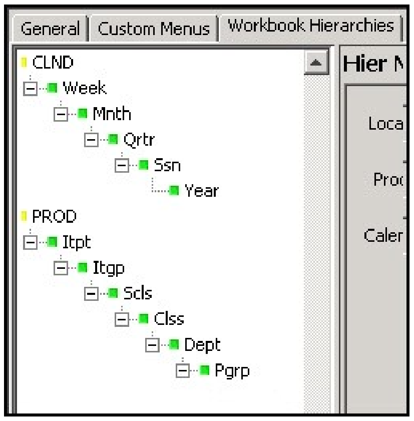
3. Right-click the dimension you want to remove. From the right-click menu, select **Remove Dimension**.

Figure 4-103 Remove Dimension Option



The dimension no longer displays and the hierarchy name becomes italicized to show that the hierarchy has been modified.

Figure 4-104 Removed Dimension



4. Repeat Step 3 for other dimensions if desired.

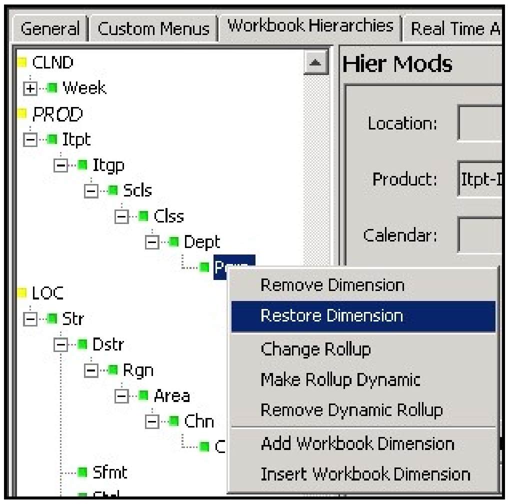
   To bring back a removed dimension, use the Restore Dimension option described in the next section.

Restore Dimension

To restore a removed dimension, complete the following steps:

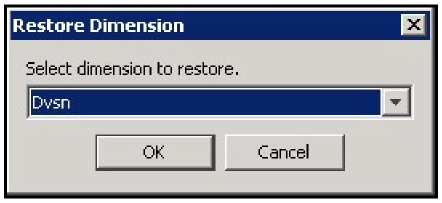
1. Right-click the hierarchy or any dimension within the hierarchy of the removed dimension.
2. In the right-click menu, click **Restore Dimension**.

Figure 4-105   Restore Dimension Option



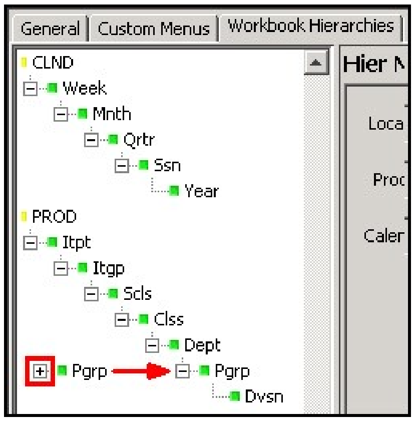
3. The Figure 4-106 opens. From the list, select the dimension you want to restore. Click **OK**.

Figure 4-106   Restore Dimension Window



The Figure 4-106 closes. In the Workbook Hierarchies pane, the dimension higher than the restored dimension has an expand icon next to it. Click the icon to see the restored dimension.

Figure 4-107   Restored Dimension



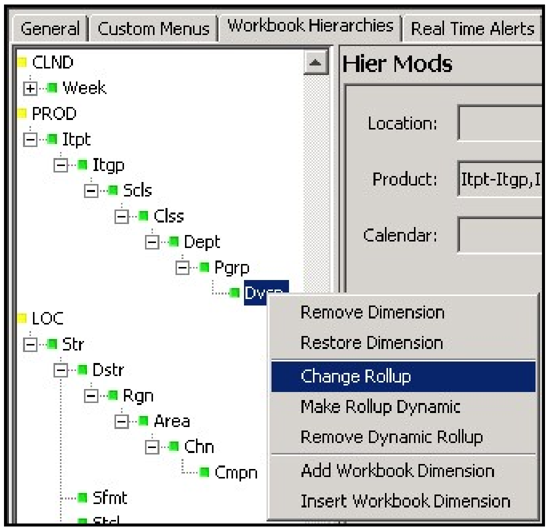
Change Rollup

You can make a dimension rollup to a different dimension by using the Change Rollup feature. For instance, if you want the company dimension to roll up from subclass rather than division, complete the following steps:

Right-click the dimension you want to change.

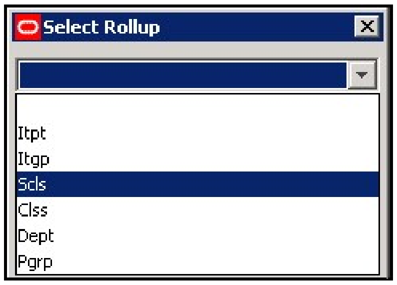
In the right-click menu, click **Change Rollup**.

Figure 4-108 Change Rollup Option



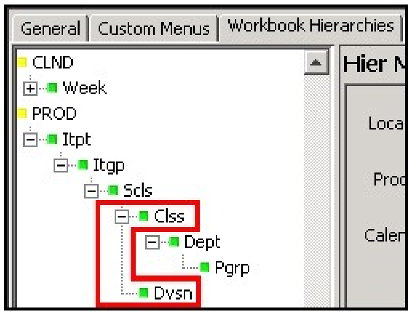
The Figure 4-109 opens. Select the dimension that you want the original dimension to roll up from. (Only dimensions lower than the original are available.) Click **OK**.

Figure 4-109 Select Rollup Window



The Figure 4-109 closes. In the Workbook Hierarchies pane, the dimension now rolls up from the new dimension.

Figure 4-110 New Rollup



Make Rollup Dynamic

Use this option to create a dynamic hierarchy entry for a dimension. Dynamic hierarchical relationships are built using measure data during the workbook build process and may vary each time a workbook is built, but the hierarchical relationships within the workbook remain constant. Dynamic hierarchies can be based on two or more other hierarchies.

For example, the cluster dimension may be an alternate parent of the store dimension in the location hierarchy, and the cluster that a store belongs to may vary by the class dimension in the product hierarchy. In one workbook, a clustering process may determine the store-cluster relationships for each class, and store that information in a measure. A second workbook could then use that relationship to build a dynamic hierarchy. In this example, if the rollup of store to cluster is different for each class, and the user brings more than one class into the workbook, the rollup of store to cluster used in the workbook is based on the data from the first class in the hierarchy.

The dynamic hierarchy process cannot invent a new dimension; it can only change the parent-child relationships of the existing dimensions. In the example, the cluster dimension must be a normal dimension defined through the Hierarchy Definition window and maintained through the loadHier or user-defined dimension processes. The dimension is normal, so it may be used in the base intersection of measures.

Important Dynamic Hierarchy Notes

If the branch of a hierarchy that has parent-child relationships defined by the dynamic hierarchy process only has a business meaning when the dynamic hierarchy process is used, you must use the Remove Dimension option to remove the dimensions in other workbook templates. For example, in the previous cluster example, if the store-cluster relationship only exists in the context of a class, use the Remove Dimension option to hide that relationship in a workbook template that does not include the product hierarchy.

It is your responsibility to ensure that the position names contained in the measure that drives the dynamic hierarchy are real positions that exist in the parent dimension. If not, positions with those names are present in the workbook, but data for them cannot be committed to the PDS, and it is lost when the workbook is deleted.

The resulting dynamic hierarchy is created at the end of the wizard selection process and before the actual workbook build. Therefore, the end product is only visible inside the workbook and not in the wizards.

Note:

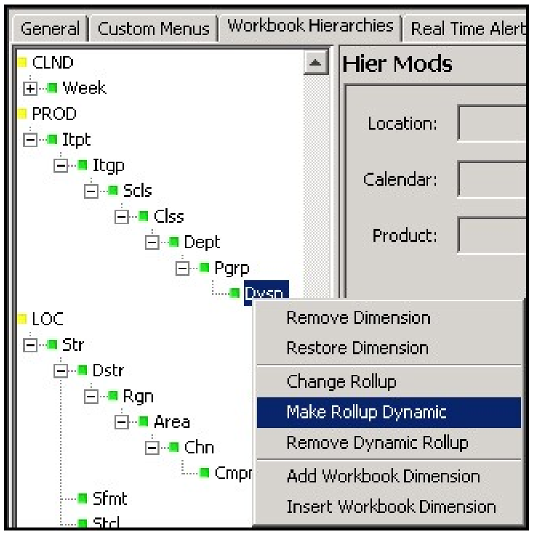
For in-depth examples and explanations about dynamic hierarchies, see Appendix – Dynamic Hierarchies.

Make a Rollup Dynamic

To make a rollup dynamic, complete the following steps:

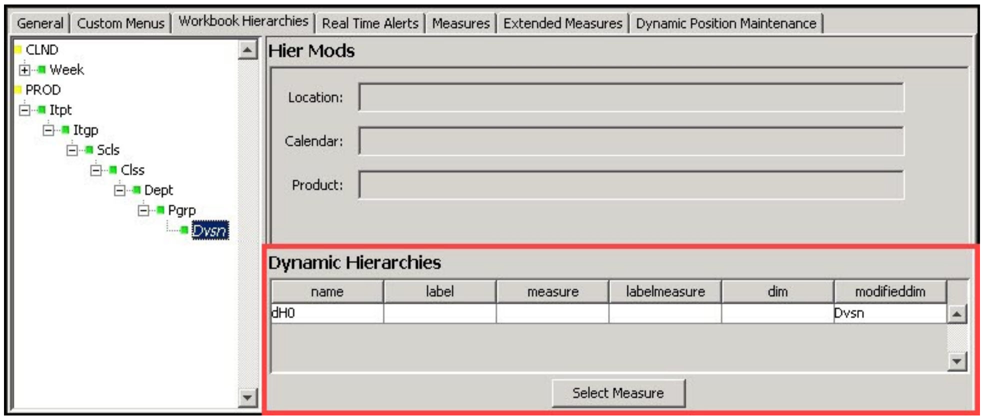
1. Select the dimension you want to change.
2. In the right-click menu, click **Make Rollup Dynamic**.

Figure 4-111 Make Rollup Dynamic Option



The selected dimension is now italicized, and a dynamic hierarchy displays in the **Dynamic Hierarchy** pane. The dimension you selected in Step 1 is populated in the `modifieddim` attribute.

Figure 4-112 New Dynamic Rollup



name	label	measure	labelmeasure	dim	modifieddim
dH0					Dvsn

3. In the **Dynamic Hierarchy** pane, set the following properties:
   - **name** – The name of the dynamic hierarchy. Duplicate names are not allowed. This is an internal name used as a handle to the dynamic hierarchy. It is not visible to the end user.
   - **label** – The label of the dynamic hierarchy entry that represents the workbook-only dimensions that can be configured with an external label. This external label is displayed within the RPASCE Client in place of the dimension name. Only dynamic hierarchy entries that represent workbook-only dimensions have editable label properties.
   - **measure** – This is the name of the measure that holds the name of the parent position. Click **Select Measure** to see the list of the measures that are used in the solution. Type the first few characters of the measure name in the box at the top of the list to go to the required measure or scroll to find it. Double-click to select the desired measure. The measure's base intersection must contain the dimension that is the child of the child-parent relationship (dvsn in the example) or is lower than it (dept, clss and scls in the example) and one or more dimensions which the dynamic position assignment is dependent on (such as comp). The content of the measure is the name of the parent position in the relationship (in the example, this would be the name of the comp that the dvsn belongs to for this comp). This measure may or may not be included in the workbook. If the measure is included in the workbook, changes to the measure within the workbook do not change the parent-child relationships within the workbook, which are static after the workbook is built.
   - **label measure** – This is the name of the measure that holds the label of the parent position. The process to select the label measure is the same as the process to select the measure. The base intersection of the label measure must be the same as the measure. The content of the label measure is the label of the parent position in the relationship.

     Note:

     If a given parent position name has different labels specified for different child positions, the label for the parent position used in the built workbook is one of the different labels, which are arbitrarily selected.

   - **dim** – This is the name of the dimension that is the child in the parent-child relationship. It is the dimension in the modified hierarchy that is used to resolve the modified roll-up relationship represented by the dynamic hier entry. It is stored within the measure and the label measure. The hierarchy that the dynamic hierarchy belongs to is derived from this property.

     Note:

     A dimension that is the modified dimension of one dynamic hierarchy entry cannot be used as the dim attribute of another dynamic hierarchy.

   - **modified dim** – This is the name of the dimension that is the parent in the parent-child relationship. It is the dimension within the modified hierarchy whose roll-up behavior is being modified by the dynamic hierarchy entry. Note that, for dynamic hier entries that represent workbook-only dimensions, the modified dim attribute is the name of the dynamic hier entry and the name of the workbook only dimension that this entry represents.

If you want to remove this new hierarchy, use the instructions described in Remove Dynamic Rollup.

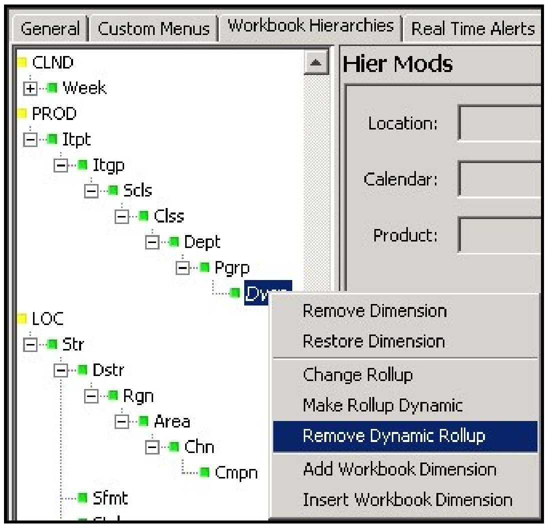
Remove Dynamic Rollup

If you have added a dynamic rollup with the Make Rollup Dynamic option, you can remove it with the Figure 4-113.

To remove a dynamic rollup, complete the following steps:

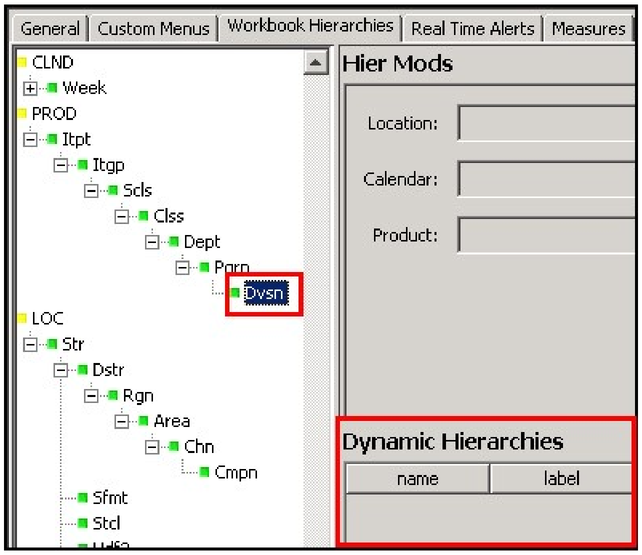
1. Right-click the dimension with the dynamic rollup. The dimension's name is in italics.
2. In the right-click menu, select **Remove Dynamic Rollup**.

Figure 4-113 Remove Dynamic Rollup Option



The dynamic rollup is removed from the **Dynamic Hierarchies** pane and the dimension name is no longer in italics.

Figure 4-114 Removed Dynamic Hierarchy



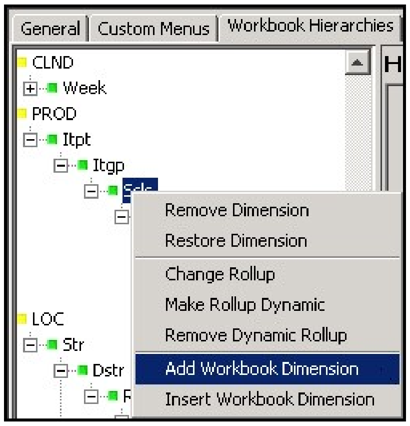
Add Workbook Dimension

Use this option to add a workbook-only dimension to the workbook. Workbook-only dimensions are driven by measures just like regular dynamic dimensions. The difference is that workbook-only dimensions exist only in the workbook and never in the PDS. They can also be used to support a dynamic number of roll-ups and levels. If the workbook dimension does not have data in the measures, it is not displayed as a dimension in the workbook.

To add a workbook dimension, complete the following steps:

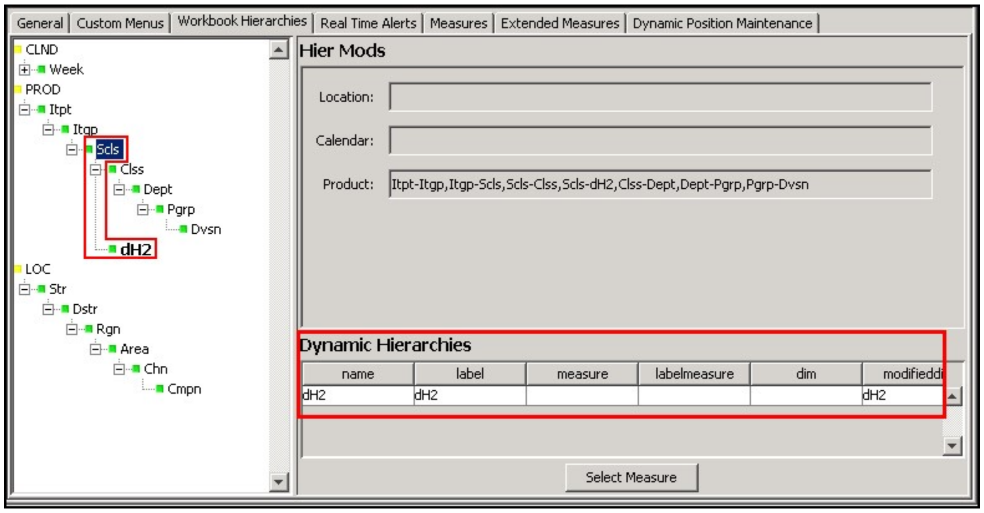
1. Right-click the dimension that you to add a higher dimension to.
2. In the right-click menu, click **Add Workbook Dimension**.

Figure 4-115 Add Workbook Dimension Option



The new workbook dimension displays underneath the selected dimension as well as in the Dynamic Hierarchies table.

Figure 4-116   New Workbook Dimension



Location:

Calendar:

Product: Itpt-Itgp,Itgp-Scls,Scls-Clss,Scls-dH2,Clss-Dept,Dept-Pgrp,Pgrp-Dvsn

Dynamic Hierarchies

name	label	measure	labelmeasure	dim	modifieddi
dH2	dH2				dH2

To continue editing the workbook dimension hierarchy, see the instructions in Make Rollup Dynamic. To remove it, see the Remove Dimension.

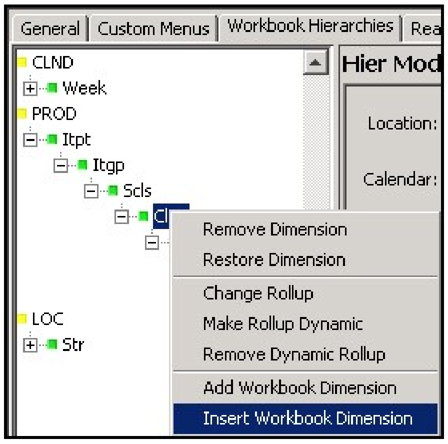
Insert Workbook Dimension

Use this option to insert a workbook-only dimension between existing dimensions in the workbook. Workbook-only dimensions are driven by measures just like regular dynamic dimensions. The difference is that workbook-only dimensions exist only in the workbook and never in the PDS. They can also be used to support a dynamic number of roll-ups and levels. If the workbook dimension does not have data in the measures, it is not displayed as a dimension in the workbook.

To insert a workbook dimension, complete the following steps:

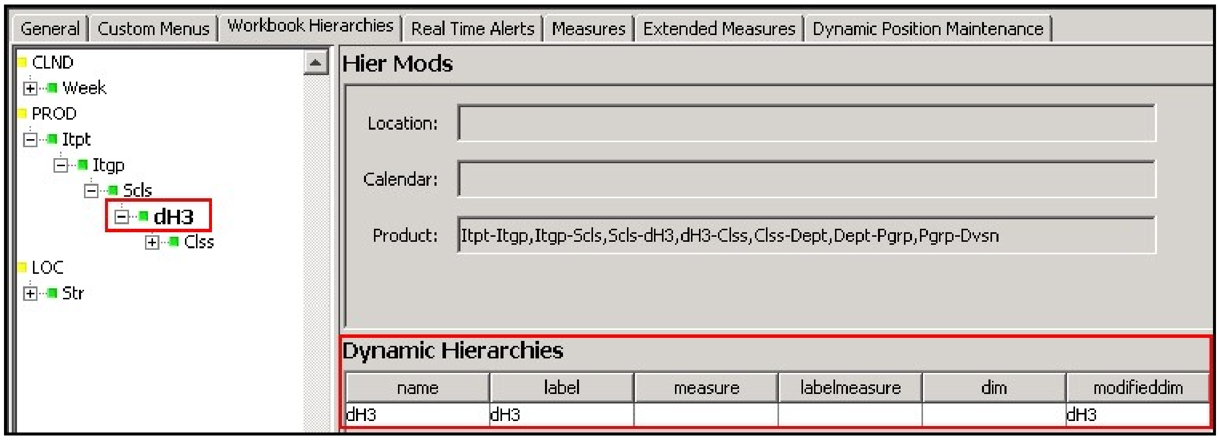
1. Right-click the dimension where you want to insert a lower dimension.
2. In the right-click menu, click **Insert Workbook Dimension**.

Figure 4-117   Insert Workbook Dimension Option



The new dimension displays higher than the selected dimension as well as in the **Dynamic Hierarchies** table.

Figure 4-118 New Workbook-Only Dimension



Location:

Calendar:

Product: Itpt-Itgp,Itgp-Scls,Scls-dH3,dH3-Clss,Clss-Dept,Dept-Pgrp,Pgrp-Dvsn

name	label	measure	labelmeasure	dim	modifieddim
dH3	dH3				dH3

To continue editing the workbook dimension hierarchy, see the instructions in Make Rollup Dynamic. To remove it, see Remove Dimension.

Real Time Alerts Tab

The use of this tab is optional. It allows you to create real time alerts within a workbook that update dynamically. Real time alerts will re-evaluate every time the measures upon which they depend are modified. Real Time Alerts are not two-state (either a hit or not a hit); any number of discrete conditions can be defined for a Real Time Alert. When the alert is evaluated, whichever condition applies will be used (for example, a Real Time Alert defined on an inventory measure could define one condition for low stock and a more severe condition for no stock). The RPASCE Client provides the following support for real time alerts:

- A Real Time Alert can be set as the active alert. When an alert is active within the RPASCE Client, users can navigate between hits for that alert.
- A Real Time Alert can define one or more styles for a Real Time Alert. Every cell of the target measure which evaluates as an alert hit for the Real Time Alert will use the alert's style for that cell in place of the measure's default style. As Real Time Alerts evaluate, the cell styles will update to reflect changes in which cells are hits and which are not.
- One or more messages can be defined for a Real Time Alert. These messages will be displayed in the info pop-up for the current active alert.
- The layout and view positioning used in RPASCE Alert Navigation mode can be defined for Real Time Alerts. If not defined, the RPASCE client will use the default layout based on the number of Alert Worksheets configured, and the Primary view used for navigation is always the one in the first position.

The default layout for the number of selected alert worksheets is as follows:

Table 4-2 Default Layout for the Number of Selected Alert Worksheets

Number of Worksheets	Default Layout
1	Full View

Table 4-2 (Cont.) Default Layout for the Number of Selected Alert Worksheets

Number of Worksheets	Default Layout
2	2 Horizontal
3	1X2 Horizontal
4	4 Tiled

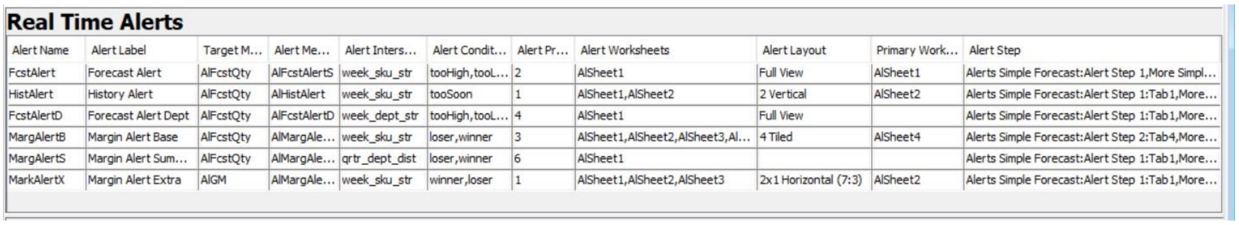
The Real Time Alert configuration specifies which layout is to be used in Navigation Mode and which of the Alert Worksheets listed is the primary one.

The number of selected Alert Worksheets determines which layouts are available for selection.

Table 4-3 Layouts for the Number of Alert Worksheets

Number of Alert Worksheets	Layout Label
1	Full View
2	2 Vertical 2 Horizontal 7:3 Vertical 3:7 Vertical
3	1x2 Vertical 2x1 Vertical 1x2 Horizontal 2x1 Horizontal 2x1 Horizontal (7:3) 2x1 Horizontal (3:7) 1x2 Horizontal (7:3) 1x2 Horizontal (3:7)
4	4 Tiled

Figure 4-119 Real Time Alerts Tab



Real Time Alerts

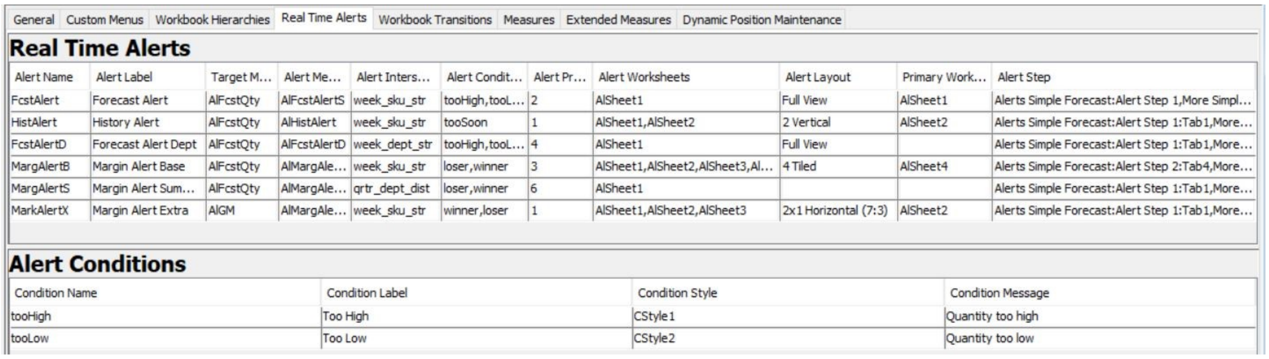
Alert Name	Alert Label	Target M...	Alert Me...	Alert Inters...	Alert Condit...	Alert Pr...	Alert Worksheets	Alert Layout	Primary Work...	Alert Step
FcstAlert	Forecast Alert	AlFcstQty	AlFcstAlertS	week_sku_str	tooHigh,tooL...	2	AlSheet1	Full View	AlSheet1	Alerts Simple Forecast:Alert Step 1,More Simpl...
HistAlert	History Alert	AlFcstQty	AlHistAlert	week_sku_str	tooSoon	1	AlSheet1,AlSheet2	2 Vertical	AlSheet2	Alerts Simple Forecast:Alert Step 1:Tab1,More...
FcstAlertD	Forecast Alert Dept	AlFcstQty	AlFcstAlertD	week_dept_str	tooHigh,tooL...	4	AlSheet1	Full View		Alerts Simple Forecast:Alert Step 1:Tab1,More...
MargAlertB	Margin Alert Base	AlFcstQty	AlMargAle...	week_sku_str	loser,winner	3	AlSheet1,AlSheet2,AlSheet3,Al...	4 Tiled	AlSheet4	Alerts Simple Forecast:Alert Step 2:Tab4,More...
MargAlertS	Margin Alert Sum...	AlFcstQty	AlMargAle...	qrtr_dept_dist	loser,winner	6	AlSheet1			Alerts Simple Forecast:Alert Step 1:Tab1,More...
MarkAlertX	Margin Alert Extra	AlGM	AlMargAle...	week_sku_str	winner,loser	1	AlSheet1,AlSheet2,AlSheet3	2x1 Horizontal (7:3)	AlSheet2	Alerts Simple Forecast:Alert Step 1:Tab1,More...

Condition Definition Interface

The Condition table allows users to create, remove and modify the conditions that are used in alerts for the current workbook. This table contains columns to allow the specification of the following properties:

- **Condition Name** – The name is the value populated within the alert measure to identify which condition, if any, has been triggered for a given cell in the target measure. The condition name will also be used to create a key to allow translation of condition messages.
- **Condition Label** – The label is the externalized label used to represent a condition to users of the client. This label is used to identify the condition within the client. The condition label is internationalized so that it can be translated using the RPASCE multi-language functionality.
- **Condition Style** – The style is the name of a style configured within the Style Definition tool that describes the formatting that should be applied to those cells which are evaluated as hits for the condition.
- **Condition Message** – The message is the text message that is supplied to describe the condition that has been triggered when the workbook alert evaluates a ‘hit' on a given cell of the base measure. This message is internationalized so that it can be translated using the RPASCE multi-language functionality.

Figure 4-120 Condition Definition Table



Real Time Alerts

Alert Name	Alert Label	Target M...	Alert Me...	Alert Inters...	Alert Condit...	Alert Pr...	Alert Worksheets	Alert Layout	Primary Work...	Alert Step
FcstAlert	Forecast Alert	AlFcstQty	AlFcstAlertS	week_sku_str	tooHigh,tooL...	2	AlSheet1	Full View	AlSheet1	Alerts Simple Forecast:Alert Step 1,More Simpl...
HistAlert	History Alert	AlFcstQty	AlHistAlert	week_sku_str	tooSoon	1	AlSheet1,AlSheet2	2 Vertical	AlSheet2	Alerts Simple Forecast:Alert Step 1:Tab1,More...
FcstAlertD	Forecast Alert Dept	AlFcstQty	AlFcstAlertD	week_dept_str	tooHigh,tooL...	4	AlSheet1	Full View		Alerts Simple Forecast:Alert Step 1:Tab1,More...
MargAlertB	Margin Alert Base	AlFcstQty	AlMargAle...	week_sku_str	loser,winner	3	AlSheet1,AlSheet2,AlSheet3,Al...	4 Tiled	AlSheet4	Alerts Simple Forecast:Alert Step 2:Tab4,More...
MargAlertS	Margin Alert Sum...	AlFcstQty	AlMargAle...	qrtr_dept_dist	loser,winner	6	AlSheet1			Alerts Simple Forecast:Alert Step 1:Tab1,More...
MarkAlertX	Margin Alert Extra	AlGM	AlMargAle...	week_sku_str	winner,loser	1	AlSheet1,AlSheet2,AlSheet3	2x1 Horizontal (7:3)	AlSheet2	Alerts Simple Forecast:Alert Step 1:Tab1,More...

Alert Conditions

Condition Name	Condition Label	Condition Style	Condition Message
tooHigh	Too High	CStyle1	Quantity too high
tooLow	Too Low	CStyle2	Quantity too low

Alert Definition Interface

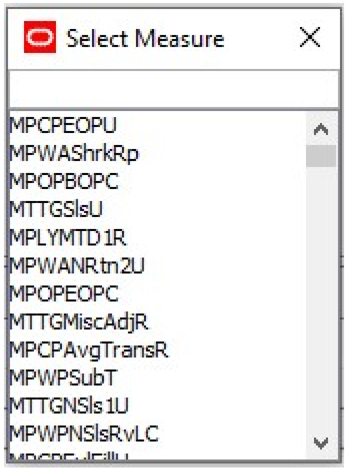
The Alert table allows users to create, remove and modify workbook alerts used within the selected workbook. Selecting the row corresponding to an alert will cause the alert conditions of that alert (if it has any defined) to be populated within the condition table. The alert table contains columns to allow the specification of the following properties:

> **Note:**
>
> If you define two or more alerts with the same alert measure and target measure, you will only be able to see the alert formatting for one of them.

- Alert Name – The name is used as an internal key by RPASCE.
- Alert Label – The label is an external identifier that is used to represent the alert to users of RPASCE.
- Target Measure – The target measure is the measure over which alert hits will be evaluated and to which the formatting will be applied. Boolean measures cannot be used as target measures for real time alerts. Target measure must be defined within the Viewable Measure Profiles of the workbook. When the user clicks the Target Measure

field within the Alert Table, a Figure 4-121 opens that is pre-populated with measures from the workbook viewable measure profiles.

Figure 4-121 Select Measure Window



MPCPEOPU
MPWAShrkRp
MPOPBOPC
MTTGSlsU
MPLYMTD1R
MPWANRtn2U
MPOPEOPC
MTTGMiscAdjR
MPCPAvgTransR
MPWPSubT
MTTGNSls1U
MPWPNSlsRvLC

- Alert Measure – The alert measure is a String measure that is used in the alert evaluation process. It is this measure that will hold the information about what cells in the target measure are considered hits and which condition a given cell has triggered. The intersection of the alert measure must be identical to the alert intersection property of the workbook alert. The alert measure must be included in the workbook rule groups, usually in the calc rule group. The alert measure can be either a hidden measure or in the workbook viewable measure profile. When the user clicks the Alert Measure field within the Alert Table, a Select Measure dialog box pops up that is pre-populated with qualified measures from this workbook. The qualified measures have following characteristics:
  – String type.
  – Measure base intersection is the same as the alert intersection.
  – The measure is used within the workbook rule groups.
  – The measure defagg type is either ambig or popcount. The measure defspread type is none.

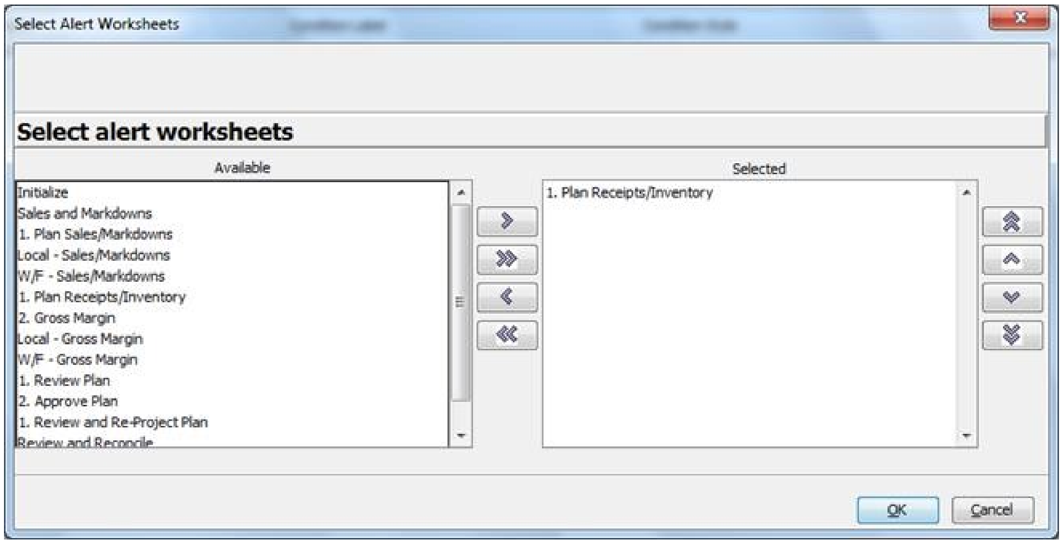
  The alert measure with defagg at popcount is used to show alert formatting on rollup cells, along with a badge of the count of alert hits at the alert intersection. This is done by using a popcount aggregated Alert Measure value at the rollup intersection to format the Target Measure cell at the same intersection. See the "Creation of Alert Badge at Rollup Intersection" section below for details on how to use the popcount alert measure to configure alert badge.
- Alert Intersection – The alert intersection is the intersection on which the workbook alert is defined. It is possible to define multiple workbook alerts over a single target measure, each at a different intersection. Because an alert can only register hits at the intersection of the alert measure, it may be desirable to create multiple alerts for a single target measure so that users can evaluate the target measure at different intersections within the client.
- Alert Conditions – The alert conditions field contains the list of conditions (of the full set of defined conditions for a workbook) that are associated with a given workbook alert. This list is used to identify which conditions should be grouped with an alert when a workbook is registered or built.

- Alert Priority – The alert priority is validated to be unique for the same target measure. The alert priority controls which real time alert formatting is shown in RPASCE Client if multiple alerts are triggered on the same measure. When more than one alert has been created for a single target measure, some cells of the target measure may trigger alert hits for more than one alert. When this is the case, the alert with the lower priority is applied to that cell of the target measure.

  The alert priority also controls the order that real time alerts are displayed in the Alert Navigation drop-down list in the RPASCE Client. The alerts are displayed in the ascending order of the alert priority, with the lower priority (that is, lower number) at the top of the drop-down list.

- Alert Worksheets - The alert worksheets field contains the name of one or more worksheets contained within the alert step that should be displayed when the user enters alert navigation mode for this alert in the RPASCE Client. Between one and four worksheets may be selected and ordered within the view layout. The first worksheet will be the primary view for purposes of alert navigation if Primary Worksheet is not specified in the other column.

Figure 4-122 Select Alert Worksheets Window



Select Alert Worksheets

Select alert worksheets

Available

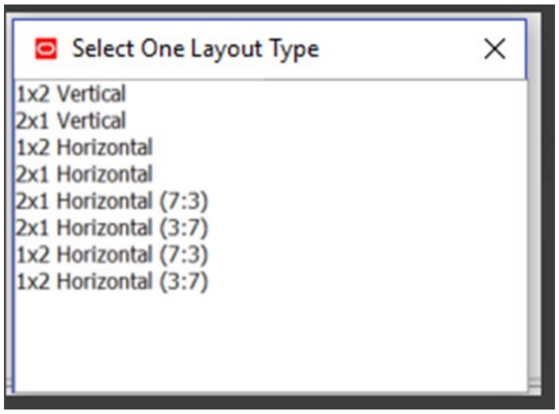
Initialize
Sales and Markdowns
1. Plan Sales/Markdowns
Local - Sales/Markdowns
W/F - Sales/Markdowns
1. Plan Receipts/Inventory
2. Gross Margin
Local - Gross Margin
W/F - Gross Margin
1. Review Plan
2. Approve Plan
1. Review and Re-Project Plan
Review and Reconcile

Selected

1. Plan Receipts/Inventory

- Alert Layout - The alert layout field contains the layout that RPASCE utilizes in the alert navigation mode for this alert. If not specified, the default layout type will be used in RPASCE, as described in the prior section. In the Select One Layout Type window, the drop-down list contains the labels of the available Alert Layouts that are appropriate for the number of selected alert worksheets.

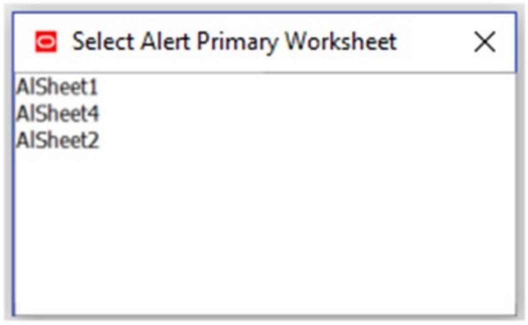
Figure 4-123   Select One Layout Type Window



1x2 Vertical
2x1 Vertical
1x2 Horizontal
2x1 Horizontal
2x1 Horizontal (7:3)
2x1 Horizontal (3:7)
1x2 Horizontal (7:3)
1x2 Horizontal (3:7)

- Primary Worksheet - The primary worksheet specifies which of the selected alert worksheets is the primary one. If not specified, the first worksheet in the Alert Worksheets field will be used as the primary one. The drop-down list in the Select Alert Primary Worksheet window contains the names of the selected alert worksheet. The Primary Worksheet must contain the Target Measure.

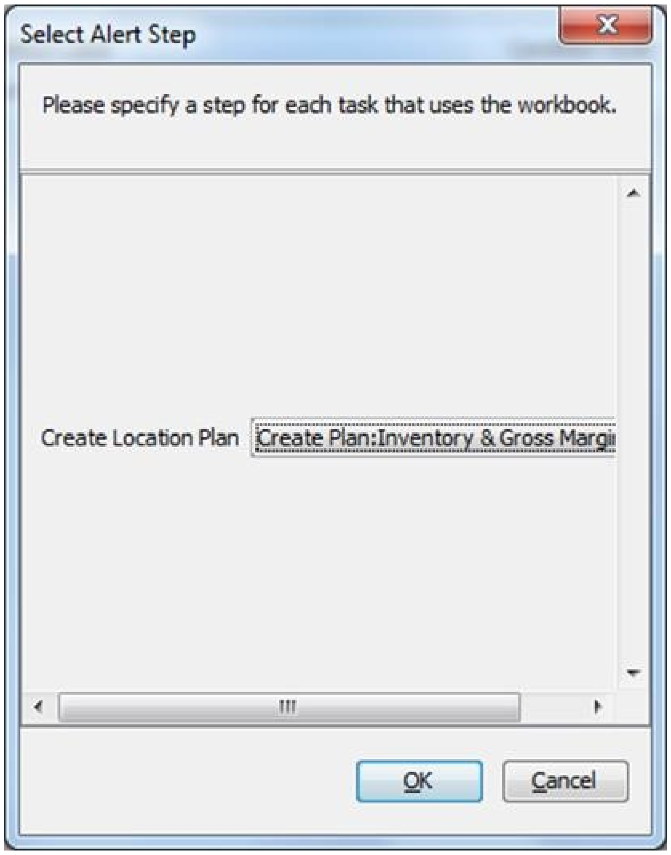
Figure 4-124   Select Alert Primary Worksheet Window



AlSheet1
AlSheet4
AlSheet2

- Alert Step - The alert step field contains the name of the step which the RPASCE Client should move to when entering alert navigation mode for this alert. If more than one task makes use of this workbook, an alert step must be selected for each task. In the Figure 4-125, the drop-down list contains the labels of the available Alert Steps.

Figure 4-125   Select Alert Step Window



Select Alert Step

Please specify a step for each task that uses the workbook.

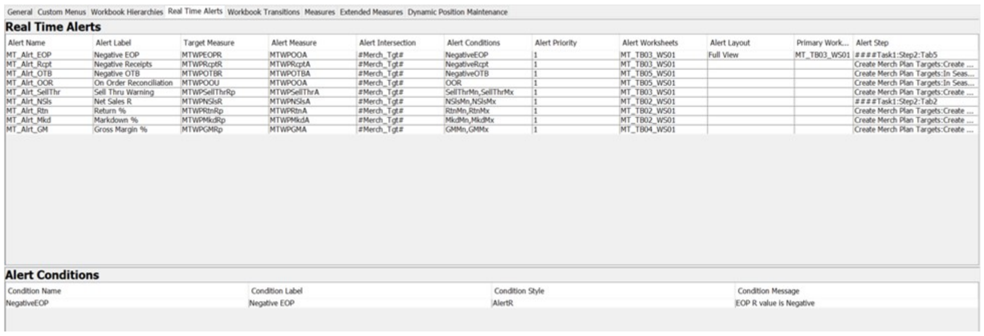
Create Location Plan   Create Plan:Inventory & Gross Margi

Once the Alert Steps are selected, their names are displayed in the Alert Step field, prefixed with identification keyword “####".

Note:

In older versions of RPAS, the Alert Step field displays the labels of selected Alert Steps. This is still supported in the current version. PDS patching does not require re-selecting Alert Steps for existing real time alerts. Since only newly selected Alert Steps have their names displayed, it is possible that the Alert Step column in one workbook may contain both names and labels for different real time alerts.

Figure 4-126 Real Time Alert Definition Table



General | Custom Menus | Workbook Hierarchies | Real Time Alerts | Workbook Transitions | Measures | Extended Measures | Dynamic Position Maintenance

Real Time Alerts

Alert Name	Alert Label	Target Measure	Alert Measure	Alert Intersection	Alert Conditions	Alert Priority	Alert Worksheets	Alert Layout	Primary Work...	Alert Step
MT_Alrt_EOP	Negative EOP	MTWPEOPR	MTWPOOA	#Merch_Tgt#	NegativeEOP	1	MT_TB03_WS01	Full View	MT_TB03_WS01	####Task1:Step2:Tab5
MT_Alrt_Rcpt	Negative Receipts	MTWPRcptR	MTWPRcptA	#Merch_Tgt#	NegativeRcpt	1	MT_TB03_WS01			Create Merch Plan Targets:Create ...
MT_Alrt_OTB	Negative OTB	MTWPOTBR	MTWPOTBA	#Merch_Tgt#	NegativeOTB	1	MT_TB05_WS01			Create Merch Plan Targets:In Seas...
MT_Alrt_OOR	On Order Reconciliation	MTWPOOU	MTWPOOA	#Merch_Tgt#	OOR	1	MT_TB05_WS01			Create Merch Plan Targets:In Seas...
MT_Alrt_SellThr	Sell Thru Warning	MTWPSellThrRp	MTWPSellThrA	#Merch_Tgt#	SellThrMn,SellThrMx	1	MT_TB03_WS01			Create Merch Plan Targets:Create ...
MT_Alrt_NSls	Net Sales R	MTWPNSlsR	MTWPNSlsA	#Merch_Tgt#	NSlsMn,NSlsMx	1	MT_TB02_WS01			####Task1:Step2:Tab2
MT_Alrt_Rtn	Return %	MTWPRtnRp	MTWPRtnA	#Merch_Tgt#	RtnMn,RtnMx	1	MT_TB02_WS01			Create Merch Plan Targets:Create ...
MT_Alrt_Mkd	Markdown %	MTWPMkdRp	MTWPMkdA	#Merch_Tgt#	MkdMn,MkdMx	1	MT_TB02_WS01			Create Merch Plan Targets:Create ...
MT_Alrt_GM	Gross Margin %	MTWPGMRp	MTWPGMA	#Merch_Tgt#	GMMn,GMMx	1	MT_TB04_WS01			Create Merch Plan Targets:Create ...

Alert Conditions

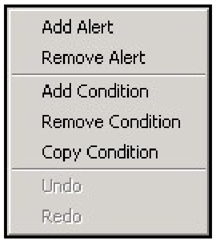
Condition Name	Condition Label	Condition Style	Condition Message
NegativeEOP	Negative EOP	AlertR	EOP R value is Negative

Operations within the Real Time Alerts Panel

The Real Time Alerts panel supports the following operations through the right-click menu:

- Add Alert
- Remove Alert
- Add Condition
- Remove Condition
- Copy Condition

Figure 4-127 Real Time Alerts Menu



Add Alert

Selecting the Add Alert action will create a new Real Time Alert and allow specification of properties within the Alert Table.

Remove Alert

Selecting the Remove Alert action will remove the currently selected alert and its conditions from the workbook. If no alert is currently selected, this option will be unavailable.

Add Condition

Selecting the Add Condition action will create a new Alert Condition within the currently selected real time alert and allow specification of properties within the Condition Table. If there is no currently selected Real Time Alert, this option will be unavailable.

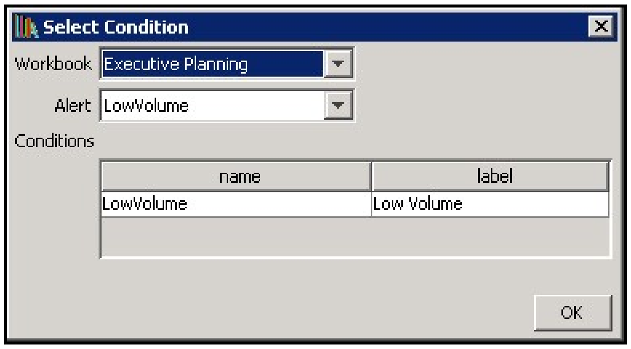
Remove Condition

Selecting the Remove Condition action will remove the currently selected condition from the alert. If no condition is currently selected, this option will be unavailable.

Copy Condition

Selecting the Copy Condition action will allow users to copy an alert condition defined in another real time alert so that it can be used in the currently selected real time alert. When using the Copy Condition action, users must specify solution (for multi-solution configurations), workbook and source alert from which the condition will be copied. The user can then select the condition to be copied to the currently selected alert. If there is no currently selected Real Time Alert, this option will be unavailable.

Figure 4-128 Copy Condition Window



Workbook: Executive Planning

Alert: LowVolume

Conditions

name	label
LowVolume	Low Volume

Example of Workbook Configuration

The process of configuring a workbook alert requires multiple steps across several functional areas within the Configuration Tools. For this reason, an example provides details regarding the end-to-end process of the workbook alert configuration. For more information on Real Time Alert, refer to the Real Time Alerts chapter in *Oracle Retail Predictive Application Server Cloud Edition User Guide*.

For purposes of this example, assume the existence of an inventory measure. The desire is to create an alert to warn of low stock. Assume further that the process defines two conditions that would be considered as a low stock situation: first, if the inventory count falls below a projected sales figure for the period, and, second, if the inventory count falls below some static threshold. For our purposes, we will assume that the failure to meet projected sales is considered more important than falling below the static threshold.

Given these assumptions assume that the following three measures already exist in the application and have processes in place to be calculated or seeded:

- InvU - base inventory count
- ProjSls - projected sales
- MinThrsh - low stock threshold

Assume also that the workbook in which the alert will be defined is already configured to contain all three of the above measure and that processes exist to load and/or calculate the values of the three measures. Given these inputs, the following steps illustrate the configuration of a workbook alert to evaluate low stock conditions.

Definition of the Condition Styles

Within the Style Definition tools, styles must be configured to represent the formatting to be applied for each condition within the alert. Given that we have defined two distinct conditions, it would be desirable to define two distinct styles so that it is possible to visually distinguish which condition is triggered for any given alert hit. Note that it is not necessary to create new styles for each condition; styles may be reused between conditions and may be used both for static formatting of measures and for alert hits.

For our example, we will assume that the following new styles are created.

Warning: Text color yellow, Font type: bold

Error: Text color red, Font type: bold

For more information on style configuration, see the section "Styles" earlier in this document.

Definition of Alert Measures

Within the Measure Definition tool, an alert measure must be defined for the Workbook alert. This measure must be a String-type measure and its base intersection should be identical to the intersection at which the alert is going to be evaluated. In addition, the naValue of the alert measure should always be the empty string, the aggregation type of the alert measure should be either ambig or popcount and the spread type of the alert measure should be none.

The following measure will be created within the Measure Definition tool:

LowStkAlrt - type: String, naValue: ""

Note that the new measure can be configured as by identical processes to normal measure configuration. For more information, see the section "Measures and Components" earlier in this document.

Creation of Alert Calculation Rule

Within the Rule Definition tool, a rule must be defined to evaluate the alert measure. This rule should be added to the calc group for the workbook so that it will be re-evaluated as a part of every calc cycle to allow the workbook alert to update as the data within the workbook changes. Because the alert in question contains two conditions, the rule must be able to evaluate either of the conditions.

The general form of alert calculation rules is one or more if statements that assign the name of a condition to the alert measure if the condition representing the alert is triggered. The assignment of a condition is handled by using a string literal that contains the name of the condition. For the example, the alert rule would be:

LowStkCalc:

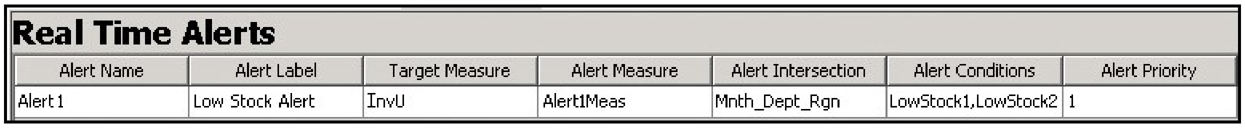
LowStkAlrt = if (InvU < ProjSls, "Condition1", if (InvU < MinThrsh, "Condition2", ""))

For more information, see the section "Rules" earlier in this document.

Creation of the Workbook Alert

Within the Workbook Definition tool, a workbook alert must be created. This new alert should be configured according to the configuration already performed.

Figure 4-129 Configuration of Workbook Alert



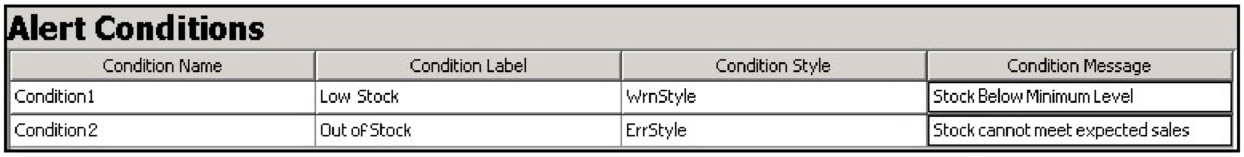
Real Time Alerts

Alert Name	Alert Label	Target Measure	Alert Measure	Alert Intersection	Alert Conditions	Alert Priority
Alert1	Low Stock Alert	InvU	Alert1Meas	Mnth_Dept_Rgn	LowStock1,LowStock2	1

Creation of Alert Conditions

Within the Workbook Definition tool, conditions must be defined for the workbook alert. These two conditions will correspond to the two triggers that will be a part of the alert. Note that it is not necessary to configure conditions before the creation of the alert rule but since the rule will make use of the condition names, it is suggested that they be created prior to the rule.

Figure 4-130 Configuration of Alert Conditions



Alert Conditions

Condition Name	Condition Label	Condition Style	Condition Message
Condition1	Low Stock	WrnStyle	Stock Below Minimum Level
Condition2	Out of Stock	ErrStyle	Stock cannot meet expected sales

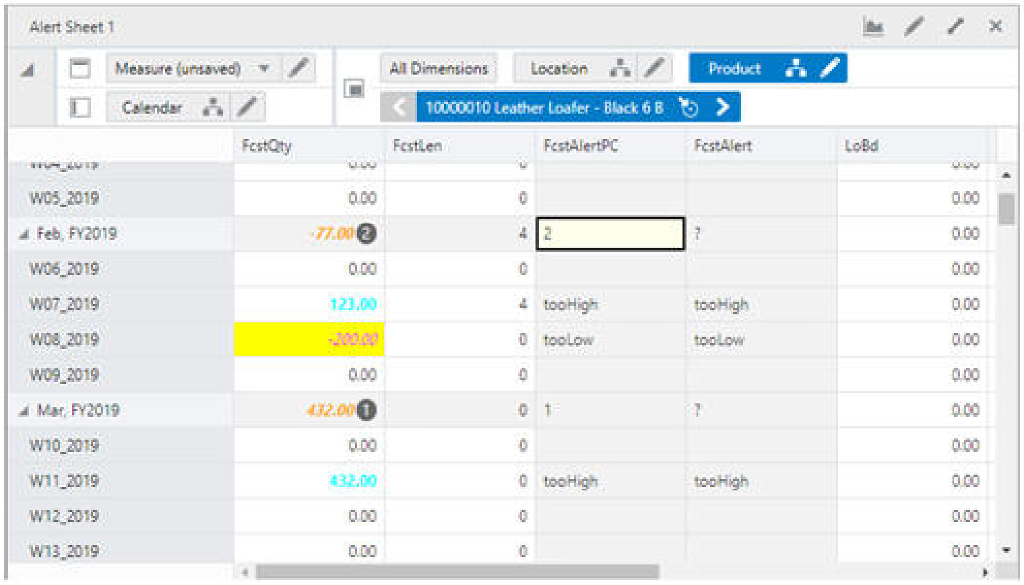
At this point, the workbook alert has been configured. Upon the execution of the calculation cycle, the alert measure will be populated with the appropriate condition for any cell that is evaluated as an alert hit. This information will then be used by the client to display information about the workbook alert.

Creation of an Alert Badge at the Rollup Intersection

To show alerts at a rollup intersection in addition to the alert intersection, the alert measure must have popcount as the default agg type and the alert condition rollup must be defined.

Here is an example of the alert badge display in RPASCE UI:

Figure 4-131 Alert Badge at the Rollup Intersection



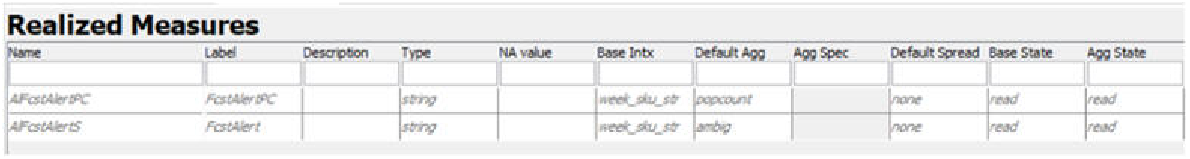
	FcstQty	FcstLen	FcstAlertPC	FcstAlert	LoBd
W05_2019	0.00	0			0.00
Feb, FY2019	-77.00 2	4	2	?	0.00
W06_2019	0.00	0			0.00
W07_2019	123.00	4	tooHigh	tooHigh	0.00
W08_2019	-200.00	0	tooLow	tooLow	0.00
W09_2019	0.00	0			0.00
Mar, FY2019	432.00 1	0	1	?	0.00
W10_2019	0.00	0			0.00
W11_2019	432.00	0	tooHigh	tooHigh	0.00
W12_2019	0.00	0			0.00
W13_2019	0.00	0			0.00

In this example, FcstAlertPC is the badge alert measure and shows the popcount of alert hits (2) at the higher rollup intersection using a badge of the number 2. FcstAlert is a regular alert measure that has ambig as the default agg and only shows alert hits at the alert intersection week_sku_str. The orange color of the alert hits -77 at the higher level is controlled by the style defined in the alert condition rollup, which is used for the badge alert FcstAlertPC at the higher rollup intersection.

Here are the configuration steps for the badge alert measure.

1. Configure the alert measure (AlFctAlertPC) with the default agg as the popcount.

Figure 4-132 Realized Measures

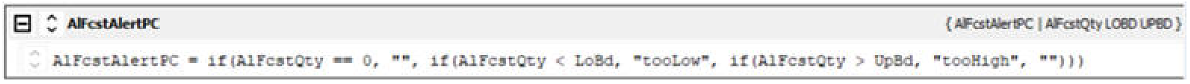


Realized Measures

Name	Label	Description	Type	NA value	Base Intx	Default Agg	Agg Spec	Default Spread	Base State	Agg State
AlFcstAlertPC	FcstAlertPC		string		week_sku_str	popcount		none	read	read
AlFcstAlertS	FcstAlert		string		week_sku_str	ambig		none	read	read

2. The alert calc rule still produces the tooHigh and tooLow conditions:

Figure 4-133 Alert Calc Rule

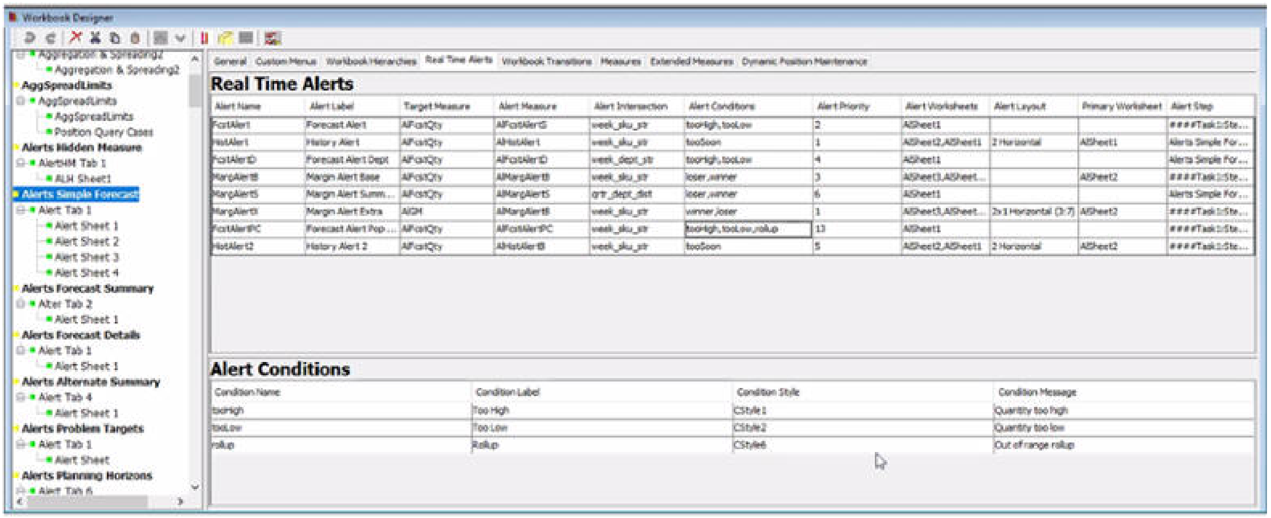


AlFcstAlertPC { AlFcstAlertPC | AlFcstQty LOBD UPBD }

```
AlFcstAlertPC = if(AlFcstQty == 0, "", if(AlFcstQty < LoBd, "tooLow", if(AlFcstQty > UpBd, "tooHigh", "")))
```

3. In the workbook's Real Time Alerts tab, an additional rollup condition must be defined to provide the styles for the aggregated cells and must be configured as one of the alert conditions for the badge alert AlFcstAlertPC.

Figure 4-134   Real Time Alerts Tab



Real Time Alerts

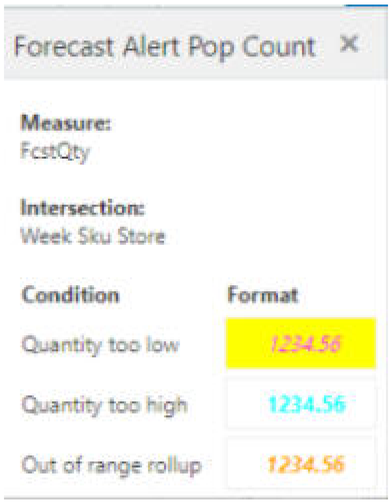
Alert Name	Alert Label	Target Measure	Alert Measure	Alert Intersection	Alert Conditions	Alert Priority	Alert Worksheets	Alert Layout	Primary Worksheet	Alert Step
FcstAlert1	Forecast Alert	AlFcstQty	AlFcstAlertS	week_sku_str	tooHigh,tooLow	2	AlSheet1			####Task1:Ste...
HistAlert1	History Alert	AlFcstQty	AlHistAlert1	week_sku_str	tooSoon	1	AlSheet2,AlSheet1	2 Horizontal	AlSheet1	Alerts Simple For...
FcstAlertD	Forecast Alert Dept	AlFcstQty	AlFcstAlertD	week_dept_str	tooHigh,tooLow	4	AlSheet1			Alerts Simple For...
MargAlertB	Margin Alert Base	AlFcstQty	AlMargAlertB	week_sku_str	loser,winner	3	AlSheet1,AlSheet...		AlSheet2	####Task1:Ste...
MargAlertS	Margin Alert Summ...	AlFcstQty	AlMargAlertS	qrtr_dept_dist	loser,winner	6	AlSheet1			Alerts Simple For...
MargAlertE	Margin Alert Extra	AlGM	AlMargAlertE	week_sku_str	winner,loser	1	AlSheet3,AlSheet...	2x1 Horizontal (3:7)	AlSheet2	####Task1:Ste...
FcstAlertPC	Forecast Alert Pop ...	AlFcstQty	AlFcstAlertPC	week_sku_str	tooHigh,tooLow,rollup	13	AlSheet1			####Task1:Ste...
HistAlert2	History Alert 2	AlFcstQty	AlHistAlertB	week_sku_str	tooSoon	5	AlSheet2,AlSheet1	2 Horizontal	AlSheet2	####Task1:Ste...

Alert Conditions

Condition Name	Condition Label	Condition Style	Condition Message
tooHigh	Too High	CStyle1	Quantity too high
tooLow	Too Low	CStyle2	Quantity too low
rollup	Rollup	CStyle6	Out of range rollup

All of these defined conditions will appear in the Alert Info popup in RPASCE UI so that the user can match the styles to the condition.

Figure 4-135   Forecast Alert Pop Count



Forecast Alert Pop Count

Measure:
FcstQty

Intersection:
Week Sku Store

Condition	Format
Quantity too low	1234.56
Quantity too high	1234.56
Out of range rollup	1234.56

At the alert intersection, the alert measure and its rule calculate the condition strings (tooLow, tooHigh, and so on) for the cells that have alert hits. These are used to format Target Measure cells based on the style defined for those conditions.

At rollup intersections, the Alert Measure cells contain a number based on the number of child populated cells. When a nonzero number is displayed here, the Target Measure cells are formatted with a special rollup style and display the numeric count as a badge on the cell. Normally, the cell's value is used to look up the condition name in the alert definition, and that style is used. For a popcount aggregation, the numeric value will not be found, so a condition name rollup is searched for in order to obtain the styles. If that is not found, the cell will be formatted in red like all alert measure values that do not match a configured condition.

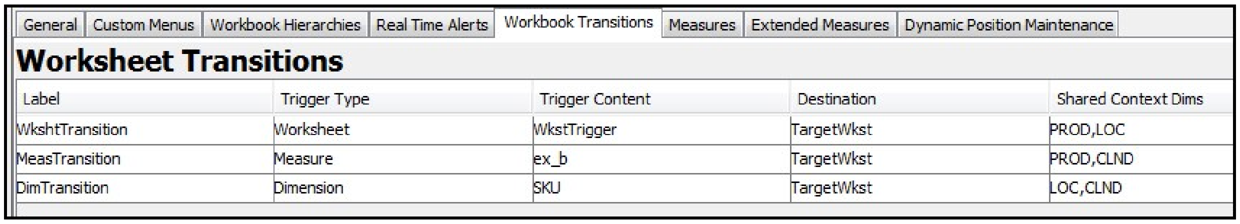
In all cases, the alert display is subject to the following:

- The alert must be selected for display in the Alert Popup dialog box on RPASCE UI.
- When a cell is a hit for more than one alert, the priority is used and only the winner is displayed.

Workbook Transitions Tab

The use of this tab is optional. It allows you to define automatic transitions between the worksheets of a workbook within the RPASCE Client. This transition is initiated through selecting an item in the context menu. When a transition is selected, the current worksheet will be deactivated and a target worksheet will be activated.

Figure 4-136 Worksheet Transitions Tab



Worksheet Transitions

Label	Trigger Type	Trigger Content	Destination	Shared Context Dims
WkshtTransition	Worksheet	WkstTrigger	TargetWkst	PROD,LOC
MeasTransition	Measure	ex_b	TargetWkst	PROD,CLND
DimTransition	Dimension	SKU	TargetWkst	LOC,CLND

Defining a Worksheet Transition

Complete the following steps to define a worksheet transition.

1. Select the Workbook Transitions tab.
2. Right-click in the transitions table area and select **Add**.
3. Enter the following information:
   - **Label** – The label for a worksheet transition will be the text displayed within the right-click menu of the RPASCE Client.
   - **Trigger Type** – The trigger type is used to define the type of content that acts as the trigger for this transition. There are three possible types of triggers: worksheets, measures, and dimensions.
   - **Trigger Content** – The trigger of a worksheet transition determines under which circumstances the menu item for a transition will be available in the context menu. When the trigger is a worksheet, the item will be available from any selection of the specified worksheet. If the trigger is a measure, the item will be available whenever the selection includes the specified measure. If the trigger is a dimension, the item will be available when a position of the specified dimension is selected.
   - **Destination** – The worksheet that will be activated when a worksheet transition is selected from the context menu of the RPASCE Client.
   - **Shared Context Dims** – When making the transition from the source to destination worksheets, the RPASCE Client is capable of automatically applying the selection context of the source worksheet to the destination. Hierarchies listed within the Shared Context Dims property will have this automatic filtering applied.

Removing a Worksheet Transition

In order to remove a worksheet definition from the workbook:

1. Select the Workbook Transitions tab.
2. Select an existing worksheet transition from the worksheet transitions table.
3. Right-click and select **Remove**.

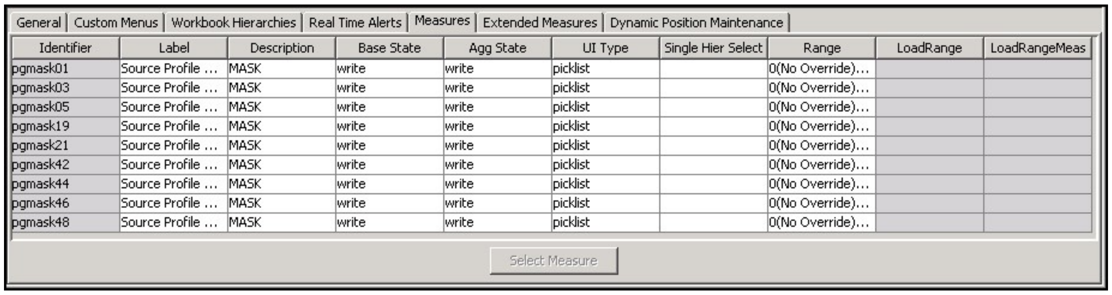
Measures Tab

The use of this tab is optional. It allows you to override certain properties of measures at the workbook level. Use this tab in cases when it is necessary to configure multiple workbooks for a solution and the processes implied by those workbooks require different measure behavior. For example, a measure must be writable in one workbook in a solution but read only in all other workbooks. By using this tab, you can override the following standard measure properties at the workbook level:

- Label, Description
- Base State
- Agg State
- UI Type
- Single Hier Select
- Range

In addition, there are two properties, LoadRange and LoadRangeMeas that are not standard measure properties that may only be set though the Measures tab.

Figure 4-137 Measures Tab



Identifier	Label	Description	Base State	Agg State	UI Type	Single Hier Select	Range	LoadRange	LoadRangeMeas
pgmask01	Source Profile ...	MASK	write	write	picklist		0(No Override)...		
pgmask03	Source Profile ...	MASK	write	write	picklist		0(No Override)...		
pgmask05	Source Profile ...	MASK	write	write	picklist		0(No Override)...		
pgmask19	Source Profile ...	MASK	write	write	picklist		0(No Override)...		
pgmask21	Source Profile ...	MASK	write	write	picklist		0(No Override)...		
pgmask42	Source Profile ...	MASK	write	write	picklist		0(No Override)...		
pgmask44	Source Profile ...	MASK	write	write	picklist		0(No Override)...		
pgmask46	Source Profile ...	MASK	write	write	picklist		0(No Override)...		
pgmask48	Source Profile ...	MASK	write	write	picklist		0(No Override)...		

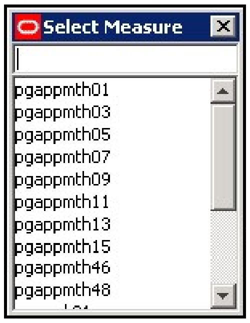
Defining Measure Properties Override Settings for a Workbook

Complete the following steps to define the Measure Properties override settings for a workbook.

1. Click the **Measures** tab.
2. Right-click in the measure table area and select **Add**.
3. Select the newly added row.

4. Click **Select Measure** to get a list of the measures used in the workbook. The Figure 4-138 opens.

Figure 4-138 Select Measure Window



Select Measure

pgappmth01
pgappmth03
pgappmth05
pgappmth07
pgappmth09
pgappmth11
pgappmth13
pgappmth15
pgappmth46
pgappmth48

5. Type the first few characters of the measure name in the box at the top of the list to go to the required measure or scroll to find it. Double-click to select the desired measure. Measures that already have an entry in the Measures tab are listed but unavailable and cannot be selected.

Modifying Measure Properties Override Settings for a Workbook

Complete the following steps to modify the Measure Properties override settings for a workbook.

1. Click the **Measures** tab.
2. Modify the appropriate property information for the measure.

Removing a Measure Override Settings from a Workbook

Complete the following steps to remove the measure property override settings.

1. Click the **Measures** tab.
2. Right-click on the measure row you want to delete and select **Remove**.

Defining the LoadRange and LoadRangeMeas Properties

The **LoadRange** and **LoadRangeMeas** property fields can only be set in the Measure tab. They are used to specify dynamic picklists. Dynamic picklists are picklists whose valid values do not vary within a workbook but can be set dynamically during the workbook build process.

The **LoadRange** and **LoadRangeMea**s properties are retained for backwards compatibility purposes. In most cases context-sensitive picklists (that is, picklists using the `measurerange = measS` syntax in the range property) and single select wizards will be used instead.

Dynamic picklists are of two forms:

- The first form has values that are set according to the data values in cells for another measure (using the same format as for the Range property of static picklists). As with static picklists, the value shown to the user in the UI is the label for the value. It is more usual to use context-sensitive picklists, where picklist values can vary according to context in the workbook.

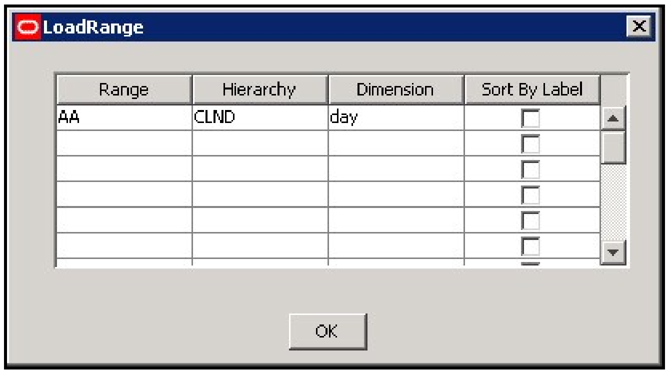
- The second form is to have values that are positions in a branch of a hierarchy. The value shown to the user in the UI is the label of the position, but the content of the cell is the name of the position. The single select wizard provides an alternative method for selecting a position, which is more commonly used.

Defining a Measure with a Dynamic Picklist

Complete the following steps to define a measure with a dynamic picklist.

1. Select the row for the measure.
2. Right-click in the row for the measure and select the Set Dynamic Picklist option.
3. If the picklist measure is to show hierarchy positions, an entry is required in the **LoadRange** property field (and not in the **LoadRangeMeas** property field). Click in the **LoadRange** field. The Figure 4-139 opens.

Figure 4-139 Load Range Window



Range	Hierarchy	Dimension	Sort By Label
AA	CLND	day	☐
			☐
			☐
			☐
			☐
			☐

a. In the Range column, enter a name for the range. This name is used internally by RPASCE and needs to be unique across the application.
b. In the Hierarchy column, select the hierarchy from which the user is to select a position.
c. In the Dimension column, select the dimension from which the user is to select a position. The positions along this dimension, which are brought into the workbook, will be the available choices in the picklist in RPASCE Client.
d. Select the **Sort by Label** check box if the positions in the picklist are to be sorted alphabetically by their label. If this is not selected, the positions will be shown in their internal order, which is the order in which they were defined.
e. Click **OK** to save your changes and close the window.

> **Note:**
>
> When creating a dynamic picklist based on a dimension, you must ensure that the position labels for the source dimension do not contain opening or closing parentheses. Because dynamic picklists cannot parse label names that have parenthesis in them, using them in a label name will cause the system to encounter a run time failure when building or opening workbooks.

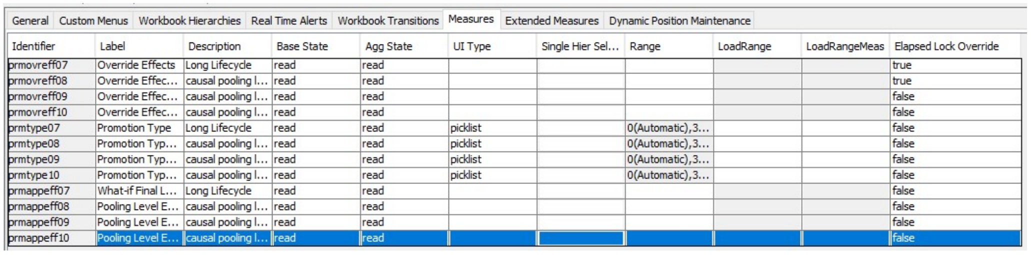
4. If the picklist measure is to display values based on the data values for a cell, an entry is required in the **LoadRangeMeas** property and in the **LoadRange** property fields. Click in the **LoadRangeMeas** field and click **Select Measure**. Type the first few characters of the measure name in the box at the top of the list to go to the required measure or scroll to find it. Double-click the measure whose contents are the valid picklist values. In addition, there should be an entry in the **LoadRange** field for each hierarchy in the base intersection of the selected **LoadRangeMeasure**. Each of the entries in the **LoadRange** field needs a name, hierarchy, and dimension as described previously, but the value for sort label is ignored. If the scope of the workbook is such that it covers multiple cells of the **LoadRangeMeasure**, the available picklist options in the workbook will be constructed from the content of the first cell of the **LoadRangeMeasure** when the dimensions are ranged to the positions selected during the workbook build process.

Make a Measure Editable during Elapsed Period

In RPAS CE, the elapsed lock functionality locks historical cells of all measures along the calendar dimension based on the **r_elapsed** value, across the whole workbook. However, some applications such as RDF require that certain measures be editable even in past or historical periods, while other measures must still be elapsed locked. The measures that can be edited and those that cannot are specific to the Workbook Template, so this setting is a Workbook Template customization rather than a measure property that applies across the whole application. The Workbook Measure Override table provides the Elapsed Lock Override column for users to achieve this goal.

By default, the Elapsed Lock Override column property is false. If users want to override the default value, users must add the measure into the Workbook Measure Override table if this measure row is not configured and set this new column field value to True.

Figure 4-140 Workbook Measure Override Table



Identifier	Label	Description	Base State	Agg State	UI Type	Single Hier Sel...	Range	LoadRange	LoadRangeMeas	Elapsed Lock Override
prmovreff07	Override Effects	Long Lifecycle	read	read						true
prmovreff08	Override Effec...	causal pooling l...	read	read						true
prmovreff09	Override Effec...	causal pooling l...	read	read						false
prmovreff10	Override Effec...	causal pooling l...	read	read						false
prmtype07	Promotion Type	Long Lifecycle	read	read	picklist		0(Automatic),3...			false
prmtype08	Promotion Typ...	causal pooling l...	read	read	picklist		0(Automatic),3...			false
prmtype09	Promotion Typ...	causal pooling l...	read	read	picklist		0(Automatic),3...			false
prmtype10	Promotion Typ...	causal pooling l...	read	read	picklist		0(Automatic),3...			false
prmappeff07	What-if Final L...	Long Lifecycle	read	read						false
prmappeff08	Pooling Level E...	causal pooling l...	read	read						false
prmappeff09	Pooling Level E...	causal pooling l...	read	read						false
prmappeff10	Pooling Level E...	causal pooling l...	read	read						false

Make a measure editable during Elapsed Period using the Elapsed Lock Override column.

> **Note:**
>
> The configuration for Elapsed Lock Override is optional. If it is not configured, it defaults to false and the measure is locked within the Elapsed period and cannot be edited during the workbook. For automation, Applications such as RDF can use Tools APIs to set the Elasped Lock Override property for their EditedMeasure object, to specify the set of measures that they are allowed to edit in history periods.

Extended Measures Tab

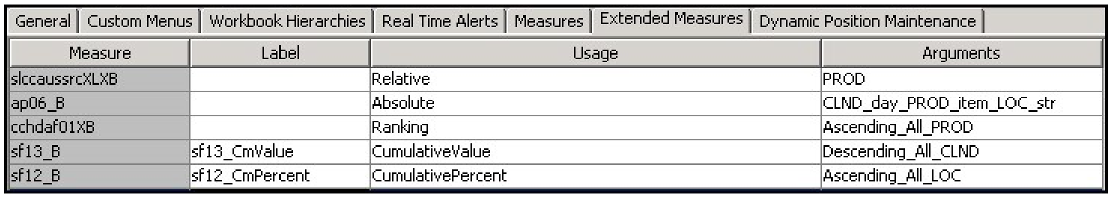
The use of this tab is optional. It allows for the configuration of extended measures that represent different usages of the underlying base measure. The following usages are supported for extended measures:

- Creation of a measure that aggregates data using an alternate aggregation method from the default aggregation type of the measure. The aggregation types available are those configured as the Allowed Aggs of the measure.
- Creation of participation measures, such as absolute and relative percent-to-parent measures.
- Creation of a ranking measure, which assigns a rank in either ascending or descending order to the values of a measure.
- Creation of a cumulative total measure, which contains a running total of a measure's values based upon an ascending or descending ranking.
- Creation of a cumulative percentage measure, which contains a running percent-to-parent contribution total based upon an ascending or descending ranking.

Once defined, these extended measures can be added onto worksheet profiles to be viewed in the RPASCE Client.

> **Note:**
>
> Hybrid is not supported for extended measures.

Figure 4-141 Extended Measures Tab



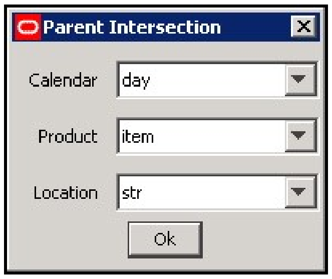
Measure	Label	Usage	Arguments
slccaussrcXLXB		Relative	PROD
ap06_B		Absolute	CLND_day_PROD_item_LOC_str
cchdaf01XB		Ranking	Ascending_All_PROD
sf13_B	sf13_CmValue	CumulativeValue	Descending_All_CLND
sf12_B	sf12_CmPercent	CumulativePercent	Ascending_All_LOC

Adding an Extended Measure

Complete the following steps to add an extended measure.

1. To add a measure, right-click in the table area and select **Add**. The row is inserted into the tab and is highlight in red until you define the **Measure**, **Usage**, and **Argument** fields.
2. Click **Select Measure**. The Select Measure window opens.
3. Double-click a measure to select it.
4. For Relative, Absolute, Ranking, Cumulative value and Cumulative percent extended measures, a label can be specified. This label will be used to display the extended measure within the workbook. If no label is entered, the label of the base measure will be used
5. In the Usage field, select **Relative**, **Absolute**, **Ranking**, **Cumulative value**, **Cumulative percent**, or an alternate aggregation type.
6. Click in the **Arguments** field and select the appropriate options. If the Usage is defined as Absolute the Figure 4-142 opens, set the appropriate options from the window and click **OK**. If the Usage is defined as Relative, select appropriate hierarchy from the list displayed. If the Usage is defined as Ranking, Cumulative value or Cumulative percent, the Figure 4-143 opens, set the appropriate options and click **OK**.

Figure 4-142 Parent Intersection Window



Calendar: day
Product: item
Location: str

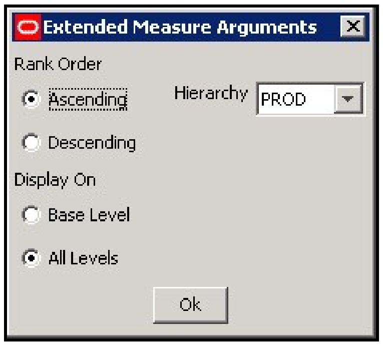
   a. Within the Figure 4-142, the dimension that is used to determine the percent-to-parent contribution of an Absolute extended measure may be set. All dimensions higher than the dimension of the measure's base intersection for that hierarchy are available as well as the value AllDim, which represents all positions within the workbook.
   b. Update the Figure 4-143 for these arguments that may be set for Ranking, Cumulative value and Cumulative percent extended measures:

      Rank Order – Whether the ranking must be in ascending or descending order.

      Display Type – Whether the ranking must be completed only on the base intersection of the measure, or at all aggregate intersections also.

      Hierarchy – The hierarchy along which the values of the measure should be ranked or totaled.

Figure 4-143 Extended Measure Arguments Window



Rank Order: [✓ Ascending] Descending
Hierarchy: PROD
Display On: Base Level [✓ All Levels]

> **Note:**
>
> Alternate aggregation extended measures do not require arguments.

Removing an Extended Measure

To remove an extended measure, right-click on the measure you want to remove and select **Remove**.

Dynamic Position Maintenance Tab

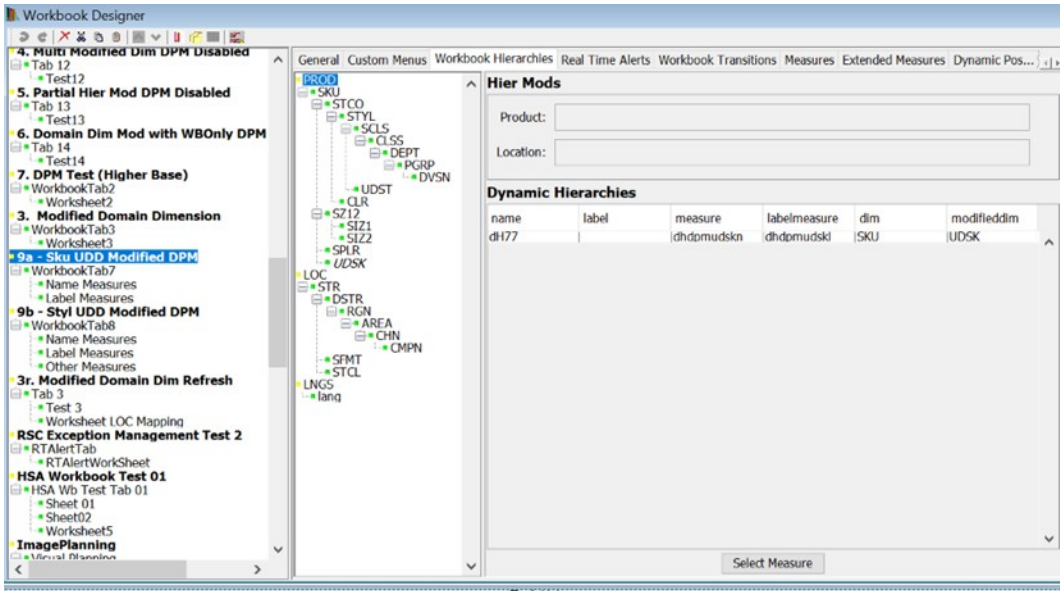
If dimensions are enabled to support Dynamic Position Maintenance (DPM) in the Dimensions pane within the Hierarchy Definition tool, the configuration administrator will see those dimensions in the Dynamic Position Maintenance tab.

> **Note:**
>
> Dynamic Position Maintenance is referred to as Placeholder Maintenance in the *Oracle Retail Predictive Application Server Cloud Edition User Guide* and several other documents. Both terms refer to the same functionality.

The DPM Impact of the Dynamic Hierarchy User-Defined Dimension

When a user-defined dimension (UDD) is the modified dimension of a dynamic hierarchy definition in the workbook, the immediate child dimension of the UDD can be DPM-enabled if this immediate child dimension is DPM-enabled in the Hierarchy Definition configuration. See the section Workbook Hierarchies Tab and the Appendix – Dynamic Hierarchies, for more information on dynamic hierarchies. For example, USDK is both a user-defined dimension and a modified dimension of a dynamic hierarchy dH77. Its immediate child dimension SKU is DPM-enabled in the PDS, and so SKU and its parent dimensions (except USDK because UDD cannot be DPM-enabled) can be DPM-enabled in this workbook, as evidenced by the following Dynamic Position Maintenance figure.

Figure 4-144 Configuration of Dynamic Hierarchies in Workbook Hierarchies Tab

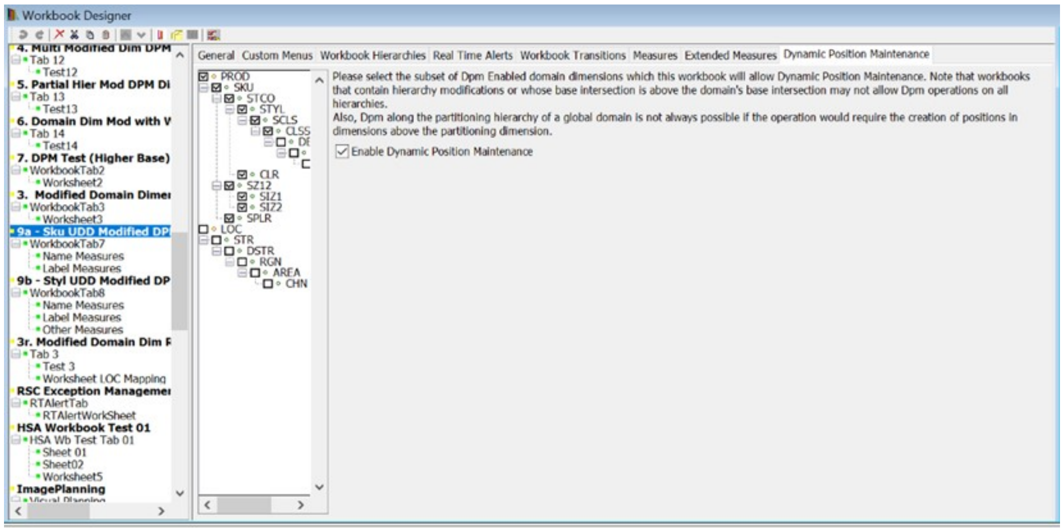


Product:

Location:

name	label	measure	labelmeasure	dim	modifieddim
dH77		dhdpmudskn	dhdpmudskl	SKU	UDSK

Figure 4-145 The SKU and Its Parent Dimensions in the Dynamic Position Maintenance Tab



Please select the subset of Dpm Enabled domain dimensions which this workbook will allow Dynamic Position Maintenance. Note that workbooks that contain hierarchy modifications or whose base intersection is above the domain's base intersection may not allow Dpm operations on all hierarchies.
Also, Dpm along the partitioning hierarchy of a global domain is not always possible if the operation would require the creation of positions in dimensions above the partitioning dimension.

[✓] Enable Dynamic Position Maintenance

Depending on how the workbook has been configured, some dimensions that support DPM within the PDS may not be available for DPM within the workbook. The following conditions will disqualify a dimension for DPM within a workbook:

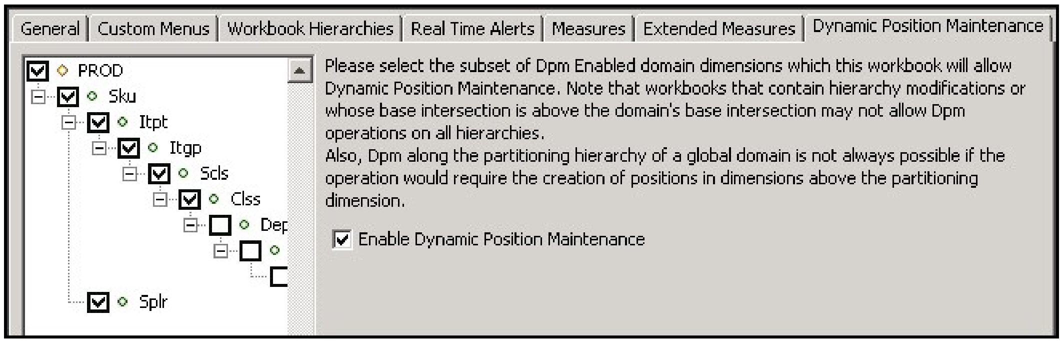
- Dimensions lower than the base intersection of the workbook will not be available for DPM. The base intersection of a workbook is the lowest base intersection of all measures within the workbook.
- Dimensions belonging to a hierarchy that has been modified by the Remove Dimension or Change Rollup operations within the Workbook Hierarchies panel will not support DPM.

- Dimensions more than one level higher than the modified dimension of a dynamic hierarchy will not support DPM.
- Dimensions that are the modified dimension of a dynamic hierarchy will not support DPM if their immediate parent dimension (where week is the parent of day) does not support DPM.

In addition, if the workbook has been configured to contain workbook-only dimensions, those workbook-only dimensions support DPM if the dimension they aggregate supports DPM. See the “Workbook Hierarchies" section and the Dynamic Hierarchies Appendix for more information on workbook-only dimensions.

To enable DPM functionality in the workbook, select **Enable Dynamic Position Maintenance**. The configuration administrator may then select the highest dimension in the hierarchy in which the end user will add positions. See the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide* and *Oracle Retail Predictive Application Server Cloud Edition User Guide* for more information on Dynamic Position Maintenance.

Figure 4-146 Dynamic Position Maintenance Tab



Please select the subset of Dpm Enabled domain dimensions which this workbook will allow Dynamic Position Maintenance. Note that workbooks that contain hierarchy modifications or whose base intersection is above the domain's base intersection may not allow Dpm operations on all hierarchies.
Also, Dpm along the partitioning hierarchy of a global domain is not always possible if the operation would require the creation of positions in dimensions above the partitioning dimension.

☑ Enable Dynamic Position Maintenance

Note:

Users cannot import positions from alternate hierarchies that are not already in the workbook to be used as parents for new DPM positions.

Remove a Workbook

Complete the following steps to remove a workbook.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Workbooks**. The Figure 4-71 opens in the workspace.

1. Select the workbook to remove.
2. From the toolbar, click **Delete**, or select **Remove** from the right-click menu.
3. Click **Yes**. The associated worksheets and tabs are removed.

Working with the Rule Group Simulator

The RPASCE calculation engine is powerful and complex. The rule group approach means that there are very many potential calculation paths. However, during any configuration exercise, there is a significant design verification cost to ensure that the behavior is as would be expected by an end user. The rule group simulator enables the verification of the interaction between measures from within the Configuration Tools. It cannot, however, enable the verification of the calculations themselves because that requires a full PDS to be built.

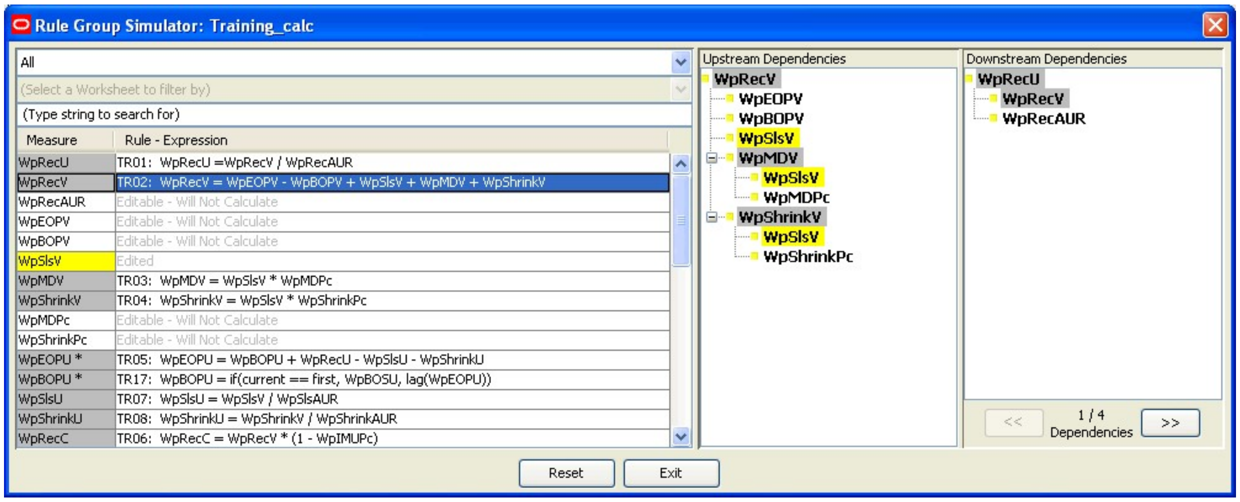
The Rule Group Simulator is integrated into the workbook tool, and it uses all of the measures used in the rule set in the workbook, which may be more than those mentioned in the rule group being simulated. Users of the rule group simulator are expected to understand the calculation cycle, especially with respect to measure protection processing and the process that determines which expressions will be evaluated. See Appendix – Calculation Engine User Guide for more information.

> **Note:**
>
> The rule group simulator is not able to simulate the expressions that will be evaluated as the result of a rule group transition, nor simulate the calculations that will follow if a rule group is evaluated in full mode, such as when evaluated from the mace utility, or the evaluation of the load rule group when a workbook is built.

About the Rule Group Simulator

The Rule Group Simulator feature is provided in a separate window with two areas: a measure table, and a tree view with Upstream and Downstream Dependencies panes.

Figure 4-147 Rule Group Simulator Window



Measure	Rule - Expression
WpRecU	TR01: WpRecU =WpRecV / WpRecAUR
WpRecV	TR02: WpRecV = WpEOPV - WpBOPV + WpSlsV + WpMDV + WpShrinkV
WpRecAUR	Editable - Will Not Calculate
WpEOPV	Editable - Will Not Calculate
WpBOPV	Editable - Will Not Calculate
WpSlsV	Edited
WpMDV	TR03: WpMDV = WpSlsV * WpMDPc
WpShrinkV	TR04: WpShrinkV = WpSlsV * WpShrinkPc
WpMDPc	Editable - Will Not Calculate
WpShrinkPc	Editable - Will Not Calculate
WpEOPU *	TR05: WpEOPU = WpBOPU + WpRecU - WpSlsU - WpShrinkU
WpBOPU *	TR17: WpBOPU = if(current == first, WpBOSU, lag(WpEOPU))
WpSlsU	TR07: WpSlsU = WpSlsV / WpSlsAUR
WpShrinkU	TR08: WpShrinkU = WpShrinkV / WpShrinkAUR
WpRecC	TR06: WpRecC = WpRecV * (1 - WpIMUPc)

The measure table displays all of the measures in the scope of the simulation. The Measure column displays the measure name. The measure status is reflected by color coding. A tooltip also displays the measure status when the mouse is placed over the measure name. All measures can be shown, or the list of measures can be filtered. Editable measures can have their status toggled (to or from Edited), and the simulator immediately updates all statuses, calculations, and trees. Figure 4-4 explains the meaning of the color coding used in the Figure 4-147.

Table 4-4   Color Coding Used in the Rule Group Simulator Window

Color	Meaning
Yellow	Edited. If it is a recalc measure, it will be calculated by indirect spreading of another measure through a mapping rule and recalculation at aggregated levels. If the measure has another aggregation type, it will be calculated by spreading and aggregation.
Pale Gray	Editable. Although the measure is not forced, and thus is still editable, it will be calculated through the calculation engine having to select an expression in an affected rule.
White	Editable. Will not be calculated, so it will not change at all.
Pale Blue	Protected by protection processing. Although the measure is protected (usually this will be because it is the measure on the left-hand side of the only expression in a rule), it is not forced because none of the right-hand side measures are changed, so it does not need to be calculated, and it will not change at all.
Mid Blue	Protected by protection processing. Is forced, so it will be calculated.
Dark Blue	Read-only. The measure is set as being read only in the measure properties, so it will not change at all. A measure that is read-only but is going to be calculated will be shown as mid-blue. That status takes priority over read-only.

Note:

The status of a measure encapsulates two concepts that are not as closely linked as they may seem:

Whether or not the measure can be edited (shades of blue = no, white/gray/yellow = yes)

Whether or not the measure will be calculated.

It is possible for a measure to be editable, but it would be calculated if a calculate were issued. Similarly, it is possible for a measure to be protected by protection processing that would not be calculated if a calculate were issued.

The Rule - Expression column of the table shows the calculation for each measure. For those measures that would be calculated if the end user issued a *calculate* with the current collection of edited measures, the rule and expression that would be used to calculate the measure is shown. For non-calculated measures, this column displays the measure status.

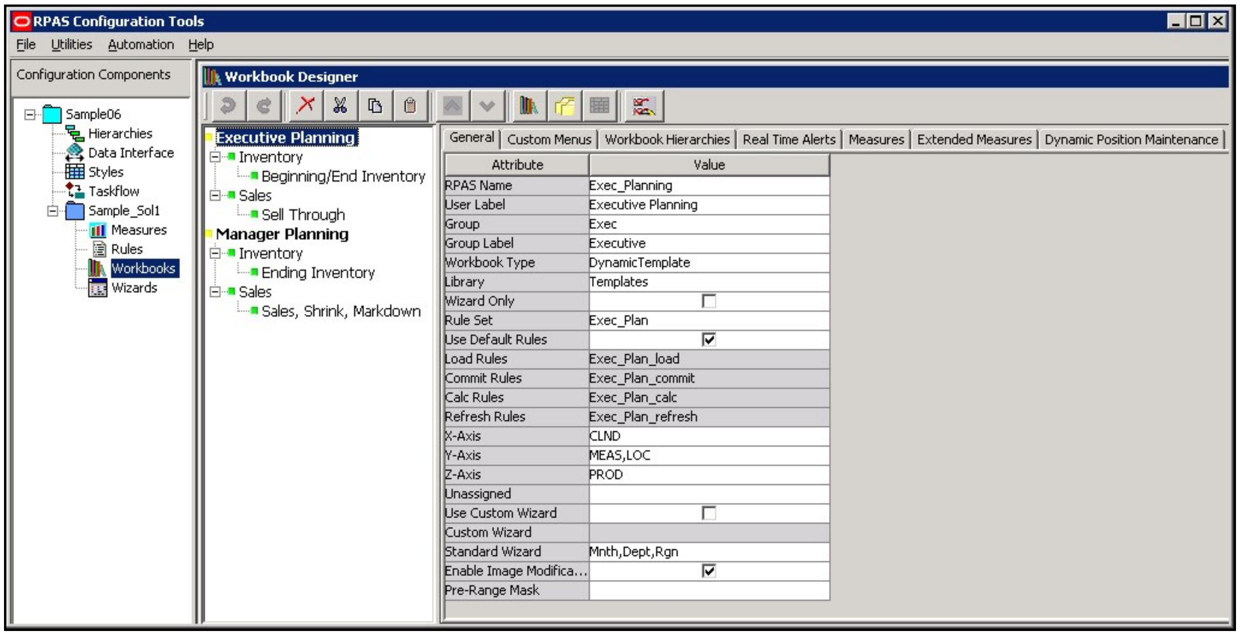
The tree view shows (in separate panes) the upstream and downstream measure relationships (that is, the expressions that will be evaluated) for the measure with focus. Measures in the panes are also color coded. If the measure with focus would be calculated, the upstream pane shows the expression to calculate it and all measures that it is dependent upon (calculated from) with their expressions, if appropriate. The downstream pane similarly shows measures that are dependent upon (calculated from) the measure with focus, if there are any. If the measure with focus is on the right-hand side of several expressions that will be calculated, each of the expressions can be viewed using the forward and backward arrows.

Invoking the Rule Group Simulator

Complete the following steps to open the Rule Group Simulator window.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Workbooks**. The Figure 4-148 opens in the workspace.

Figure 4-148 Workbook Designer Window



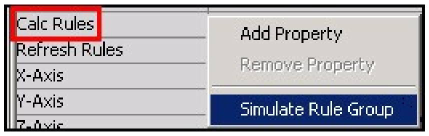
- Sample06
  - Hierarchies
  - Data Interface
  - Styles
  - Taskflow
  - Sample_Sol1
    - Measures
    - Rules
    - Workbooks
    - Wizards

- Executive Planning
  - Inventory
    - Beginning/End Inventory
  - Sales
    - Sell Through
- Manager Planning
  - Inventory
    - Ending Inventory
  - Sales
    - Sales, Shrink, Markdown

Attribute	Value
RPAS Name	Exec_Planning
User Label	Executive Planning
Group	Exec
Group Label	Executive
Workbook Type	DynamicTemplate
Library	Templates
Wizard Only	☐
Rule Set	Exec_Plan
Use Default Rules	☑
Load Rules	Exec_Plan_load
Commit Rules	Exec_Plan_commit
Calc Rules	Exec_Plan_calc
Refresh Rules	Exec_Plan_refresh
X-Axis	CLND
Y-Axis	MEAS,LOC
Z-Axis	PROD
Unassigned	
Use Custom Wizard	☐
Custom Wizard	
Standard Wizard	Mnth,Dept,Rgn
Enable Image Modifica...	☑
Pre-Range Mask	

1. Select one of the workbooks in the Workbook Designer tree display.
2. Select the **General** tab of the Workbook properties table.
3. Right-click on **Calc Rules** and select **Simulate Rule Group**.

Figure 4-149 Simulate Rule Group Menu Option



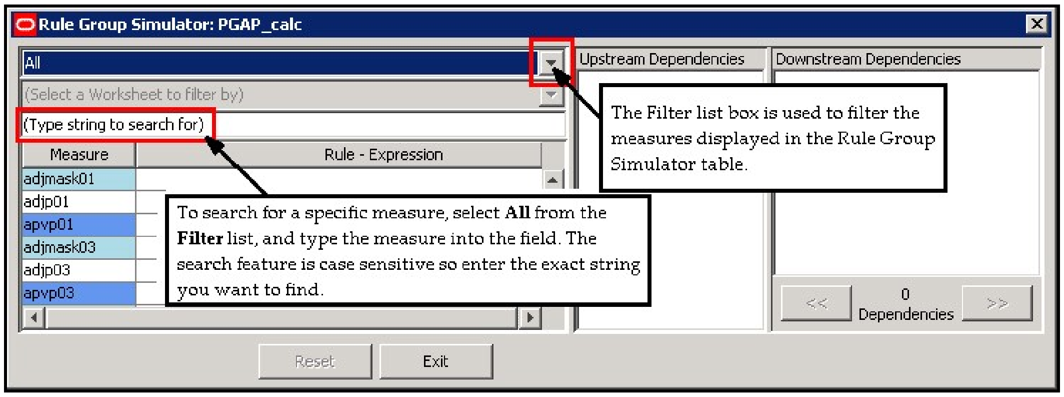
4. The Rule Group Simulator window opens.

Filtering the Measures Table

Complete the following steps to filter measures displayed in the rule group simulator.

1. Open the Rule Group Simulator. See Invoking the Rule Group Simulator.

Figure 4-150 Rule Group Simulator Showing Filter and Search Features



The Filter list box is used to filter the measures displayed in the Rule Group Simulator table.

To search for a specific measure, select **All** from the **Filter** list, and type the measure into the field. The search feature is case sensitive so enter the exact string you want to find.

2. To filter the measures table, select the one of the following options from the **Filter** list:
   - Select **All** to display all measures in the workbook.
   - Select **Will Calculate (According to Calculation Order)** to display only those measures that will be calculated. The sequence of the measures displayed is the sequence in which they will be calculated. Measures that are edited are not shown.
   - Select **Will Not Calculate** to display only those measures that will not be calculated will be shown. Measures that are edited are not shown.
   - Select **Read Only** to display only those measures whose status is read-only (that is, have a Base State and Agg State of read).
   - Select **Contains String** to type a case-sensitive string to filter by in the text box underneath the **Filter** list option. Only measures that include the string entered are displayed.
   - Select **By Worksheet** to select a worksheet from the current workbook using the list option underneath the filter list option. Only the default measures from that worksheet are shown. The Worksheet list option is unavailable until By Worksheet is selected from the **Filter** list.
3. If searching for a specific measure, set the filter to **All**, and enter a search string (case-sensitive) in the box underneath the **Filter** list. The first measure that includes the string will be shown and will become the measure with focus.

Changing the Edited Status of Measures

Complete the following steps to change the edited status of measures.

> **Note:**
>
> Only measures that can be edited (colors gray, white, and yellow) may have their status changed.

1. Select the name of the measure in the measure table.
   - If the status was previously **Editable** (gray or white), the status of the measure changes to **Edited** (yellow).
   - If the status was previously **Edited** (yellow), the status of the measure changes to **Editable** (either gray or white, depending on the rule group and the other edits currently applied).

   After any change in status, the simulator updates all necessary statuses, calculations, and tree views.
2. To change the status of all **Edited** measures back to **Editable** (gray or white), click **Reset**.

   > **Note:**
   >
   > Measures that are calculated in a cycle, which typically includes BOP and EOP inventory values, are indicated with an asterisk ***** next to their names in the measure table and Upstream and Downstream Dependencies panes.

Using the Upstream and Downstream Panes

Complete the following steps to use the upstream and downstream panes

1. To change the measure with focus for the upstream and downstream panes:
   a. With the Filter list option set to All, enter a search string in the field under the Filter list option. The first measure that contains the search string will get focus.

      > **Note:**
      >
      > Remember, when searching by measures, the text entered in the search text field is case-sensitive.

   b. Click in the Calculation Column of the measure table for the measure.
   c. Click on any occurrence of the measure in the Upstream or Downstream Dependencies panes.

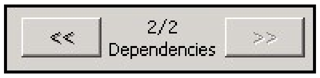
When the focus changes, the tree panes are refreshed as appropriate based on the measure which currently has focus, and the measure table scrolls so the measure with focus is shown.

The measure with focus always displays at the top of the Upstream Dependencies pane. If it will be calculated, the Upstream pane shows the measures that it is dependent upon (calculated from, directly and indirectly). This is displayed using a parent-child tree structure with the measures used to calculate an individual measure showing as children of it. If the children are also calculated, they display with their dependent measures, and so on. Therefore, the expanded Upstream Dependencies tree view displays all the measure relationships that affect the measure with focus.

The Downstream Dependencies pane shows measures that are dependent upon (calculated from) the measure with focus, if there are any. Measure relationships (expressions) display in a parent-child tree structure. If the measure with focus is on the right-hand side of several expressions that will be calculated, the relationships cannot all be shown at the same time in a simple tree structure, so a single relationship is displayed. The number of such relationships, and the one being shown, is indicated at the bottom of the pane.

2. To collapse the detail of the dependencies for a measure in the Upstream or Downstream panes, click **Minus** next to the measure name. The **Minus** changes to a **Plus**, and the detail is collapsed. To expand the detail of the dependencies for a measure in the Upstream or Downstream Dependencies panes, click the **Plus** next to the measure name. The **Plus** changes to a **Minus** and the detail is expanded.
3. To change which measure relationship for the measure with focus is shown in the downstream pane, click the back or forward arrows beneath the Downstream Dependencies pane.

Figure 4-151 Back and Forward Arrows



Note:

The Rule-Expression column of the measures table will display multiple result expressions with a note beside the rule name saying that it is a multiple result. Furthermore, the entire expression will be displayed showing all of the left-hand side measures that comprise the multiple results. If a measure that has focus is one of the multiple result measures, it will be shown in the Upstream and Downstream Dependencies panes as MeasA [+MeasB][+MeasC] where MeasA is the measure with focus and MeasB and MeasC are the other multiple result measures.

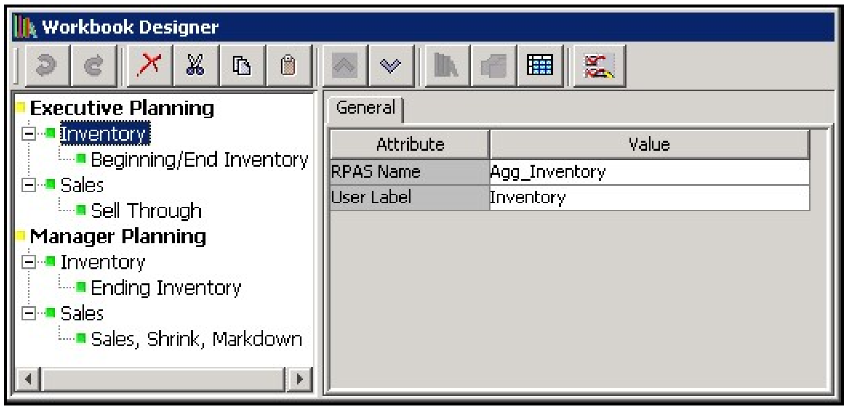
Exiting the Rule Group Simulator

To exit the rule group simulator, click **Exit**.

Working with Workbook Tabs

Workbook tabs are a feature in the RPASCE Client that enables the workflow to be separated into steps or business processes. The RPASCE Client can have tabs, but those are set up using the taskflow. Each workbook must have at least one tab. Users select the appropriate tab to use depending on the stage they have reached in the business process. A tab may contain one or more worksheets that allow the users to interact with the data in the workbook. The measures available, the orientation of the hierarchies, and the base intersection that data is available for may vary by worksheet within the tab.

Figure 4-152 Workbook Designer Window



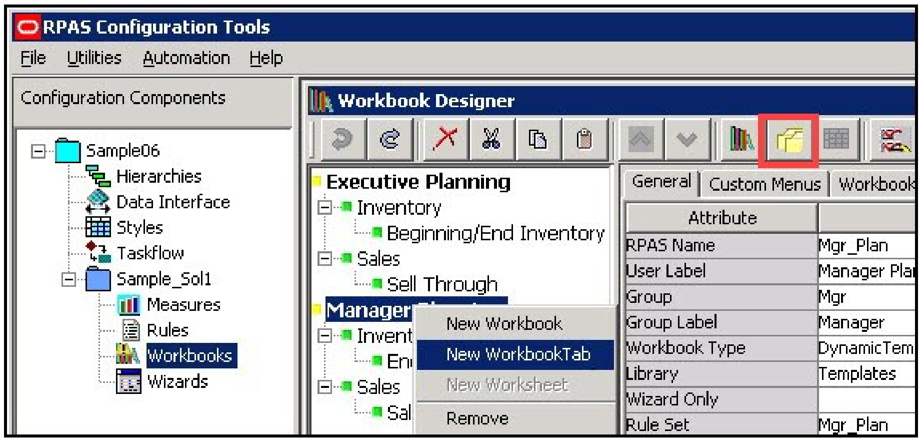
Attribute	Value
RPAS Name	Agg_Inventory
User Label	Inventory

Create a Workbook Tab

Complete the following steps to create a workbook tab.

Navigate: In the Configuration Components pane, select **Project** then **Solution**, and then **Workbooks**. The Figure 4-200 opens in the workspace.

Figure 4-153 Workbook Design Window with Create New Workbook Tab



Attribute	
RPAS Name	Mgr_Plan
User Label	Manager Plan
Group	Mgr
Group Label	Manager
Workbook Type	DynamicTem
Library	Templates
Wizard Only	
Rule Set	Mgr_Plan

1. Select the workbook in which to create a new workbook tab.
2. Choose one of the following methods:
   - Click **New Workbook Tab**.
   - Right-click and select the **New Workbook Tab.**
   - Select an existing workbook tab in the workbook, and press **Insert**.

   A new workbook tab is created.
3. In the **RPAS Name** field, enter RPAS internal name of the workbook tab.
4. In the **User Label** field, enter a description of the workbook tab that users will see on the tab in the RPASCE Client.

Edit Workbook Tab Properties

Complete the following steps to edit workbook tab properties

Navigate: In the Configuration Components pane, select **Project** then **Solution**, and then **Workbooks**. The Figure 4-200 opens in the workspace.

- Select the workbook tab whose properties are to be edited.
- In the General tab, type the RPAS Name and the User Label.
- Complete one of the following to alter the order in which the tab is displayed in the RPASCE Client:
  - To change the order of the tabs, select the up button or down button as necessary.
  - Drag and drop the tab in the Workbook Designer tree display.

Remove a Workbook Tab

Complete the following steps to remove a workbook tab.

Navigate: In the Configuration Components pane, select **Project** then **Solution**, and then **Workbooks**. The Figure 4-200 opens in the workspace.

1. Select the workbook tab to remove.
2. Choose one of the following methods:
3. From the toolbar, click **Delete**.
4. Select **Remove** from the right-click menu.
5. Press **Delete**.
6. Click **Yes**.

The workbook tab and associated worksheets are removed.

Comprehensive Workbook Validation

In addition to the real-time validation of user inputs, the Workbook Designer has the capability to perform a comprehensive validation of all Workbook content in the solution. Upon performing comprehensive validation, the Task List will be updated with any validation issues present in the Workbooks of the Configuration.

Complete the following steps to complete comprehensive workbook validation.

Navigate: In the Configuration Components pane, select **Project** then **Solution**, and then **Workbooks**. The Figure 4-200 opens in the workspace.

1. From the toolbar, click **Comprehensive Validation**.
2. All Workbook content for the solution will be validated and any issues detected will be entered in the Task List.

Working with Worksheets

The following sections provide descriptions of the worksheet components.

Measures and Worksheets

A worksheet is a specific window into the data in a workbook. Worksheets are placed on workbook tabs. Using the Configuration Tools, you define the measures on a worksheet, the base intersection the worksheet uses, and the orientation of the hierarchies on the worksheet. The workbook measures can be organized in following categories for a worksheet:

- Selected profile
- Viewable profile
- Hidden
- Extended

Selected Profile Measures

The selected profile contains the list of measures that are initially displayed for this profile in the RPASCE Client. All worksheets have one profile that is marked as the default profile. This default profile is the profile that displays when a workbook is initially built. There must be at least one measure on each of the selectable profiles.

Viewable Profile Measures

The viewable profile contains the full list of measures that the RPASCE Client user can view in the worksheet by using the Show/Hide functionality within the RPASCE Client. It must contain all of the measures in the default profile, but it often includes further measures that are not initially displayed.

Hidden Measures

Hidden measures are those that are used in the rule set assigned to a workbook, but that are not assigned to any of the profiles in any of the worksheets contained in that workbook. This might include measures that are used purely for calculation purposes and would have no usefulness to the RPASCE Client user.

Extended Measures

Extended measures, which represent different usages of the underlying base measures, can be added to the default or viewable worksheet profile. You can add extended measures that are aggregated based on different aggregation methods. The aggregation methods available for selection are based on the Allowed Aggs of the base measure. The same base measure can have multiple extended measures based on different aggregation methods.

You can also add extended measures that represent the relative and absolute percent-to-parent contributions. The same base measure can have multiple extended measures based on different selections for relative and absolute percent-to-parent contributions.

You can add extended measures that rank the values of a measure in either ascending or descending order or that contain a running total of a measure's values based on an ascending or descending ranking.

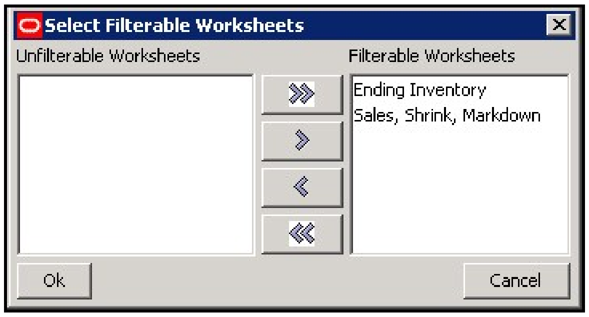
Worksheet Types

There are two types of worksheets supported. The Pivot/Chart type worksheet type is supported for all Fusion Clients and RPASCE Clients.

Filterable Sheets

The RPASCE Client provides the capability to filter the visible positions of a worksheet based upon the selection context of a different worksheet. By default, this filtering is enabled for all worksheets. It is possible to exclude worksheets from the list of filterable worksheets. Selecting this attribute in the table opens the Figure 4-154.

Figure 4-154 Select Filterable Worksheets Window



Ending Inventory
Sales, Shrink, Markdown

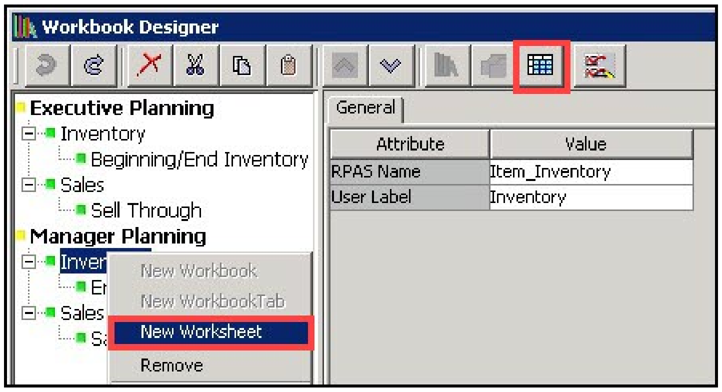
The Figure 4-154 contains the list of all candidates for filtering from the worksheet being edited. By default, all worksheets will be in the Filterable Worksheet column. By selecting one or more worksheets and moving them to the Unfilterable Worksheets column, those worksheets will not be able to be filtered from the worksheet being edited.

Create a Worksheet

Complete the following steps to create a worksheet.

Navigate: In the Configuration Components pane, select **Project** then **Solution**, and then **Workbooks**. The Figure 4-200 opens in the workspace.

Figure 4-155 Workbook Design Window with Create New Worksheet Option



Attribute	Value
RPAS Name	Item_Inventory
User Label	Inventory

1. Choose one of the following methods:

   Click **New Worksheet** to create a new worksheet.

   Right-click and select **New Worksheet.**

   Select another worksheet on the same tab.

2. Press **Insert**.

   A new worksheet is created.

3. Assign the appropriate properties using the worksheet tabs (General, Position Queries, Measure Profiles, Style Overrides, and Window Formatting).

Defining Worksheet Properties for Pivot/Chart Worksheets

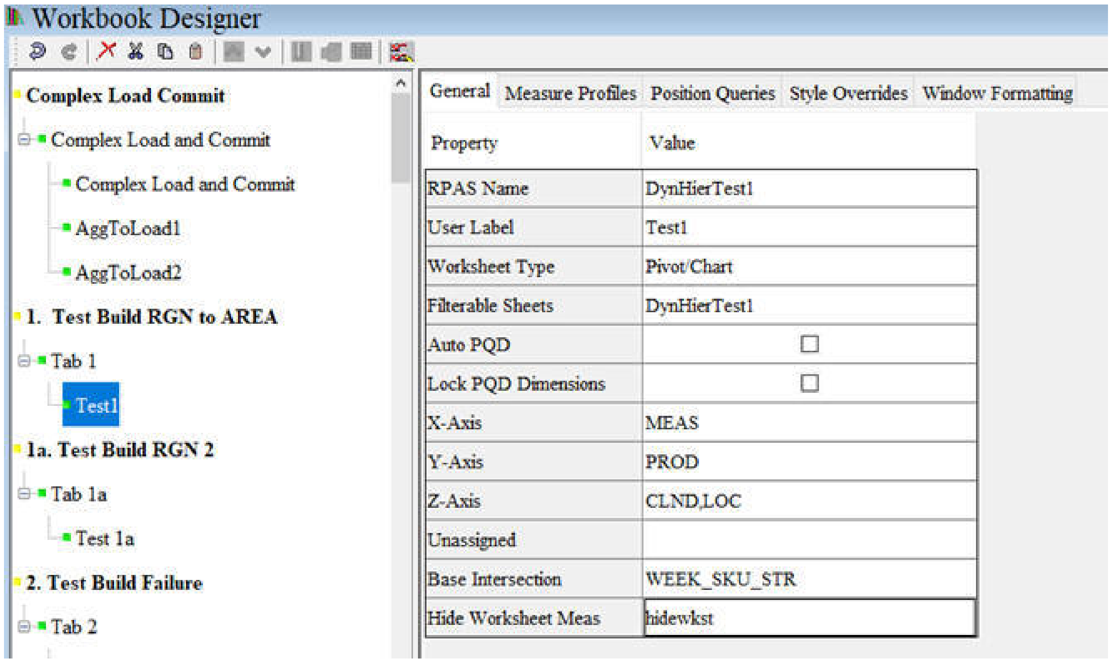
When the Pivot/Chart worksheet type is selected from the Workbook Designer window, the following tabs display in the workspace:

- General Tab
- Measure Profiles Tab
- Position Queries Tab
- Style Overrides Tab
- Window Formatting Tab

General Tab

Complete the following steps to update the General tab.

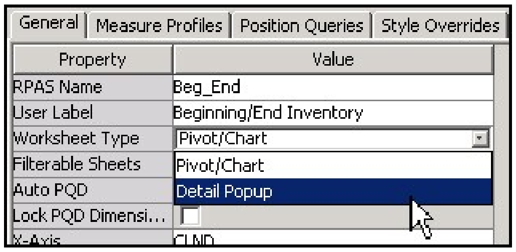
Figure 4-156 General Tab for a Worksheet



Property	Value
RPAS Name	DynHierTest1
User Label	Test1
Worksheet Type	Pivot/Chart
Filterable Sheets	DynHierTest1
Auto PQD	☐
Lock PQD Dimensions	☐
X-Axis	MEAS
Y-Axis	PROD
Z-Axis	CLND,LOC
Unassigned	
Base Intersection	WEEK_SKU_STR
Hide Worksheet Meas	hidewkst

1. In the **RPAS Name** field, enter the RPAS internal name of the worksheet.
2. In the **User Label** field, enter a description of the worksheet that users will see.
3. In the **Worksheet Type** field, select **Pivot/Chart**.

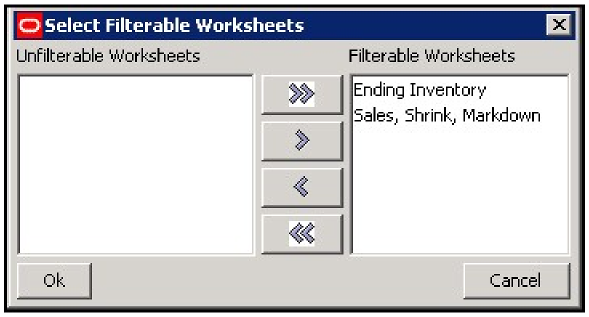
Figure 4-157 Worksheet Type Pick List



Property	Value
RPAS Name	Beg_End
User Label	Beginning/End Inventory
Worksheet Type	Pivot/Chart
Filterable Sheets	
Auto PQD	
Lock PQD Dimensi...	☐
X-Axis	CLND

4. Determine candidates for filtering from the worksheet, if any.

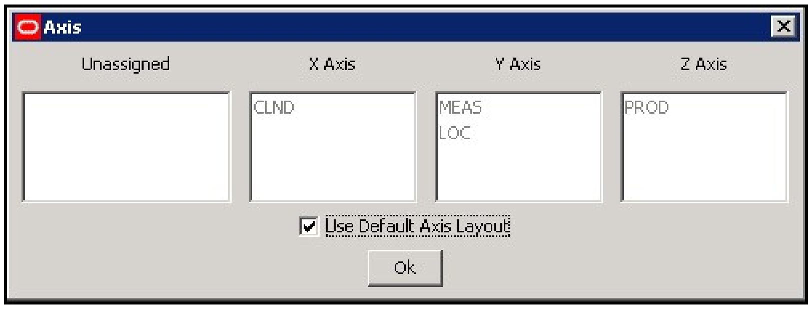
Figure 4-158 Select Filterable Worksheets Window



Ending Inventory
Sales, Shrink, Markdown

5. Define the axis layout of the worksheet.
   a. Click the **X-axis**, **Y-axis**, **Z-axis**, or **Unassigned** field. The Axis window opens.

Figure 4-159 Axis Window



Axis

Unassigned	X Axis	Y Axis	Z Axis
	CLND	MEAS LOC	PROD

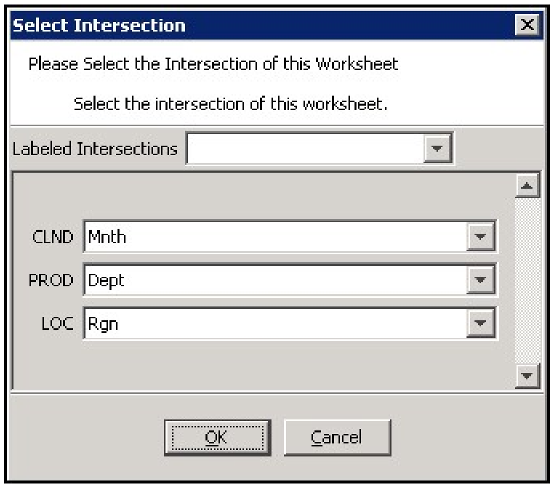
☑ Use Default Axis Layout

   b. Drag the hierarchies to the appropriate axis column.
   c. Click **OK** to save any changes and close the window.

> **Note:**
>
> The Hierarchies that appear in this process are those used by measures placed on the default and viewable profiles for the worksheet. If no measures have yet been placed in those profiles, no hierarchies appear in this process.

6. Click in the **Base Intersection** field. The Figure 4-160 opens.

Figure 4-160 Select Intersection Window



Select Intersection

Please Select the Intersection of this Worksheet

Select the intersection of this worksheet.

Labeled Intersections

CLND	Mnth
PROD	Dept
LOC	Rgn

Note:

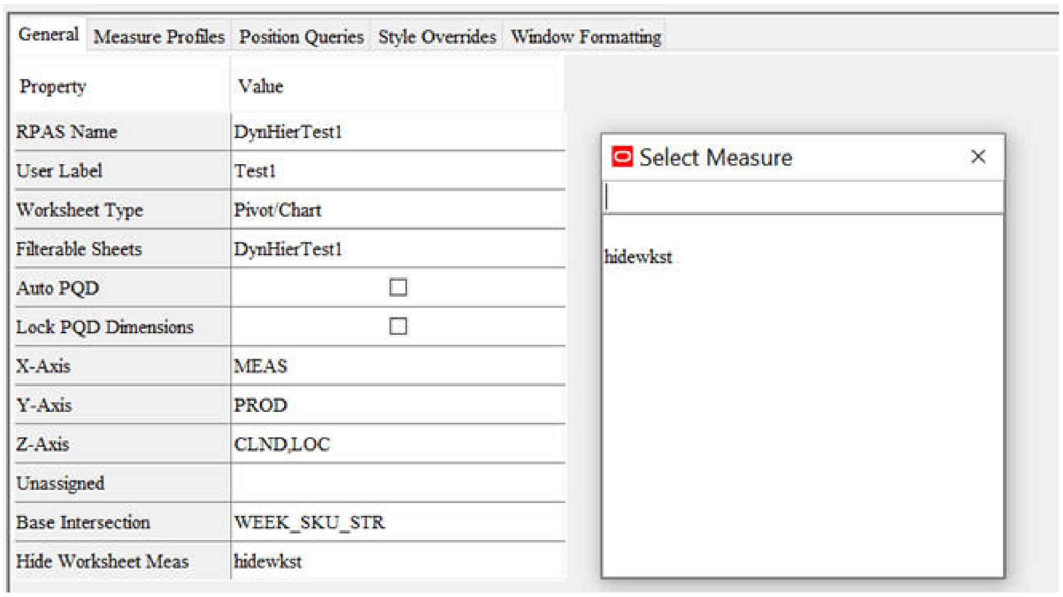
The hierarchies and dimensions that appear in this process are those used by measures placed on the default and viewable profiles for the worksheet. If no measures have yet been placed in those profiles, no hierarchies appear in this process.

7. Select the dimension for each hierarchy. The base intersection of the sheet represents the base set of dimensions at which the window could be displayed. Data on the window can be viewed at any dimension/intersection higher than this. If the base intersection of a measure on the sheet is underneath the base intersection of the sheet, the measure's values are shown aggregated to the displayed intersection. If measure's base intersection is higher than the sheet intersection, the measures values are hashed out at all intersections lower than the base intersection of the measure.
8. Define the Hide Worksheet Meas property. This property is optional and can only be configured within the Grid/Pivot worksheets. Once the Hide Worksheet Measure is configured and has a value of TRUE, the worksheet is hidden within the RPASCE UI.

   When the user clicks the Hide Worksheet Meas field, a dialog box is displayed where the user can select a measure from the pre-populated candidate measure list. The candidate measures are Boolean, Scalar measures that are defined within this workbook rule set. Whether or not this measure has storage database is not required.

   This measure’s value must be computed at the workbook build time and remain static. The Hide Worksheet Measure can be within the workbook load rules and calculated then.

Figure 4-161 Select a Hide Worksheet Measure



Property	Value
RPAS Name	DynHierTest1
User Label	Test1
Worksheet Type	Pivot/Chart
Filterable Sheets	DynHierTest1
Auto PQD	☐
Lock PQD Dimensions	☐
X-Axis	MEAS
Y-Axis	PROD
Z-Axis	CLND,LOC
Unassigned	
Base Intersection	WEEK_SKU_STR
Hide Worksheet Meas	hidewkst

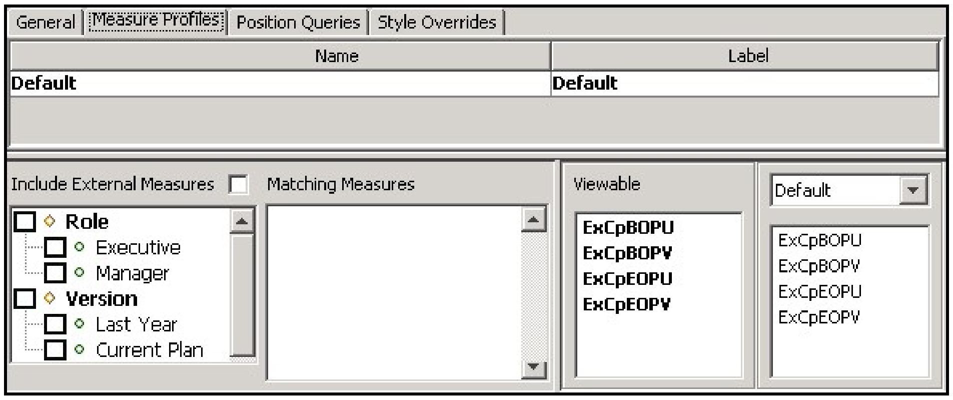
9. Click **OK**.

Measure Profiles Tab

Complete the following steps to update the Measure Profiles tab.

Usage of this tab is optional. This tab allows users to configure additional measure profiles beyond the Default profile for the worksheet. These additional profiles may be populated with measures in the same manner as the default profile. When the workbook is built, these additional profiles will be available through the RPASCE Client to provide users a different set of measures or a different measure ordering for the measures of the worksheet.

Figure 4-162   Measure Profiles Tab



Name	Label
Default	Default

Viewable: ExCpBOPU, ExCpBOPV, ExCpEOPU, ExCpEOPV

Default: ExCpBOPU, ExCpBOPV, ExCpEOPU, ExCpEOPV

Adding a Measure Profile

Complete the following steps to add a measure profile.

1. Click the **Measure Profiles** tab.
2. Right-click in the table area and select **Add**.
3. Enter the following information:
    - **Name** – The name for the measure profile used by RPASCE. This name must be unique across the worksheet.
    - **Label** – The label for the measure profile that is used internally by RPASCE. This label must be unique across the worksheet.

Note:

Duplicate label names are not allowed within the scope of the worksheet.

Copying a Measure Profile

Complete the following steps to copy a measure profile.

1. Click the **Measure Profiles** tab.
2. Select an existing measure profile.
3. Right-click in the table area and select **Copy**.

4. A new measure profile will be created that contains the same measure membership an ordering as the selected profile. This new measure profile must be saved under a new profile name.

Marking a Measure Profile as Default

Complete the following steps to mark a measure profile as default.

1. Click the **Measure Profiles** tab.
2. Select an existing measure profile that is not the default (the first profile in the table is always the default).
3. Right-click in the table area and select **Make Default**.
4. The selected measure profile becomes the default profile for the worksheet and moves to the first row in the measure profile table.

Removing a Measure Profile

Complete the following steps to remove a measure profile.

1. Click the **Measure Profiles** tab.
2. Selected an existing measure profile that is not the default.
3. Right-click in the table area and select **Remove**.
4. The selected measure profile is removed from the list of profiles for the worksheet.

Position Queries Tab

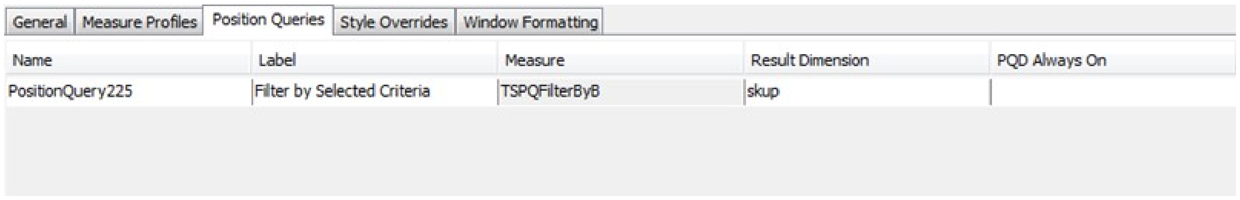
Usage of this tab is optional. This tab allows you to specify a worksheet where the positions that are shown in a query dimension are based on the current position in driving dimensions. The driving dimensions can be absent, such as in RDF applications. If the driving dimensions exist, they must be in the slice area. The process uses a position query that is a Boolean measure dimensioned on the query dimension and the driving dimensions. Only positions in the query dimension that have the value TRUE for the position query measure for the positions in the driving dimensions are shown in the worksheet. All other positions are automatically hidden.

When more than one driving dimensions are present, all the driving dimensions have to be in Z-axis for the position query to execute. If one or more driving dimensions are taken out of the Z-axis and placed in X or Y axes, associated position queries will not be executed. A given window can have more than one position query, driven by one or more dimensions in the Z-axis and driving different dimensions in the X, Y, and Z axes.

Note:

Position Queries are referred to as Special Filters in the *Oracle Retail Predictive Application Server Cloud Edition User Guide* and several other documents. Both terms refer to the same functionality.

Figure 4-163   Position Queries Tab for a Worksheet


General | Measure Profiles | Position Queries | Style Overrides | Window Formatting

Name	Label	Measure	Result Dimension	PQD Always On
PositionQuery225	Filter by Selected Criteria	TSPQFilterByB	skup	

Complete the following steps to update a Position Queries tab.

1. Click the **Position Queries** tab.
2. Right-click in the table area and select **Add**.
3. Enter the following information:
   - Name – The name for the position query used by RPASCE. This name must be unique across the project.
   - Label – The label for the position query that is used internally by RPASCE. This label must be unique across the project.

     Note:

     Duplicate label names are not allowed.

   - Measure – This defines the position query measure. This must be a Boolean type measure. Click in the field, and then click **Select Measure** to view a list of the Boolean measures used in the workbook. Type the first few characters of the measure name in the box at the top of the list to go to the required measure or scroll to find it. Double-click to select the desired measure.
   - Result Dimension - This defines the query dimension. Select a dimension from the list of dimensions for the selected measure.

     Note:

     While configuring position queries, it is important that the Boolean mask measure that drives the position queries be referenced in the workbook (by either referencing it in the load rule group or by setting it through the calc rule groups). Currently, there is no validation in Configuration Tools that checks for this, and no error is thrown at configuration time/PDS build time or workbook build time.

     RPASCE does not support multiple PQDs with the same resulting dimension active at one time. If this condition exists, these PQDs overwrite each other rather than combine, so the results are not desirable.

   - PQD Always On – When this property is set to true, users cannot disable the PQD through the Special Filters filter list in the RPASCE client. It must be used for queries that are central to the process being performed in a worksheet to the extent that the process cannot be performed without the PQD.

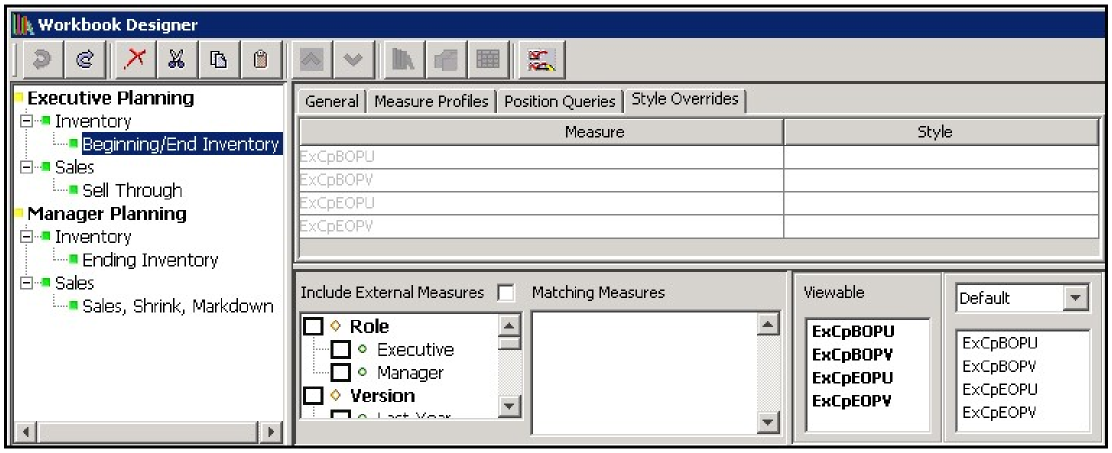
4. To remove a row from the tab, select the row, right-click, and select **Remove**. The row is deleted from the Position Queries tab.

Style Overrides Tab

Usage of this tab is optional. This tab allows you to override the style property for a measure, so that measure uses a different formatting style in the worksheet. The Style Overrides tab can also be used to apply a formatting style to an attribute defined on any of the dimensions used in the worksheet. This formatting style will then be applied to the attribute where it displays within the worksheet.

> **Note:**
>
> The measures that appear in this process are those placed on the profiles of the worksheet. Attributes that appear are those that are defined at or higher than the intersection of the worksheet. If no measures have been placed on profiles and the worksheet has not been configured with an intersection, no entries appear in the Style Overrides tab.

Figure 4-164 Style Overrides Tab for a Pivot/Chart Worksheet



Measure	Style
ExCpBOPU	
ExCpBOPV	
ExCpEOPU	
ExCpEOPV	

Viewable	Default
ExCpBOPU	ExCpBOPU
ExCpBOPV	ExCpBOPV
ExCpEOPU	ExCpEOPU
ExCpEOPV	ExCpEOPV

Complete the following steps to update the Style Overrides tab.

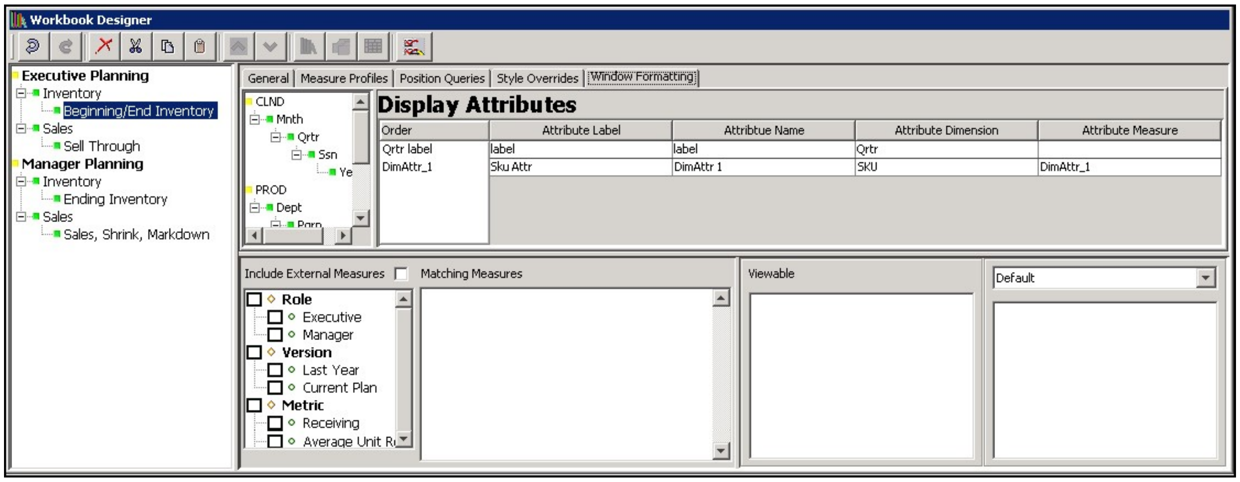
1. Click the **Style Overrides** tab.
2. Select an override formatting style for a measure. Measures whose styles have been overridden display in black. Those whose styles are defaulting display in gray.

Window Formatting Tab

Usage of this tab is optional. This tab allows you to specify the display attributes for the dimensions used within the worksheet. By default, dimensions display the position label within the RPASCE Client. If there are any additional attributes defined for a dimension, the client can display those attributes in addition or in place of position labels. Within the

RPASCE Client, if a media UI Type attribute is added to the set of display attributes, the RPASCE Client will display the appropriate image in the position display area.

Figure 4-165 Window Formatting Tab for Worksheet



Workbook Designer

General | Measure Profiles | Position Queries | Style Overrides | Window Formatting

Display Attributes

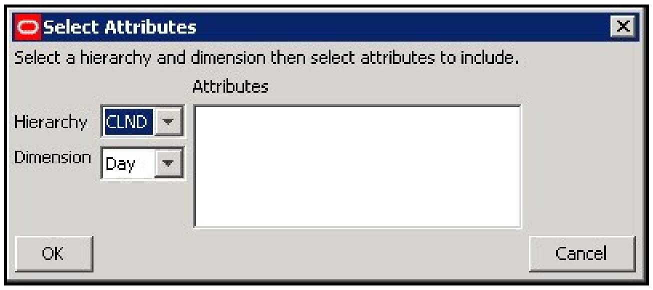
Order	Attribute Label	Attribtue Name	Attribute Dimension	Attribute Measure
Qrtr label	label	label	Qrtr	
DimAttr_1	Sku Attr	DimAttr 1	SKU	DimAttr_1

Add Additional Display Attributes for a Dimension

Complete the following steps to add additional display attributes for a dimension.

1. Click the **Window Formatting** tab.
2. Right-click and select Add from the menu.
3. Within the Figure 4-166, specify the dimension to modify.
4. Select desired attributes from the list of available attributes for the dimension.
5. Click **OK**.

Figure 4-166 Select Attributes Window



Select Attributes

Select a hierarchy and dimension then select attributes to include.

Hierarchy: CLND

Dimension: Day

Modify Order of Existing Attributes

Complete the following steps to modify order of existing attributes.

1. Click the **Window Formatting** tab.

2. Select the dimension to modify from the hierarchy tree.
3. Within the Order column, use drag-and-drop to modify the attribute order.

Remove a Display Attribute

Complete the following steps to remove a display attribute.

1. Click the **Window Formatting** tab.
2. Select the dimension to modify from the hierarchy tree.
3. Select an attribute doe the dimension in the Order column or attribute table.
4. Right-click and select **Remove** from the menu.

> **Note:**
>
> All dimensions have the label attribute set by default and all dimensions must retain at least one display attribute.

Defining Worksheet Properties for Worksheets Tiled View

RPASCE supports a view within the RPASCE Client, called the Tiled View, that supplements the Pivot, Chart and Detail Pop-up views also supported by the RPASCE Client. This view allows for the display of a series of tiles. Each tile represents a position of the Tile Axis hierarchy, shown at the intersection formed by row and page axis positions. Each tile is displayed as a single visual block that contains the visible attributes for the tile position, and cells for visible dimensional measures, if any, at that tile's intersection.

For example, configuring the Product hierarchy to be the Tile Axis hierarchy will result in each position along the Product hierarchy being represented by a tile. This tile can display attribute values for the Product position. The tile may also display Measure values for the intersection of this Product position with whatever positions are on the row and page axis.

The Tiled View may be configured so that a tile is only displayed for an intersection based on the value of a Boolean tile measure. Drag and Drop and Remove operations can be configured to manipulate this Boolean value accordingly.

Tiled View Worksheet

The Workbook Designer of the RPASCE Configuration Tools allows the creation and modification of Tiled View worksheets. Users of the Configuration Tools are able, by using the Worksheet Type attribute located within the General Properties tab of a worksheet, to set a worksheet from the default of Pivot/Chart worksheet to the new Tile worksheet.

Once a worksheet has been set to a Tile Worksheet, the Workbook Tool user interface updates to present the user of the Configuration Tools with the options available for Tiled View worksheets in the same manner that it changes when a worksheet is configured to be a Detail Pop-up worksheet.

Properties of the Tiled View Worksheet

Tiled View Worksheets support a set of properties in the same manner as Pivot/Chart worksheets and Detail Pop-up worksheets. The configuration of these properties is managed through the General Properties tab of the Workbook Designer through the same table interface used to configure properties for existing worksheet types.

From a configuration standpoint, Tiled View worksheets are very similar to Pivot/Chart worksheets. The two views share many of the same sets of properties and optional configuration elements (implemented through the various tabs available within the Workbook Designer when a worksheet is selected.) However, some Pivot/Chart specific properties may not be configurable for Tiled View worksheets (as mentioned later). In addition, Tiled View worksheets support additional, Tiled View specific properties.

The set of configurable properties for the Tiled View worksheet and summary information on each property can be found in this section.

RPAS Name

The RPAS Name attribute is the internal name used by RPASCE to identify the worksheet. The value of this property must be a valid RPASCE name and is validated according to the rules for RPASCE names.

User Label

The User Label attribute is an external name used to identify the worksheet to the user of the RPASCE Client. The value of this property is displayed within the RPASCE Client for single language applications and is the resource exported for translation in multi-language applications.

Worksheet Type

The worksheet type attribute must be set to Tiled View in order to configure a Tiled View worksheet.

Filterable Sheets

As with Pivot Table Views, Tiled View worksheets support the ability to filter the visible positions of a worksheet based upon context supplied by the worksheet. The Tiled View supports only the existing show/hide position-based filtering; the application does not allow Tile Filtering based upon the Tile Measure to other views.

Tiled Axis, Row Axis, Page Axis and Unassigned Axis

The Tiled View worksheet contains an axis layout that is similar but not identical to the X-, Y- and Z-Axis layout schema used by Pivot worksheets. Like traditional axis layouts, the Tiled View axis layout requires the hierarchies present in the base intersection of the worksheet to be assigned to one and only one axis.

The Tiled View axis layout differs in the names and meanings of the axes used. Tiled View axis layout consists of:

- **Tile Axis** – This Axis supports only one hierarchy and is the axis that corresponds to tiles; each visible position in the Tile Axis is represented by a single tile in the view.
- **Row Axis** – The Row Axis may contain a single hierarchy. When configured, this hierarchy is used to create groupings of tiles where multiple blocks of tiles representing the Tile Axis hierarchy are organized according to the positions of the Row Axis hierarchy. Use of the Row Axis is optional; when no Row Axis hierarchy is configured there will be only a single block of tiles as determined by the Tile Axis hierarchy.
- **Page Axis** – All hierarchies present in the base intersection of the worksheet that are not assigned to either the Tile Axis or Row Axis must be assigned to the Page Axis. This axis is functionally equivalent to the Page Axis, or Z-Axis, of Pivot Table views. Hierarchies assigned to the Page Axis are navigable using a set of controls that page the view as the selected position is modified by the user.
- **Unassigned Axis** – As with Pivot Views, there are no valid cases in which hierarchies may be assigned to the Unassigned Axis. This attribute provides information to the user of the Configuration Tools that one or more hierarchies has not yet been assigned to a valid axis and details which hierarchies require axis assignment.

Base Intersection

As with Pivot Views, a Tiled View worksheet has a base intersection. Unlike Pivot View worksheets, which allow a number of valid intersections for any given view, Tiled View worksheets require the base intersection of the worksheet to be identical to the base intersection of the Tile Measure, if one is defined. Tiled View worksheets that do not make use of a Tile Measure support the configuration of a base intersection based upon the dimensional measures present in the view profiles.

Tile Measure

The Tile Measure is a measure used by the Tiled View worksheet to manage several behaviors associated with the view and its support of filtering and drag and drop operations within the RPASCE Client. Prior to using the Tiled View worksheet in conjunction with these functionalities, the Tile Measure must be configured with the name of the Boolean measure that will be used.

Tile Measure is an optional property. A Tiled View can be configured that does not specify a Tile Measure. However, Tiled View worksheets that do not specify a Tile Measure do not support the following behaviors:

- Filtering of tiles based upon Tile Measure cell values
- Drag and Drop operations
- Formatting the Background Color of the Entire Tile

When a tile measure is specified, the cell contents of the measure are used to drive tile filtering. For any given page/row slice, the tile axis hierarchy positions whose cell values are equal to the Tile Value is displayed and those whose cell values are different from the Tile Value are filtered. Drag and drop operations, if they are enabled, modify the cell values of the tile measure with the resulting change to the visible set of positions at the time of the operations (and not as a result of a subsequent calculation cycle.)

Note:

The background color of the entire tile is taken from the formatting of the Tile Measure, if present. This can be set via Measure/Cell styles, Boolean Exception Formatting, or by being the target measure for a Real Time Alert.

Tile Value

When used in conjunction with the RPASCE Client drag and drop functionality, the Tile Value attribute contains the value to be set into the Tile Measure as a part of a drop operation. It must be either true or false in order to be contained by the Boolean Tile Measure

Filter Tiles

This attribute specifies whether the values of the Tile Measure are used to drive filtering of visible intersections within the Tiled View worksheet. If checked, only intersections for which the Tile Measure's value for that tile is equal to the configured Tile Value will be visible.

Enable Drop

When Drop is enabled, then the worksheet can serve as a drop location within the constraints set by the Drag Source Hierarchies attribute.

Drag Source Hierarchies

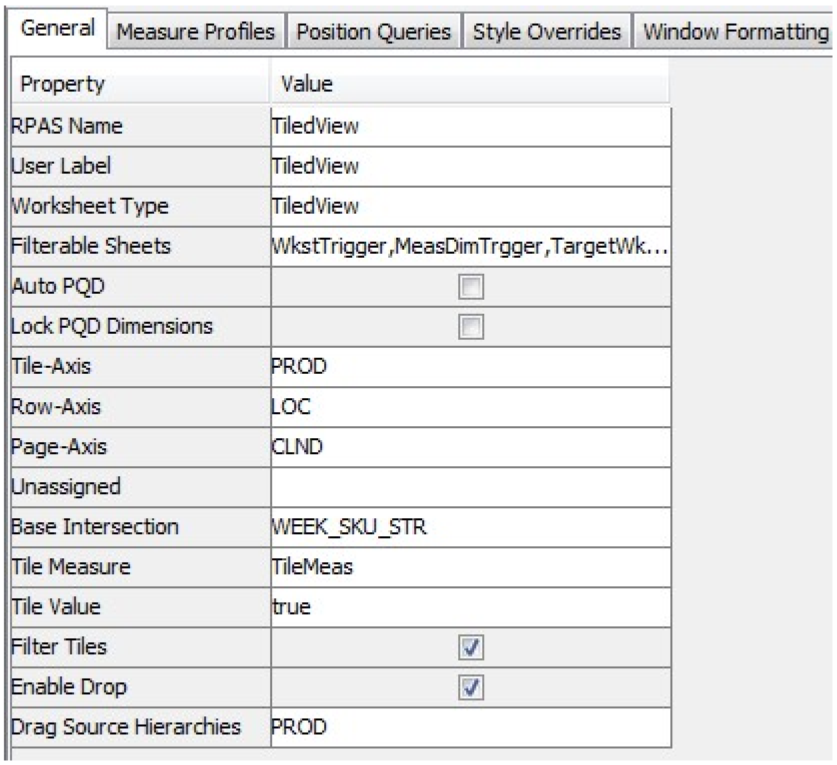
This attribute is used in conjunction with the drag and drop functionality enabled by configuring a Tile Measure, Tile Value, and checking the Enable Drop attribute. When drag and drop is enabled, a drag and drop action transmits the selection from the dragged tile to the Tiled View worksheet upon which is the drop target.

Those hierarchies which are configured through drag source hierarchies have their selection context transmitted, those hierarchies which are not configured as drag source hierarchies instead use the position context of the page and/or row upon which the tile is dropped. Drag Source Hierarchies must include the Tile Axis hierarchy; it can optionally include any other hierarchy in the base intersection of the Tile Measure.

For example, assume a drag and drop action between two Tiled View windows configured along CLND, PROD and LOC. In both windows, PROD is the Tile Axis hierarchy, LOC is the Row Axis hierarchy and CLND is the Page Axis hierarchy. Assume also that the drag source hierarchies attribute of the target view is PROD.

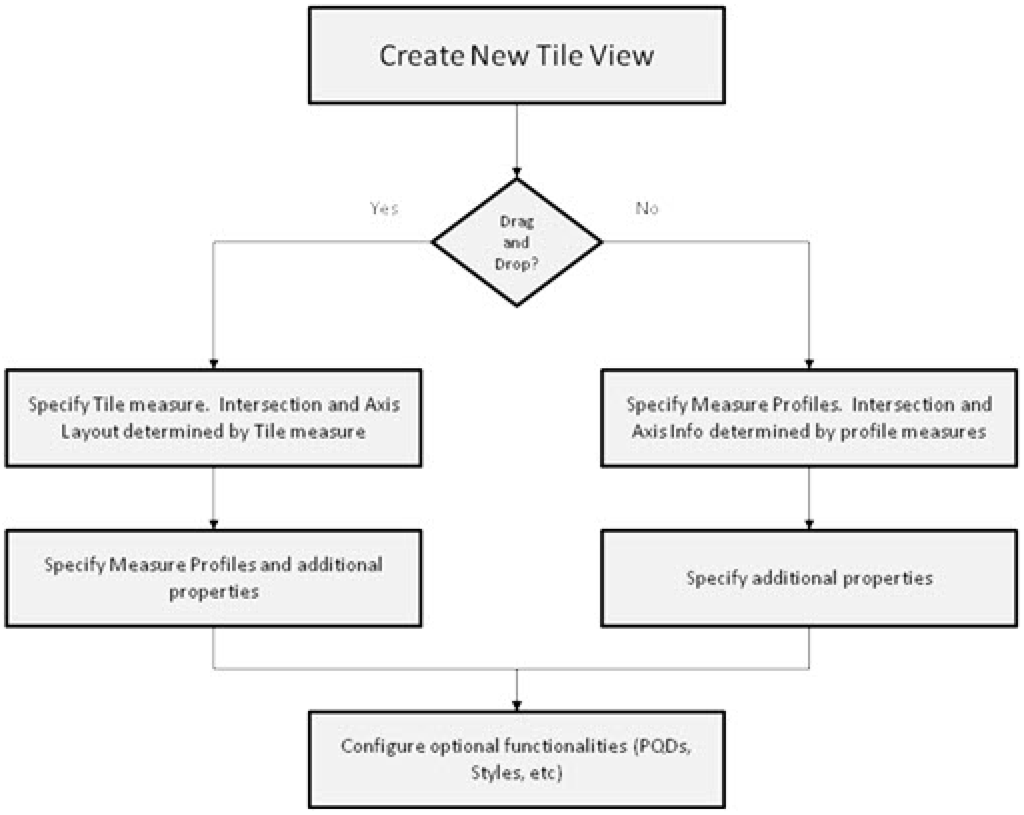
When processing the drag and drop action, the position along PROD corresponding to the tile that was dragged is combined with the LOC and CLND positions based upon the row and page where the tile is dropped to determine the relevant intersection. If, instead, drag source hierarchies had been configured to be PROD and LOC, then the PROD and LOC positions would be those of the dragged tile and only the CLND position would be determined by the page upon which the tile was dropped.

Figure 4-167 General Properties Tab of a Tiled View Worksheet



Property	Value
RPAS Name	TiledView
User Label	TiledView
Worksheet Type	TiledView
Filterable Sheets	WkstTrigger,MeasDimTrgger,TargetWk...
Auto PQD	☐
Lock PQD Dimensions	☐
Tile-Axis	PROD
Row-Axis	LOC
Page-Axis	CLND
Unassigned	
Base Intersection	WEEK_SKU_STR
Tile Measure	TileMeas
Tile Value	true
Filter Tiles	✓
Enable Drop	✓
Drag Source Hierarchies	PROD

Figure 4-168 Process Flow for Configuring a Tiled View Worksheet


Create New Tile View
Yes
Drag and Drop?
No
Specify Tile measure. Intersection and Axis Layout determined by Tile measure
Specify Measure Profiles. Intersection and Axis Info determined by profile measures
Specify Measure Profiles and additional properties
Specify additional properties
Configure optional functionalities (PQDs, Styles, etc)

Additional Tabs of the Tiled View Worksheet

Tiled View worksheets support the configuration of additional options through tabs present in the Workbook Designer when a Tiled View worksheet is loaded into the User Interface. The additional tabs supported when working with a Tiled View worksheet are:

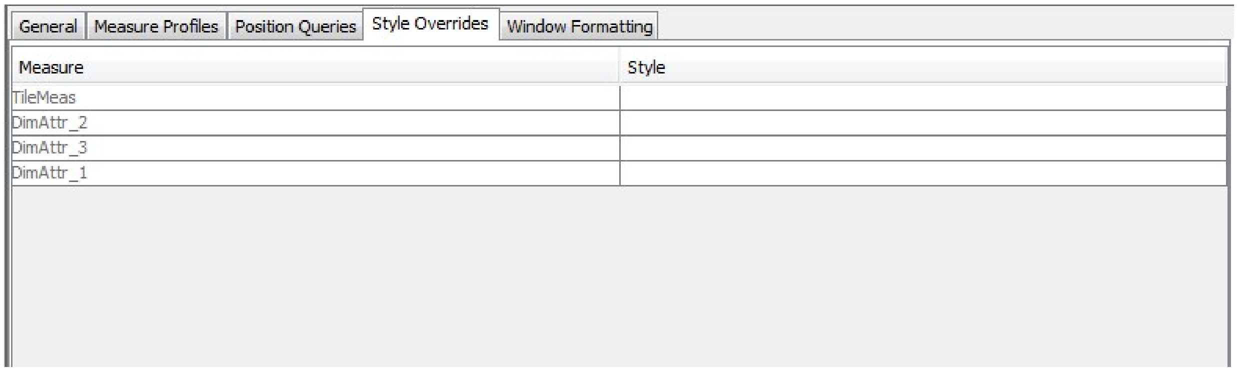
- Style Overrides
- Measure Profiles
- Window Formatting
- Position Queries

Style Overrides

Tiled View worksheets support the ability to configure styles for measures displayed within the view. The Style Override tab will therefore be available when configuring a Tiled View worksheet and it functions in the same manner as in Pivot and Detail Pop-up Views.

Additionally, it is possible to configure a set of formatting properties that are applied to the tile itself in a Tiled View worksheet. The primary property supporting configuration is the color of the tile. In order to allow the configuration of tile format properties, the Style Override panel will also include an entry for the Tile Measure. Setting a style for the Tile Measure within the Style Override panel will cause the set style to be used for tile formatting and not the base configured style of the measure, if a style has been configured for the measure.

Figure 4-169 Style Override Panel for a Tiled View Worksheet Showing a Tile Measure



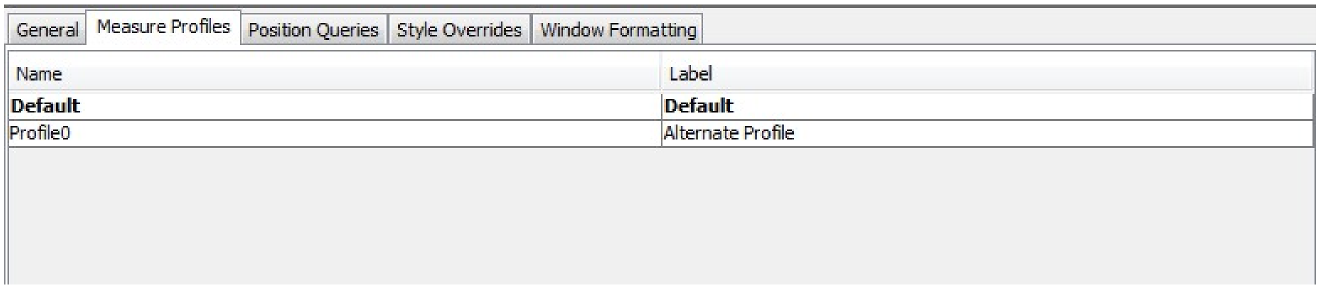
Measure	Style
TileMeas	
DimAttr_2	
DimAttr_3	
DimAttr_1	

Measure Profiles

Tiled View worksheets support the ability to configure multiple user profiles for the measure present in the view. The configuration of these profiles and the process of defining the measures present in each will follow the behavior of measure profile configuration for Pivot View worksheets with the exception that it will be considered valid for a measure profile to contain no measures.

This is intended to support a case in which the tiles of the Tiled View contain only dimension attributes configured through Window Formatting. One scenario is a tile that contains only a single media measure attribute to display an image in the tile or potentially an image and the position label for the tile.

Figure 4-170 Measure Profiles Panel for the Tiled View Worksheet

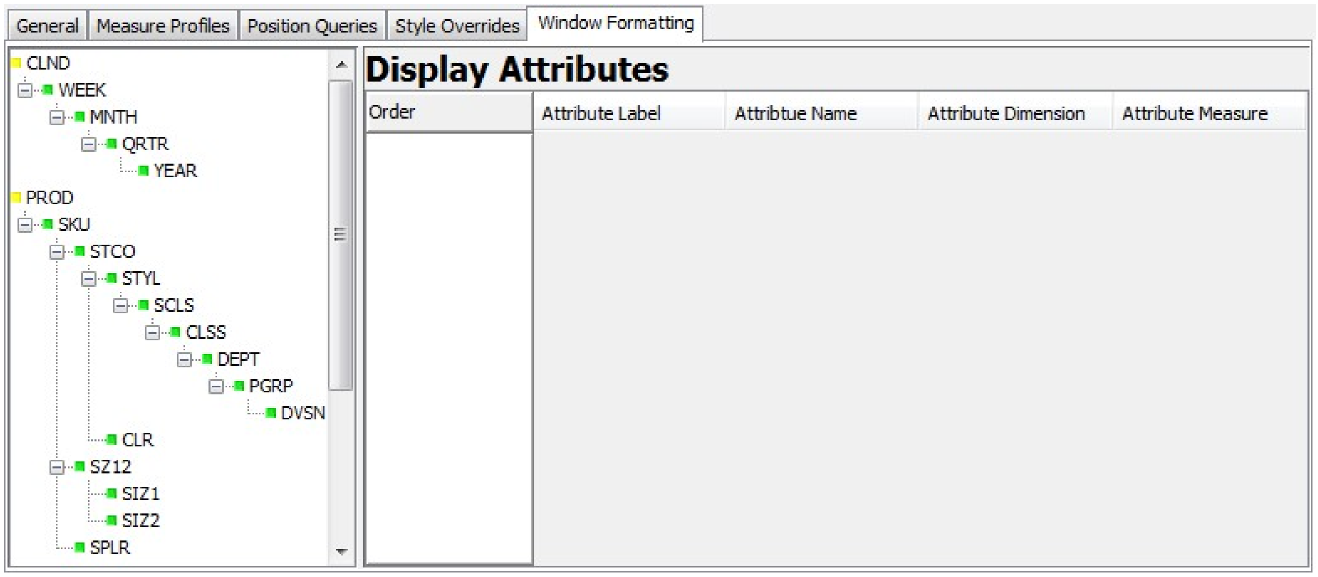


Name	Label
Default	**Default**
Profile0	Alternate Profile

Window Formatting

Like Pivot Views, Tiled Views support worksheet style configuration through a Window Formatting tab. Here you can define which display attributes you would like to be visible for each dimension of the worksheet. Attributes defined for dimensions of the Tile Axis hierarchy appears within each tile in addition to any dimensional measures defined for the worksheet's profiles. If label is the first visible attribute it appears in the tile's header.

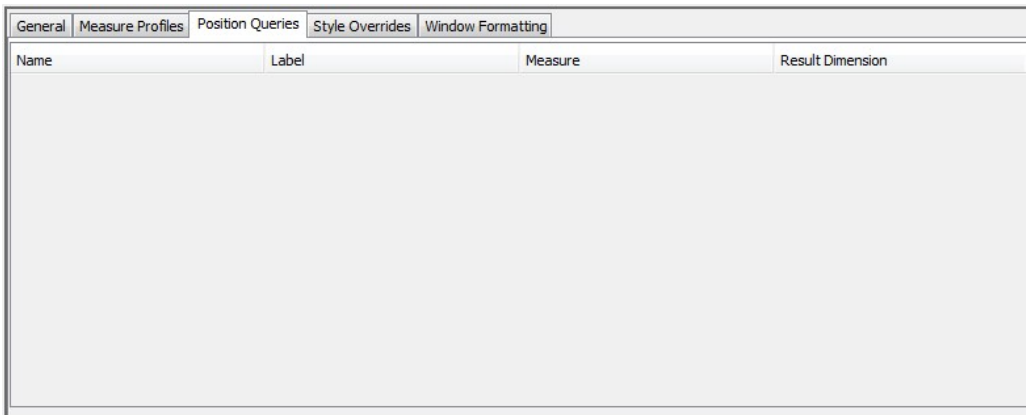
Figure 4-171 Window Formatting Panel for a Tiled View Worksheet



Position Queries

Tiled View does support the use of Position Queries. So, the Position Queries panel of the Tiled View worksheet functions in the same manner as Pivot View worksheets.

Figure 4-172 Position Queries Panel for a Tiled View Worksheet



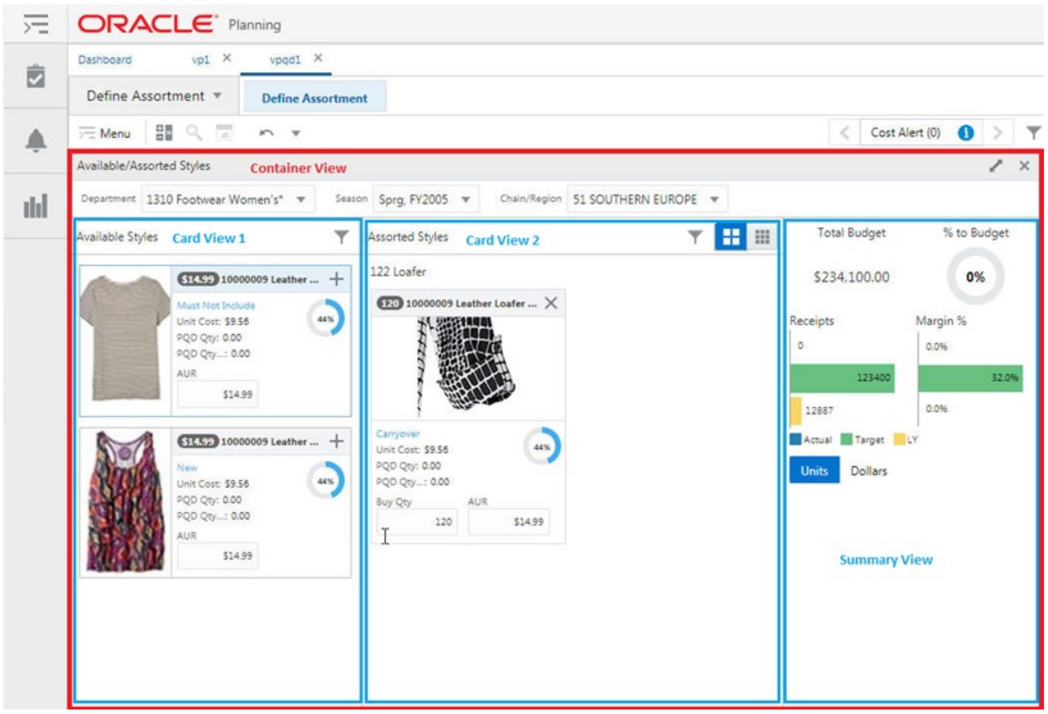
Defining Worksheet Properties for the Worksheets Container View

RPASCE supports a Visual Planning capability within the RPASCE Client. Three views are introduced to support the Visual Planning capability.

- In the RPASCE client, the **Container View** type groups the sub-views, two Card views and one Summary View, together within a given layout, sharing a top filter that acts as a page edge. This is the view that displays in the View Manager. The sub-views specified here generally do not display directly in the view manager. The sub-views are expected to conform to the requirements of the container.
- The **Card View** is like the Tiled View in Fusion Client but allows one or more platform-supported card types (list, small, large, image w/popup) to be selected and configured to specify how they are populated. This can be configured to support drag/drop and add/remove functionality.
- The **Summary View** is a summary panel to display some key metrics and a breakdown. Like the Card View, this has platform-supported summary types (only one so far), each paired with a summary configured that specifies how this is populated. The metrics here are displayed at the intersection defined by the top filter, except for the breakdown part, where the attribute choice changes one of the dimensions to the configured level and shows a breakdown of those positions.

Here is how this relates to an Assortment example:

Figure 4-173 Container View Worksheet



The Workbook Designer of the RPASCE Configuration Tools can be used in the creation and modification of the Container View worksheets. Users of the Configuration Tools are able, by using the Worksheet Type attribute located within the General Properties tab of a worksheet, to set a worksheet from the default of the Pivot/Chart worksheet to the Container View worksheet.

Once a worksheet has been defined as a Container View Worksheet, the Workbook Tool user interface updates to present the user of the Configuration Tools with the options available for Container View worksheets in the same manner that it changes when a worksheet is configured to be a Detail Pop-up worksheet.

Features of the Container View Worksheet

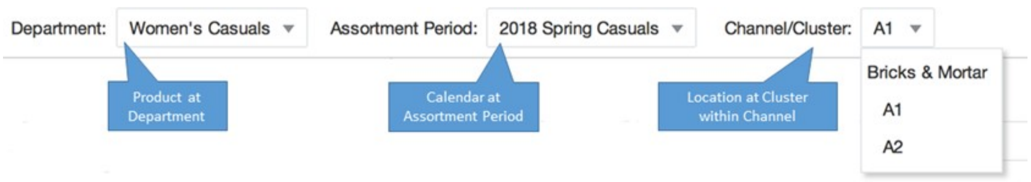
As mentioned in the overview, the Container View groups other views for a common purpose and a specific layout. The Container View worksheet defines the common Top Filter Dimensions, sub-views specified by Worksheets, a card hierarchy, and a Layout Type that specifies how these are arranged.

Top Filter Dimensions

The Top Filter Dimensions is used to specify the top filter dimensions separated by comma. For each hierarchy in the container worksheet intersection, one or more dimensions can be configured for the filter. In the following example, the top filters are Department, Assortment Period, Channel, and Cluster. The two dimensions Channel and Cluster belong to LOC hierarchy and are part of Top Filter.

The top filter acts as a kind of shared page edge. In the RPASCE client, the user chooses a position on each drop-down, and each sub-view is constrained by the selection. The filter sets up an intersection (in this example Women's Casuals/2018 Spring Casuals/A1) that is used to restrict the sub-views. The user's choice here can change the levels used by the sub-views. For example, changing from A1 to Bricks & Mortar changes the intersection from Department/Assortment Period/Cluster to Department/Assortment Period/Channel. Changing the dimension position in the Card Hierarchy displays only those cards that roll up to that dimension position in RPASCE client.

Figure 4-174 Define the Axis Layout of the Worksheet


Department: Women's Casuals
Assortment Period: 2018 Spring Casuals
Channel/Cluster: A1
Bricks & Mortar
A1
A2
Product at Department
Calendar at Assortment Period
Location at Cluster within Channel

Define the Axis Layout of the Worksheet

The Axis layout in the Container View is set automatically in the backend, based on configuration for Intersection and Card Hierarchy.

The Container View worksheet contains an axis layout that is similar but not identical to the X-, Y-, and Z-Axis layout schema used by the Pivot worksheets. Like traditional axis layouts, the Container View axis layout requires the hierarchies present in the base intersection of the worksheet to be assigned to one and only one axis.

- **X-Axis** –This axis is not assigned in the Container View.
- **Y-Axis** –This axis is assigned to the Card hierarchy only and is the axis that corresponds to cards.
- **Z-Axis** –All hierarchies present in the base intersection of the worksheet that are not assigned to Y-Axis are assigned to the Z-Axis. This axis is like the Page Axis, or Z-Axis, of the Pivot Table views. Hierarchies assigned to the Page Axis are navigable using a set of controls that page the view as the selected position is modified by the user.
- **Unassigned Axis** –As with Pivot Views, there are no valid cases in which hierarchies may be assigned to the Unassigned Axis. This attribute provides information to the user of the Configuration Tools that one or more hierarchies has not yet been assigned to a valid axis and details which hierarchies require axis assignment.

Support for Images, Attributes, and Measures

The Container View can use its sub-views such as Card View to provide a representation of information in which images, attributes, and dimensional measures may all be displayed in a single informational unit. For details, see the Card View configuration section.

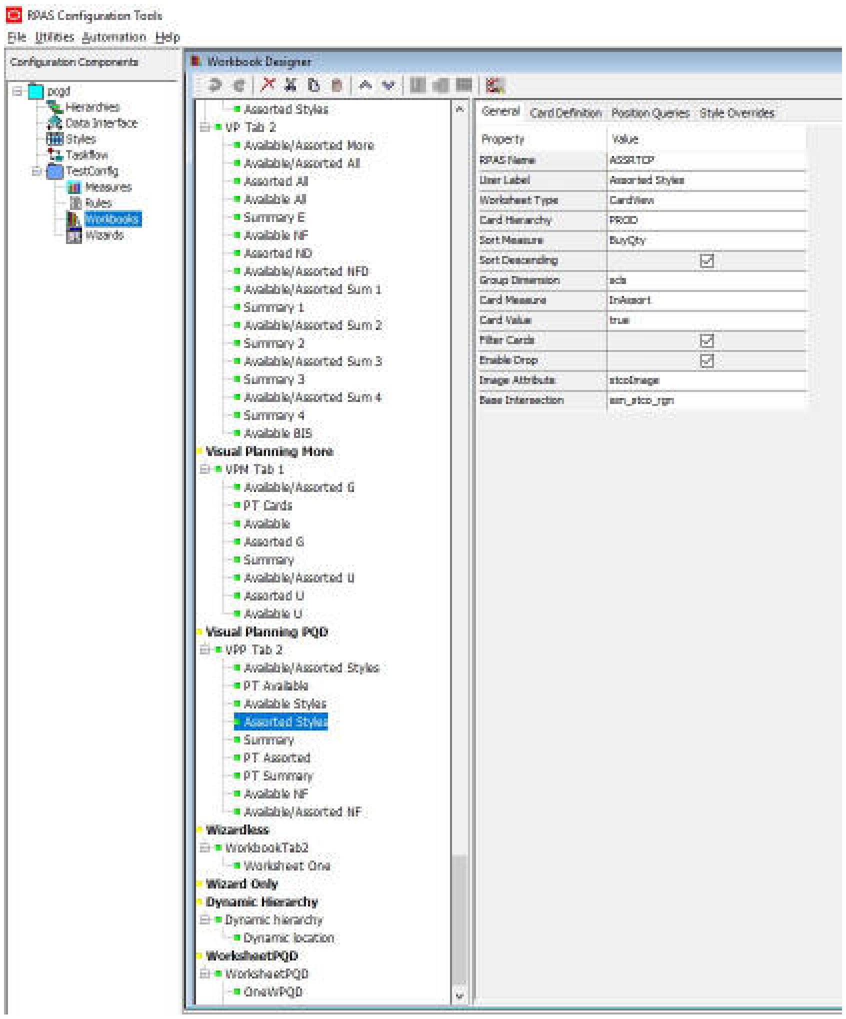
Alert Navigation

Neither the Container View nor any of the sub-views may be configured as an Alert Worksheet for Real Time Alerts.

Support for Measure Profiles

Unlike the Pivot worksheet, the Container View configuration itself does not allow users to select measures included in Measure Profiles since the Container View is basically a grouping of sub-views. The sub-views have populated Measure Profiles based on the measure specification, but in the Container View the Viewable Measure Profile, Default Measure Profile, and Selectable Measure Profile are all empty and cannot be changed by user configuration.

Figure 4-175   Workbook Designer Showing a Container View Worksheet



Property	Value
RPAS Name	ASSRTCP
User Label	Assorted Styles
Worksheet Type	CardView
Card Hierarchy	PROD
Sort Measure	BuyQty
Sort Descending	☑
Group Dimension	scls
Card Measure	InAssort
Card Value	true
Filter Cards	☑
Enable Drop	☑
Image Attribute	stcoImage
Base Intersection	ssn_stco_rgn

Properties of the Container View Worksheet

Container View Worksheets support a set of properties in the same manner as Pivot/Chart worksheets and Detail Pop-up worksheets. The configuration of these properties is managed through the General Properties tab of the Workbook Designer through a similar table interface used to configure properties for existing worksheet types.

From a configuration standpoint, Container View worksheets are similar to Pivot/Chart worksheets, except for some Pivot/Chart specific properties and optional configuration elements that do not apply to Container View.

The set of configurable properties for the Container View worksheet and summary information on each property can be found in this section.

RPAS Name

The RPAS Name attribute is the internal name used by RPASCE to identify the worksheet. The value of this property must be a valid RPASCE name and is validated according to the rules for RPASCE names.

User Label

The User Label attribute is an external name used to identify the worksheet to the user of the RPASCE Client. The value of this property is displayed within the RPASCE Client for single language applications and is the resource exported for translation in multi-language applications.

Worksheet Type

The worksheet type attribute must be set to Container View in order to configure a Container View worksheet.

Layout Type

The Layout Type determines the visual layout of the container and its sub-views, and potentially the requirements and relationship of the sub-views. Currently, only the DragDropSum Layout Type is supported. This assumes that three views are supplied and are assigned to the layout in from, to, and summary roles.

Worksheets

The Worksheets configuration lists the worksheets in the order in which they are assigned to the position/role of the Layout Type.

Note that the string entered in the Worksheets row must represent the existing worksheets defined in the current workbook, and in the order of Card View, Card View, and Summary View, with the worksheet names separated by a comma.

For example, in Figure 4-175, the string `AVAILCP, ASSRTCP, ASUMP` is entered for Worksheets, with the worksheets AVAILCP, ASSRTCP, and ASUMP assigned to the from, to, and summary role, respectively. The worksheets AVAILCP and ASSRTCP are Card Views, and ASUMP is a Summary View that has already been defined in the same workbook. If these requirements are not meet, the Worksheet row will be highlighted in red and the validation error will be displayed in the Task List panel.

Intersection

As with Pivot Views, a Container View worksheet has a base intersection. Unlike Pivot View worksheets, which allow some valid intersections for any given view, a Container

View requires the base intersection of the worksheet to be identical to the base intersection of its sub-views, especially the Card View worksheets.

Top Filter Dimensions

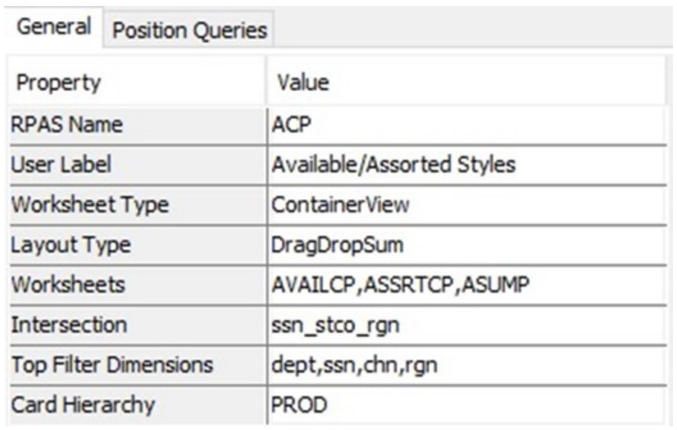
The Top Filter Dimensions is used to specify the top filter dimensions separated by comma.

> **Note:**
>
> No multiple dimensions along the Card Hierarchy can be specified in the Top Filter Dimensions.

Card Hierarchy

The Card Hierarchy is like the Tile Axis Hierarchy in the Tiled View. The Container View and its sub-views must have an identical Card Hierarchy. In the Container View and its sub-views, the Card Hierarchy is always on the Y-Axis, while the other hierarchies within the worksheet Intersection are on the Z-Axis.

Figure 4-176 General Properties Tab of a Container View Worksheet



Property	Value
RPAS Name	ACP
User Label	Available/Assorted Styles
Worksheet Type	ContainerView
Layout Type	DragDropSum
Worksheets	AVAILCP,ASSRTCP,ASUMP
Intersection	ssn_stco_rgn
Top Filter Dimensions	dept,ssn,chn,rgn
Card Hierarchy	PROD

Additional Tabs of the Container View Worksheet

The Container View worksheets support the configuration of additional options through tabs present in the Workbook Designer when a Container View worksheet is loaded into the User Interface. The additional tab supported when working with a Container View worksheet is Position Queries.

Position Queries

The Container View supports the use of Position Queries (PQD). The Position Queries panel of the Container View worksheet behaves in a similar way to the Pivot View worksheets.

The Container View and its sub-views must have the same Position Queries, as they share the Top Filter and must operate seamlessly in concert. Only the Container View shows as Special Filters/Applies to choices, but the grouping is treated as a single entity.

Since all hierarchies of all these views are on the Z-Axis except for the Card Hierarchy and Measures, only the Card dimension effectively serves as the Result dimension of a PQD, driven by choices made in the Top Filter for Z-Axis dimensions.

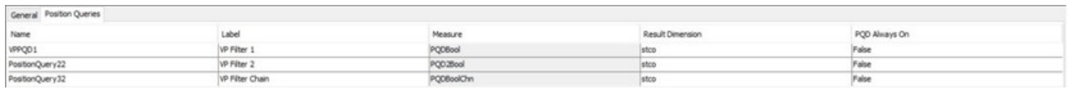
Top Filter choices use the PQDs that are applicable and enabled to show/hide positions of the Card Dimension. This can:

- Hide some cards in the Card Views.
- Restrict the choices available for the Card Dimension in the Top Filter.

Note that, as in all of RPASCE, hiding positions does not exclude them from contributing to rollups. This requires special attention to how the Summary View's measures and rules are designed. While measures can be designed to use the Boolean Card Measure to roll up only values that have been added to the assortment, it will not care whether they are hidden.

It is possible that the PQD will filter the choices available for the Card Dimension in the Top Filter. For example, a PQD could filter out a whole department either directly or by filtering out all its Style/Colors.

Figure 4-177 Position Queries Panel for a Container View Worksheet



Name	Label	Measure	Result Dimension	PQD Always On
VPPQD1	VP Filter 1	PQDBool	stco	False
PositionQuery22	VP Filter 2	PQD2Bool	stco	False
PositionQuery32	VP Filter Chain	PQDBoolChn	stco	False

Defining Worksheet Properties for Worksheets Card View

RPASCE supports the Card View, which is like the Tiled View used in the Fusion Client, with some key differences:

The Card View is used in conjunction with the Top Filter of a Container View. It has no rows or columns or page edge in the UI, only cards. The Top Filter provides the intersection used for the cards, except for the Card Dimension. For the Card Dimension, the Top Filter provides a position at a parent level, and the child positions at the base level of the Card View are used for each card.

RPASCE provides a small number of Card Types that can be used to display the cards. The Card View configuration specifies which Card Type to use and provides the attributes and measures to populate that Card Type. When more than one Card Type is configured, the user can switch between them at will.

Card View Worksheet

The Workbook Designer of the RPASCE Configuration Tools is used in the creation and modification of Card View worksheets. Users of the Configuration Tools are able, by using the Worksheet Type attribute located within the General Properties tab of a worksheet, to set a worksheet from the default of the Pivot/Chart worksheet to the Card worksheet.

Once a worksheet has been set to a Card Worksheet, the Workbook Tool user interface updates to present the user of the Configuration Tools with the options available for Card View worksheets in the same way that it changes when a worksheet is configured to be a Pivot worksheet.

Features of the Card View Worksheet

As mentioned in the overview, the Card View provides an enhanced method for displaying information in comparison to the Tiled View. Additionally, Card Views support other behaviors not found in other view types.

Card Types and Card Definitions

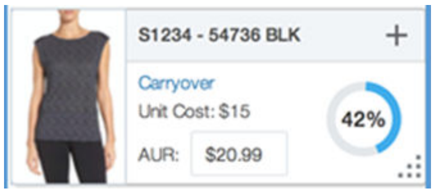
RPASCE supports List, Small, Large, and Image w/Popup card types (the Image w/Popup is to be released in a later version of RPAS CE). The configuration for a Card view enables one or more of these Card Types and provides the associated Card Configuration for each type in the Card Definition optional Tab.

The following sections show examples of each card type.

List Card Type

Figure 4-178 shows a List Card type example. This type has no icon.

Figure 4-178 List Card Example



Small Card Type

Figure 4-179 shows a Small Card type example. Figure 4-180 shows its icon.

Figure 4-179 Small Card Example

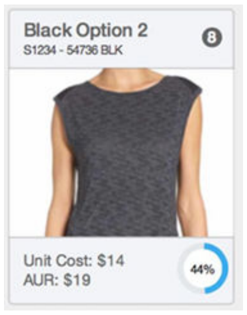
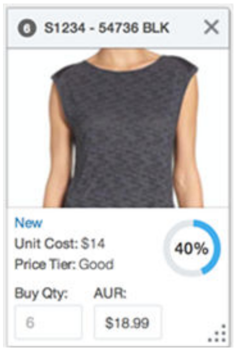


Figure 4-180 Small Card Icon

Large Card Type

Figure 4-181 shows a Large Card type example. Figure 4-180 shows its icon.

Figure 4-181 Large Card Example



S1234 - 54736 BLK

New
Unit Cost: $14
Price Tier: Good

40%

Buy Qty: AUR:

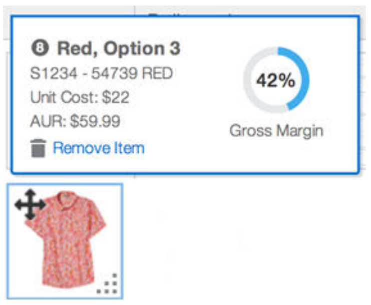
6 $18.99

Figure 4-182 Large Card Icon

Image with Pop-up Card Type

Figure 4-183 shows an Image with Pop-up Card type example. This type has no icon.

Figure 4-183 Image with Pop-up Example



Red, Option 3
S1234 - 54739 RED
Unit Cost: $22
AUR: $59.99
Remove Item

42%
Gross Margin

If more than one card type is enabled for the view, icons display to the right of the view's title bar that the user can use to toggle between them.

In the Assortment example, the Available Styles view uses the List card type, while the Assorted Styles view uses Small and Large.

The Card Definition provides attributes, measures, and translatable strings that are used by the platform's implementation of the Card Type to display the card. Each Card Definition can specify an attribute, up to three read-only measures, up to two editable measures, plus an optional gauge measure and an optional badge measure. All measures referenced here (except Option Attribute) are included in the Viewable Profile of the Card View configuration. The Card Type defines which parts of the Card Definition are used, which are ignored, and which are considered optional.

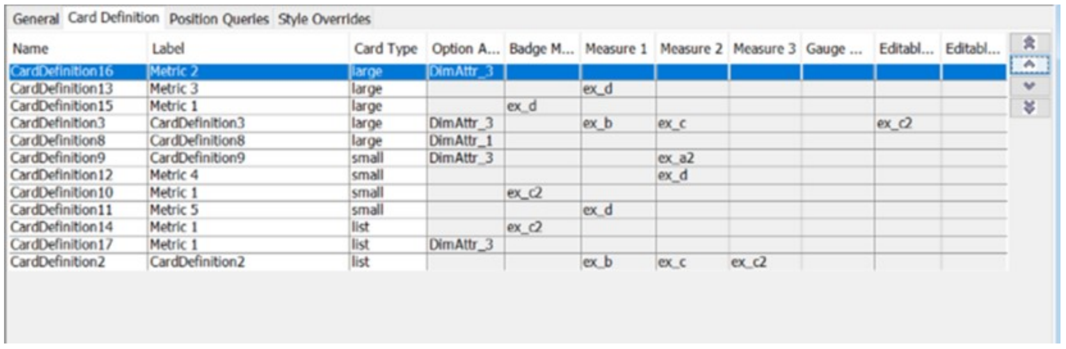
The configurer can define multiple Card Definitions per Card Type to increase the metrics visible to the user. The order of the definitions is the order in which they appear in the RPASCE Client. The RPASCE client allows the user to switch between the Card Definitions, first by choosing the Label, and then by switching between the Card Types. This capacity allows the grouping together of small sets of key metrics in a meaningful way with meaningful labels.

To change the order of Card Definitions during configuration, select the row of definition that you want to move and use the arrow buttons to the right of the Card Definition pane. Click a single arrow to move it one level up or down in the list. Click the double arrow to move it to the top or bottom of the list.

The number of Card Definition is limited to 12 across all card types per Card View worksheet, in order to keep configurations reasonable.

Here is an example with multiple card types with multiple card definitions.

Figure 4-184   Multiple Card Types with Multiple Card Definitions



Name	Label	Card Type	Option A...	Badge M...	Measure 1	Measure 2	Measure 3	Gauge ...	Editabl...	Editabl...
CardDefinition16	Metric 2	large	DimAttr_3							
CardDefinition13	Metric 3	large			ex_d					
CardDefinition15	Metric 1	large		ex_d						
CardDefinition3	CardDefinition3	large	DimAttr_3		ex_b	ex_c			ex_c2	
CardDefinition8	CardDefinition8	large	DimAttr_1							
CardDefinition9	CardDefinition9	small	DimAttr_3			ex_a2				
CardDefinition12	Metric 4	small				ex_d				
CardDefinition10	Metric 1	small		ex_c2						
CardDefinition11	Metric 5	small			ex_d					
CardDefinition14	Metric 1	list		ex_c2						
CardDefinition17	Metric 1	list	DimAttr_3							
CardDefinition2	CardDefinition2	list			ex_b	ex_c	ex_c2			

Add/Remove and Drag/Drop Behavior

Whether the card has a **+**, an **X**, or a Boolean toggle icon in RPASCE UI depends on the view's add/remove/drag/drop configuration.

The Add/Remove and Drag/Drop behavior in RPASCE UI are determined by configuration settings for Card Measure, Card Value, Filter Cards, and Enable Drop.

These are the same settings used in the Fusion Client's Tiled View but are interpreted by the RPASCE UI in its own way. Specifically:

- Card Value is used to signify whether this is a from (false) or to (true) view, for the purposes of add/remove and drag/drop.
- Enable Drop is used to enable the Drag/Drop and Add/Remove behavior appropriate for the Card Value.

The Card Measure and Card Intersection are expected to be the same between a from (Available) and a to (Assorted) view.

For the Available Styles as shown in Table 4-5.

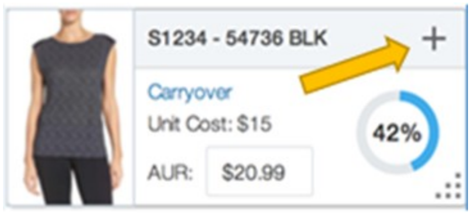
Table 4-5 Card Measure and Card Intersection for the Available Styles

Available	Assorted
Card Measure	inAssort
Card Value	False
Filter Cards	Yes
Enable Drop	Yes

Because Filter Cards is on, this view only shows cards where the Card Measure is equal to the Card Value, that is, when inAssort is False.

Because Enable Drop is on and Card Value is False, Cards have an add (**+**) operation that sets the Card Measure to True. The Card is then filtered out of the Available Style view and shows up in the Assorted Styles view.

Figure 4-185 Add Sets the Card Measure to True



For the Assorted Styles as shown in Table 4-6.

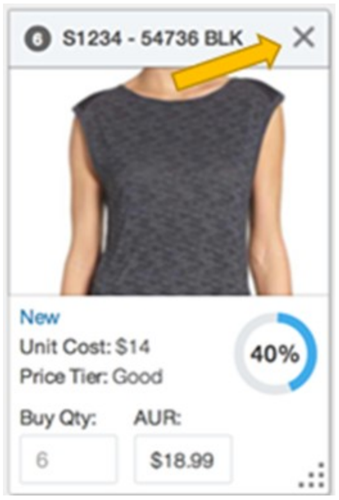
Table 4-6 Card Measure and Card Intersection Assorted Styles

Available	Assorted
Card Measure	inAssort
Card Value	True
Filter Cards	Yes
Enable Drop	Yes

Because Filter Cards is on, this view only shows tiles where the Card Measure is equal to the Card Value, that is, when inAssort is True.

Because Enable Drop is on and Card Value is True, Cards have a remove (**X**) operation that sets the Card Measure to False. The Card is then filtered out of the Assorted Styles view and shows up in the properly configured Available Styles view.

Figure 4-186   Remove Sets the Card Measure to False



Because Enable Drop is on and Card Value is True, this view has an add (**+**) operation that brings up an Figure 4-187.

Figure 4-187   Add Window

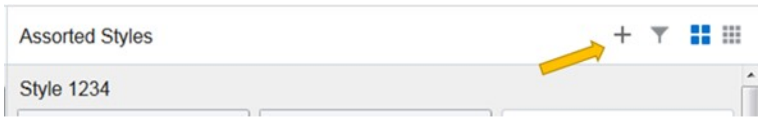
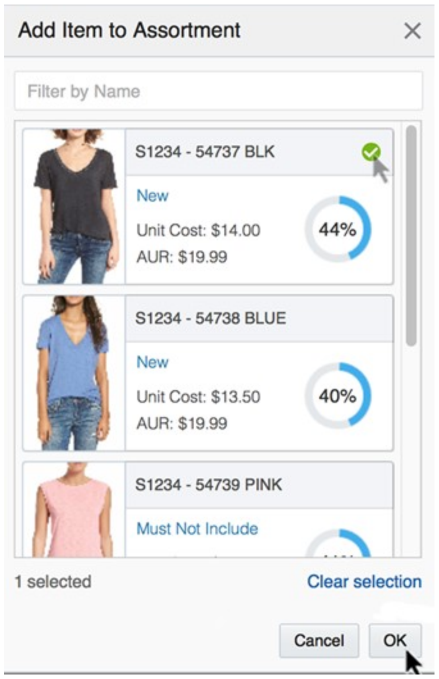


Figure 4-188 shows Available Styles on List cards.

Figure 4-188   Available Styles on List Cards



When you click **OK**, the Card Measure is set to True for the selected cards. This window is built using the List Card Type. A Card Definition is configured into the view specifically for Add and is populated by position where the Card Value is False.

Note that all these add and remove operations result in an edit to the Card Measure at the intersection of the visible cards. Like all cell edits, a Calculate is required to aggregate and spread the edit to other levels.

Define the Axis Layout of the Worksheet

The Axis layout in the Card View is set automatically in the backend, based on configurations to Base Intersection and Card Hierarchy.

The Card View worksheet contains an axis layout that is similar but not identical to the X-, Y-, and Z-Axis layout schema used by the Pivot worksheets. Like traditional axis layouts, the Card View axis layout requires that the hierarchies present in the base intersection of the worksheet be assigned to one and only one axis.

- **X-Axis** –This axis is assigned to Measure In the Card View.
- **Y-Axis** –This axis is assigned to the Card hierarchy only and is the axis that corresponds to cards.
- **Z-Axis** –All hierarchies present in the base intersection of the worksheet that are not assigned to the X-Axis or Y-Axis are assigned to the Z-Axis. This axis resembles the Page Axis of the Pivot Table views. Hierarchies assigned to the Page Axis are navigable using a set of controls that page the view as the selected position is modified by the user.
- **Unassigned Axis** –As with Pivot Views, there are no valid cases in which hierarchies may be assigned to the Unassigned Axis. This attribute provides

information to the user of the Configuration Tools that one or more hierarchies has not yet been assigned to a valid axis and details which hierarchies require axis assignment.

Support for Images, Attributes, and Measures

The Card View supports dimensional attributes by specifically providing an Image Attribute in the General Properties table and Option Attributes in the Card Definition panel. For details, see the Properties of the Card View Worksheet and Card Types and Card Definitions sections.

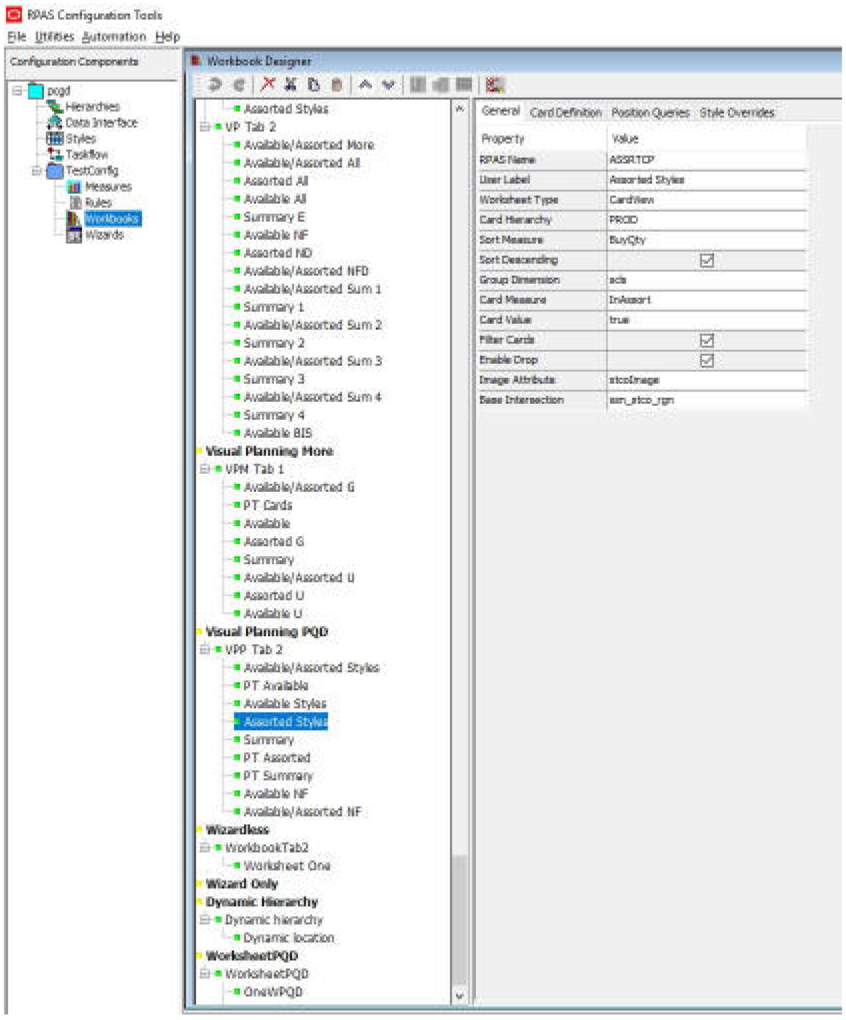
Support for Measure Profiles

Unlike the Pivot worksheet, the Card View automatically builds Measure Profiles in the backend, based on the user selection of measures in the Card View configuration. The Viewable Measure Profile, Default Measure Profile, and Selectable Measure Profile include Card Measure, Sort Measure, and all measures defined in the Card Definitions. Option and Image Attribute measures are dimensional attribute measures and are not included in the previous Measure Profiles.

Window Formatting

Unlike the Pivot View, the Card View does not have a Window Formatting tab to configure the worksheet style since only the Image and Option Attributes are visible display attributes supported in the Card View. The Card View internally uses an Attribute Profile to automatically track them. Label is always the first visible attribute that displays in the card's header. Image and Option attributes, if supplied, are displayed as dimension attributes in the RPASCE UI.

Figure 4-189   Workbook Designer Showing a Card View Worksheet



Property	Value
RPAS Name	ASSRTCP
User Label	Assorted Styles
Worksheet Type	CardView
Card Hierarchy	PROD
Sort Measure	BuyQty
Sort Descending	✓
Group Dimension	scls
Card Measure	InAssort
Card Value	true
Filter Cards	✓
Enable Drop	✓
Image Attribute	stcoImage
Base Intersection	ssn_stco_rgn

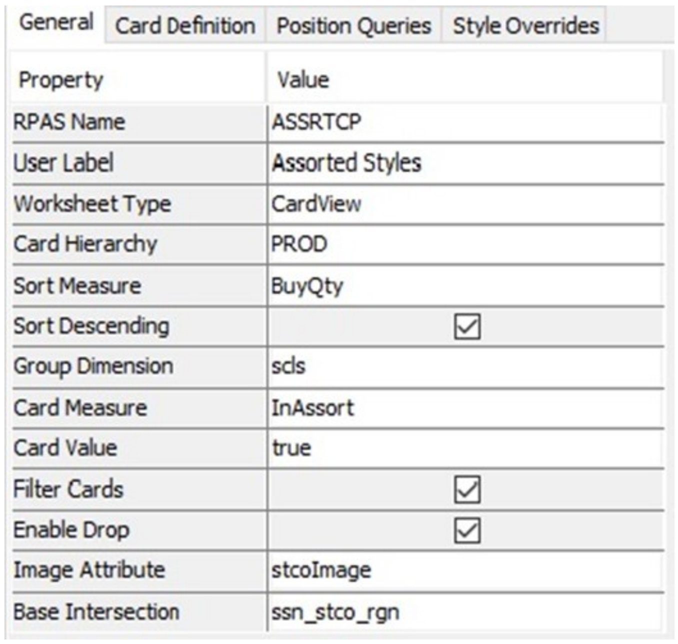
Properties of the Card View Worksheet

The Card View Worksheet supports a set of properties in the same manner as the Tiled View worksheets and the Pivot worksheets. The configuration of these properties is managed through the General Properties tab of the Workbook Designer through the same table interface used to configure properties for existing worksheet types.

From a configuration standpoint, the Card View worksheets are very similar to the Tiled View worksheets. The two views share many of the same sets of properties and optional configuration elements (implemented through the various tabs available within the Workbook Designer when a worksheet is selected). However, some Tiled View-specific properties may not be configurable for the Card View. In addition, the Card View worksheets support some additional Card View-specific properties.

The set of configurable properties for the Card View worksheet and summary information on each property can be found in this section.

Figure 4-190 General Properties Tab of a Card View Worksheet



Property	Value
RPAS Name	ASSRTCP
User Label	Assorted Styles
Worksheet Type	CardView
Card Hierarchy	PROD
Sort Measure	BuyQty
Sort Descending	✓
Group Dimension	scls
Card Measure	InAssort
Card Value	true
Filter Cards	✓
Enable Drop	✓
Image Attribute	stcoImage
Base Intersection	ssn_stco_rgn

RPAS Name

The RPAS Name attribute is the internal name used by RPASCE to identify the worksheet. The value of this property must be a valid RPAS CE name and is validated according to the rules for RPASCE names.

User Label

The User Label attribute is an external name used to identify the worksheet to the user of the RPASCE Client. The value of this property is displayed within the RPASCE Client for single language applications and is the resource exported for translation in multi-language applications.

Worksheet Type

The worksheet type attribute must be set to Card View in order to configure a Card View worksheet.

Card Hierarchy

The Card Hierarchy is similar to the Tile Axis Hierarchy in the Tiled View and is set to the Y-Axis in the Card View. The Card View must have the same Card Hierarchy as the Container View and its sub-views.

Sort Measure

The Sort Measure is optional in the Card View. At most, the Card View can specify one single Sort Measure. Users can select the Sort Measure via the pop-up measure selection window, which is pre-limited to numeric measures defined in the workbook rule sets when you click the **Sort Measure** field.

Sort Descending

The Sort Descending is an optional specification in the Card View. By default, it is set to False. The value of Sort Descending can only be specified when Sort Measure is selected.

Card Measure

The Card Measure is like the Tile Measure in Tiled View. See the Tile Measure in Tiled View for details. Users can select the Card Measure via the pop-up measure selection dialog box, which is pre-limited to Boolean measures defined in the workbook rule sets when clicking the Card Measure field.

The Card Measure's base dimension along the Card Hierarchy is referred to as the Card Dimension.

The default aggregation type of the card measure cannot be ambg. The aggregate value must evaluate to a true or false value for the card filtering functionality to work correctly.

Card Value

The Card Value is by default set to False. Users can use the drop-down list to set the Card Value to either True or False. When used in conjunction with the RPASCE Client drag and drop functionality, the Card Value attribute contains the value to be set into the Card Measure as a part of a drop operation. It must have a value of either True or False to be contained by the Boolean Card Measure.

Group Dimension

The Group Dimension is optional in the Card View. By default, this field is empty. A single Group Dimension may be configured. In RPASCE UI, cards are displayed grouped under their parent Group by dimension option when the Group Dimension is supplied.

Note:

The Card View can be configured with a sort order by specifying the Sort Measure and Sort Descending flag. This acts as the innermost sorting with respect to the configured Group Dimension, if present, and/or a Sort By attribute (a RPASCE UI feature) chosen by the user.

If supplied, the Group Dimension must be a level higher than the Card Dimension and lower than the Container's Top Filter dimension for the Card Hierarchy.

Filter Cards

Check box for Yes or No. This attribute specifies whether the values of the Card Measure are used to drive the filtering of visible intersections within the Card View worksheet. If checked, only intersections for which the Card Measure's value for that card is equal to the configured Card Value will be visible.

Enable Drop

Check box for Yes or No. When Drop is enabled, the worksheet can serve as a drop location.

Image Attribute

The Image Attribute is optional in the Card View. At most, the Card View can specify one single Image Attribute. You can select the Image Attribute measure using the Measure Selection window when you click the **Image Measure** field. The Image Attribute measures are pre-limited to dimensional attribute measures in the Application where they are String type measures and the UI type is set to media. The Image Attribute must be at the Card Level.

Base Intersection

As with Tiled Views, a Card View worksheet has a base intersection. The Card View worksheet intersection supports the configuration of a base intersection based upon the common base intersection of the measures within the Viewable Measure Profile. The Card View Viewable Measure Profile is automatically built from Card Measure, Sort Measure, and all measures are defined in the Card Definitions (except Option Attributes).

Note:

The Card View base intersection must be identical to the base intersection of the Card Measure. It is recommended to configure the Base Intersection after the Card Definition and the General Table Configuration in the Card View.

Additional Tabs of the Card View Worksheet

The Card View worksheets support the configuration of additional options through the tabs present in the Workbook Designer when a Card View worksheet is loaded into the User Interface. The additional tabs supported when working with a Card View worksheet are:

Card Definition Tab

The Card View worksheets support the ability to configure multiple Card Definitions for each card type. One Card View worksheet may contain one or more card types. The card types are defined inside the Card Definition panel, where each row represents one card type with its specific card definitions.

Here is the list of Card Definition Properties that users can configure:

Name

The RPAS Name attribute is the internal name used by RPASCE to identify the Card Definition. The value of this property must be a valid RPASCE name unique across the application and is validated according to the rules for RPASCE names.

Label

The User Label attribute is an external name used to identify the Card Definition to the user of the RPASCE Client. The value of this property is displayed within the RPASCE Client for single language applications and is the resource exported for translation in multi-language applications. Card Definitions can have duplicate labels.

Card Type

The Card Type attribute must be set to either small, large, list, or imagePopup type in order to configure a Card Definition.

Option Attribute

The Option Attribute is optional in the Card View and will be displayed as a visible attribute in the view if supplied. At most, one Card Type can specify one Option Attribute. Select the Option Attribute measure using the Measure Selection window when you click the **Option Attribute** field. The Option Attribute measures are pre-limited to dimensional attribute measures in the application. The Option Attribute measure must be at the Card Level.

Badge Measure

The Badge Measure is optional. You can set the badge measure using the Measure Selection window when you click this field. Badge Measures are pre-limited to measures defined in the workbook rule sets.

Measure 1, Measure 2, and Measure 3

Measures 1, 2, and 3 are optional. Users define them based on measures they plan to show in the card. They are pre-limited to measures defined in the workbook rule sets.

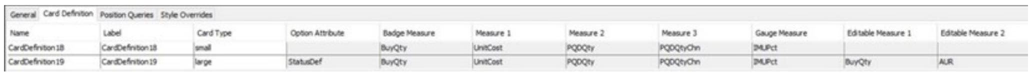
Editable Measure 1 and Editable Measure 2

Editable Measures 1 and 2 are optional. Users define them based on measures they plan to edit on the cards. In the assortment example, in the Assorted Styles card view, users can edit the number for the Buy Qty and AUR measures in the large card. In the Card Definition configuration for the large card, Editable Measure 1 is set to the BuyQty measure and Editable Measure 2 is set to the AUR measure. Both Editable Measure 1 and 2 are pre-limited to the measures defined in the workbook rule sets in the pop-up measure selection window.

Gauge Measure

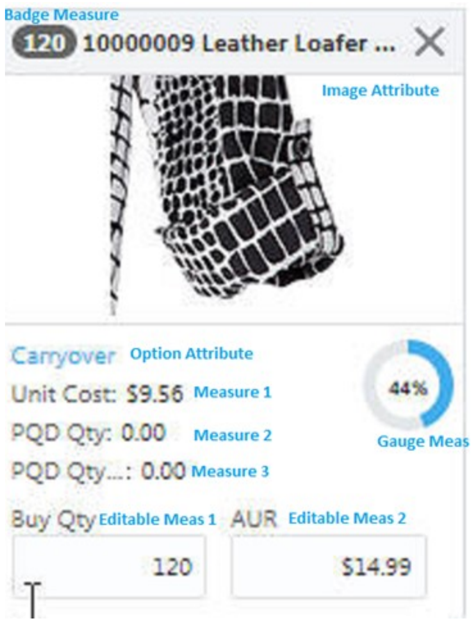
The Gauge Measure is optional. Users define it to represent the gauge. Currently, only the circular percent gauge type is supported. The gauge measure is pre-limited to measures defined in the workbook rule sets.

Figure 4-191 Card Definition Tab for a Card View Worksheet



Name	Label	Card Type	Option Attribute	Badge Measure	Measure 1	Measure 2	Measure 3	Gauge Measure	Editable Measure 1	Editable Measure 2
CardDefinition18	CardDefinition18	small		BuyQty	UnitCost	PQDQty	PQDQtyChn	IMUPct		
CardDefinition19	CardDefinition19	large	StatusDef	BuyQty	UnitCost	PQDQty	PQDQtyChn	IMUPct	BuyQty	AUR

Figure 4-192 Card Definition Tab for a Card View Worksheet



120 10000009 Leather Loafer ...

Carryover

Unit Cost: $9.56

PQD Qty: 0.00

PQD Qty...: 0.00

44%

Buy Qty 120

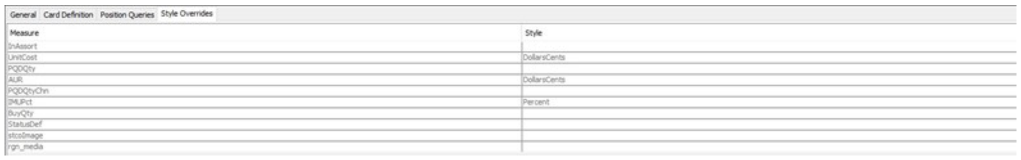
AUR $14.99

Style Overrides

The Card View worksheets support the ability to configure styles for measures displayed within the view (all viewable measures and dimension attributes). The Style Override tab is therefore available when configuring a Card View worksheet, and it functions in the same way as in Pivot and Detail Pop-up Views.

Additionally, it is possible to configure a set of formatting properties that are applied to the card itself in a Card View worksheet. The Style Override panel also includes an entry for the Card Measure, Image Attribute, and Option Attributes. Setting a style for the Card Measure within the Style Override panel will cause the set style to be used for card formatting and not the base configured style of the measure, if a style has been configured for the measure.

Figure 4-193 Style Override Tab for a Card View Worksheet



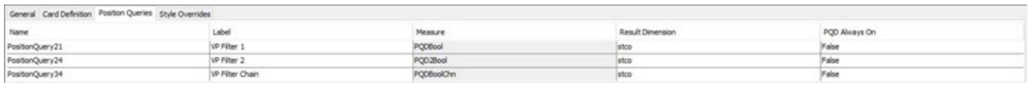
Measure	Style
InAssort	
UnitCost	DollarsCents
PQDQty	
AUR	DollarsCents
PQDQtyChn	
IMUPct	Percent
BuyQty	
StatusDef	
stcsImage	
rgn_media	

Position Queries

The Card View supports the use of Position Queries. The Position Queries panel of the Card View worksheet and the Tiled View worksheets function in the same way.

Note: The Position Queries defined in Card View must be identical to the Position Queries defined in the Container View and its sub-views. See the Position Queries of the Container View for details.

Figure 4-194   Position Queries Tab for a Card View Worksheet



Name	Label	Measure	Result Dimension	PQD Always On
PositionQuery21	VP Filter 1	PQDBool	stco	False
PositionQuery24	VP Filter 2	PQD2Bool	stco	False
PositionQuery34	VP Filter Chain	PQDBoolChn	stco	False

Defining Worksheet Properties for the Worksheets Summary View

RPASCE supports the Summary View within the RPASCE Client. This view is used to display a small number of metrics in a relatively standard way with a configured Summary Definition. The Summary View is always used together with the Card View within the Container View.

Summary View Worksheet

The Workbook Designer of the RPASCE Configuration Tools is used for the creation and modification of the Summary View worksheets. Users of the Configuration Tools can, by using the Worksheet Type attribute located within the General Properties tab of a worksheet, set a worksheet from the default of the Pivot/Chart worksheet to the new Summary worksheet.

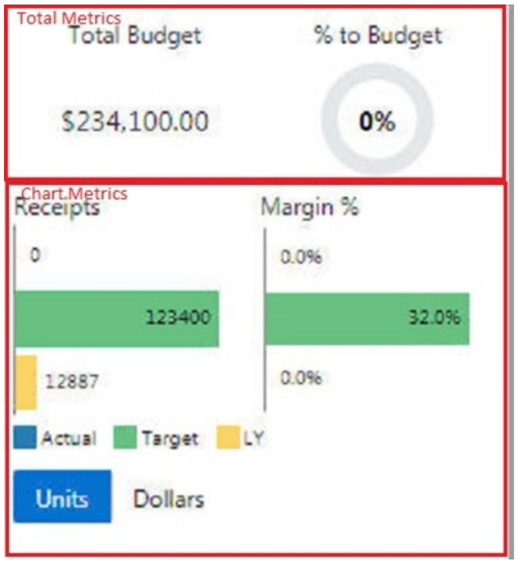
Once a worksheet has been set as a Summary Worksheet, the Workbook Tool user interface updates and presents the user of the Configuration Tools with the options available for the Summary View worksheets in the same way that it changes when a worksheet is configured to be a Detail Pop-up worksheet.

Features of the Summary View Worksheet

Figure 4-195 shows an example of a Summary View in the RPASCE Client. The view includes Total Metrics and Chart Metrics sections.

The two sections (Total Metrics and Chart Metrics) are all measures shown at the intersection defined by the Container View's Top Filter selections.

Figure 4-195   Summary View Worksheet



Total Metrics

RPASCE supports two measures here, one displayed as a quantity and one as a percentage via gauge. Label strings are also specified, as it is unlikely that the measure label would be ideal in this context. Currently, the gauge is displayed as a circular percent gauge.

Table 4-7 Total Metrics Configuration

Item	Description
Total Quantity Measure	Measure
Total Quantity Label	Translatable String
Total Gauge Measure	Measure
Total Gauge Label	Translatable String

Chart Metrics

RPASCE supports up to four metrics here. Each has a percent, units, and currency variation, for a total of 12 measures. Users can configure translatable strings to use as the labels in this presentation. Each is displayed at the intersection defined by the Top Filter of the container view.

Table 4-8 Chart Metrics Configuration

Item	Description
Percent Label	translatable string
Quantity Label	translatable string
Units Label	translatable string
Currency Label	translatable string
Metric 1 Label	translatable string
Metric 1 Units Measure	measure
Metric 1 Currency Measure	measure
Metric 1 Percent Measure	measure
Metric 2 Label	translatable string
Metric 2 Units Measure	measure
Metric 2 Currency Measure	measure
Metric 2 Percent Measure	measure
Metric 3 Label	translatable string
Metric 3 Units Measure	measure
Metric 3 Currency Measure	measure
Metric 3 Percent Measure	measure
Metric 4 Label	translatable string
Metric 4 Units Measure	measure

Table 4-8 (Cont.) Chart Metrics Configuration

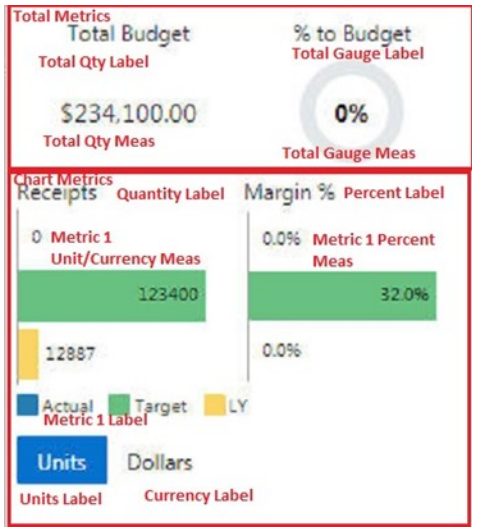
Item	Description
Metric 4 Currency Measure	measure
Metric 4 Percent Measure	measure
Units Chart Type	only horizontal bar available at this time
Currency Chart Type	only horizontal bar available at this time
Percent Chart Type	only horizontal bar available at this time

The Units, Currency, and Percent Metrics are displayed as horizontal bars. The Legend is used for metric labels. In the assortment example, the colors blue, green, and yellow are used to represent Metric 1 (Actual), Metric 2 (Target), and Metric 3 (LY) labels, respectively. The Metric 1 row is not displayed in blue in the example because of chart bar scaling. If switched to Dollars, Metric 1 row will be displayed in blue. The configuration settings listed in Table 4-8 display in the following layout:

Table 4-9 Layout of Configuration Settings

Quantity Label	Percentage Label
Metric 1 Units or Currency Measure	Metric 1 Percent Measure
Metric 2 Units or Currency Measure	Metric 2 Percent Measure
Metric 3 Units or Currency Measure	Metric 3 Percent Measure
Metric 4 Units or Currency Measure	Metric 4 Percent Measure

Figure 4-196 Example Summary Views



Support for Images, Attributes, and Measures

The Summary View does not provide a representation of information in which images, attributes, and dimensional measures may all be displayed in a single informational unit.

Define the Axis Layout of the Worksheet

The Axis layout in the Summary View is set automatically in the backend, based on configurations to the Base Intersection and the Card Hierarchy.

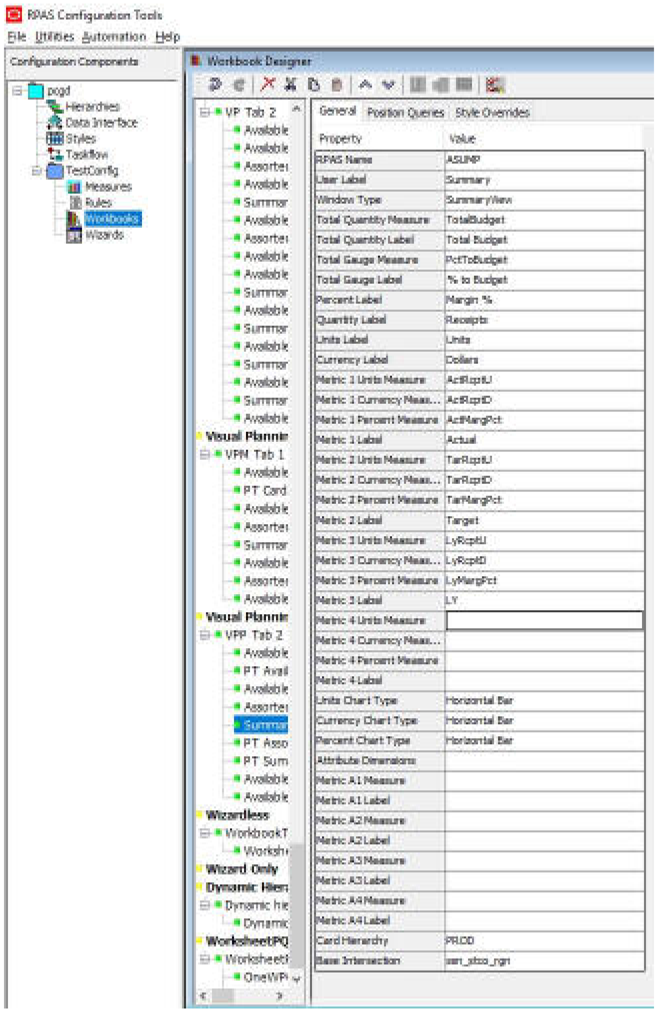
The Summary View worksheet contains an axis layout that is similar but not identical to the X-, Y-, and Z-Axis layout schema used by Pivot worksheets. Like traditional axis layouts, the Summary View axis layout requires the hierarchies present in the base intersection of the worksheet to be assigned to one and only one axis.

- **X-Axis** –This Axis is assigned to MEASURE in the Summary View.
- **Y-Axis** –This Axis is assigned to the Card Hierarchy only and is the axis that corresponds to cards.
- **Z-Axis** –All hierarchies present in the base intersection of the worksheet that are not assigned to the X-Axis, Y-Axis are assigned to the Z-Axis. This axis is like the Page Axis of Pivot Table views. Hierarchies assigned to the Page Axis are navigable using a set of controls that page the view as the selected position is modified by the user.
- **Unassigned Axis** –As with Pivot Views, there are no valid cases in which hierarchies may be assigned to the Unassigned Axis. This attribute provides information to the user of the Configuration Tools that one or more hierarchies has not yet been assigned to a valid axis and details which hierarchies require axis assignment.

Support for Measure Profiles

Like the Card View worksheet, the Summary View automatically builds the Measure Profiles in the backend, based on measure selection in the Summary Configuration. All measures selected in the Summary Definitions are included in the Viewable Measure Profile, Default Measure Profile, and Selectable Measure Profile.

Figure 4-197 Workbook Designer Showing a Summary View Worksheet



Property	Value
RPAS Name	ASUMP
User Label	Summary
Window Type	SummaryView
Total Quantity Measure	TotalBudget
Total Quantity Label	Total Budget
Total Gauge Measure	PctToBudget
Total Gauge Label	% to Budget
Percent Label	Margin %
Quantity Label	Receipts
Units Label	Units
Currency Label	Dollars
Metric 1 Units Measure	ActRcptU
Metric 1 Currency Meas…	ActRcptD
Metric 1 Percent Measure	ActMargPct
Metric 1 Label	Actual
Metric 2 Units Measure	TarRcptU
Metric 2 Currency Meas…	TarRcptD
Metric 2 Percent Measure	TarMargPct
Metric 2 Label	Target
Metric 3 Units Measure	LyRcptU
Metric 3 Currency Meas…	LyRcptD
Metric 3 Percent Measure	LyMargPct
Metric 3 Label	LY
Metric 4 Units Measure	
Metric 4 Currency Meas…	
Metric 4 Percent Measure	
Metric 4 Label	
Units Chart Type	Horizontal Bar
Currency Chart Type	Horizontal Bar
Percent Chart Type	Horizontal Bar
Attribute Dimensions	
Metric A1 Measure	
Metric A1 Label	
Metric A2 Measure	
Metric A2 Label	
Metric A3 Measure	
Metric A3 Label	
Metric A4 Measure	
Metric A4 Label	
Card Hierarchy	PROD
Base Intersection	sen_stco_rgn

Properties of the Summary View Worksheet

The Summary View Worksheets support a set of properties in the same manner as the Card View worksheets. The configuration of these properties is managed through the General Properties tab of the Workbook Designer through the same table interface used to configure properties for existing worksheet types.

From a configuration standpoint, the Summary View worksheets are like the Card View worksheets. The two views share many of the same sets of properties and optional configuration elements (implemented through the various tabs available within the Workbook Designer when a worksheet is selected). However, some Card View-specific properties may not be configurable for the Summary View worksheets. In addition, the Summary View worksheets support some additional, Summary View-specific properties.

The set of configurable properties for the Summary View worksheet and summary information on each property can be found in this section.

RPAS Name

The RPAS Name attribute is the internal name used by RPASCE to identify the worksheet. The value of this property must be a valid RPASCE name and is validated according to the rules for RPASCE names.

User Label

The User Label attribute is an external name used to identify the worksheet to the user of the RPASCE Client. The value of this property is displayed within the RPASCE Client for single language applications and is the resource exported for translation in multi-language applications.

Worksheet Type

The worksheet type attribute must be set to the Summary View in order to configure a Summary View worksheet.

Total Quantity Measure

The Total Quantity Measure is pre-limited to numeric measures in the workbook rule sets in the pop-up measure selection window.

Total Quantity Label

The Total Quantity Label is a translatable string entered by users. This label is displayed above the Total Quantity Measure value in the view. In the assortment example, Total Budget is configured.

Total Gauge Measure

The Total Gauge Measure is a measure that is displayed as a circular percent gauge in the RPASCE client. Users can select the Total Gauge Measure via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Total Gauge Label

The Total Quantity Label is a translatable string entered by users. This label is displayed above the Total Gauge Measure value in the view. In the assortment example, % to Budget is configured.

Percent Label

The Percent Label is a translatable string entered by users. This label is displayed above the Percent Metric column in the view. In the assortment example, this label is configured to be Margin %.

Quantity Label

The Quantity Label is a translatable string entered by users. This label is displayed above the Units/Currency Metric column in the view. In the assortment example, this label is configured to be Receipts.

Units Label

The Units Label is a translatable string entered by users. This label is displayed at the bottom of the Units/Currency Metric column in the view. In the assortment example, this label is configured to be Units.

Currency Label

The Currency Label is a translatable string entered by users. This label is displayed at the bottom of the Units/Currency Metric column in the view. In the assortment example, this label is configured to be Dollars.

Metric 1 Units Measure

The Metric 1 Units Measure is displayed in the first row, first column of the Chart Metrics section if the Units Label, for example, Units, is selected. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 1 Currency Measure

The Metric 1 Currency Measure is displayed in the first row, first column of the Chart Metrics section if the Currency Label, for example, Dollars, is selected. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 1 Percent Measure

The Metric 1 Percent Measure is displayed in the first row, second column of the Chart Metrics section, under the Percent Label. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 1 Label

The Metric 1 Label is a translatable string entered by users. This label is used to set the label legend in the view. In the assortment example, this label is configured to be Actual and the legend indicates that the first row is displayed as blue if the chart bar scaling allows.

Metric 2 Units Measure

The Metric 2 Units Measure is displayed in the second row, first column of the Chart Metrics section if the Units Label, for example, Units, is selected. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 2 Currency Measure

The Metric 2 Currency Measure is displayed in the second row, first column of the Chart Metrics section if the Currency Label, for example, Dollars, is selected. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 2 Percent Measure

The Metric 2 Percent Measure is displayed in the second row, second column of the Chart Metrics section. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 2 Label

The Metric 2 Label is a translatable string entered by users. This label is used to set the label legend in the view. In the assortment example, this label is configured to be Target and the legend indicates that the second row must be displayed as green if the chart bar scaling allows.

Metric 3 Units Measure

The Metric 3 Units Measure is displayed in the third row, first column of the Chart Metrics section if the Units Label, for example, Units, is selected. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 3 Currency Measure

The Metric 3 Currency Measure is displayed in the third row, first column of the Chart Metrics section if the Currency Label, for example, Dollars, is selected. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 3 Percent Measure

The Metric 3 Percent Measure is displayed in the third row, second column of the Chart Metrics section. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 3 Label

The Metric 3 Label is a translatable string entered by users. This label is used to set the label legend in the view. In the assortment example, this label is configured to be LY and the legend indicates that the second row must be displayed as yellow if the chart bar scaling allows.

Metric 4 Units Measure

The Metric 4 Units Measure is displayed in the fourth row, first column of the Chart Metrics section if the Units Label, for example, Units, is selected. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 4 Currency Measure

The Metric 4 Currency Measure is displayed in the fourth row, first column of the Chart Metrics section if the Currency Label, for example, Dollars, is selected. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 4 Percent Measure

The Metric 4 Percent Measure is displayed in the fourth row, second column of the Chart Metrics section. Users can set it via the pop-up measure selection window, which is populated with numeric measures defined in the workbook rule sets.

Metric 4 Label

The Metric 4 Label is a translatable string entered by users. This label is used to set the label legend in the view.

Note:

Up to four Chart Metrics can be configured. If one metric definition is not completely provided, that is, if either the Units Measure, Currency Measure, Percent Measure, or Label is not defined by users, then this chart metric row will not be displayed in the RPASCE Client.

Units Chart Type

Only Horizontal Bar is supported for Units Chart Type.

Currency Chart Type

Only Horizontal Bar is supported for Currency Chart Type.

Percent Chart Type

Only Horizontal Bar is supported for Percent Chart Type.

Card Hierarchy

The Card Hierarchy is like the Card Hierarchy in the Card View and is set to the Y-Axis in the Summary View. The Summary View must have the same Card Hierarchy as the Container View and its other sub-views.

Base Intersection

As with the Card View, a Summary View worksheet has a base intersection. The Summary View worksheet intersection supports the configuration of a base intersection based upon the common base intersection of the measures within the Viewable Measure Profile. The Summary View Viewable Measure Profile is automatically built from all measures selected in the Summary definition.

Note:

The Summary View base intersection must be identical to the base intersection of the Container View and Card Views. The Base Intersection must be set after all necessary measures have been defined for the Summary Definition.

The other properties in the General Properties table such as Attribute Dimensions, Metric A1 Measure, Metric A1 Label, Metric A2 Measure, Metric A2 Label, Metric A3 Measure, Metric A3 Label, Metric A4 Measure, and Metric A4 Label are reserved for Attribute Summary metrics, which are to be supported in later releases.

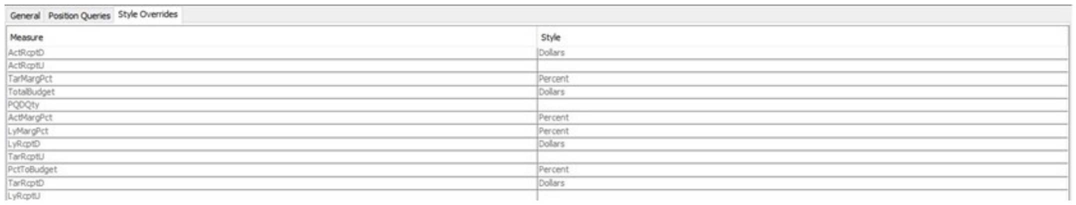
Additional Tabs of the Summary View Worksheet

The Summary View worksheets support the configuration of additional options through tabs present in the Workbook Designer when a Summary View worksheet is loaded into the User Interface. The additional tabs supported when working with a Summary View worksheet are:

Style Overrides

The Summary View worksheets are used to configure styles for measures displayed within the view. The Style Override tab is available when configuring a Summary View worksheet and it functions in the same way as in the Pivot and Detail Pop-up Views.

Figure 4-198 Style Override Tab for a Summary View Worksheet Showing Summary Measures



Measure	Style
ActRcptD	Dollars
ActRcptU	
TarMargPct	Percent
TotalBudget	Dollars
PQDQty	
ActMargPct	Percent
LyMargPct	Percent
LyRcptD	Dollars
TarRcptU	
PctToBudget	Percent
TarRcptD	Dollars
LyRcptU	

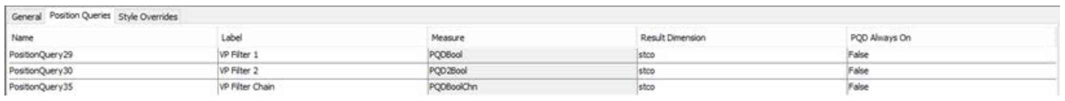
Position Queries

The Summary View does support the use of Position Queries. The Position Queries panel of the Summary View worksheet functions in the same way as the Pivot View worksheets.

Note:

If Position Queries are defined for the Summary View, they must be identical to the Position Queries defined in the Container View and Card Views. See the Position Queries of Container View for details.

Figure 4-199 Position Queries Tab for a Summary View Worksheet



General | Position Queries | Style Overrides

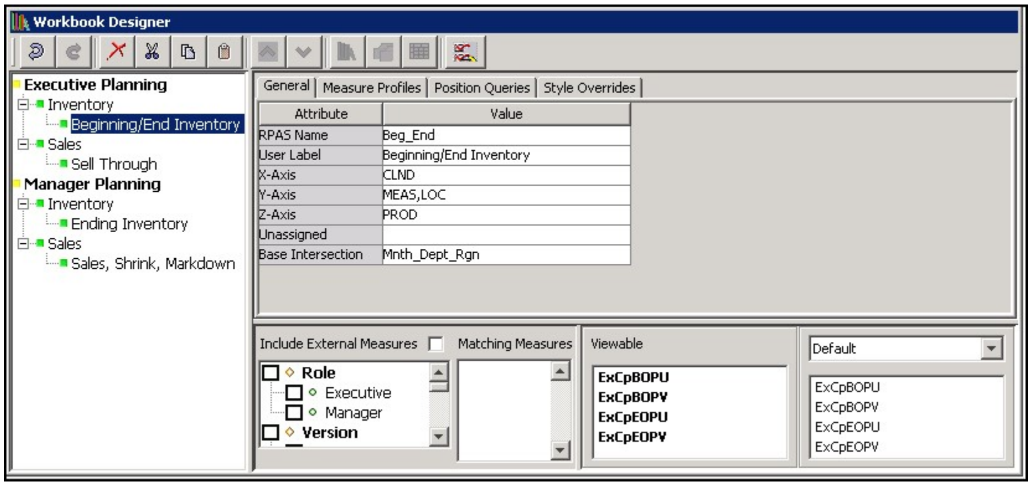
Name	Label	Measure	Result Dimension	PQD Always On
PositionQuery29	VP Filter 1	PQDBool	stco	False
PositionQuery30	VP Filter 2	PQD2Bool	stco	False
PositionQuery35	VP Filter Chain	PQDBoolChn	stco	False

Specifying Which Measures Display in a Worksheet

Complete the following steps to specify which measures display in a worksheet.

Navigate: In the Configuration Components pane, select **Project** then **Solution**, and then **Workbooks**. The Figure 4-200 opens in the workspace.

Figure 4-200 Workbook Designer Window



Attribute	Value
RPAS Name	Beg_End
User Label	Beginning/End Inventory
X-Axis	CLND
Y-Axis	MEAS,LOC
Z-Axis	PROD
Unassigned	
Base Intersection	Mnth_Dept_Rgn

Viewable: ExCpBOPU, ExCpBOPV, ExCpEOPU, ExCpEOPV

Default: ExCpBOPU, ExCpBOPV, ExCpEOPU, ExCpEOPV

1. Select the worksheet to be used to specify measures.
2. Select the profile to be used to specify measures. When the user has specified multiple profiles for a worksheet using the Measure Profiles tab, each of these profiles may have a different set of measures defined for it. In cases where there is more than one configured profile, one profile is used as the Selected profile. It is this profile that will have its measure content modified through interaction within the Profile Panel. When there are multiple configured profiles for a worksheet, the profile that is currently Selected may be changed using the drop-down list in the upper right of the Profile Panel.
3. In the column that displays measure components (under Include External Measures), select the check boxes next to the Measure components. The

matching measures display in the Matching Measures column as the components are selected. Only realized measures that are used in the rule set that is assigned to the selected workbook display.

4. If External Measures are to be available for placement on the worksheet, select the Include External Measures check box.
5. To add measures from the matching measures column to the Viewable or Selected columns, do one of the following options:
   - Select the measures to add from the Matching Measures column. Drag the measures to the Viewable or Selected column.
   - Select the measures to add from the Matching Measures column and press **Ctrl+C**, and then click in the Viewable or Selected column and press **Ctrl+V**.
   - Right-click in the Matching Measures column and select **Copy**, and then select **Paste** from the right-click menu in the Viewable or Selected column.
   - To add all measures from the Matching Measures column, right-click in the Viewable or Selected column and select Add Matching from the menu.
6. To add measures from the Viewable column to the Selected column, do one of the following options:
   - Select measures in the Viewable column and drag the measures to the Selected column.
   - Select measures in the Viewable column and press **Ctrl+C**, and then click in the Selected column and press **Ctrl+V**.
   - Right-click in the **Viewable** column and select **Copy**. Right-click in the **Selected** column and select **Paste**.

   Note:

   Adding a measure to the **Selected** column also adds it to the **Viewable** column if it is not already in the **Viewable** column.

7. To remove measures from the **Viewable** or **Selected** columns: do one of the following options:
   - Select the measures to remove and press **Delete** or **Ctrl+X**, or right-click and select **Cut**.
   - Right-click and select **Remove Matching** to remove all measures that are also in the Matching Measures column or select **Remove All** to remove all measures.

   Note:

   Removing a measure from the Viewable column also removes it from all profiles in the **Selected** column.

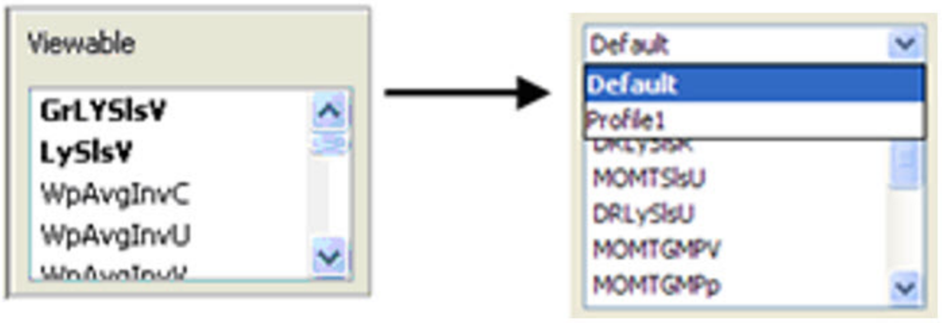
Specify the Sequence of Measures on a Worksheet

Complete the following steps to specify the sequence of measures on a worksheet.

Navigate: In the Configuration Components pane, select **Project** then **Solution**, and then **Workbooks**. The Figure 4-200 opens in the workspace.

1. Select the worksheet that will be used to sequence measures.
2. To sequence measures manually, drag and drop the measures from the **Viewable** column to the **Selected** column.

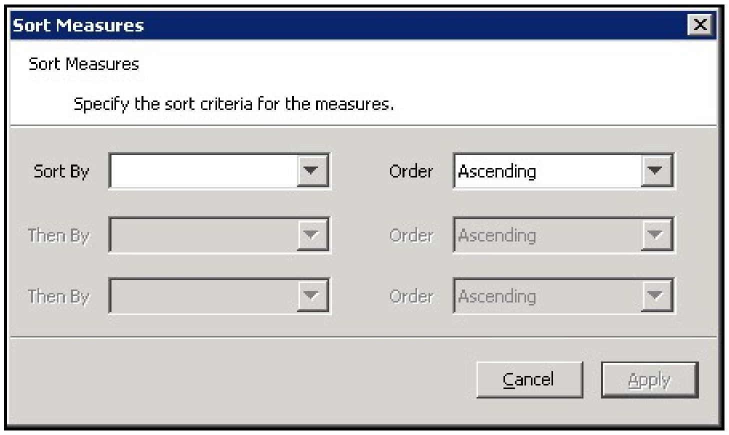
Figure 4-201 Viewable Column to Selected Column



3. To sort the measures:
   a. Right-click and select Sort in the **Selected** column. The Figure 4-202 opens.

> **Note:**
>
> Sorting measures are based on the internal component name (not the label).

Figure 4-202 Sort Measures Window


Sort Measures

Sort Measures

Specify the sort criteria for the measures.

Sort By | Order: Ascending
Then By | Order: Ascending
Then By | Order: Ascending

   b. Measures can be sorted based on their major components. Select the sort sequence in which the major components are to be applied, and whether the sort should be ascending or descending
   c. Click **Apply**.

The measures are sorted into the specified sequence. This is a one-time sort. If new measures are added to the Selected column, or measures are manually sequenced, the sort sequence previously specified will no longer apply. You would need to resort the measures again.

Edit Worksheet Properties

Complete the following steps to specify the sequence of measures on a worksheet.

Navigate: In the Configuration Components pane, select **Project** then **Solution**, and then **Workbooks**. The Figure 4-200 opens in the workspace.

1. Click the worksheet to edit.
2. Update the information as appropriate.
3. To remove information from the Position Queries table:
   a. Select the field.
   b. Right-click and select **Remove**.

Remove a Worksheet

Complete the following steps to remove a worksheet:

1. Select the worksheet to remove.
2. From the toolbar, click **Delete**, or right-click and select **Remove**.
3. Click **Yes**.

Wizards

Note:

The following section contains descriptions of features that are not supported in the current version of RPASCE. However, in order to assist in the process of migration of a pre-CE solution to the CE platform, the configuration components associated with these deprecated features remain present within the RPASCE Configuration Tools and therefore continue to be described within this document. For additional information, see Appendix – Calculation Engine User Guide in this document.

This section describes the tasks you can perform by using the wizard designer. The wizard designer supports the graphical layout for custom wizards.

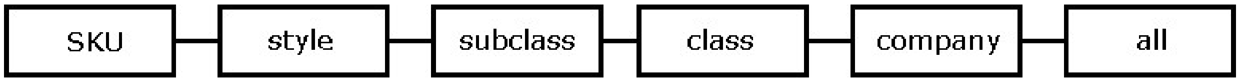
When the workbook build process only involves selecting the scope of the workbook by selecting positions from the standard hierarchies, you can use standard wizards as described in the General Tab of the Workbook. Most of the workbooks can be built using the standard wizards, and no coding is required – just configuration.

If the wizard process needs to do additional processing, you need to use custom wizards. The following two examples will help to clarify the need for custom wizards.

A Simple Example

Consider the product hierarchy:

Figure 4-203 Product Hierarchy



If the workbook builder needs to select just a few SKUs from an application that contains many SKUs, that process could be tedious if the builder is presented with a wizard with huge numbers of SKUs. You may want to include a two-step process to select the SKUs:

- In the first step, the builder selects a class.
- In the second step, the builder selects SKUs from just the SKUs in that class.

A More Complex Example

The workbook build process may need the builder to make choices from some predefined options. The choices that the builder makes could determine what further selections or choices the builder must make. The workbook that is eventually built could therefore be of several different ‘subtypes.'

Neither of the previous examples are configurable as Standard Wizards, so custom wizards must be used. The custom wizards can be designed in the custom wizard designer, but must be accompanied with code that:

- Describes the sequence of the wizards
- Collects and processes the information from the wizards
- Generates the content of the next wizard
- Describes the content of the workbook that is generated

In the first example, the workbook designer lays out two 2-tree wizards (one for class and the other for SKU). Then to support the wizard process, the designer would write the code to read the selection from the class wizard and range down the available SKUs in the second wizard.

In the second example, the designer lays out the first wizard page as a group of check boxes (or list boxes if there are too many options). The designer would also lay out the other wizard pages that are required. The designer writes the code to collect the selections made in the first wizard, and to display (or not) the other wizard pages that are dependent on the selections made.

In summary, when using custom wizards, the designer is responsible for:

- Laying out the wizards.
- Writing code to control the transition between the wizards and out of the wizards.
- Writing code to initiate the building of the workbook.

The Wizard Tool only helps the designer in the first aspect of this work. The designer (the one who establishes the layout of these wizards) is expected to have knowledge of the process of controlling, collecting, and processing the information from these wizards.

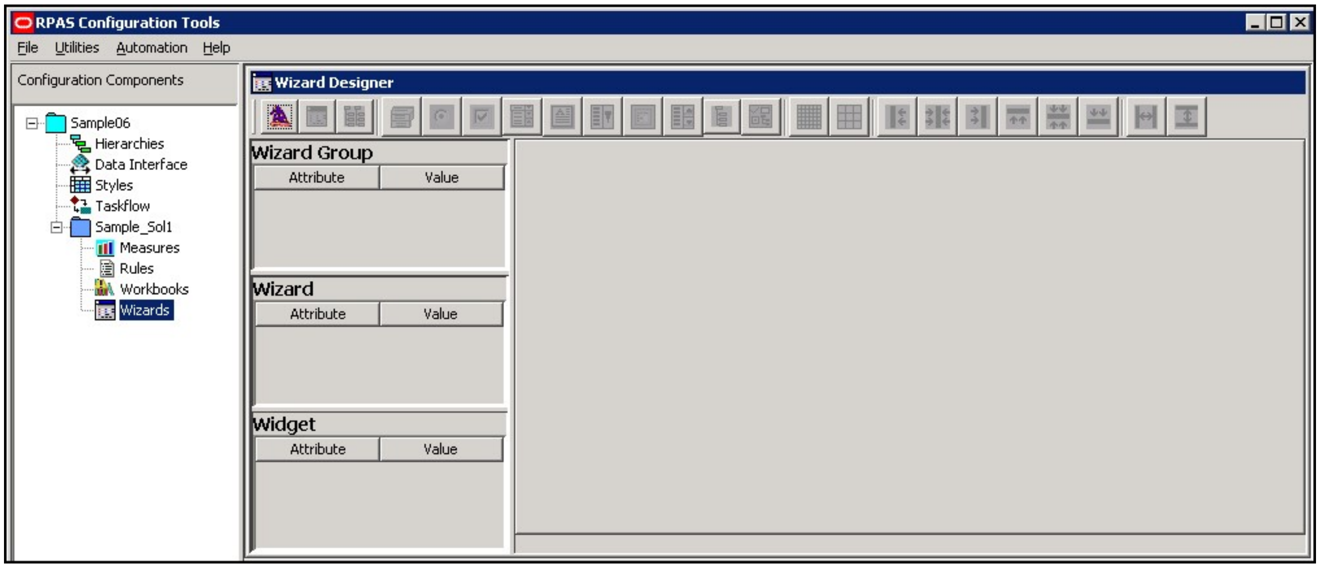
The following section describes working with layout of the custom wizards.

Create a Wizard Group

Complete the following steps to create a Wizard Group.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Wizards**. The Figure 4-204 opens in the workspace.

Figure 4-204 Wizard Designer Window



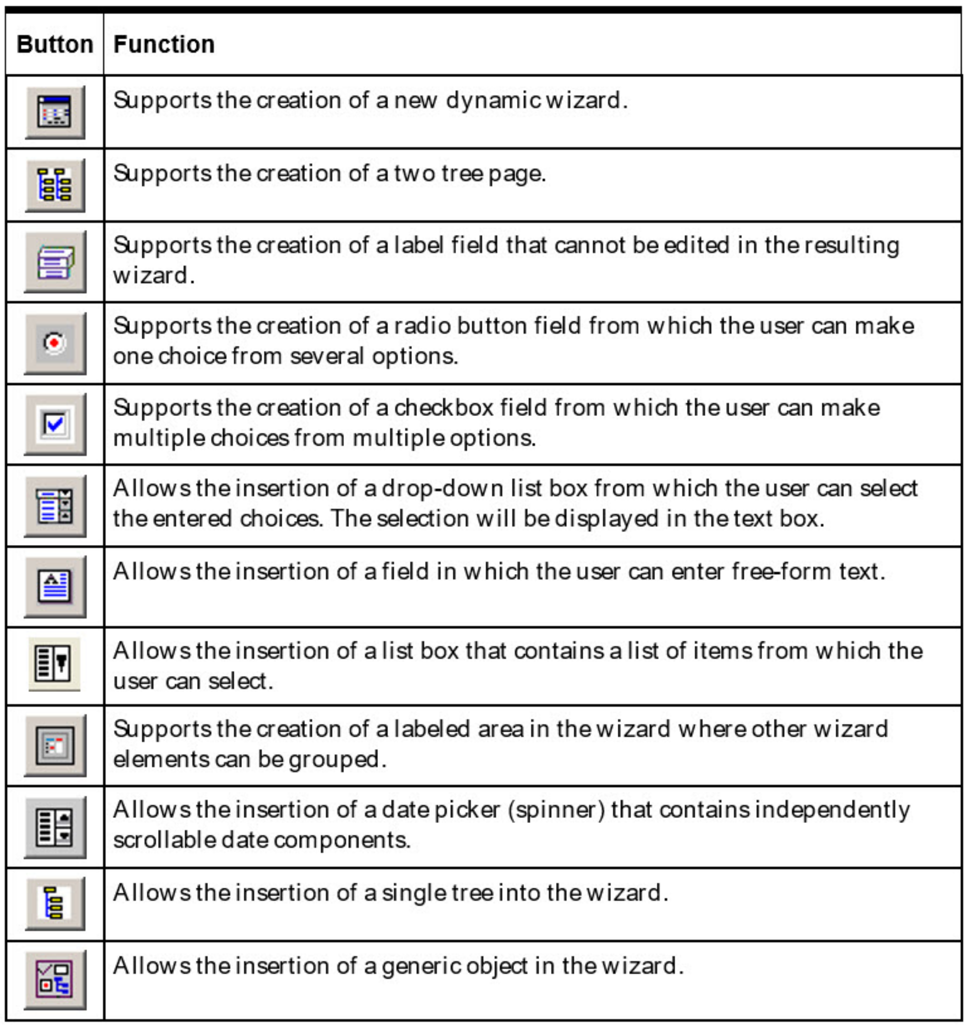
1. In the Wizard Designer window, click **New Wizard**.
2. In the Wizard Group area, click in the Value field, and enter the name of the Wizard Group.

Create a Wizard Page

Complete the following steps to create a Wizard Page.

1. Click one of the buttons shown in Figure 4-205 to add the appropriate control:

Figure 4-205 Wizard Buttons



Button	Function
	Supports the creation of a new dynamic wizard.
	Supports the creation of a two tree page.
	Supports the creation of a label field that cannot be edited in the resulting wizard.
	Supports the creation of a radio button field from which the user can make one choice from several options.
	Supports the creation of a checkbox field from which the user can make multiple choices from multiple options.
	Allows the insertion of a drop-down list box from which the user can select the entered choices. The selection will be displayed in the text box.
	Allows the insertion of a field in which the user can enter free-form text.
	Allows the insertion of a list box that contains a list of items from which the user can select.
	Supports the creation of a labeled area in the wizard where other wizard elements can be grouped.
	Allows the insertion of a date picker (spinner) that contains independently scrollable date components.
	Allows the insertion of a single tree into the wizard.
	Allows the insertion of a generic object in the wizard.

2. Click on the wizard page grid to place the selected control on the page.
3. If necessary, drag the control to the appropriate place on the grid to reposition it.
4. In the Widget area, enter the following information:
   - **Name** – The RPASCE internal name of the control.
   - **Type** – The control type.
   - **Text** – The text to be displayed on the control label.
   - **Time_Format** – The format of the time. There are two valid inputs for this attribute: 12 and 24. Use 12 if you want to use a 12-hour clock (for example, the time displays like 10:58PM). Use 24 for a 24-hour clock (for example, the time displays like 22:58).
   - **Align** – The same as the style attribute. Valid values are left, right, center, multiline, and flip depending on the type of widget being created.
   - **Func** – Indicates whether the widget will be dynamic or static. These are the only valid values for this attribute.

- **Locx** – The x coordinate of the control on the wizard page. The value of this field is automatically changed when the control is moved using your mouse.
- **Locy** – The y coordinate of the control on the wizard page. The value of this field is automatically changed when the control is moved using your mouse.
- **Width** – The width of the control in pixels.
- **Height** – The height of the control in pixels.

Edit Wizard Control Properties

Complete the following steps to edit wizard control properties.

Navigate: In the Configuration Components pane, select **Project**, then **Solution**, and then **Wizards**. The Figure 4-204 opens in the workspace.

1. Select the wizard group tab that contains the wizard to edit.
2. Select the wizard tab that contains the widget to edit.
3. Select the widget.
4. In the Widget area, update the information as necessary.

5 Accessibility

This chapter documents support for accessibility in the RPASCE Configuration Tools. It describes the support for accessibility and assistive technologies within the underlying platform used by the Configuration Tools. Additionally, it details accessibility support and considerations built into the application beyond the capabilities of the underlying platform.

Java Swing and the Java Access Bridge

The RPASCE Configuration Tools is built using the Java Swing platform as its underlying technology. The Java Swing platform, which form a part of the Java Foundation Classes framework, provide robust support for accessibility and assistive technologies.

This support takes the form of full support for the Java Accessibility API, a set of APIs that provide the ability for assistive technologies to interface with an application programmatically and for an application to adapt to accessibility settings made within the Windows operating system.

This support includes:

- Support for alternate input devices
- Support for keyboard-only operation of the application
- Support for high contrast and larger text display
- Support within each UI component for accessible roles, labels, and descriptions to support assistive technologies such as screen readers

In order to provide integration with assistive technologies and accessibility settings of the Windows environment, Java contains a supplemental library called the Java Access Bridge. This supplemental library must be enabled in order for it function.

Information about the Java Access Bridge - including instructions on how to enable it – can be found at:

`https://www.oracle.com/technetwork/java/javase/tech/index-jsp-136191.html`

Accessibility Support with the RPASCE Configuration Tools

In addition to the accessibility support provided by the Java Foundation Classes framework, the RPASCE Configuration Tools provide the following accessibility features:

- All components have a text identifier, either in the form of a component label or an accessible tooltip.
- Color contrast has been evaluated across the application in order to ensure a contrast ratio of 4.5:1. In places where a ratio of 4.5:1 could not be met, alternative cues such as the use of an italic font style provide a secondary visual indicator.
- The Configuration Tools supplement visual indications of errors with a centralized task list containing records of validation issues to assist in error identification and correction.

- A consistent keyboard focus navigation policy has been ensured across the Configuration Tools (See note on traversal mechanics in next section.)

Keyboard Focus Traversal Considerations

The RPASCE Configuration Tools has a complex component layout, consisting of internal windows and multiple compound components such as tables and trees. In order to provide a consistent mechanism for keyboard traversal, the following conventions exist.

- The focus traversal key (Tab) will iterate in a logical fashion through the components according to their layout.
- When the focus enters a compound component type such as a table, the focus traversal key (Tab) will retain focus within the component, wrapping from the last component in the inner sequence to the first. Use of **Ctrl+Tab** moves the focus out of the inner focus cycle group to the next component in the outer cycle group (for example, from an embedded table to the component following the table according to the traversal order).
- Within certain compound components such as trees, menus, and tab groups, use of the arrow keys allows internal navigation, while Tab moves the focus to the next component according to the traversal order.
- Once focus has entered one of the internal frames of the application such as the Measure Tool or Rule Tool, it cycles through the components of that tool. In order to transition out of the embedded frame back to the enclosing workbench, make use of **Ctrl+Tab** when the focus is on the initial component of the embedded frames traversal order (in most cases the initial button in the toolbar).
- At all times, the Alt key activates the main menu bar while retaining keyboard focus on the currently focused component. Navigation of the menu can be accomplished using the arrow key or, when applicable, mnemonics or accelerators.

Keyboard Shortcuts

Keyboard shortcuts enable users to work efficiently. Instead of using a mouse, users with mobility or vision disabilities can use keyboard shortcuts to navigate through the software. While the previous Keyboard Focus Traversal section describes shortcuts related to focus traversal, this section lists additional frequently used shortcuts for accessing ConfigTools GUI components.

Note the following:

- A plus sign (+) in a shortcut means that users need to press the multiple keys at the same time.
- If the focus is inside an editable text component or selectable component such as button etc, use Shift +Tab to move focus to the previous component, and Tab to the next component.
- When the focus is on a menu, users can use Mnemonics to activate its menu item. The menu item mnemonics is the underscored letter in the menu item name.
- To open any menu item directly, use Alt + Menu mnemonic + Menu Item Mnemonic. For example, to open the Domain configuration dialog directly, use Alt + F + W shortcuts.

- When a button has the focus, typing the Space key is equivalent to a mouse click on the button.

Configuration Tools Menu Bar

In the Configuration Tools menu bar, use the left or right arrow key to navigate the File, Utilities, Automation and Help menus.

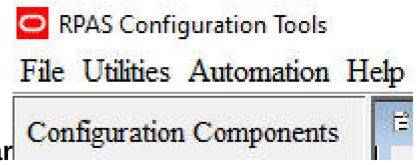


Figure 5-1 Configuration Tools Menu Bar

Table 5-1 File Menu

Action	Keyboard
Open the File menu	Alt + F
In the File menu, open the New dialog box	W
In the File menu, open the Domain Configuration	O
In the File menu, open the Integration Configuration	I
In the File menu, close the File menu	C
In the File menu, save the current configuration	S
In the File menu, save as for the current configuration	A
In the File menu, open the Tools Preferences dialog box	R
In the File menu, open the Configuration Properties window	P
In the File menu, open the recently closed configuration	the seq number of the configuration
In the File menu, close the ConfigTools	X

In the New Dialog Box

Action	Keyboard Shortcut
Open the Project dialog box	P
Open the Solution dialog box	S
Open the Integration Configuration	I

Table 5-2 Utilities Menu

Action	Keyboard Shortcut
Open the Utilities menu	Alt + U
In the Utilities menu, open the Configuration Converter dialog box	C

Table 5-2 (Cont.) Utilities Menu

Action	Keyboard Shortcut
In the Utilities menu, open the Deployment Tool dialog box	D
In the Utilities menu, open the Generate Reports dialog box	G
In the Utilities menu, open the Function Library Manager dialog box	F

Automation Menu

The menu items for Automation are enabled when the applications plug-ins are provided. See the application user guide for details.

Table 5-3 Help Menu

Action	Keyboard Shortcut
Open the Help menu	Alt + H
In the Help menu, open the Content dialog box	C
In the Help menu, open the About dialog box	A

Table 5-4 Measures Manager

Action	Keyboard Shortcut
Add a major measure component	Alt + Insert
Add a minor measure component	Insert
Move the currently selected measure component up in the navigation tree	Ctrl + P
Move the currently selected measure component down in the navigation tree	Ctrl + D
Show the currently highlighted measure component in the Measure Components tab	Space
Remove the currently selected measure component from the Measure Components tab	Space
Show all measure components of the solution in the Measure Components tab	Ctrl + S
Remove all measure components of the solution from the Measure Components tab	Ctrl + H
Realize native measures based on currently selected measure components	Ctrl + L
Unrealize native measures from the currently selected measure components	Ctrl + N
Find a component/measure/realized measure/ external measure within the content displayed in the Measure Components tabs	Ctrl + F
Undo	Ctrl + Z
Redo	Ctrl + Y

Table 5-5 Rules

Action	Keyboard
Edit the expressions of the selected rule	Ctrl + E
Undo a previous operation	Ctrl + Z
Redo a previous operation	Ctrl + Y

6

Configuring Dashboards in RPASCE

The Dashboard is a feature inroduced in RPASCE that is loaded when a user first logs into the system. Dashboards serve both as a home page to allow users to launch into the tasks they perform in the system as well as a place to provide summary information on defined metrics or exceptions as they exist in the application data schema. This allows the user to identify areas of the data that require attention, and it provides them with information to help them organize their tasks effectively.

Within the dashboard, a user has access to one or more dashboard profiles. Each profile provides information geared towards a particular role-based set of responsibilities of the user. For example, a financial planner may have profiles related to Pre-Season or In-Season activities. Additionally, administrative users have access to an Administrative profile in the dashboard that provides information on administrative tasks that have run in the system.

Within the system, every user's dashboard is backed by a workbook. This workbook serves as a data source for the information displayed in metric and exception profiles. However, the view components of the dashboard are different from traditional workbook views and must be defined in a resource external to the configuration that is housed in the RPASCE client app server tier. As a result, the process of Dashboard definition is a hybrid of traditional configuration activities and the creation and maintenance of this resource.

This chapter details the operations required to build a dashboard configuration. It describes the requirements of configuration within the Configuration Tools and details the process of creating the appropriate dashboard definition file to drive the view components of the RPASCE client.

Note:

In addition to configuring the dashboard, you must build the dashboard workbook for the dashboard to display when users log into the UI. Admin users have access to an OAT task that builds the dashboard workbooks for all users in the system. For the process for building the dashboard workbook, see *Schedule Dashboard Build* in the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide*.

Determining the Dashboard Requirements and the Approach

Many of the most important decisions involving a RPASCE Dashboard implementation occur before the configuration process begins. These decisions take place while you are gathering requirements and determining the functionality to be incorporated into a customer-specific implementation.

The purpose of this section is to provide guidance to help that process that includes both the overall functional goals of a dashboard (what is and what is not best practice dashboard functionality) and the impact of these decisions regarding the scope and level-of-detail on the performance and size of the dashboard workbooks.

Functional Considerations for Efficient Dashboard Design

The RPAS CE Dashboard presents a novel view of a user's data that is different than the views the user interacts with when working with taskflow-based segment workspaces. This view is designed to support data browsing; a user can locate individual data points through the dimension filters and can drill down into the numbers by selecting positions of differing levels within those filters.

These capabilities, along with the overall layout information, are presented within the dashboard (summary metrics coupled with supporting charts containing related metrics) Such capabilities can make the dashboard an appealing option for a large number of activities. When you are determining the requirements for a dashboard implementation, you may find yourself trying to include a great deal of information into the dashboard.

However, while the motivations behind such requests are valid, in actual practice, the use of the dashboard for these activities is both impractical and potentially counterproductive for various reasons.

Dashboard Interaction Pattern

The overall functional philosophy behind a dashboard is to provide a place where users can orient themselves when entering the application. The goals of an effective dashboard are to provide users with notifications (either explicitly or implicitly through viewing data) of conditions that require attention and to provide tools to help users determine how best to allocate their time within the application.

Dashboards do not support and are not intended to support the performance of actions; they are meant to quickly take users to the places where they should take actions. In the RPASCE platform, the only actions users can take on the dashboard are those relating to the creation and maintenance of segment workspaces.

As such, every minute users spend in the dashboard can realistically be considered a minute they spend not working on their plans, forecasts, and so on. If this time is spent creating an effective and efficient plan for what work the users need to perform in the application, then it is time well spent. However, an hour spent examining every hit of a low-level alert within the dashboard is an hour spent not actually working on resolving any of those alert hits and adjusting their plans to account for the performance of their business.

Data Partitioning and the Dashboard

The work users perform in the application is contained within segment workspaces. It is generally the case that an individual user is responsible for more information than is contained within any single workspace; the user divides her or his time between a number of segments, each containing a plan, forecast, and so on, that represents a portion of the information under their responsibility.

This is done not only for technical reasons but also to divide a user's full set of information into a number of manageable self-contained chunks. By reducing the overall set of data into these smaller workspaces, the application attempts to make it easier to focus on individual tasks and not flood the user with more information than that user can effectively process.

This partitioning of a user's data is not present in the dashboard. Within the single view of a dashboard profile, a user is interacting with the full scope of the data, limited only by position-level security settings. This presents the real problem of the application providing the user with more information than the user can effectively process, resulting in information overload.

An effective dashboard design that uses data aggregation counteracts the risk of information overload. While users can see the full breadth of their data, they interact with it at a summary level. They can inspect the overall performance of their plans at this summary level and then launch into individual segments where the lower-level data is once again partitioned into manageable portions.

System Responsiveness and Sizing

The technical implementation of the RPASCE Dashboard is designed to prioritize the following goals.

First and foremost is responsiveness; the goal is to make both the initial loading of the dashboard as well as the reaction of the dashboard to user actions as close to instantaneous as possible.

Second is familiarity of structure; information within the dashboard must be presented in a way that is compatible with how information is presented elsewhere in the application so that users do not need to translate what they are seeing in the dashboard into the information scheme within the workspaces where they work.

Last is ease of configuration; the goal is to simplify the process of creating a dashboard by utilizing existing processes where possible so that the configurator does not have to learn an entirely new process.

For these reasons, the decision was made to create a workbook for each user that would back the user's dashboard and serve the data required by that dashboard. Because of this decision, the RPAS CE Dashboard can meet the previously stated goals.

For the first goal, dashboard workbooks can be built in batch, allowing time when the user is offline to be traded for faster access of data when they are online using the dashboard.

For the second goal, the dashboard organizes information according to standard RPASCE indexing schemes; there is a one-to-one correspondence to the context users interact with in the dashboard and those used to build or find information in workbooks. It is even then possible to programmatically apply context in the form of automatic building of new workbooks or filtering existing ones when they are launched from the exception dashboard.

For the third goal, the creation of workbooks, measures, and rules is a common RPASCE configuration activity. As is detailed in the following sections, only the specification of custom-view related information required by the dashboard UI requires new process support.

These goals have motivated the creation of a dashboard workbook. However, as a tradeoff, the dashboard is affected by the same considerations regarding performance and sizing that affect traditional taskflow-based workbooks. As more and more information is included within the workbooks, their overall responsiveness declines and their storage requirements grow.

Because dashboards display all the data that is available to a user and not simply a single partitioned application’s slice, the growth can easily exceed the reasonable upper limits for workbook size. This problem is especially prevalent for users with admin rights who are not limited by position security or users whose role and responsibilities result in position rights for large portions of the overall application data.

Best Practices for Dashboard Configuration

Keeping the previous considerations in mind, note the following:

The overall pattern for dashboard implementations is to provide at-a-glance access to information about the health of the overall business.

The purpose of a dashboard is to help users orient themselves and plan out their work but not to perform their work.

The level of detail available in the dashboard must be balanced, considering not only the appropriate volume of information to allow the users to make good decisions without overloading them, but also the impact of the amount of information on overall system responsiveness and storage requirements.

Determining What Levels to Include in Dashboard Workbooks

The following guidelines can be thought of as best practices for the purpose of determining the scope of action and level-of-detail to include within a dashboard configuration.

Filter Level Determination

The first decision to make in order to determine the correct levels to include in a dashboard profile is what levels to make available in the filters that determine what data is displayed in tiles.

Because the purpose of the dashboard is to help users identify which segments require attention, the filters should contain the level at which a user's segments are defined. For example, if a profile contains in-season information and the tasks associated with in-season activities are defined by class (that is, class selections are made in the segment wizards for those tasks), then class should be available for filter selections.

Failure to include class in the previous example could result in a situation in which a user can see that there is a problem but fail to be able to identify which plans contained the problem. This can result in the necessity to open the segments one-by-one to find where each one needs to be.

Conversely, it is often counterproductive to include levels far lower than the level of task definition (class, in the previous example) within the filters. Top filters provide only a single data point for each combination of selections; this makes them a poor choice for examining lower-level data due to the need to manipulate the filter control and wait for the system to update for each selection.

Additionally, the controls for making filter selections are not capable of supporting a comprehensive display of low-level positions, necessitating a sequential position-by-position search when there are some options. While a filter approach works very well when a user is trying to locate a specific known position, it is a poor mechanism when the user does not know in advance which position has a problem.

For example, if a user opens the dashboard and sees that one of the plans is underperforming, the user may already know where the problem is. It may be the case that some style/color combination has been underperforming in previous weeks and so the user would most likely select that style/color in the filters to confirm that it was still the problem area.

However, if a user opens the dashboard and sees that a plan is underperforming this week when it had been performing to expectations previously, the user faces a different problem. Although the user knows that one or more style/color combination is underperforming, the user does not have the history to suggest which style/color has the problem. In this scenario, finding the style/color with the problem would be cumbersome; the user must select each in turn in the filters and examine it in isolation before loading the next, and so on.

In the second scenario, the use of the dashboard to inspect the data would be counter-productive. Although the data is present in the dashboard, it is faster and easier to locate the problems by launching into the user's segment where the pivot table allows a much more efficient side-by-side comparison of all the style/color combinations and where the search can be facilitated by a real time alert.

For this reason, the incorporation of low-level data into the dashboard is generally not as effective at providing concise and clear summary data in the dashboard and allowing users to transition into their segments to drill into the low-level data.

Note that the calendar dimension is an exception to the previous guidelines. Users often need to view their data as a series of weeks (weeks in current period, year-to-date, and so on.) Since this is how they interact with their data, it should be included within the charts of the dashboard even though it is potentially a much lower level than is used in segment definition.

In the case of the calendar, it is better to focus on setting reasonable limits on the length of time included in the dashboard than on the level-of-detail. Although users may see value in having several periods or even years of data available, this sort of historical analysis would be better performed in a task than in the dashboard.

Use of Seasonless Profiles

One note to consider when configuring an application in which segments are defined at a low level of detail is that, due to the cost of selection traversal mentioned previously, it may for these applications be better for users to make use of a seasonless profile and arrange to see the low level on the x-axis of a chart. Such an approach would allow a larger number of positions to be displayed side-by-side and provide a better mechanism for users to see the multiple segments' data at once for easy comparison.

The trade-off is that such an approach prevents users from viewing an individual segment's data displayed as a series of weekly values (the way charts in an in-season or pre-season profile display data). As a result, the need to easily process the low-level data as discrete values (to align with user segments) must be evaluated against the need to easily see the week-by-week performance for any single position to determine if such an approach is valid for any implementation.

Choosing to use a seasonless profile does not appreciably impact system responsiveness or dashboard sizing, so the decision can be primarily based upon the needs of the user. Is it more important to see multiple data points at once within the charts to make processing larger numbers of positions or is it more important to have a week-by-week breakdown of the performance of any given position available directly in the dashboard?

Dashboard Measure Definition

Once the set of levels that will be available within the dashboard is determined, an examination of the measures containing the metrics and exceptions to be displayed is warranted. The metrics are often defined with a low base-intersection (such as item/store/week) to support their use within the tasks defined for the application. However, when these measures are incorporated into a dashboard in which the available filter levels are higher than the measure's base intersection, the benefit of that low level-of-detail is unavailable to the user.

In such cases, a new measure should be defined with an intersection matching the filter levels so that the application measure can be aggregated at the time of the dashboard workbook build. For example, in a dashboard that supports sub-class/region/week, the item/store/week measure's data can be loaded into a sub-class/region/week measure defined within the workbook.

Performing this aggregation of the data will not impact the performance of the dashboard beyond this initial access after a build or refresh. (The first access of a dashboard's data can take longer if low-level data needs to be aggregated to the level of the filters.) At the same time, it has no drawback for the user, as the lower level-of-detail is not available in the dashboard, so its removal does not limit them. This aggregation does, however, result in smaller dashboard workbooks. This in turn means more storage is available to support the tasks and segments in which the users work.

Use of the Dashboard for Tasks

A final note should be made concerning the use case of trying to support activities and tasks directly within the dashboard. There may be a desire to support a user activity from the dashboard. These activities may take the form of a data analysis task or a report, and the dashboard may seem to be a likely location to perform them.

Sometimes this is the case because the dashboard seems to avoid one or more of the perceived limitations of the RPASCE system, such as the need to create multiple workbooks for low-level data due to application partitioning.

It is firmly the case that such an approach is not recommended. That is not to say that the goal is not valid or that the customer must use such approaches are not real. It is simply that the approach of using the dashboard to work-around the constraints imposed by RPASCE is flawed.

Although it can seem to be a distinct entity with different mechanics from traditional RPASCE workbooks, the dashboard is based on just such a workbook and, as such, cannot be used to overcome the limitations of workbooks such as the impact on workbook size and performance when large amounts of data are included in a single workbook.

Moving a task into the dashboard will not overcome system limitations; it will only circumvent protections such as application partitioning that have been put in place to protect customers and users from implementations that do not perform at scale. Attempts to circumvent such protections will result in implementations that are ultimately unsatisfactory and potentially unsustainable.

Summary

Previous sections have contained some suggestions regarding designing effective and efficient dashboards for RPASCE applications. This section provides a high-level summary of that guidance in a single location.

The dashboard interaction pattern better supports the display of the summary health-of-business information than it does low level data.

The goal of a dashboard implementation is to quickly and easily allow users to determine where to spend their time; the dashboard does not allow users to modify their data and so is not a good location for performing the tasks users need to perform.

Because the dashboard is backed by a workbook within the PDS, it is as sensitive to large position counts as other workbooks regarding performance and sizing.

The recommended lowest level for the top filters in a profile is the level at which the tasks performed from that dashboard use to define segments.

The inclusion of lower levels in the chart filters may seem to give great value; however, in practice, those filters are not an effective way to browse through low-level information. Consider including only one level lower than what is used in the top filters within the chart filters.

The previous statement does not necessarily apply to the calendar dimension. Users need to view their data as a time-series to represent year-to-date or current period performance and so week must be included within the dashboard. Consider instead limiting the total time horizon included in the dashboard to only the periods needed to orient the user and determine where to work in the application. Historical analysis is best performed in traditional workbooks.

In cases where data is stored in the system at a low level (for example, item_store_week), that data can be loaded into measures with a higher base intersection in the dashboard workbook. The lower level of detail cannot be viewed in the dashboard, and this aggregation can markedly decrease the size of the dashboard workbook, leaving more space available for the workbooks required by users' tasks.

Creation of a Dashboard Workbook Template

In order to support the RPASCE Dashboard, the solution implementor must define a workbook template within the application configuration. Because the view components of the dashboard are not driven by the configuration of worksheets, profiles, and other traditional view-related configuration points, most of the effort involves the specification of the measures and rules required for the workbook.

Measure and Rule Configuration Requirements

The dashboard workbook requires measures to support the data requests coming out of the dashboard UI and the tiles contained within it. The requirements for supporting a metric profile are simpler than those for supporting an exception profile; both are described later.

For most of the metric tiles, these measures may already exist within the application and must simply be included in the dashboard workbook. Some KPI metrics may require calculation based upon PDS data, as they are not persisted within the application but are instead calculated on demand. Finally, metric tiles backed by low intersection data may show better performance if a higher base intersection measure is defined within the workbook. In this way, aggregations can be performed one time at load or refresh and not multiple times in response to fetch requests from the UI.

Implementers must carefully consider the base intersection of measures when designing a dashboard workbook. The worksheet intersection for a metric profile must be the highest common base intersection level of all the measure used in the tiles of that profile. For example, a dashboard tile that compares Merch Planner working plan (week_scls_chn) to LY (week_scls_str) and Merch Manager Targets (year_dept_chn) as variances must be displayed at the year_dept_chn level. Measures whose base intersection is lower than that level are only available for display at that higher level in the dashboard profile. This may result in undesirable charts or results, depending on pre/inseason type. Therefore, multiple profiles at various base intersections may be desired.

Note that the dashboard does support workbook refresh in order to allow the tiles within the dashboard's profiles to update to reflect changes to the data in the PDS. For this reason, dashboard workbooks must be configured with both load and refresh rules, like task-based workbooks. Additionally, rules that calculate derived measures such as variances must still be incorporated into the calc group for the workbook even though the dashboard does not allow edits and propagation of edits through an explicit calc operation.

An exception profile has greater requirements in terms of measure definition for the dashboard workbook. In order to support the hit count functionality of an exception tile, every exception must be represented by a numeric measure with an aggregation type of total. These measures can then be populated as a part of the load or refresh process to provide a record of individual exception hits that can drive the summary information provided within the exception tiles.

The exception dashboard does not explicitly use either pre-RPASCE batch alerts or the real time alerts used within non-dashboard workbooks, but the exception dashboard equivalent of either of these is simple to configure.

Support for Real Time Alert Functionality

When desiring to provide summary information about an exception managed through a real time alert, a new rule can be created as follows:

Define a new integer-type measure to hold hits within the dashboard workbook. This measure must have the same intersection as the alert measure for the real time alert and must have an aggregation type of total.

Copy the alert rule used to evaluate the real time alert from its source workbook's calc group to the calc group of the dashboard workbook.

Update the left-hand side measure of this copied rule to be the newly defined integer-type measure.

Modify the alert expression so that the strings representing alert conditions in the alert rule become the integer value '1' and empty strings used to clear the real time alert hit become the integer value '0'.

Include the necessary load or calc rules to pull all right-hand side measures of the modified rule into the workbook.

To provide an example of this transformation: consider a hypothetical real time alert to calculate low stock. This alert evaluates against two conditions: zero units in inventory (the outOfStock condition) or inventory lower than a threshold value supplied by a secondary measure (the lowStock condition). This real time alert can be evaluated by the following alert rule:

LowStockAlert = if (InvU < 0, “outOfStock", if (InvU < Threshold, “lowStock", “"))

After defining a new integer measure LowStockHitCount for use in the dashboard workbook and copying the alert rule to the calc rule group of the dashboard workbook, the expression can be modified to be:

LowStockHitCount = if (InvU < 0, 1, if (InvU < Threshold, 1, 0))

The LowStockHitCount measure is now available to back an exception tile in the exception profile of the dashboard workbook. Its aggregation type of total allows it to provide the number of exception conditions within the scope of any level the user sets the dashboard, while the values at its base intersection make it possible to determine exactly where those conditions are.

Note that it is also necessary to include load rules for the InvU and Threshold measures in the load group of the dashboard workbook to support the new calculation.

Support for Batch Alert Functionality

Batch alert functionality does not exist in the RPASCE platform, as the exception dashboard replaces that functionality. However, the types of operations performed by batch alerts can be converted to exception dashboard functionality in a straightforward manner to support their use cases or - in the case of the migration of a pre-RPASCE application to the RPASCE platform – to migrate them to the new system.

1. Define a batch process to evaluate the exception. For migrating batch alerts, this may be as simple as running the batch alert rule or rule group as part of a scheduled batch job.
2. Include a rule within the load rule group of the dashboard workbook to pull the measure populated in batch into the dashboard workbook.
3. Define a new integer-type measure to hold alert hits within the dashboard workbook. This measure must have the same intersection as the measure evaluated in the batch job and an aggregation type of total.
4. Create a rule within the calc rule group of the dashboard workbook to populate the new hit count measure with a 1 for cells in the source measure with a value of true and 0 for cells in the source measure with a value of false.

Batch Alert Functionality

To give an example of this process, consider an exception represented by the Boolean measure BatchAlert that is evaluated by a scheduled batch process. A new integer-type measure named BatchHitCount is defined to back an exception tile in the dashboard workbook.

The following rule can then be included in the load rule group of the dashboard workbook:

```
BatchAlert = BatchAlert.master
A new rule can be defined in the calc rule group of the dashboard workbook:
BatchHitCount = if (BatchAlert, 1, 0)
```

The BatchHitCount measure is now available to back an exception tile in the exception profile of the dashboard workbook. Its aggregation type of total allows it to provide the number of exception conditions within the scope of any level the user sets the dashboard, while the values at its base intersection make it possible to determine exactly where those conditions are.

Note that the example assumes a desire to store alert hits with a Boolean value within the PDS to be used for other processes such as to back a pre-range mask or to be included in subsequent calculations. If is not necessary, then the exception can be evaluated as an integer **1** or **0** in the initial batch process and this measure can be used directly in the dashboard workbook.

Specifying the Wizards for the Dashboard Workbook

Dashboard workbooks are not managed directly by the users through segments but are instead created through an OAT task. As a result, individual users do not go through the wizard process for their dashboards.

However, for the OAT task that creates dashboard workbooks to correctly include the positions desired in the dashboard, some wizard configuration is required for the dashboard workbook template.

In general, it is not necessary to configure individual wizards. For most dimensions, such as Location, the dashboard must contain all positions in the PDS. For other dimensions, such as Product, the selections to be included in the dashboard must be limited by position security; those positions a user has access to must be included in the dashboard.

In both cases, the behavior desired is achieved in the absence of a wizard definition for the dimension. However, dashboard configurations that include Calendar must include a wizard definition for the Calendar dimension. Should the dashboard workbook template not include a wizard definition for Calendar, then the logic used to define included seasons in the Schedule Dashboard Build OAT task cannot be properly applied and users will not see the positions they are expecting.

In order to support all levels (year, half, and quarter) available in the Schedule Dashboard Build task, the level specified in the Calendar wizard page must be quarter or lower. Selecting a higher level will prevent the OAT task from making selections at any level lower than that specified in the wizard definition and could result in user dashboards that include unexpected positions.

Specifying the Formatting for Metrics in Dashboard Tiles

The dashboard does not contain a user formatting functionality such as is present in segment workspaces. As a result, the formatting used to display values for measures displayed in metric tiles must be set through the configuration.

The mechanism by which this formatting is set is the same mechanism used to configure default formatting information for measures within segment workspaces. Within the Style Tool of the RPAS CE Configuration Tools, a style can be created that specifies settings for formatting values such as precision, prefix, suffix, and so on.

Then that defined style can be associated with the metric being displayed in the dashboard. As with formats for traditional workspaces, this association can be done either within the Workbook Designer tool in the StyleORs panel for the dashboard view or it can be done a single time within the Measure Tool, in which case the format will be used wherever the measure is displayed (unless overridden in a specific context). The dashboard provides dynamic scaling of measure values. If dynamic scaling is used, the precision is 1.

Creation of the Dashboard View Configuration File

The specification of a view-related configuration for the RPASCE Dashboard is accomplished through the creation of a configuration file. This file is created outside of the RPASCE Configuration Tools and is deployed directly to the RPASCE Client application server. The contents of this configuration file are used by the RPASCE Client to determine how to organize and display the information configured into the dashboard workbook.

Dashboard configuration files follow the naming convention of prefixing the application name to the string `DashboardSettings.json`. For example, the dashboard configuration file for an mfprcs application instance would be `mfprcsDashboardSettings.json`.

Although a dashboard configuration file can be created from scratch, in most cases the simpler alternative is to modify the GA version of the file to incorporate any desired changes.

Dashboard Settings Configuration in the Deployment Tool

The Deployment Tool of the RPASCE Configuration Tools contains a view intended to support the creation and modification of the dashboard settings resource. This tool implements a familiar navigation/content method for defining the entities present in the resource and allows users to inspect and modify the properties of these entities through a set of UI controls such as text fields.

With the support of the Deployment Tool, users can expect to perform most if not all work in dashboard settings configuration through the new view. The following information is therefore only intended for users who wish to perform advanced operations not supported within the dashboard settings resource view.

For more information on the dashboard settings view, see the DashboardSettings.json View section.

Use of JSON in the Dashboard Settings Configuration File

The contents of the dashboard settings configuration file are formatted as a JSON (JavaScript Object Notation) object. JSON is a common flexible information encoding notation used frequently in cloud applications; it is more compact and, when properly formatted, more readable than the XML format. (Importantly, it is also not subject to some security concerns that are present when using XML for information encoding.)

JSON is a fairly simple and straightforward format; information about the specifics of the format is readily available online. Here is a brief summary of the format as it is used to create the dashboard settings file and a definition of some of the terms used in following sections.

JSON Structures

The JSON format is based upon two structures, the object and the array.

An object is a collection of name/value pairs. It is represented as a comma-separated set of name/value pairs enclosed in curly braces.

An array is a list of values. It is represented as a comma-separated list of values enclosed in square brackets.

A name/value pair consists of a string representing the name (contained within quotes), followed by a colon, followed by a value.

A value is one of a string (enclosed in quotes), a number, the Boolean values true or false, an object, and array or the special value null.

JSON Structures

This section contains a set of examples of JSON structures.

The first example has an object (denoted by curly braces) containing three key/value pairs (with keys: key1, key2, and key3). The first value is the string value1, the second value is the Boolean value false, and the third value is the number 3.

```
{key1:value1,key2:false,key3:3}
```

The second example is an array (denoted by square brackets). This array contains a list of the values used in the preceding example. Note the absence of the keys; the array contains only values:

```
[value1,false,3]
```

The final example involves an object containing a set of key/value pairs. One of the values of the key/value pairs is an array and another is a second object:

```
{key1:value1,key2:[0,1,2,3],key3:{key4:value4,key5:value5}}
```

The final example can be formatted into the more readable format:

```
{
               "key1":"value1",
                   "key2":[0,1,2,3],
                   "key3":{
                           "key4":"value4",
                       "key5":"value5"
              }
}
```

Use of JSON in the Dashboard Settings Configuration File

While the preceding is not an exhaustive description of the JSON format and its structure, it should suffice to explain the use of JSON in the remainder of this document and define the terms used in all following examples.

Top-Level Properties of the Dashboard Settings Configuration File

Most of the work involved in creating a dashboard settings configuration file involves specifying the information required to display the profiles of the dashboard and describe the tiles (metric or exception) contained within those profiles. However, a few of the properties defined within the dashboard settings file are not contained within a profile. The following properties must be defined at the highest level of the dashboard settings configuration file.

Table 6-1 Top-Level Properties of the Dashboard Settings Configuration File

Property	Value Type	Description
templateName	JSON string	The RPAS CE name of the workbook template used to build the dashboard workbook.
currentProfile	JSON string	The name of the profile to be displayed by default when the dashboard is loaded into the UI.

Table 6-1 (Cont.) Top-Level Properties of the Dashboard Settings Configuration File

Property	Value Type	Description
profiles	JSON array	A JSON array of one or JSON objects defining the profiles to be used in the dashboard. Consult following sections for information on profile objects.

Here is an example of a JSON object containing the previous properties to define the top-level information of a dashboard settings file. Note that the array containing profile information is empty in the example; information on the contents of the profile array and examples of profile configurations are contained in subsequent sections.

Top-Level Properties

The following example shows the top-level configuration of the dashboard used in the MFP application. In this example, the dashboard is backed by a workbook template named `mp_dp`. Note that this example does not contain any profile information, as profile specification is covered later.

```
{
           "templateName":"mp_db",
           "currentProfile":"ADMIN",
           "profiles":[]
}
```

Configuration of Dashboard Profiles

A dashboard profile defines a set of metrics that may be displayed within the dashboard. Each profile defines a set of common view settings that are shared by all metrics within the profile as well as a list of tiles representing individual metrics.

The Administration Profile

In order to allow administrative users to access the Task Status profile, a DashboardSettings file must contain an entry to describe the profile. This profile does not support modification through configuration and so must be included (if one is not present) unchanged.

Administration Profile

The Task Status profile, like all dashboard profiles, must be embedded within the profiles list described previously. An example of the description for the administrative profile follows:

```
{
      "name": "ADMIN",
      "label": "Administration",
      "type": "other",
      "viewName": "PlnDashVw",
      "roles": ["admin"],
      "tiles": [
        {
          "id": "admin-tasks",
          "type": "admin",
          "name": "AdminTasks",
          "title": "Admin Task Status",
          "description": "Admin task statuses",
          "inserted": true,
          "contentModuleBinding": {
            "name": "admin/adminTaskStatus"
          }
        }
      ]
    }]
```

Metric Dashboard Profiles

Metric profiles provide users with a set of tiles that display summary information about the performance of their business. These tiles can be used to display a key performance indicator or other meaningful metric value, along with other supporting metrics. The metric profile configuration defines which metrics will be available, what comparison metrics (such as variances) will be provided with a metric and how the metric information will be displayed in the detail charts.

Because dashboard profiles define what data is available to the users and how it is displayed, the bulk of time spent modifying a dashboard settings file is spent modifying existing profiles or adding new ones.

The RPASCE dashboard supports three types of metric profile:

- inseason
- preseason
- metric

Inseason and preseason profiles both incorporate the concept of seasons into the display and organize the data around the definition of seasonality given in the profile configuration. The RPASCE dashboard also supports a seasonless metric profile that allows more flexibility in how the data will be presented to the user, but it does not include support for seasonality.

The following list contains the properties shared by all types of metric profiles, with information on each:

- **name**–a unique identifier for the profile
- **label**–the display label identifying the profile to the user. The label is displayed within the profile selector to allow changing profiles and so should be brief.
- **type** – preseason, inseason, or metric. For inseason profiles, the season containing today will be context for displaying time in charts. For preseason profiles, the season coming after the current season will be the context. Metric, or seasonless, profiles do not organize information around the concept of seasons.
- **viewName**–the name of the view that is used by the dashboard for performing fetch operations on the dashboard workbook. This view must contain all measures that are used in tiles within its default profile and must have an intersection compatible with the lowest level available in the profile.
- **role**s–an array of roles that have access to this profile. Only users whose RPASCE usergroup is listed in the roles will see the profile. If this property is omitted, all users will have access to the profile. An administrative task is used to assign roles to profiles and dashboard settings.
- **chartLevel**–the default level to be used in the row axis of charts displayed in this profile.

- **filters**–an array of objects describing what dimensions must be available as filters in the dashboard. A filter object supports the properties:

Table 6-2 Filter Object Properties

Property Name	Property Type	Property Description
name	JSON string	The name of the dimension
width	JSON number (optional)	The desired width of the control in em, a logical measurement of width used in webpage layout.) By default, the control will attempt to resize to fit the text it contains but a specific size can be configured if this behavior is not desired.

- **rowDim, colDim, pagedDims–**a JSON object (or array of JSON objects in the case of pagedDims) that describe how to allocate dimensions in metric charts.

  Note that the current implementation of the dashboard requires that the MEAS dimension be assigned to the rowDim. Other assignments can cause errors when displaying the dashboard in the UI.

  All season profiles, where type is inseason or preseason, require the clnd dimension to be in the colDim. It is not an option to put any other dimension on the colDim for a season profile. Additionally, the chartLevel must be one of the levels specified in the colDim. Refer to the Dashboard Metric Profile JSON example in next several pages.

  Use the profile type 'metric' to arrange the dimensions the way you want that does not limit the clnd dimension to inseason or preseason. It displays what is in the workbook. It is still possible to use the filter to limit the clnd dimension.

  The properties of the dim object are:

Table 6-3 Dim Object Properties

Property Name	Property Type	Property Description
name	JSON string	The name of the dimension to be assigned
levels	JSON array of strings	The list of levels that can be used to display chart data
hierarchy	JSON string	The hierarchy (for dimensions with multiple hierarchies) to be used within the filters of the dashboard.
defaultLevel	JSON string (optional)	Sets a default level from the list in levels for display of data in the chart

- **tiles** - an array of metric tile objects (as described later) representing the metrics available in the profile.
- **title** - An optional a resource key for translation or the measures label will be used.
- **description** - An optional a resource key for translation or the measures label will be used.
- **unitOfMeasure** - An optional a resource key for translation for the unit of measure or nothing will display.

Additionally, seasonal profiles (inseason and preseason) have a set of properties to define how the profile should represent seasons to the user. These properties were originally intended to align with a yearly calendar, but it is now possible to display seasons that do not align evenly with years or to include more than a year within a season.

For example, the settings:

- seasonLevel: year
- numPosInSeason: 2

will result in a season containing two calendar years. For such configurations, logs will include a warning because the validation preformed based on numSeasonInYear is violated. (This is to facilitate finding problems when a season that is aligned with years is desired.) However, the dashboard will build and display data as expected for the settings.

Note that when a seasonal profile is being configured, the seasonLevel, numPosInSeason and numSeasonInYear properties must all be set. If seasonality settings are not required, then a seasonless Metric profile can be used instead of a seasonal profile.

- **seasonLevel** - the level in the calendar dimension used to define a season in conjunction with the numPosInSeason and numSeasonInYear attributes (for example, week).
- **numPosInSeason** - the number of positions in the seasonLevel that make up a season (for example, 26).
- **numSeasonInYear** - the number of seasons in a calendar year (for example, 2).

Note:

In all cases where a reference is made to a component of the application configuration (for example, a measure, dimension, or level), the name of the component must display in the `DashboardSettings.json` file in lower case. Use of upper case or mixed case can result in errors when the RPASCE client tries to use the component.

Metric Profile

Here is an example of a metric profile as it displays in the dashboard settings file. Note that this example does not contain any tile definitions; they are described later:

```
"profiles": [{
      "name": "MFP-IN",
      "label": "MFP: In-season",
      "type": "inseason",
      "viewName": "PlnDashVw",
      "seasonLevel": "week",
      "numPosInSeason": 26,
      "numSeasonsInYear": 2,
      "chartLevel": "week",
      "filters": [{"name": "clnd"}, {"name": "loc"}, {"name": "prod",
"width":24}],
      "rowDim": {
        "name": "meas",
        "levels": [
          "info"
        ],
        "defaultLevel": "info"
      },
      "colDim": {
        "name": "clnd",
        "levels": [
          "half",
          "qrtr",
          "mnth",
          "week"
        ],
        "defaultLevel": "mnth"
      },
      "pagedDims": [
        {
          "name": "prod",
          "levels": [
            "dept",
            "clss",
            "scls"
          ]
        },
        {
          "name": "loc",
          "levels": [
            "chan",
            "chnl"
          ]
        }
      ],
      "tiles": []
}]
```

The Metric Profile example above defines the In Season profile used by the MFP application. As such, it is defined as an inseason profile (using the type property). The single view PlnDashVw defined within the dashboard workbook is set as the viewName.

This profile defines seasons based upon the week level (seasonLevel) by specifying periods of 26 weeks (numPosInSeason) with two periods a year (numSeasonsInYear). Note that additional periods will be added to the final season of the year; this allows, for example, a 53-week year. Additionally, the default level of CLND that must be used in charts is specified as week (chartLevel).

The set of filters available in the dashboard is given, in this case CLND, PROD and LOC. The width property has been used in conjunction with the PROD dimension to increase the width from the default of twenty to twenty-four (note the absence of quotes around the value; it is a number and so is not quoted the way a string would be). This should result in less truncation for longer position labels.

A common layout of dimensions is used by all metric detail charts contained within a profile. In this example, CLND is represented on the colDim, PROD and LOC are pagedDims. In each case, the name of the dimension, a list of available levels, and a default level are provided.

Finally, the tiles property, in a fully specified profile, would contain an array of tile objects representing the tiles available in that profile.

Metric Tile Configuration

Each tile displayed in a metric dashboard profile is defined by an entry in the tiles list of the profile. The dashboard provides support for single metric tiles that display a single value with an accompanying chart as well as metric tiles and chart that provide a comparison between a metric and one or more supporting metrics. The metric dashboard profile supports two types of metric tiles: the variance tile and the comparison tile.

Variance Tiles

Variance tiles provide additional information represented by a variance measurement in the tile and display of the metrics used to calculate the variance within the chart. When this is the case, the use of configured formats in the dashboard workbook template can allow for variance formatting (for example, a scale factor of 100 and suffix of '%').

When displaying a comparison value in a metric tile, the dashboard will make use of the first word in the measure label of comparison metrics. This string can be modified for the dashboard by including an edited measure entry for the comparison metric that modifies the label property within the dashboard workbook.

There are two different methods for defining the thresholds that will trigger the tile and variance to be displayed as either medium or high severity. Use of the mediumThresholds and highThresholds within the variance object allows a single set of threshold values to be used for both supporting values but only supports alerting on negative values. This approach has been deprecated and may be removed in a future release.

Alternatively, the left and right objects within the variance can make use of the severity, comparison, and value properties of the thresholds object that provide a more flexible definition of thresholds and can be assigned different values for the left and right variances.

Metric tiles support the following properties:

- **measure** - the name of the measure that is the primary metric displayed in the tile.
- **inserted** - true or false (as a Boolean value, quotes are not required for the value). Tiles with a value of true will be displayed by default in the profile. Tiles with a value of false will initially be hidden but can be displayed by the user.
- **id** - a string that serves as a unique identifier for the tile. This property can be used when more than one metric tile contains information for a single measure (for example, the same base value with different comparison metrics in each tile). This property is optional; if a value is not provided, the value for the measure will be used. However, when multiple tiles use a single measure, the UI may not be able to provide the correct information for things such as tile description.
- **description** - a short description of the information provided by the tile that can be displayed in the tile info window.
- **variance** - an object defining either one or two comparison metrics for the primary metric. The comparisons are defined as either left or right, and a tile can contain both, either, or none of these variances. In cases where there are no comparison metrics, the variance property can be omitted. Variance objects support the following properties listed in Table 6-4.

Table 6-4 Variance Objects Properties

Property Name	Property Type	Property Description
mediumThreshold	JSON number	A threshold value that, if exceeded, will cause a comparison to be displayed as a warning. This is a deprecated property. It has been replaced by the threshold object used in the left and right objects for more flexible severity threshold control. The property field can be blank if the variance contains left and/or right objects which have thresholds defined, and thus the threshold definition at the variance level is not necessary.
highThreshold	JSON number	A threshold value that, if exceeded, will cause a comparison to be displayed as a warning. This is a deprecated property. It has been replaced by the threshold object used in the left and right objects for more flexible severity threshold control. The property field can be blank if the variance contains left and/or right objects which have thresholds defined, and thus the threshold definition at the variance level is not necessary.
left	JSON object	See subsequent table for properties.
right	JSON object	See subsequent table for properties.

The values of left and right are themselves objects with the following properties listed in Table 6-6.

Table 6-5 Left and Right Objects Properties

Property Name	Property Type	Property Description
measure	JSON string	The name of the measure whose value should be displayed in the tile.
chartMeasure	JSON string	The name of the measure whose values should be displayed in the detail chart for this tile.
thresholds	Array of JSON object	One or more objects providing an alternative method for providing threshold information for the supporting value.

The thresholds object contains the following properties listed in Table 6-6.

Table 6-6 Thresholds Object Properties

Property Name	Property Type	Property Description
severity	JSON string	Either high or medium.
comparison	JSON string	One of the following comparison types: >, >=, <, <=, =, !=
value	JSON number	The value to use in evaluating the threshold.

Threshold Object

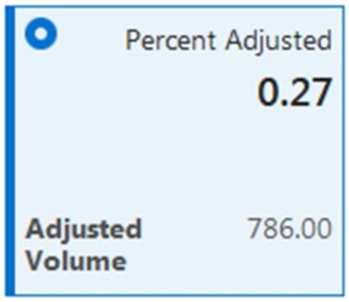
Here is an example of a threshold as it would display in the settings file.

```
"thresholds": [{
    "severity": "high",
    "comparison": "<",
    "value": 0.0
  }, {
    "severity": "medium",
    "comparison": "<",
    "value": 0.1
  }]
```

Comparison Tiles

Comparison Tiles provide an alterative method for displaying information about a metric. Like variance tiles, comparison tiles support the definition of a main metric and some secondary metrics, both within the tile and within the associated detail chart. Unlike variance tiles, the supporting metrics are not assumed to be variances and so the tile does not support evaluating thresholds. However, the comparison tile does allow the additional flexibility of support for dual-y charting, in which supporting charts can define two different scales for the y-axis.

Figure 6-1 Comparison Tile with One Sub-Measure Entry



Comparison Tiles support the following properties:

- **measure** - the name of the measure that is the primary metric displayed in the tile.
- **inserted** - true or false (as a Boolean value, quotes are not required for the value). Tiles with a value of true will be displayed by default in the profile. Tiles with a value of false will initially be hidden but can be displayed by the user.
- **id** - a string that serves as a unique identifier for the tile. This property can be used when more than one metric tile contains information for a single measure (for example, the same base value with different comparison metrics in each tile). This property is optional; if a value is not provided, the value for the measure will be used. However, when multiple tiles use a single measure, the UI may not be able to provide the correct information for things such as tile description.
- **description** - a short description of the information provided by the tile that can be displayed in the tile info window.
- **subMeasures** – an optional list of measures to be displayed underneath the main metric within the tile. Space on the tile constrains the maximum number of subMeasures to three.

- **contentMeasures** – an optional list of measures to be used within the supporting chart for this tile. If no value is given, then the measures used in the tile will be displayed within the chart.
- **chartType** – the type of chart to display for this metric tile. Supported values are bar and line
- **subMeasureLabels** – Options array of resourced labels instead of the measure labels. This must match the subMeasures array.
- **contentTitle** – An optional title to display in the content area when the tile is selected. If not specified, the title of the tile is used.
- **yAxisLabel** – An optional y-axis label in content area, uses resource bundle.
- **sortMeasures** – an optional list of measures that can be used to sort the information displayed within the chart. The contents of the sortMeasures array are a list of JSON objects with the following properties listed in Table 6-7.

Table 6-7 sortMeasures Array Properties

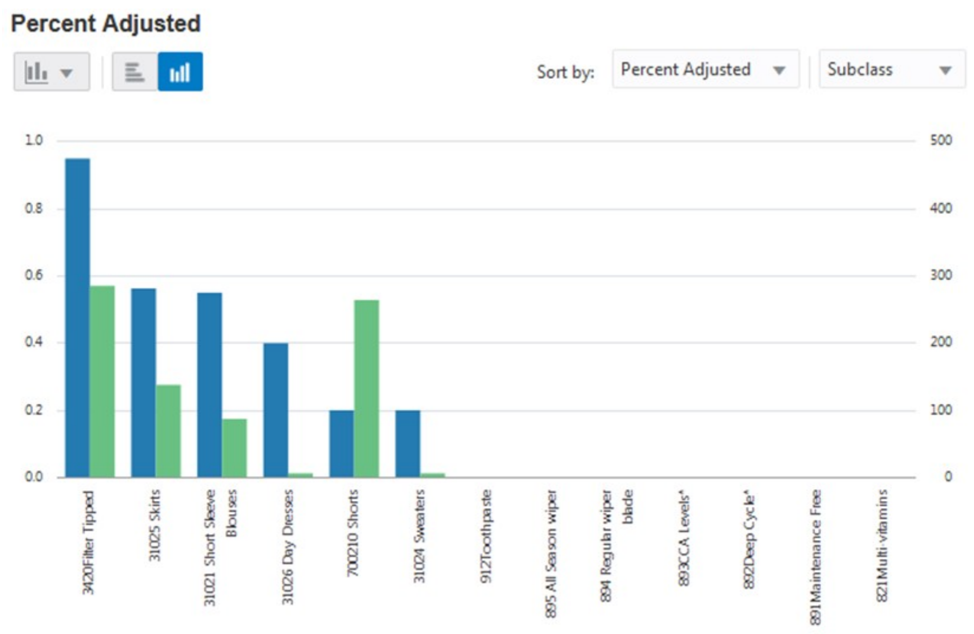
Property Name	Property Type	Property Description
measure	JSON string	The name of the measure that is used to order the chart.
ascending	Boolean	True or false, depending on whether the ordering should be ascending (true) or descending (false).
yAxisLabel	JSON string	An optional y-axis label in content area, uses resource bundle.

- **dualY** - a JSON object used to support the use of dual-Y axis scales. By specifying the dualY property, it is possible to display a secondary set of values with independent scaling, either within the same chart as other measures or in a secondary chart that displays underneath the primary chart.

Table 6-8 dualY Properties

Property Name	Property Type	Property Description
measures	JSON string	A list of measures to display in a splitY chart.
splitDualY	JSON string	Either on or off. When on, the dualY measures display in s separate chart. When off, a single chart with two y-axis scales displays.
splitterPosition	JSON measure	When splitDualY is set to on, this value provides the ratio of space on the y axis to be given to each of the displayed charts. Valid values are 0 to 1.
yAxisLabel	JSON string	An optional y-axis label in content area, uses resource bundle.

Figure 6-2 Dual-Y Axis Displaying Both Percentage and Integer Values



Example Metric Tile Configuration

For example, consider a tile displaying sales information. A comparison of working plan sales to current plan sales and to last year's sales might involve the measures:

- WpSlsR – working plan sales
- CpSlsR – current plan sales
- LySlsR – last year sales
- WpSlsRvC – variance of working plan sales to current plan sales
- WpSlsRvL – variance of working plan sales to last year sales

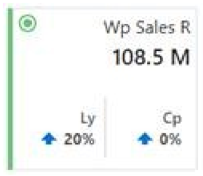
In the metric tile, the working plan sales is the primary metric and the two variance metrics WpSlsRvC and WpSlsRvLforms the left and right variance values displayed in the tile. In the detail chart, however, the values of the base metrics CpSlsR and LySlsR are used to show the details.

In order to provide the example, a metric tile is defined within the profile as:

Metric Tile

```
"tiles" : [{
          "measure": "mpwpslsr",
          "inserted": true,
           "id": "slsvlycp",
          "description": "Variance of Sales to Last Year and Current Plan",
          "variance": {
            "mediumThreshold": 1.0,
            "highThreshold": 2.0,
            "left": {
              "measure": "mpwpslsrvl",
              "plan": "Ly",
              "chartMeasure": "mplyslsr",
               "thresholds": [{
                  "severity": "high",
                  "comparison": "<",
                  "value": -0.2
                  }, {
                  "severity": "medium",
                  "comparison": "<",
                ""value": -0.1
                }]
            },
            "right": {
              "measure": "mpwpslsrvc"
              "plan": "Cp",
              "chartMeasure": "mpcpslsr"
            }
          }
        }]
```

Figure 6-3   Metric Tile



This details the properties of a metric tile that displays information on sales. The primary metric is `mpwpslsr` (making use of the measure property). This tile is by default shown in the profile (inserted is set to true); if inserted had been set to false, the tile would initially be hidden but would be available within the Edit Dashboard window.

The tile is intended to be a comparison tile and so contains a variance specification. That variance specification sets both a mediumThreshold and a highThreshold (1.0 and 2.0, respectively). Should the left or right comparison measure's values exceed these values, the tile will be marked as either a warning (for medium) or error (for high).

The left and right properties provide information about the two comparison values in the tile. In the left, the measure mpwpslsrvl contains the value to be displayed in the tile and the mplyslsr measure contains the values that will be used in the chart shown when the tile is selected. In the right value, the measure mpwpslsrvc contains the value for the tile while mpcpslsr will be displayed within the chart.

Exception Dashboard Profiles

Like a metric dashboard profile, an exception dashboard profile contains some properties that define how the information present in that profile will be displayed in the RPASCE client. Exception dashboard profiles share the same set of properties as metric dashboard profiles. However, some the values of some properties affect the display of the information in the dashboard differently in exception profiles.

Here is a listing of the properties of an exception profile:

- **name** - a unique identifier for the profile.

- **label** - the display label identifying the profile to the user. The label is displayed within the profile selector to allow changing profiles and so should be brief.
- **type** - exception profiles must use value exceptions for the profile type. Exception profiles contain both in-season and pre-season time frames and so it is unnecessary to distinguish between them.
- **viewName**- the name of the view that is used by the dashboard for performing fetch operations on the dashboard workbook. This view must contain all measures that are used in tiles within its default profile and must have an intersection compatible with the lowest level available in the profile.
- **role**s - an array of roles that have access to this profile. Only users whose RPASCE usergroup is listed in the roles will see the profile. If this property is omitted, all users will have access to the profile. An administrative task is used to assign roles to profiles and dashboard settings.
- **seasonLevel** - the level in the calendar dimension used to define a season in conjunction with the numPosInSeason and numSeasonInYear attributes (for example, week)
- **numPosInSeason** - the number of positions in the seasonLevel that make up a season (for example, 26).
- **numSeasonInYear** - the number of seasons in a calendar year (for example, 2).
- **season** – the seasonality timeframe that this Exception dashboard profile can show. This season property can be set to either inseason, preseason, or both. If none is specified, then this property defaults to inseason.
- **filters** - an array of objects describing what dimensions must be available as filters in the dashboard. A filter object supports the properties:

Table 6-9 Filter Object Properties

Property Name	Property Type	Property Description
name	JSON string	The name of the dimension
Width	JSON number (optional)	The desired width of the control in em, a logical measurement of width used in webpage layout (the em dash). Increasing the value will prevent clipping off the end of labels of positions displayed in the filter control. The default values for width are 12 for the CLND dimension and 20 for all others.

- **rowDim, colDim, pagedDims** - a JSON object (or array of JSON objects in the case of pagedDims) that describe how to allocate dimensions in exception charts. Individual exception tiles may specify dimension assignments separately through the topDim and byDim exception tile properties. If a tile does not provide assignments, the default assignments of the profile will be used. The properties of the dim object are:

Table 6-10 Dim Object Properties

Property Name	Property Type	Property Description
name	JSON string	The name of the dimension to be assigned

Table 6-10 (Cont.) Dim Object Properties

Property Name	Property Type	Property Description
levels	JSON array of strings	The list of levels that can be used for displaying chart data
hierarchy	JSON string	The hierarchy (for dimensions with multiple hierarchies) to be used within the filters of the dashboard.
defaultLevel	JSON string (optional)	Sets a default level from the list in levels for display of data in the chart

- **tiles** - an array of exception tile objects (as described later) representing the exceptions available in the profile.

Exception Profile in the Dashboard Settings File

Here is an example of an exception profile as it displays in the dashboard settings file. Note that this example does not contain any tile definitions; exception tiles are described later.

```
{
      "name": "RTA",
      "label": "IP: Exceptions",
      "type": "exceptions",
      "viewName": "alerts",
      "numPosInSeason": 26,
      "numSeasonsInYear": 2,
      "filters": [{"name": "clnd"}, {"name": "loc"}, {"name": "prod"}],
      "rowDim": {
        "name": "prod",
        "levels": [
            "cmpp",
            "dvsn",
            "pgrp",
            "dept",
            "clss",
            "scls"
        ],
        "defaultLevel": "scls"
      },
      "colDim": {
        "name": "loc",
        "levels": [
            "comp",
            "chan",
            "chnl",
            "regn",
            "dstr",
            "stor"
        ],
        "defaultLevel": "stor"
      },
      "pagedDims": [
        {
          "name": "meas",
          "levels": [
            "info"
          ]
        },
        {
          "name": "clnd",
          "levels": [
            "year",
            "half",
            "qrtr",
            "mnth",
            "week"
          ],
          "defaultLevel": "year"
        }
      ],
      "tiles": []
}
```

This defines the exception profile used by the Item Planning (IP) application. As such, it is defined as an exceptions profile (using the type property). The view alerts defined within the dashboard workbook is set as the viewName.

This profile defines seasons based upon the week level (seasonLevel) by specifying periods of 26 weeks (numPosInSeason) with two periods a year (numSeasonsInYear). Note that additional periods will be added to the final season of the year; this allows, to give an example, for a 53-week year.

The set of filters available in the dashboard is given, in this case, CLND, PROD and LOC.

A default layout of dimensions will be used by all exception detail charts that do not define individual assignments. In this example, PROD is represented on the rowDim, LOC on the colDim and MEAS and CLND are assigned to the pagedDims. In each case, the name of the dimension, a list of available levels, and a default level are provided.

Exception Tile Configuration

Each tile displayed in an exception dashboard profile is defined by an entry in the tiles list of the profile. The dashboard provides support for exceptions to break down exceptions to provide global and local exception information. Exception dashboard tiles also provide the ability to launch directly into the appropriate task and segment that supports resolving the exception represented by the tile.

Exception tiles support the following properties:

- **id** - a string containing an identifier for the tile. This name must be unique across all tiles in the profile.
- **hitMeasure** - a numeric measure in the profile's view that contains information on exception hits.
- **inserted** - true or false (as a Boolean value, quotes are not required for the value). Tiles with the value true are displayed by default in the profile. Tiles with false are initially hidden but can be shown by the user.
- **topDim** - a string containing the name of a dimension to use as the outer grouping of values displayed in exception detail charts.
- **byDim** - a string containing the name of a dimension to use as the inner grouping of values displayed in exception detail charts.
- **byMeasure** - a string containing the name of a measure whose values are used to sort the information provided in exception detail charts.

- **sortMeasures** - an array of JSON objects defining measures that may be used to sort the information provided in exception detail charts. The sortMeasures property is an alternative to the byMeasure property, providing the ability to specify multiple options as well as to sort in either an ascending or descending order. The following properties are supported for a sortMeasure entry:

Table 6-11 sortMeasure Entry Properties

Property Name	Property Type	Property Description
measure	JSON string	The name of the measure that is used to order the chart.
ascending	Boolean	True or false, depending on whether the ordering should be ascending (true) or descending (false).

- **topCount** - a number defining the number of charts to be displayed for topDim in the exception detail pane. This property is optional; the default value is 5.
- **byCount** - a number defining the number of charts to be displayed for byDim in the exception detail pane. This property is optional; the default value is 10.
- **openTemplate** - a string containing the name of the workbook template that must be used for contextual launch for this exception. The value of the RPAS CE name of the workbook template as configured in the workbook tool is the expected value.
- **openAlertName** - a string containing the name of a real time alert in the openTemplate that must be used for contextual launch for this exception.
- **openTo** - a string containing the name of the step that must be used for contextual launch for this exception. The expected value is the fully qualified step identifier as it displays in the taskflow resources (for example, MFP.Activity1.Activity1.Task2.Step3). This value is optional and may be omitted if the openAlertName property is specified.
- **dualY** – a JSON object used to support the use of dual-Y axis scaling. By specifying the dualY property, it is possible to display a secondary set of values with independent scaling, either within the same chart as the exception hit counts or in a secondary chart displays underneath the primary chart. The following set of properties are supported by the dualY object:

Table 6-12 dualY Object Properties

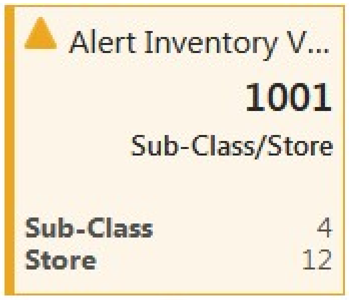
Property Name	Property Type	Property Description
measure	JSON string	A list of measures to display in a splitY chart.
splitDualY	JSON string	On or off. When on, the dualY measures displays in s separate chart; when off, a single chart with two y-axis scales displays.
splitterPosition	JSON number	When splitDualY is set to on, this value provides the ratio of space on the y axis to be given to each of the displayed charts. Valid values are 0 to 1.

Example Dashboard Tile in the Dashboard Settings

```
"tiles": [
        {
            "id": "EOPAlert",
            "inserted": true,            "hitMeasure": "isalinvvldct",
            "topCount": 5,
            "topDim": "prod",
            "byDim": "loc",
            "openTemplate": "IP_IE",
            "openAlertName": "EOPAlert"
        }]
```

This displays as shown by the following tile (Figure 6-4) and charts (Figure 6-5) in the dashboard.

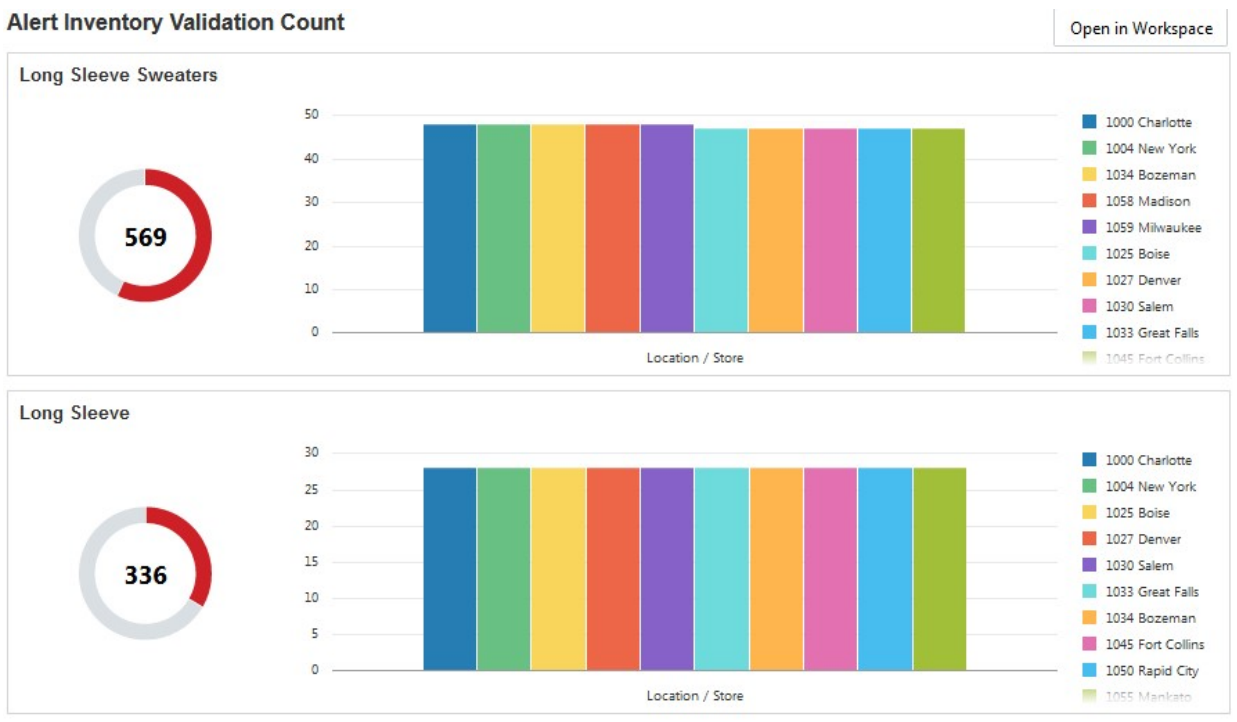
Figure 6-4   Exception Dashboard Tile in the Dashboard View



Alert Inventory V...
1001
Sub-Class/Store

Sub-Class	4
Store	12

Figure 6-5   Alert Inventory Validation Count Charts



This defines an exception tile that displays an inventory alert. It is identified as EOPAlert and is set to be visible by default in the dashboard profile (by setting inserted to true).The tile makes use of the `isalinvvldct` measure to hold exception hits; this measure has values calculated through the load, refresh, and calc rules of the dashboard workbook. The tile is configured to display five charts for the product dimension (by specifying a topCount of 5 and topDim of prod) with an inner nesting of up to ten charts for the location dimension (specifying byDim as loc and making use of the default value for byCount, which is 10). For purposes of contextual launch, the alert tile is configured to launch into segments for the IP_IE workbook template and transition to the EOPAlert defined in that workbook to allow exception resolution.

Configuring Attribute Roll-Up Dynamic Hierarchies

For applications that support more attribute-driven processes, it has become common to make use of dynamic hierarchies to create hierarchical roll-ups based upon the attribute values of positions. This allows users to view data organized not only by the standard product organization schema of styles, subclasses, and so on, but also grouped based upon the attributes of the products (for example, all styles with a certain brand or material). However, in some cases, there is a need for this attribute-based dynamic hierarchy to be available within the dashboard as well as in taskflow workspaces.

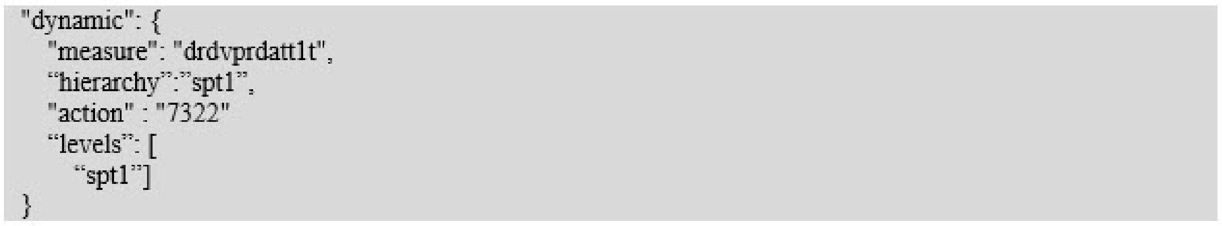
In order to support attribute roll-ups in the dashboard, the dashboard workbook must be configured to perform the creation of the dynamic hierarchy. This is accomplished using the same mechanism that allows the operation in taskflow workbooks. A custom menu is defined that, based upon a user selection, populates appropriate data within a set of mapping measures, commits those measures to the PDS, and then makes use of the dynhierrefresh function to use the committed mapping data to update the hierarchies of the workbook.

Additionally, the `dashboardSettings.json` file that defines the view-related aspects of the dashboard must be updated. Every profile in which the attribute roll-up is to be available must contain information about the measures and levels used in the process so that it can allow users to specify the desired attribute and incorporate the refreshed hierarchies into the filter set used by the profile. This specification is done through a new dynamic property, which is represented as a JSON object with the following properties:

- **measure**–the name of the measure that is meant to hold the user's selection of the attribute to be used in the attribute roll-up.
- **hierarchy**–the hierarchy that contains the attribute roll-up. The dashboard only supports the use of a single hierarchy (for dimensions with multiple hierarchies) within the chart filters.
- **levels**–a JSON array of the levels present within the hierarchy created by the dynamic hierarchy operation.
- **defaultLevel**– the default level to display in the filters of the dashboard when the dynamic hierarchy is active.
- **action**–the menuId property of the menu item that performs the dynamic hierarchy operation in the workbook. This value is accessible in both the tmpl.cfg file generated for the dashboard workbook as well as in the .XML file that contains the configuration of the dashboard workbook.

Here is an example of the dynamic property from a dashboard profile:

Figure 6-6   Dynamic Property from a Dashboard Profile



```
"dynamic": {
   "measure": "drdvprdatt1t",
   “hierarchy”:”spt1”,
   "action" : "7322"
   “levels”: [
      “spt1”]
}
```

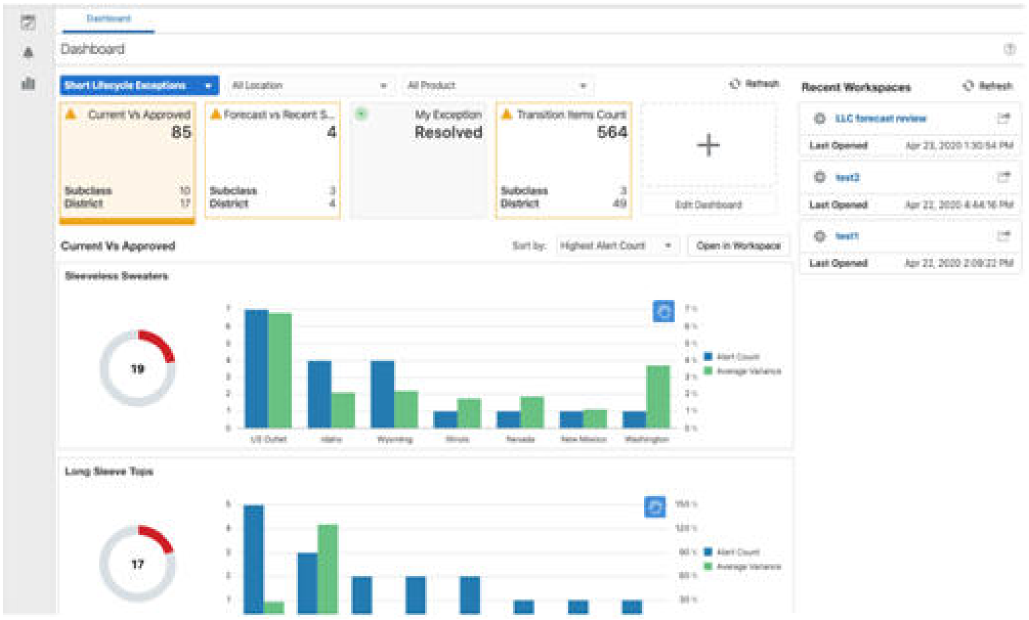
In this example, the rules of the dashboard workbook use the measure `drdvprdattlt` to hold the user's specification of the attribute to use in the attribute roll-up. The level created by the dynamic hierarchy operation is named spt1, as is the hierarchy that displays within the UI to identify the roll-up hierarchy. Finally, the custom menu item defined within the dashboard workbook that performs the dynamic hierarchy operation has the menuId 7322.

Configuring the Unit of Measurement Display in Dashboards

RPASCE allows users to control the unit of measure display in the Dashboard. The unit of measure display formatting can be applied to major Dashboard profile tiles (Metric tile, Comparison Metric tile or Exception tile), sort measures, or dual y-axis.

Here is an example of the unit of measure display used for the dual y-axis in RPASCE:

Figure 6-7   Unit of Measure Display for Second Y-Axis in Dual Y-Axis



The charts in Figure 6-7 support the dual y-axis. The first y-axis on the left side represents the number of exceptions in a subclass. The second y-axis on the right side displays the variance that is a quantitative measure of how much it is for certain exceptions.

If the exception is triggered when the sales are less than, for example, $10, then the following scenarios can happen:

- Scenario 1. Sales are 11 or higher: no exception
- Scenario 2. Sales are 9: exception is triggered
- Scenario 3. Sales are 0: exception is triggered

Scenarios 2 and 3 both trigger an exception, but users will probably want to address scenario 3 first because it differs from the target to a greater degree than scenario 2. The unit of measure display for the second y-axis can be formatted to represent the variance in a more intuitive way. The variance can be measured by using either a percentage that indicates how much it differs from the target, or currency, or a customized format in which users control the specific formatting specification for the second y-axis. Such specification is done through a new yAxisFormat property, which is represented as a json object with the following properties:

- type - yAxisFormat supports three types: percent, currency, and custom. The percent type displays the '%' character after the actual value in the Dashboard. In the Dashboard Deployment Tool, the user selects a type via the drop-down list next to the yAxis Format type label.
- currency - the label of the currency, such as USD. It is an optional property to be used, together with the yAxisFormat type currency.
- prefix - this property indicates the unit of measure that precedes the actual value. For example, if one Y-Axis value is $42, then the prefix should be "$".
- suffix - this property indicates the unit of measure after the actual value. For example, if one y-axis value is 25%, then the suffix must be %.
- numDecimalPlaces - the property indicates the number of decimal places and must be an integer.
- scaleFactor - this property indicates the scale factor and is a float.

These four properties, prefix, suffix, numDecimalPlaces, and scaleFactor, are only used for the custom yAxisFormat type. Here are short examples of yAxisFormat Json Objects:

Type Percent

```
"yAxisFormat": {
  "type": "percent"
}
```

Type Currency

```
"yAxisFormat": {
  "type": "currency",
  "options": {
    "currency": "USD"
  }
}
```

Type Custom

```
"yAxisFormat": {
  "type": "custom",
  "options": {
    "prefix": "$",
    "numDecimalPlaces": 2,
"suffix": "USD"
```

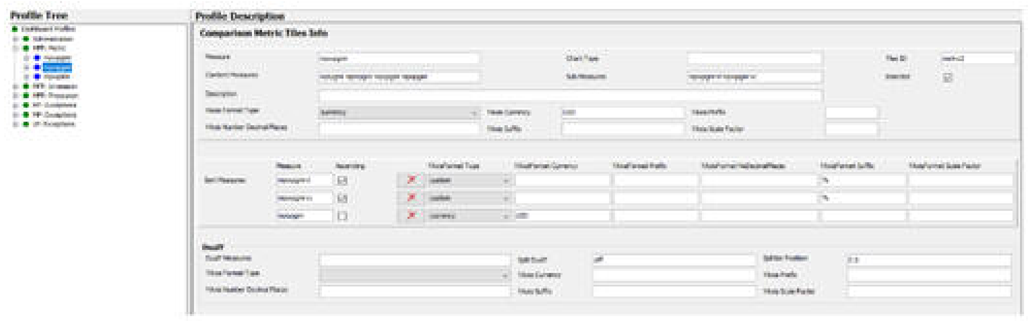
```
  }
}
```

When the YAxisFormat is used within any major Dashboard Profile Tile section, it is used to format that Tile chart.

When the YAxisFormat is used within the Sort Measures section, each sort measure can have its own yAxisFormat. When that sort measure is selected from the Sort By drop-down list on RPASCE UI, its YAxisFormat will be applied to the second y-axis for that sort measure.

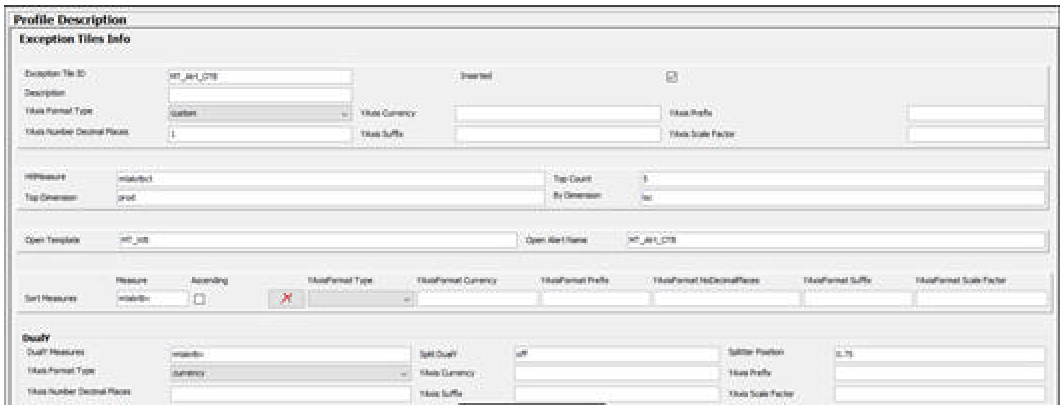
When the YAxisFormat is used within the dual-y section, it will be applied to the second y-axis format for all dualY measures, as in the example above.

Figure 6-8   Example 1: Used in the Comparison Metric Tile and Sort Measures



```
{
"id": "metric2",
"measure": "mpwpgmr",
"inserted": true,
"yAxisFormat": {"type": "currency", "options": {"currency": "USD"}},
"subMeasures": [
"mpwpgmrvl", "mpwpgmrvc"
],
"subMeasureLabels": [
"Ly", "Cp"
],
"contentMeasures": [
"mplygmr", "mpwpgmr", "mpcpgmr", "mpopgmr"
],
"sortMeasures": [
{
"measure": "mpwpgmrvl",
"yAxisFormat": {"type": "custom","options": {"suffix": "%"}},
"ascending": true
},
{
"measure": "mpwpgmrvc",
"yAxisFormat": {"type": "custom","options": {"suffix": "%"}},
"ascending": true
},
{
"measure": "mpopgmr",
"yAxisFormat": {"type": "currency","options": {"currency": "USD"}},
"ascending": false
}
],
"dualY": {
"splitDualY": "off"
}
},
```

Figure 6-9 Example 2: Used in Exception Tile and Dual Y



Profile Description

Exception Tiles Info

Exception Tile ID	MT_Alrt_OTB	Inserted	☑		
Description					
YAxis Format Type	custom	YAxis Currency		YAxis Prefix	
YAxis Number Decimal Places	1	YAxis Suffix		YAxis Scale Factor	

HitMeasure	mtalotbct	Top Count	5
Top Dimension	prod	By Dimension	loc

Open Template	MT_WB	Open Alert Name	MT_Alrt_OTB

	Measure	Ascending		YAxisFormat Type	YAxisFormat Currency	YAxisFormat Prefix	YAxisFormat NoDecimalPlaces	YAxisFormat Suffix	YAxisFormat Scale Factor
Sort Measures	mtalotbv	☐	✗						

DualY

DualY Measures	mtalotbv	Split DualY	off	Splitter Position	0.75
YAxis Format Type	currency	YAxis Currency		YAxis Prefix	
YAxis Number Decimal Places		YAxis Suffix		YAxis Scale Factor	

```
{
"id": "MT_Alrt_OTB",
"inserted": true,
"hitMeasure": "mtalotbct",
"topCount": 5,
"topDim": "prod",
"byDim": "loc",
"openTemplate": "MT_WB",
"openAlertName": "MT_Alrt_OTB",
"byCount": 10,
"description": "",
"yAxisFormat": {"type": "custom","options": {"numDecimalPlaces": 1}},
"sortMeasures": [
{
"measure": "mtalotbv",
"ascending": false
}
],
"dualY": {
"measures": ["mtalotbv"],
"yAxisFormat": {"type": "currency"},
"splitDualY": "off",
"splitterPosition": 0.75
}
},
```

Dashboard Plans

You can configure the number of recent plans that are displayed in the dashboard using the following config.properties setting. The value you assign to the property sets the maximum number of recent plans that will be displayed. The default value is 10.

```
dashboard.plans.max=10
```

Dashboard Errors

You can configure how dashboard errors are displayed using the following property:

dashboard.errors.show=false

This setting allows only administrative users to see errors when opening the metric dashboard related to the settings file on the UI. You can open the workbook behind the dashboard as a Workspace to see its structure when you click the **Open as Workspace** icon next to **Refresh**.

Note:

To debug any issues with the dashboard workbook, Sis and Admins can use this property. This property file can be found on the client side and can be updated under the/config/client folder. After the issue is resolved, remember to disable this property.

7

Contextual Help

Contextual help is end-user help that is relevant to the current application context. This means that the help topics the user sees are related to the view(s) that are open rather than a generic list of topics to hunt through. This enables the user to find information they need much more quickly, which improves user adoption, efficiency, and, ultimately, accuracy. The system implementer partners with the retailer to create the help content that describes the business process, calculations, critical KPIs, and so on. Training content, including instructional videos, published in a location accessible by the cloud service can be linked to within the help topics.

Overview

This chapter demonstrates how to configure contextual help in solutions based on RPASCE. Contextual help is a resource to access relevant help topics for dashboards and non-administrative application workspaces.

The help topics are classified into two categories, Dashboard and Workspace.

Dashboard

The help topics for the dashboard can be divided into two categories:

- **All**–The generic topics related to the product are added to this category.
- **Report**–This consists of topics related to dashboard, such as its effective usage, the analysis of the metrics, and so on.

Figure 7-1 shows the view of a dashboard and the help icon. The help topics for the dashboard can be seen on the right-hand panel in Figure 7-2.

Figure 7-1 Contextual Help Dashboard Window

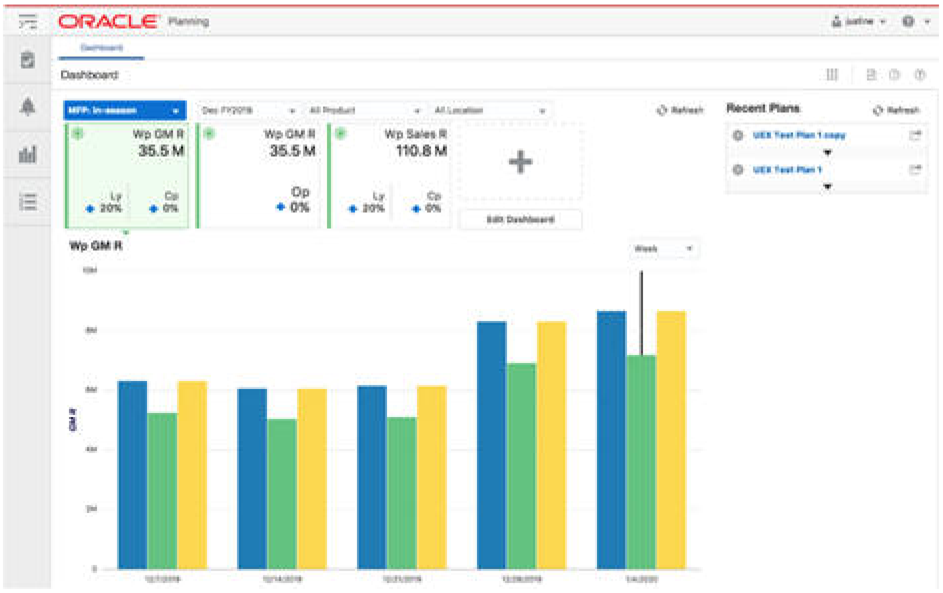
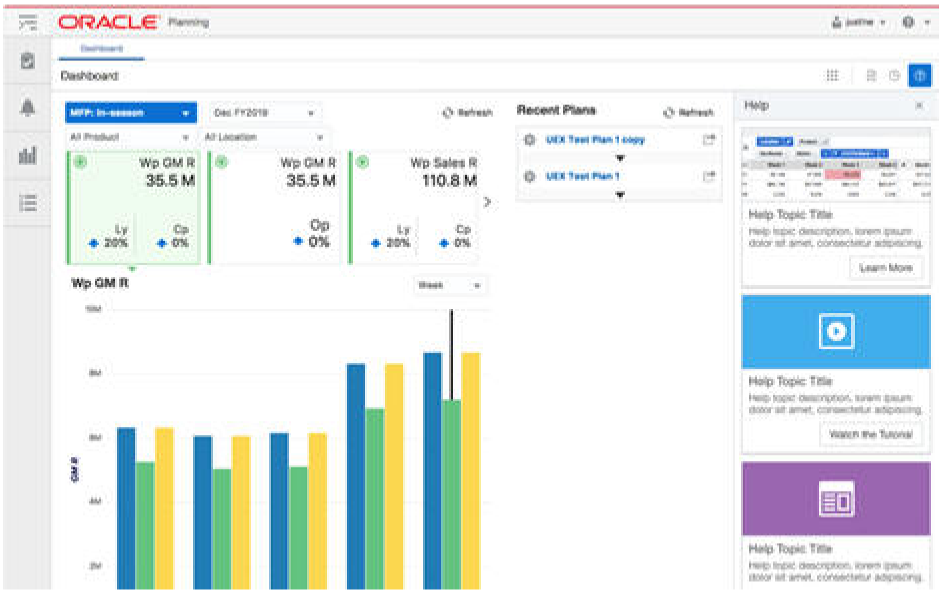


Figure 7-2 Contextual Help Topics Panel



Workspace

Workspace is defined as the space for the workbooks and their related view content. Here the topics are aligned with respect to the different levels of the taskflow. Figure 7-3 and Figure 7-4 illustrates the workspace for the product MFPRCS. Here, the Step, Tabs, and View are visible.

Figure 7-3 MFPRCS Workspace

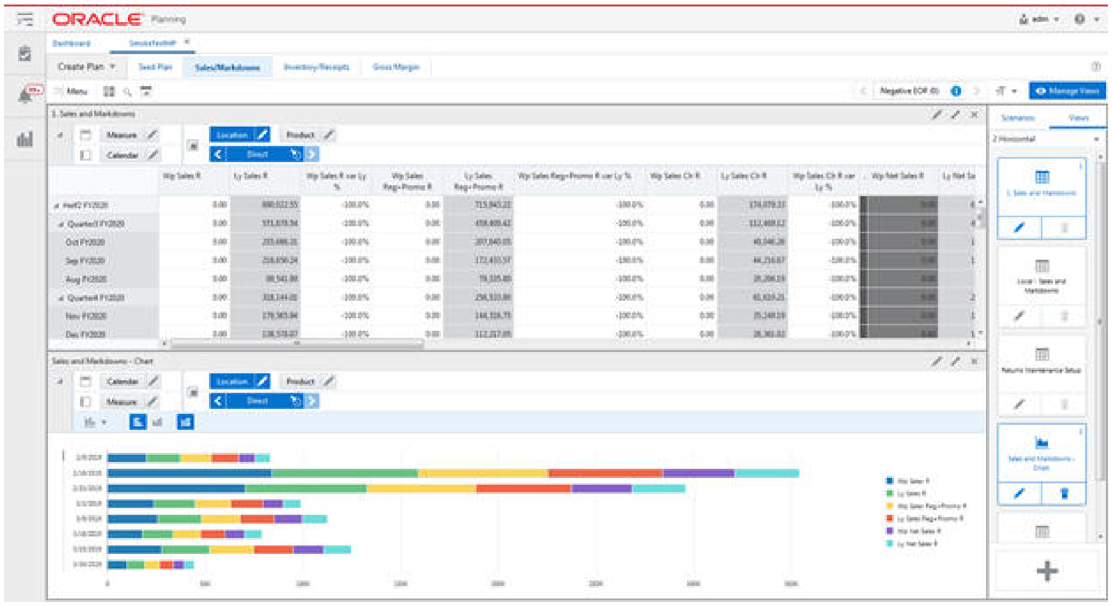
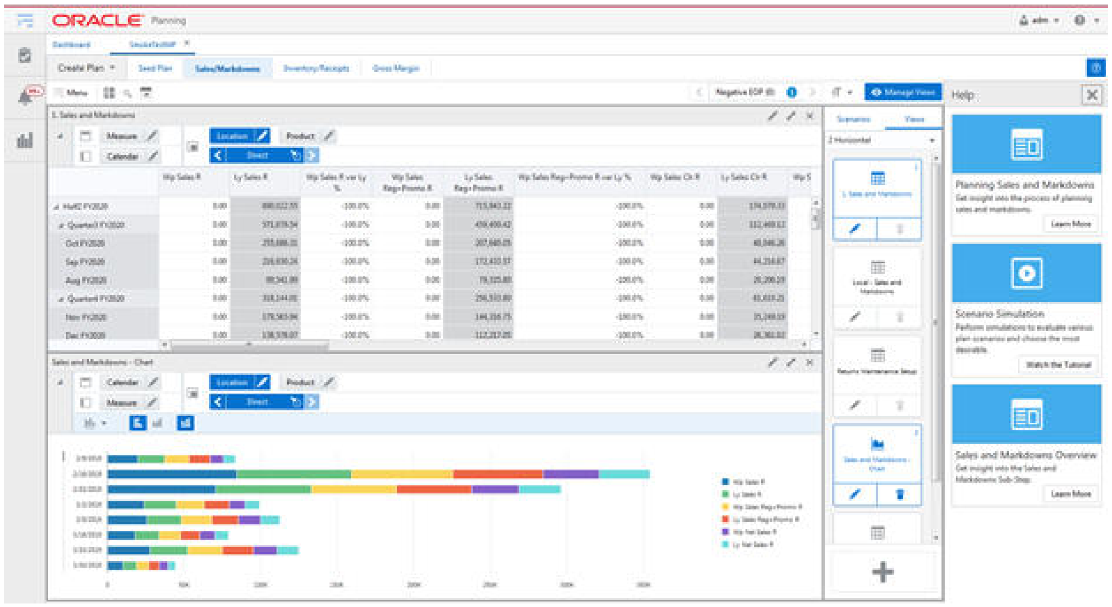


Figure 7-4 MFPRCS Workspace Displaying Help Topics



Contextual Help Configuration File

This section provides details about creating the configuration file.

Creating the Configuration File

The specifications related to contextual help for the RPASCE dashboard and workspace are achieved by creation of a configuration file. This file is created outside of the RPASCE

Configuration Tools. The contents of this configuration file are used by the RPASCE Client to determine how to organize and display the help topics dashboard and workspace.

Although a contextual help configuration file can be created from scratch, in most cases it is easier to modify the existing version of the file in order to incorporate any desired changes.

See "Retrieving or Updating the InContext Help JSON File" for instructions for uploading or modifying the file to the RPASCE Client application.

Using JSON

The contents of the contextual help configuration file are formatted as a JSON (JavaScript Object Notation) object. JSON is a common flexible information encoding notation used frequently in cloud applications; it is more compact and, when properly formatted, is more readable than the XML format.

Configuration File Levels

The configuration file is divided into three major levels: ALL, REPORTS, and WORKBOOKS. All the three levels are of type JSON Object. The ALL level is the generic level. REPORTS and WORKBOOKS are both children of level ALL.

Table 7-1 Configuration File Levels

Level	Description
ALL	This level describes the generic help topics related to the RPASCE solution in use. These topics are visible throughout the application, regardless of the view. The topics that explain the planning solution and its applications can be listed here.
REPORTS	Reports containing the help topics related to the dashboard.
WORKBOOKS	This level contains the help topics for Workbook and its sub-categories such as Task, Step, Tab, and View.

The generic JSON structure for any solution is as follows:

```
{
    "helpTopics" : [],
    "reports" : {"helpTopics" : [] },
    "workbooks" : {"helpTopics" : [] }
}
```

Help Topic Building Blocks

The help topics object as a whole is a JSON array of collection of attributes. This helpTopics object is the building block in each Level - All, Reports, and Workbooks.

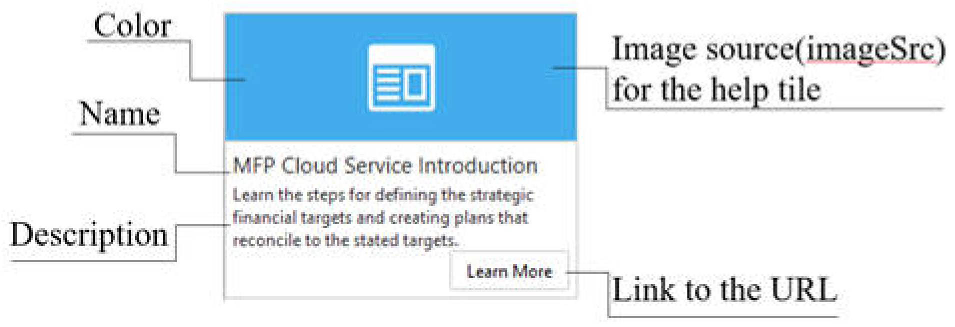
The following JSON snippet illustrates the generic helpTopics object structure:

```
"helpTopics" : [{
"name" : "Help Topic 1",
    "description" : "Description 1",
    "url" : "URL 1",
    "type" : "Type 1",
    "imageSrc" : "Image 1",
    "color" : "Color 1"
},{
    "name" : "Help Topic 2",
    "description" : "Description 2",
    "url" : "URL 2",
    "type" : "Type 2",
    "imageSrc" : "Image 2",
    "color" : "Color 2"
}]
```

Help Topic Tile

The help topic tile is a visual representation of the topic. Figure 7-5 displays the properties.

Figure 7-5 Help Topic Tile


Color
Name
Description
Image source(imageSrc) for the help tile
MFP Cloud Service Introduction
Learn the steps for defining the strategic financial targets and creating plans that reconcile to the stated targets.
Learn More
Link to the URL

The retailer and the implementer decide together which help topics should be available for each workspace or view within the customer's specific RPASCE workflow. Up to three contextual help tiles can be displayed on an individual view. More can be configured if desired; these are visible through scrolling.

Table 7-2 describes the available attributes that can be specified during the contextual help requirements definition for each help tile:

Table 7-2 Available Attributes

Property	Value Type	Description
Name	JSON String	The title of the topic.
Description	JSON String	A short description of the topic.
URL	JSON String	The link to the help topic.
Type	JSON String	The type of the topic's content. Valid options are document and video.
ImageSrc	JSON String	An optional URL for a preview image to display on the topic. The valid image types are jpg/jpeg, png, and gif.

Table 7-2 (Cont.) Available Attributes

Property	Value Type	Description
Color	JSON String	An optional color for the background of the topic's icon. The icon will be displayed if no preview image is provided. Valid colors are: lightblue, red, lightgreen, purple, blue, grey, orange, turquoise, and green.

Note that the mandatory fields for the tile are URL and Type. The remainder of the fields can be empty. Also note, on a help tile, that the document type labels the help link button as 'Learn More' & 'video' type labels it as 'Watch the Tutorial'.

Tips

- The icon (specified in ImageSrc) on the help tile should be simple and meaningful. It can be useful for visually indicating the type of resource (for example a video, workflow diagram, calculation reference, or a text description).
- Color can be another way to visually indicate the type of resource, or to highlight topics related to a particular business process or user role.

Naming Conventions for Keys

The naming convention of the key depend upon the level or sub-level of each of the element. Table 7-3 provides an example of naming conventions at different levels. You do not need to specify help topics at all levels.

You will find the keys for the level/sub-level in the Taskflow configuration file. Taskflow information can be found in the *Oracle Retail Predictive Application Service Cloud Service Implementation Guide* and the *Oracle Retail Merchandise Financial Planning Starter Kit* guide.

In this example, the solution in use is mfprcs.

Table 7-3 Naming Conventions for Keys

Level/Sub-Level	Key Example	Description
All	NA	A key is not required as help topics are added to the root of the JSON.
Reports	reports	
Reports > Dashboard	reports.dashboard.id	The key must match the name provided for reports in the Taskflow_MultiSolution.xml file.
Workbooks	workbooks	
Activity > Task	mfprcs.Activity1.Task1	The task name must match the entry provided in Taskflow_MultiSolution.xml file for the particular task.

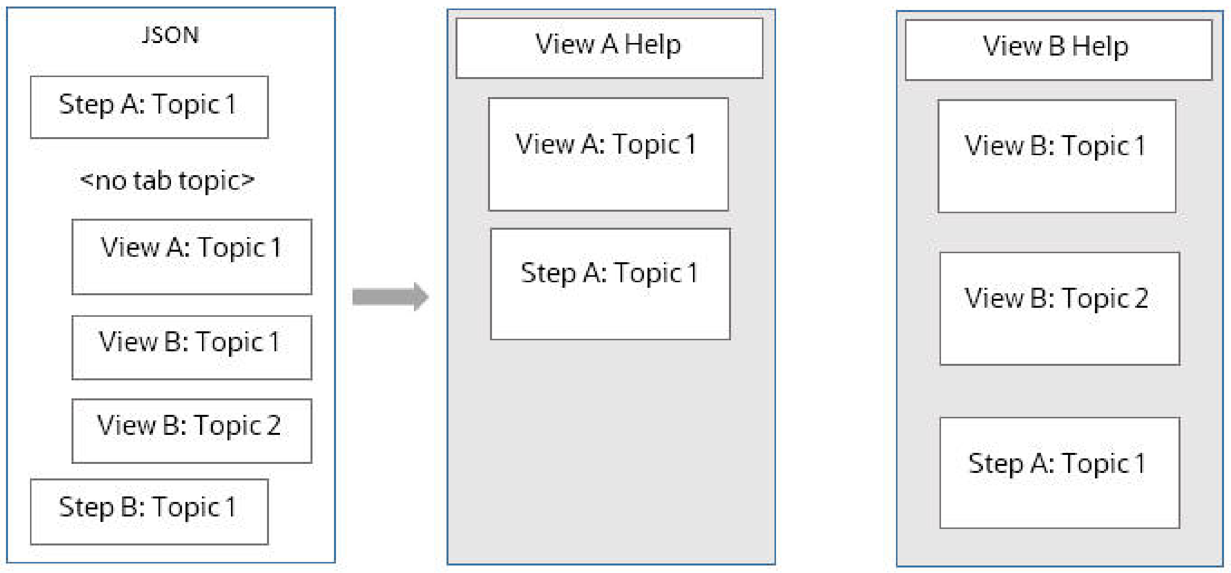
Table 7-3 (Cont.) Naming Conventions for Keys

Level/Sub-Level	Key Example	Description
Activity > Task > Step	mfprcs.Activity1.Task1.Step1	The step name must match the entry provided in Taskflow_MultiSolution.xml file for the particular step.
Activity > Task > Step > Tab	mfprcs.Activity1.Task1.Step1.Tab1	The tab name must match the entry provided in Taskflow_MultiSolution.xml file for the particular tab.
Activity > Task > Step > Tab > View	MT_TB01_WS01	The view name must match the entry provided in Taskflow_MultiSolution.xml file for the particular view. The view key name is unique as it can be added anywhere under task, step, or tab from the solution.

JSON Structure of Contextual Help Configuration File

Figure 7-6 displays an example of a JSON object containing all the three levels and the help topics related to each of them. The maxTopics define how many topics must be visible on RPASCE. If, for a particular level, there are fewer than maxTopics topics, it fetches the remaining topics from its parent. Figure 7-6 also shows that the maxTopics for workbooks is set to 2, and it overrides the maxTopics for the root, which is set to 3, for the workbooks level. Since there is no maxTopics set for reports, the value for maximum topics for this level is capped to 3, which is fetched from the root level.

Figure 7-6 Workbook Max Topics



When multiple views are displayed in a workbook, the help topics for the visible view are displayed in a specific order. The lower priority topics are dropped from the set presented to the user. The order from highest to lowest of relevant topics is as follows:

1. Application-specific information about the view currently displayed.
2. Application-specific information about the tab currently active.
3. Application-specific information about the step currently active.
4. Application-specific information about the task being performed in the workspace.
5. Platform-specific information.

```
{
  "maxTopics" : "3.0",
  "helpTopics" : [ {
    "name" : "MFP Cloud Service Introduction",
    "description" : "Learn the steps for defining the strategic financial 
targets and creating plans that reconcile to the stated targets.",
    "url" : "http://docs.oracle.com/cd/E75764_01/merchfinplan/pdf/cloud/20/html/
retail_implementer_guide/output/introduction.htm#introduction",
    "type" : "document",
    "imageSrc" : "",
    "color" : "turquoise"
  } ],
  "reports" : {
    "helpTopics" : [ ],
    "reports.dashboards.id" : {
      "helpTopics" : [ {
        "name" : "Using the dashboard",
        "description" : "Manipulate the dashboard in order to effectively 
analyze plan matrics",
        "url" : "http://docs.oracle.com/cd/E75764_01/merchfinplan/pdf/cloud/20/
html/retail_implementer_guide/output/dashboard.htm#dashboard",
        "type" : "document",
        "imageSrc" : "",
        "color" : "turquoise"
      } ]
    }
  },
  "workbooks" : {
    "maxTopics" : "2.0",
    "helpTopics" : [ ],
    "mfprcs.Activity1.Task1" : {
      "helpTopics" : [ {
        "name" : "Overview of Merch Plan Targets",
        "description" : "Learn about the steps associated with creating and monitoring 
targets",
        "url" : "http://docs.oracle.com/cd/E75764_01/merchfinplan/pdf/cloud/20/html/
retail_implementer_guide/output/
CreateMerchPlanTargets.htm#create_merch_plan_targets_task",
        "type" : "document",
        "imageSrc" : "",
        "color" : "turquoise"
      } ]
   }
}
```

Adding or Editing the Contextual Help Configuration File

The implementer can add or edit the help topics directly under the levels ALL and REPORTS. However, for level WORKBOOKS, the implementer can either add/edit under Task or can add/edit under a specific sub-level (Step, Tab, or View).

Here are examples where the implementer adds the help topics at different levels.

- **Adding/Editing the help topic for level ALL.** The implementer can add the help topic object directly under the root of the JSON under the property helpTopics. To edit, the implementer must search for the name of the help topic in JSON and edit any of the required properties.
- **Adding/Editing the help topic for level REPORTS**. In this case, the implementer must add the help topic under the reports object of the JSON. The implementer must search for the key reports and then add the help topic under the attribute helpTopics. Similarly, the implementer can edit any particular help topic by searching for the name of the help topic.
- **Adding/Editing the help topic for sub-level Step under level WORKBOOKS.** To add a topic under the sub-level Step, the implementer must search for the Step key and add the help topic. To edit, the implementer must search for a particular help topic and edit the properties as required.
- **Adding/Editing the help topic for sub-level View under level WORKBOOKS**. To add a topic under the sub-level View, the implementer must search for the View key and add the help topic. To edit, the implementer must search for a particular help topic and edit the properties as required.

Retrieving or Updating the InContext Help JSON File

You can update or retrieve the help JSON file using an OAT task. This provides the flexibility to view the list of available help resources and modify them as needed.

Make sure the following environment variables are set:

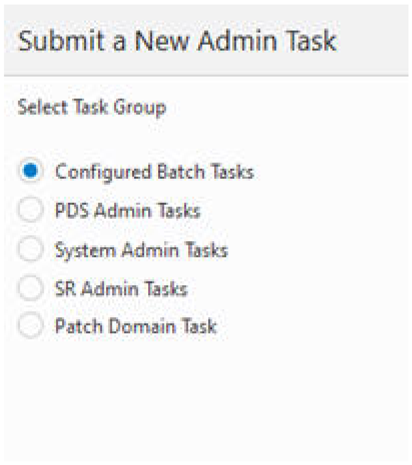
- APP_NAME

- OUTGOING_FTP_PATH
- INCOMING_FTP_PATH

Complete the following steps to submit the OAT task.

1. Open Submit a New Admin Task.
2. Select Configured Batch Tasks under Task Group and click **Next**.

Figure 7-7 List of Task Groups



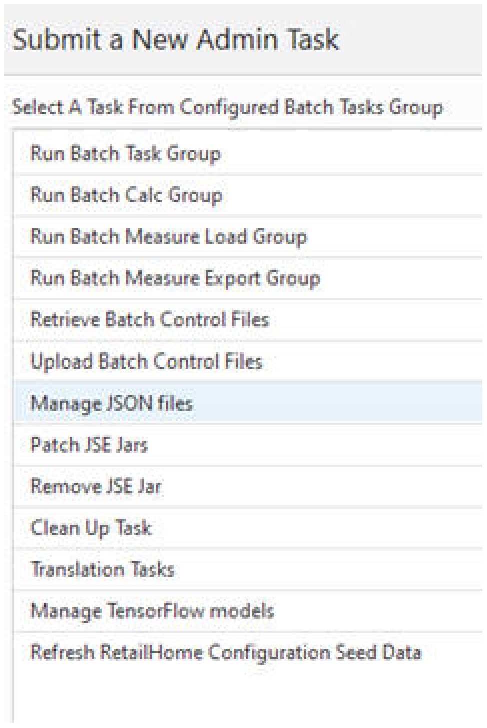
Submit a New Admin Task

Select Task Group

● Configured Batch Tasks
○ PDS Admin Tasks
○ System Admin Tasks
○ SR Admin Tasks
○ Patch Domain Task

3. Select Manage JSON files from the list of available tasks and click **Next**.

Figure 7-8 Configured Batch Task Group Tasks



Submit a New Admin Task

Select A Task From Configured Batch Tasks Group

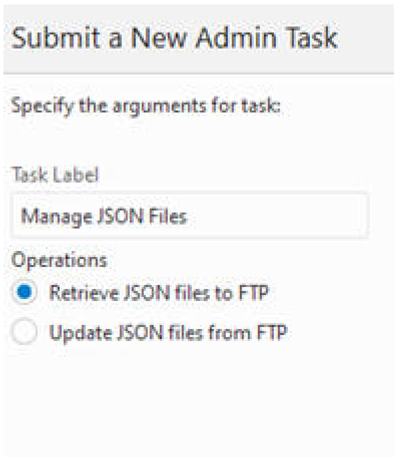
Run Batch Task Group
Run Batch Calc Group
Run Batch Measure Load Group
Run Batch Measure Export Group
Retrieve Batch Control Files
Upload Batch Control Files
Manage JSON files
Patch JSE Jars
Remove JSE Jar
Clean Up Task
Translation Tasks
Manage TensorFlow models
Refresh RetailHome Configuration Seed Data

4. Provide a task label and select the type of operation to be performed.
   - **Retrieve JSON files to FTP**. This operation fetches the help JSON file from the RPASCE application. When this task is run, the help JSON file (with other JSON files) is bundled as a tar.gz with the label <solution_name>_json.tar.gz and is placed under the OUTGOING_FTP_PATH. The help JSON file is also prefixed with the solution name <solution_name>HelpConfig.json

     Example: mfprcsHelpConfig.json
   - **Update JSON files from FTP.** After the JSON changes are complete, the file name must be prefixed with solution name <solution_name>HelpConfig.json. This JSON file must then be bundled as <solution_name>_json.tar.gz and placed under the INCOMING_FTP_PATH/config directory. Next, when the Update JSON files from FTP operation is performed on the RPASCE UI, it fetches the tar.gz bundle from INCOMING_FTP_PATH/config and updates the application with the new changes from the help JSON file. The user must re-log into the application to see the changes.

> **Note:**
>
> New URLs in the help JSON file must be whitelisted so that the links are accessible from the RPASCE UI. To whitelist the URLs, add the application names of these URLs to the property valid.url.hosts in <CONFIG_ROOT>/ config.properties file. For more information see "Images and Contextual Help" in the *Oracle Retail Predictive Application Server Cloud Edition Administration Guide*.

Figure 7-9 Task Label and Operation Type Options


Submit a New Admin Task

Specify the arguments for task:

Task Label

Manage JSON Files

Operations

[✓ Retrieve JSON files to FTP]

Update JSON files from FTP

8

System Preferences

General preferences can be set for the Configuration Tools at the workbench level, which refers to the entire tool. This includes the application type and general preference settings.

Note:

This chapter contains descriptions of features that are not supported in the current version of RPASCE. However, in order to assist in the process of migration of a prior version to the current, the configuration components associated with these deprecated features remain present within the RPASCE Configuration Tools and therefore continue to be described within this document.

As explained in Chapter 3, the general preference “Global Domain” and “Multi Language” are no longer configurable. The RPASCE PDS is always created as multi-lingual and the sandbox is always created as a simple domain. The measure data is stored as facts within the Oracle Database tables in RPASCE PDS.

For additional information, see Appendix C, "Appendix – Calculation Engine User Guide" in this document.

Global Domain (Deprecated)

A global domain environment provides the ability to view data from multiple physical domains in a single workbook, and to administer common activities centrally across the RPASCE 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 applications.
- **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.

There are two primary functional benefits in using a global domain environment:

- The ability to have a global view of data in workbooks.
- The end user can build workbooks with data from multiple 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 responsibilities and roles. Most users will only work within the local domains that contain their area of responsibilities, and they may not need to be aware of the global domain environment. For the RPASCE Client, position level security is used to guide users into the applications in which they have some access. For performance and user

contention reasons, global domain usage should be limited to relatively infrequent processes that require data from multiple local domains.

- Configuration and Administration.
- Most of the mechanisms that are required to build and administer a domain are centralized, so they need only be run in the master domain, which either propagates data to the local domains or stores it centrally so that the local domains reference it in the master.

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

Measure Data (Deprecated)

In a global domain environment, measure data can be physically stored across the local domains or in the master domain.

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

The base intersection of a measure (the dimensions that 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 higher than 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. In this scenario, data for measures that have a base intersection in the Product hierarchy at or lower than Department (other hierarchies are irrelevant for this discussion) is stored in the local domain. This is based on the Department that the underlying position in the Product hierarchy belongs to.

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 they will be accessed from there when these measures are required in workbooks.

All measures will be registered in the master domain, and they will be automatically registered in all local domains. RPASCE 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.

Multi-Language (Deprecated)

RPASCE applications are built to be used in English only or in English and other languages. Multi-lingual aplications allow for most data elements in an RPASCE application to be translated into another language. The translation process is managed by the Oracle Translation group and is handled as a separate agreement with Oracle.

Solution ID

A taskflow that includes multiple solutions creates the possibility for information in the taskflow of one configuration to collide with the information in the taskflow of another. For example, two different configurations could each contain an activity named Activity1. Because it is not possible at solution configuration time to know the full set of solutions that might be combined using a multi-solution taskflow, it is not possible to prevent such conflicts through configuration validation.

In order to prevent collisions between the taskflow information of multiple configurations, the solution ID is used. It is a property of the configuration. Each configuration can have a different Solution ID which defaults to the name of the configuration.

When generating taskflow resources, the solution ID for the configuration is used as a prefix to all keys. In addition, every task generated out of the Configuration Tools as part of the taskflow resources used by the RPASCE Client has a solution ID attribute used to identify to which solution the task is applicable. Note that multi-solution taskflow implementations that contain multiple instances of a solution are required to specify unique Solution IDs for each instance of that solution to prevent collisions.

For users upgrading to 14.0, the default value of solution name is used for the solution ID so that no additional configuration work is required for users not implementing a multi-solution taskflow.

Term Plan Labeling

There are many places in the RPASCE Client where labeling is used to identify RPASCE workbooks, tasks, and so on. Initial applications released on the RPASCE platform were planning applications and so these labels used the term “Plan", “plan", “planning", or “Plans" to describe the workspaces where users performed their activities.

As RPASCE supports additional applications that are not planning applications, such as the RDF application, it is now necessary to allow these labels to be modified as part of the application configuration. This is achieved via Term Plan labeling, which lets the user configure labels for the term “Plan" and its variants to align with the functional concepts of the application; if explicit labeling is not provided, the default term is used. For example, term Plans can be configured to have the label “Forecasts" in the RDF application so that the Most Recently Used list is referred to as “Recent Forecasts" instead of “Recent Plans".

The configured labels are added into both the server and the client translation resources and must be translated accordingly.

Note:

Configuration Tools updates the taskflowBundle.properties and r_msglabel.ovr.english files with the configured labels. Both files must be translated since the PDS is multi-lingual.

The five configuration identifiers show up in taskflowBundle.properties as follows:

<<SolutionIDValue>>.Plan.Label=<<configured label>>

<<SolutionIDValue>>.Plans.Label=<<configured label>>

<<SolutionIDValue>>.plan.Label=<<configured label>>

<<SolutionIDValue>>.Planning.Label=<<configured label>>
<<SolutionIDValue>>.WP.Label=<<configured label>>

Only two messages in the file r_msglabel.ovr.english contain configurable term. When the term “plan" is configured to have the label “workbook", for example, these two messages show up in file as:

R_WB_WRKCALCINPROGRESS english_us The workbook is Calculating. Try to open your '%1' again after a few minutes. If the problem persists, contact your administrator.

R_WB_WRKREFINPROGRESS english_us The workbook is Refreshing. Try to open your '%1' again after a few minutes. If the problem persists, contact your administrator.

Customizing Dashboard Labels

You can customize dashboard labels via the MultiSolutionBundle. Labels can be customized for the primary measure label, the variance measures labels, and the main chart title.

The highlighted areas in the following tile types can use a translated string and override the default:

- Exception tile
- Metric tile
- Variance tile

In the content area, the highlighted areas use a translated string and override the default.

In all but the variance plan, the default behavior is to use the value in the settings file if the value is not found as a key in the MultiSolutionBundle.

Profiles

To add translation for a profile, an entry must be added to MultiSolutionBundle in the following format: <<SolutionIDValue>>. dashboard.profile.<<name>>=<<configured label>>

Tiles (metric or exception)

To add custom translatable labels for tiles, an entry must be added to MultiSolutionBundle in the following format<<SolutionIDValue>>. <<key>>=<<configured label>>

Keys

The keys must be unique within each solution. Most of these settings default to the measure-related or application-related attribute; however, it can be changed to help abbreviate it so that it fits the dashboard.

Here is the list of tile settings that can be customized.

Tile Setting	By Default
title	Uses the measure label.
description	Uses measure label.
unitOfMeasure	Empty in metric profiles, in exception profiles it is the dimension labels of the top /by.
contentTitle	The metric tile label.
yAxisLabel	Empty.
dualY yAxisLabel	Empty.
left / right (variances)plan	If not found it uses the first word of the chartMeasure label.

Fact Prefix Override

Optional property. By default, the fact name prefix of a non-shared measure is the application configuration name. Users may override it by specifying a unique identifier for the Fact Prefix Override property within the Configuration Properties window. The value of the Fact Prefix Override property must be less than or equal to seven characters; it is used to prefix fact names.

Note:

Since the Fact Prefix Override affects the fact name construction, its value is not patchable. If the application is going to share the PDS instance with other applications, the Fact Prefix Override value must be unique across the shared applications.

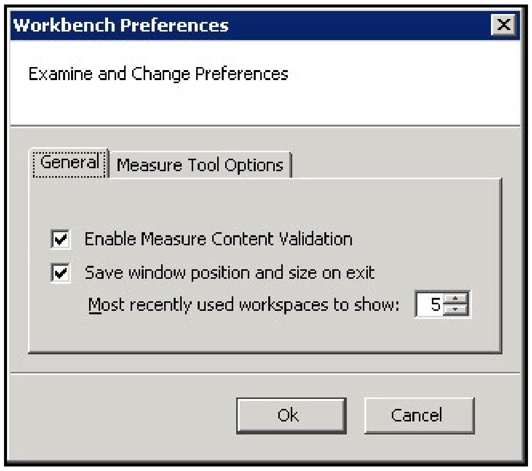
Setting Workbench Preferences

Complete the following steps to set workbench preferences.

Navigate: From the File menu, select **Tools Preferences**. The Workbench Preferences window opens.

1. Select the **General** tab and select the appropriate options.

Figure 8-1 Workbench Preferences Window– General Tab


Workbench Preferences

Examine and Change Preferences

General | Measure Tool Options

☑ Enable Measure Content Validation

☑ Save window position and size on exit

Most recently used workspaces to show: 5

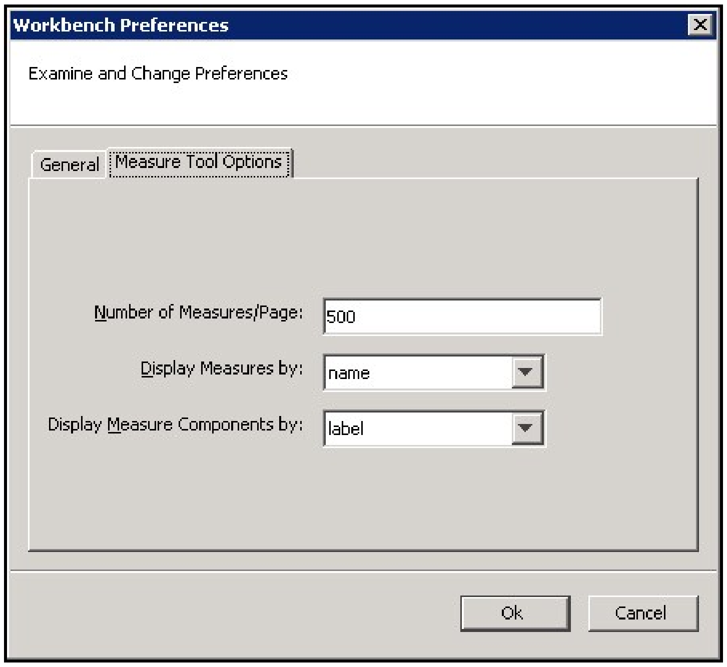
Ok Cancel

The fields for the General tab include:

- The main file menu lists configurations that were recently opened. The number of configurations is displayed in this menu. To set the number of configurations, set the **Most recently used workspaces to show** field by using the up and down arrows.
- Select the **Measure Content Validation** check box to enable measure content validation. Clear the check box to disable measure content validation. A change to the properties of a measure can affect the validity of many components both within the Rule Tool and the Workbook Tool. Whenever a measure editing session is completed (for instance, upon exiting the Measure Manager and entering a different tool), the workbench will evaluate the effects of the edits made upon the measures. This process can be time consuming. When the Measure Content Validation option is unchecked, the automatic validation of measure property edits is unavailable. This allows for rapid transitioning between tools when working on a large configuration, because it will not be necessary to await the completion of the automatic validity checking. When automatic validation is unavailable, a manual check of measure validity can be enabled from the Rule Tool. This allows you to manually update the measure content validation of the Rule and Workbook Tools (see Measure Validation within the Measure Manager for more information).

2. Select the **Measure Tool Options** tab and adjust the fields as needed.

Figure 8-2 Workbench Preferences Window– Measure Tool Options Tab



Number of Measures/Page: 500

Display Measures by: name

Display Measure Components by: label

The fields for the Measure Tool Options tab include:

- Number of Measures/Page – The Measure Manager display only a certain number of measures per page in the Measure tab. This option determines the number of measures that display on a page.
- Display Measures by – This list provides two options, name and label, and determines whether measures are displayed using their name or label in the Configuration Tools. For example, in the Workbook Tool when selecting viewable measures, the name or the label will be displayed depending on the selection here.
- Display Measure components by – This list provides two options, name and label. Your selection determines how measure components in the Measure Manager are displayed.

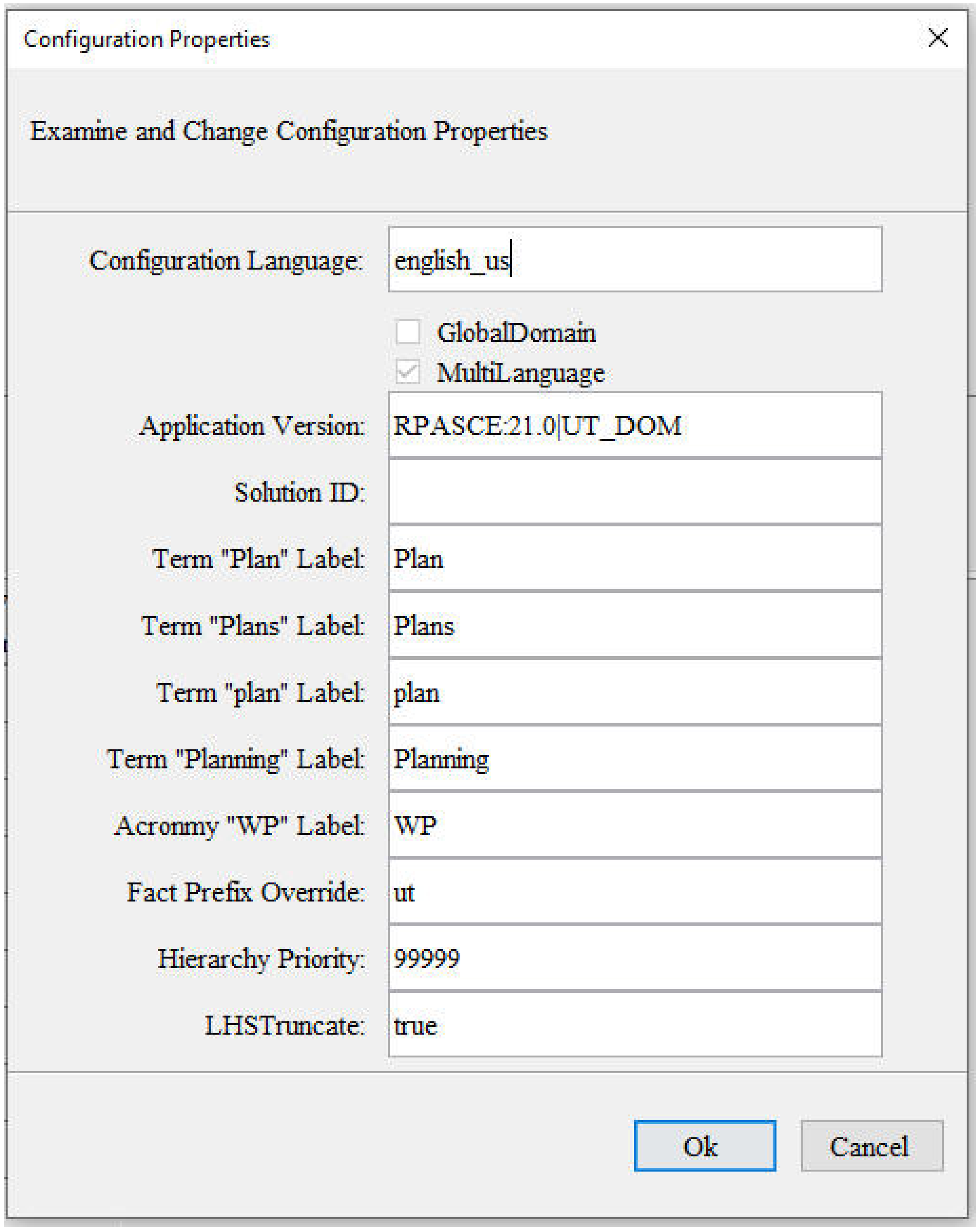
3. Click **OK** to save any changes and close the window.

Setting Configuration Properties

Complete the following steps to set Configuration Properties.

Navigate: From the File menu, select Configuration Properties. The Figure 8-3 opens.

Figure 8-3   Configuration Properties Window


Configuration Properties

Examine and Change Configuration Properties

Configuration Language:	english_us
	☐ GlobalDomain
	✓ MultiLanguage
Application Version:	RPASCE:21.0|UT_DOM
Solution ID:	
Term "Plan" Label:	Plan
Term "Plans" Label:	Plans
Term "plan" Label:	plan
Term "Planning" Label:	Planning
Acronmy "WP" Label:	WP
Fact Prefix Override:	ut
Hierarchy Priority:	99999
LHSTruncate:	true

The **Configuration Language** field is unavailable by default and displays *English*, because currently configurations can only be in English.

1. Select the following options as necessary:
   - **Global Domain– Deprecated**. Since all data is stored in PDS, the sandbox is only created and used in the PDS building and patching steps and is always a *simple* domain. The global domain setting is not configurable. ConfigTools always treats it as true.

- **MultiLanguage** – **Deprecated**. Since the PDS instance is always multi-language, this setting cannot be set to false. ConfigTools always treats it as true.

2. Enter the following properties as necessary:
   - **Application Version** – Enter the application version in the format ApplicationName:ApplicationVersion.
   - **Solution ID** – Enter a unique identifier for the configuration. The default is the name of the configuration. The solution ID is used to help integrate multiple applications using a combined taskflow.
   - **Term Plan Labelling** – For the term “Plan” and its variants, enter the desired labels that are displayed in RPASCE client.
   - **Fact Prefix Override** – Enter a unique identifier for the fact name prefix.
   - **Hierarchy Priority** – Enter an integer for the priority of this application’s hierarchies. This property defaults to 99999. During the PDS build/patching, the hierarchies from multiple applications are merged. The higher this number, the more weight this application’s hierarchies have when determining the hierarchy order of the merged PDS hierarchies. See Multi-Application Support section for more information.
   - **LHSTruncate** –This property defaults to false. If the configurator knows that all or most of the data (for example,, more than half) of the LHS measures of all batch rules within this application change in the weekly or daily batch, enter true so that the optimal SQL statements are generated for all the batch rules of this application. Configurator can turn off this optimization for a specific rule by setting its rule property LHSTruncate to FALSE in the Rule Properties window. Reference the section of “Edit Properties of a Rule” for how to add or edit a new rule property.
3. Click **OK** to save any changes and close the window.

9

Configuration Utilities

The utilities in this chapter are standalone utilities that can be run externally, or they can be launched from the Utilities menu of the Configuration Tools. The utilities provided include the Configuration Converter and the Function Library Manager, which are described in detail.

Configuration Converter

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.

Note:

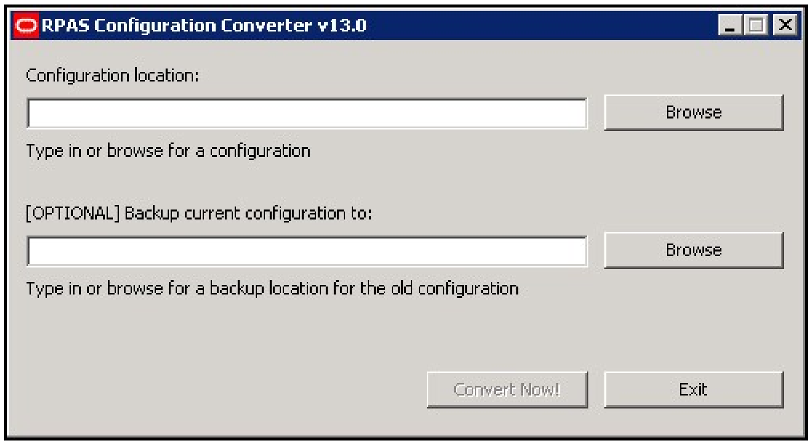
The functionality for converting a configuration is provided directly through the Configuration Tools. See the section, Open an Existing Project from an Older Version of the Configuration Tools.

Launching the Configuration Converter

The Configuration Converter can be accessed in three ways:

- From the Utilities menu in the Configuration Tools, select **Configuration Converter**. The Figure 9-1 opens.

Figure 9-1 RPAS Configuration Converter Window


RPAS Configuration Converter v13.0

Configuration location:

Type in or browse for a configuration

[OPTIONAL] Backup current configuration to:

Type in or browse for a backup location for the old configuration

- From the Windows Start menu, select Oracle, then RPASCE, then Utilities, and then select **Configuration Converter**. The Figure 9-1 opens. If this shortcut does not appear, refer to the Starter Kit Guide for the version of the Starter Kit you have installed for information about creating it.

- Go to a command prompt. Run the `RpasConverter.exe` file in the \utilities directory where the Configuration Tools were installed and run the following command:

  ```
  RpasConverter -c C:\PathToConfig\Config\Config.xml [OPTIONS]
  ```

  The following options can be used from the command line:

Table 9-1 Command Line Options

Option	Description
-b BackupDir	Use this argument to create a backup of the original configuration in BackupDir location specified.
-g	Use this argument to open the Figure 9-1
-h	Use this argument to display usage information.

Converting a Configuration

Complete the following steps to Convert a Configuration

1. In Configuration location field, enter path and configuration file name to be converted. This is the file in the configuration's directory that has the configuration name with an `.xml` extension. Click **Browse** to navigate and select the appropriate file. Make sure to provide the file extension in the Configuration location field.

   ```
   C:\Configs\MyConfig\MyConfig.xml
   ```

2. Optional: In the Backup current configuration to field, enter a directory where a copy of the original configuration will be stored. You may also click **Browse** to navigate and select a directory.

   Note:

   The directory entered must not already contain a directory whose name is the name of the original configuration. For example, to put a backup of a configuration named `MyConfig` in a directory `C:\Backups`," "`C:\Backups\MyConfig` must not exist.

3. Using the **Convert to version** list, select the version to convert to. This should always be the current version of the Configuration Tools unless there is a good reason to convert to some older version.
4. Click **Convert Now**.
5. If the conversion was successful, it may now be opened in the Configuration Tools. If there was an error while converting, an error message will be displayed, and the original configuration will remain untouched.

Note:

Refer to the Starter Kit Guide for the version of the Starter Kit you have installed for more information on the application installation and upgrade process.

Function Library Manager

The RPASCE calculation engine is designed to be extensible with support for custom functions or procedures that can be used in normal expressions. For validation purposes, the Configuration Tools are only aware of the standard RPASCE functions and procedures, so they will generate an error for any expressions that use custom functions or procedures. The Function Library Manager is used to provide validation for custom functions or procedures within the Configuration Tools. The custom functions or procedures must exist in the /applib directory of the RPASCE_HOME directory. If necessary, this utility can also be used to remove custom function libraries from being validated. There is no validation for the existence of the function libraries in RPASCE_HOME/applib directory. When a function library is removed using the Function Library Manager, it is removed only from the list of external libraries used for validation, and the contents of RPASCE_HOME/applib directory are left intact. The function libraries mentioned in this list are loaded by the Configuration Tools and will be used to perform rule validation.

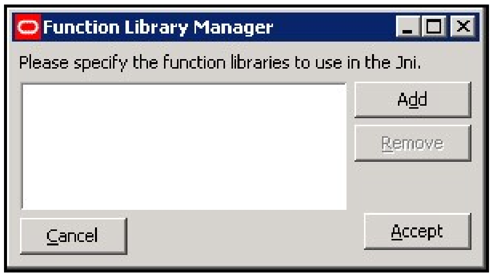
Speak to an Oracle Retail Services representative for additional information about custom functions and procedures.

Launching the Function Library Manager

Perform the following steps to Launch the Function Library Manager

Navigate: From the Utilities menu in Configuration Tools, select the Function Library Manager, or run the Function Library Manager.bat file from the utilities directory where the Configuration Tools is installed. The Figure 9-2 opens.

Figure 9-2 Function Library Manager Window

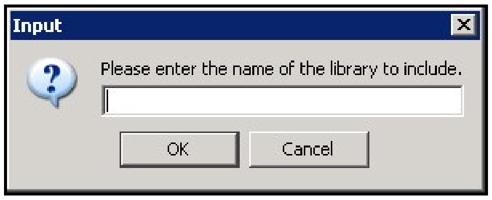

Please specify the function libraries to use in the Jni.

Adding a Function Library to Be Validated in the Configuration Tools

Perform the following steps to add a function library to the Configuration Tools:

1. Launch the Function Library Manager.
2. Click **Add**. The Figure 9-3 opens.

Figure 9-3 Input Window


Input

Please enter the name of the library to include.

3. Enter the name of the library you want recognized.

> **Note:**
>
> Enter the name without the `*.dll` or `*.so` extension.

4. Click **OK**.
5. Click **Accept** to save any changes and close the window.

Removing a Function Library from Being Validated in the Configuration Tools

Complete the following steps to Remove a Function Library from Being Validated in the Configuration Tools

1. Launch the Function Library Manager.
2. Select the Function Library you want to remove from the validation process.
3. Click **Remove**. The Function Library is removed from the list.
4. Click **Accept** to save any changes and close the window.

Report Generator

The Report Generator is a utility that may be used to extract information about a configuration for external use. The information is generated in a structured text document that is much easier to manipulate than the XML format of the configuration files that are saved and loaded by the workbench. Many of the reports correspond to files generated as a part of the installation process.

Available Reports

The following reports can be created using the Report Generator:

- **Measure Extractor** – This report generates a text file that lists the measure content of a solution.
- **Data Interface Report** – This report generates a text file that lists the properties of all the measures in the project that have been added to the Data Interface Tool.
- **Measure Description Translation** – This report generates a translation file lik the file generated as part of the installation process. It allows the extraction of measure descriptions for a project without the need to build the PDS first.

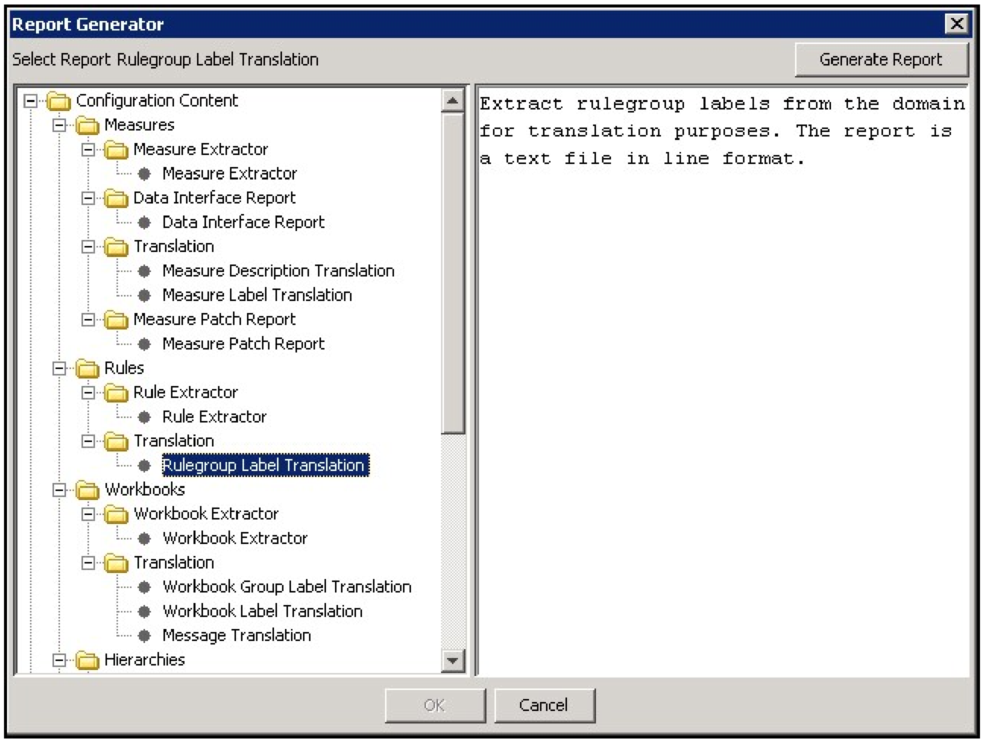
- **Measure Label Translation** – This report generates a translation file lik the file generated as part of the installation process. It allows the extraction of measure labels for a project without the need to build the PDS first.
- **Measure Patch Report** – This report examines a previous version of a configuration to determine which measure properties have changed between the two versions. It is used to determine which measures will be added, removed, or updated during a patch installation.
- **Rule Extractor** – This report generates a text file that lists the rule content of a solution.
- **Rule Group Label Translation** – This report generates a translation file that is similar to the file generated as part of the installation process. It allows the extraction of rule group labels for a project without the need to build the PDS first.
- **Workbook Extractor** – This report generates a text file that lists the workbook content of a solution.
- **Workbook Group Label Translation** – This report generates a translation file that is similar to the file generated as part of the installation process. It allows the extraction of workbook group labels for a project without the need to build the application first.
- **Workbook Label Translation** – This report generates a translation file that is like the file generated as part of the installation process. It allows the extraction of workbook labels for a project without the need to build the PDS first.
- **Messages Translation** – This report generates a translation file that is like the file generated as part of the installation process. It allows the extraction of messages issued by the RPASCE Client for a project without the need to build the PDS first.
- **Dimension Label Translation** – This report generates a translation file that is like the file generated as part of the installation process. It allows the extraction of dimension labels for a project without the need to build the PDS first.
- **Hierarchy Label Translation** – This report generates a translation file that is like the file generated as part of the installation process. It allows the extraction of hierarchy labels for a project without the need to build the PDS first.
- **Hierarchy.xml Report**– This report generates the hierarchy.xml resource file used by RPASCE during the PDS creation and the dimension patching processes.
- **Taskflow Description** – This report generates the taskflow.xml document used by the RPASCE Client like the file generated as part of the installation process. It allows the creation of the xml file without the need to build or patch the PDS first.
- **Taskflow Resources** – This report generates the Resource Bundle used by the RPASCE Client like the file generated as part of the installation process. It allows the creation of the resources bundle without the need to build or patch the PDS first.

Generate a Report

Complete the following steps to generate a report.

1. Select the project that requires the report.
2. Select Generate Reports from the Utilities menu. The Figure 9-4 opens.

Figure 9-4 Select a Report Window



Report Generator

Select Report Rulegroup Label Translation

Generate Report

- Configuration Content
  - Measures
    - Measure Extractor
      - Measure Extractor
    - Data Interface Report
      - Data Interface Report
    - Translation
      - Measure Description Translation
      - Measure Label Translation
    - Measure Patch Report
      - Measure Patch Report
  - Rules
    - Rule Extractor
      - Rule Extractor
    - Translation
      - Rulegroup Label Translation
  - Workbooks
    - Workbook Extractor
      - Workbook Extractor
    - Translation
      - Workbook Group Label Translation
      - Workbook Label Translation
      - Message Translation
  - Hierarchies

```
Extract rulegroup labels from the domain
for translation purposes. The report is
a text file in line format.
```

OK Cancel

3. Select the desired report from the list in the left pane of the Generator window. The right pane displays a short description of the currently selected report.
4. Select **Generate Report** to begin the report generation process.
5. Depending upon the report in question, there may be some options to specify in further windows. These options commonly include the location where the generated file is to be stored or the selection of a single solution from the project.
6. Once all options have been specified, click **OK** to generate the report. The button is unavailable until all options have been specified.

10 Integration Tool

The RPASCE platform supports the integration of applications with a data schema maintained within an Oracle Database instance. Once integrated, the data is automatically pushed into and pulled out of the database in order to perform operations such as batch running, workbook building, and committing. By integrating multiple RPASCE applications with a single data schema, it is possible for applications to share data automatically through transparent platform processes without the need for batch processes to synchronize information.

In order to support these behaviors, a Planning Data Schema (PDS) must be created within an Oracle database instance. The integration configuration of the objects and information within the data schema is automatically generated by RPASCE based on the application configurations.

Planning Data Schema

The Planning Data Schema (PDS) is a persistent data schema maintained within an Oracle database. The PDS is used as a central repository and storage of RPASCE data that can be produced and/or consumed by the RPASCE processes. The primary purpose of the PDS is to maintain a set of hierarchies, dimensions, RPASCE metadata, facts, and so on. These facts, which are stored within in tables of the Oracle database, are conceptually equivalent to the measures of a RPASCE application. In addition, the information within a fact is structured in a manner that is functionally equivalent to the multi-dimensional system present in a RPASCE application.

It is therefore possible to access fact information through an address composed of a set of positions along discrete dimensions. Each row of a fact table, like a cell within an RPASCE array, corresponds to a unique combination of positions along one or more dimensions (for example, a single sku, store, and week). However, unlike a RPASCE array, a fact table may contain values for multiple facts within a single table. Each fact is represented by a distinct column within the fact table.

For any given column (fact) and row (position address), a table contains a value if the fact has a populated value for that address, but contains an empty cell for that column if the fact has no value for that address.

The PDS also maintains tables to describe the hierarchies and dimensions used to structure the facts contained within the data schema. This information is stored within a separate set of tables, known as dimension tables, in a manner analogous to the dimension arrays of an RPASCE application. As with the hierarchies and dimensions of an RPASCE application, the dimension tables of a PDS can be administered in order to add, remove, and modify individual positions and can represent the child/parent relationships that describe the multiple levels represented by the dimensions of a hierarchy.

Refer to “Planning Data Schema Administration” in *Oracle Retail Predictive application Server cloud Edition Administration Guide* for more details.

Integration Configuration Components

In order to properly build and patching the PDS, the information on the application, Shared Hierarchies and Dimensions, the Facts, and the Integration Map must be supplied. The information about these components is stored in an XML document referred to as integration configuration, which is automatically created and maintained by the RPASCE Configuration Tools or rpasInstall.

Shared Hierarchies and Dimensions

The information contained within the fact tables of a PDS, such asthe information contained within the measures of a RPASCE application, is structured to represent a set of multidimensional relationships. This structure is represented by a set of dimensions that are defined along hierarchies that describe the space in which the information is relevant. These hierarchies can represent common familiar constructs such as the Calendar, Location, or Product hierarchies. They can also represent concepts that are application specific.

The Integration Configuration contains information about the hierarchies represented within the PDS. It also contains information about the structure of the dimensions that are defined within a hierarchy. As with an application configuration, dimensions are defined in a tree structure to allow the representation of roll-up information.

All hierarchies and dimensions within the application configuration are pulled into the integration configuration, although only a subset of hierarchy and dimension properties are stored within the integration configuration.

Shared Hierarchies Properties

A shared hierarchy contains following properties:

- **Hierarchy Name** – This is the name of the hierarchy. The name of a hierarchy within the PDS must correspond to the RPASCE name of an application hierarchy in order for that application hierarchy to participate in sharing data with the PDS.
- **Hierarchy Label** – The label is a user label that identifies the hierarchy in reporting and user notification.
- **Hierarchy Purge Age** – The purge age is the amount of time RPASCE continues to store a hierarchy position after the position is no longer included in the hierarchy load files. If the amount of time in days that has passed since the last hierarchy load contained an entry for a given position, that position will be marked for purging from the system.
- **Hierarchy Order** – As with hierarchies in an RPASCE application, the hierarchies of a PDS must be ordered. This ordering is used in the construction of the tables holding fact information within the PDS. For an RPASCE application to participate in sharing data with a PDS, it is necessary for the relative order of the hierarchies within that application be the same as the relative ordering of the hierarchies within the PDS. It is not necessary for the hierarchies within an application to have identical values for the order attribute as the PDS hierarchies, but the order of hierarchies relative to each other must be the same.
- **Virtual Hierarchy** – As with virtual hierarchies in a RPASCE application, this property specifies the virtual hierarchy for this shared hierarchy. The name of the virtual hierarchy must be the same as its virtual hierarchy in the application. If this

shared hierarchy has no virtual hierarchy in the application, this property must be left blank.

Shared Facts

Shared facts are entities within the PDS that correspond to measures within a RPASCE application. Like application measures, facts are defined as either scalar or multi-dimensional. Once a RPASCE application has been integrated with a PDS, it is possible to specify mappings of application measures to PDS facts. These mappings allow a RPASCE application to use the fact as it resides within the Oracle database to store the information previously associated with the application measure.

Information can be pulled from and pushed to the Oracle database automatically as a part of application operations, making the fact within the Oracle database the store-of-record for the information. When multiple applications are integrated into a single PDS, there can be several mappings for a single fact, one in each application. When this occurs, the applications can seamlessly share a single version of the fact information without requiring overt integration operations of data duplication and synchronization.

Within the PDS, facts are stored within fact tables. The PDS can store multiple facts within a single table. When this occurs, each fact is represented by a separate column within the table, while the rows of the table represent the positional addresses for the dimensional space of the fact.

By grouping multiple facts within a single table, the PDS can reduce the amount of time required to retrieve information from the Oracle database. Measures that are often read and/or written together and that contain the same fill pattern (or sets of addresses that have data as opposed to addresses for which no data is present) can be quickly accessed together when grouped in a single table.

Conversely, grouping facts together in a single table can have an adverse impact on data access if those facts are not accessed together and if those facts do not tend to have similar fill patterns. For this reason, the assignment of facts to fact tables and the grouping of facts within a table can have a large impact on system performance.

Shared Fact Properties

Shared facts are defined by certain properties. Many of the properties involve the definition of the type of data represented by the fact, while others describe where and how the fact is stored within the fact tables of the PDS. The properties of a shared fact are automatically derived from the application configuration and are read-only within the Integration Tools. The Shared Fact properties are:

Fact Name – This is the identifier of the fact. It is used by RPASCE to determine which fact is required for any given operation. One non-shared measure is mapped to one fact. If the fact name prefix is specified in the Fact Prefix Override property of the application, or in the integration configuration starting pdsappconfigs.xml file, then the fact name is the <fact-name-prefix>_<measure-name>. For shared measures, the fact name is the Shared Fact Name configured within the Data Interface Manager. Refer to the Shared Measures section there.

Fact Label – This is a user label that represents the fact in reporting and user notification and imported from the label of the measure.

Fact Intersection – The intersection of a fact, like the intersection of a measure, describes which dimensions are used to define the address space of the fact. For a application to share a measure’s data with a PDS fact, the application measure must have the same intersection as the PDS fact or a higher intersection than the PDS fact. If the application measure's intersection is higher than the PDS fact, that application measure will be read-only, meaning it cannot be calculated or committed from workbooks. The Fact intersection is derived from the measure’s base intersection unless the Shared Fact Base intersection is specified within Data Interface Manager. In that case, the value of the Shared Fact Base intersection is used as Fact intersection.

Fact Type – The fact type describes the type of data stored within the fact. These types are drawn from the set of defined RPASCE measure types (real, integer, Boolean, Date, and string). For a application to share a measure’s data with a PDS fact, the application measure must have the same type as the PDS fact.

Fact Group – The fact group is an organizational attribute that is used to distribute facts across fact tables within the PDS. A single fact table contains the information for all facts belonging to a single fact group. The efficient grouping of facts can have a large impact on system performance. During the integration configuration generation, RPASCE uses an optimized grouping algorithm to assign facts with a storage database to fact groups. Facts without storage databases are not assigned to fact groups. A discussion of best practices regarding the assignment of facts to fact groups can be found in section 3.3.

Fact Table – The fact table is the physical name of the fact table within the Oracle database that contains the data associated with the fact. This attribute is derived from the fact group and need not be configured separately.

Fact Description – The fact description is a property that describes the context of the data contained within a fact. It can be used for reporting or user notification.

Fact Na Value – The Na value of a fact is the implied value of that fact for any positional address for which the fact’s fact table does not contain a value. It is analogous to the Na value of a measure within an RPASCE application. In order for an application to share a measure’s data with a PDS fact, the application measure must have the same Na value as the PDS fact.

Fact Purge Age – The fact purge age is an attribute used by RPASCE to determine how long information loaded into a fact must be maintained by the system before it is purged as obsolete.

Fact Default Agg – Specify the default agg type of this fact. Its value is imported from the corresponding measure’s default spread type. It is used in fact data aggregation.

Fact Default Spread – Specify the default spread type of this fact. Its value is imported from the corresponding measure’s default spread type. It is used in fact data spreading.

Load Intersection – If the measure’s load intersection is the same as the fact intersection, this field is not populated. Otherwise, the measure’s load intersection value is set into this field. Used to support loading fact data.

Load Agg – If the measure’s load intersection is the same as the fact intersection, this field is not populated. Otherwise, the measure’s load agg value is set into this field. Used to support loading fact data.

Shared Fact Name – The fact name for shared measures. Its value is pulled from the Shared Fact Name configuration within Data Interface Manager. This field is not populated for non-shared measures.

Shared Fact Base Intx - The fact intersection for shared measures. Its value is pulled from the Shared Fact Base intersection within Data Interface Manager. This field is not populated for non-shared measures. Do not populate this field. This value is optional.

File Name – The file name is the name of the flat, input file of the fact. Its value is pulled from corresponding measure's file name configuration within Data Interface Manager. This property is used in both fact data loading and fact grouping.

Report Category – The report category is a string value representing the category where the fact data is reported within data visualization. Its value is pulled from the Report Category configuration in the Data Interface Manager.

Source Fact - The source fact is an attribute used by ORDS to update facts in RPASCE PDS. The Boolean value it contains indicates if the fact is a source fact that is used on the right side of the calculation expression. By default, this attribute is False. Users must set it to True to indicate it is a source fact so that ORDS can update this source fact in PDS through a web service call.

Integration Map

The integration map is the structure that describes which application measures share data with the facts contained within the PDS. The map is made up of a set of entries; each entry describes the fact within the PDS and the application and measure that participate in the sharing.

Integration Map Properties

Integration map properties include:

- **Fact Name** – The fact within the PDS that stores the data to be used by the application measure.
- **Application** – An identifier that describes the application for which the measure participating in the sharing is defined.
- **Measure** – The measure within the application that participates in the data sharing.
- **Outbound Integration** – Retail bulk data integration is used to transfer bulk data between other retail products and RPASCE PDS. The integration can be two-way, with the transfer from RPASCE PDS to other retail products called outbound and the reverse direction called inbound. The outbound integration property is a Boolean flag that indicates whether or not the shared fact is included in the outbound bulk data integration. By default, this property is false. To include the shared fact in outbound integration, this property must be explicitly set to true.

Integration Map Constraints

For a measure to be eligible to share data with a PDS fact, the measure must have compatible values for several of the properties of the fact.

- **Measure Type** – The data type of the measure must be the same as the data type of the fact.
- **Measure Na Value** – The Na value of the measure must be the same as the Na value of the fact.
- **Measure Base Intersection** – The base intersection of the measure must be the same as the fact intersection, or in the case of shared measures, the base intersection must be

either at or higher than the fact intersection. Note that, in configurations that make use of the hierarchy indirection feature of the Configuration Tools, the literal value of the intersection property of a measure may differ from the fact intersection due to use of the RPASCE Name dimension attribute or labeled intersections; in such cases, the RPASCE internal intersection is used in place of the literal value of the measure intersection property for this check.

Application Configurations

The Integration Tool uses the application configurations to identify the application configurations of the applications that are being integrated into the PDS. This information is used internally by the Integration Tool for many of its processes.

Application Configuration Properties

Application configuration properties include:

- **Application Identifier** – This is an identifier that describes the application within the integration map.
- **Configuration Location** – This is the location of the configuration that represents the application. It is used by the Integration Tool to determine which version of a configuration should be used by the Integration Tool.

Integration Tool

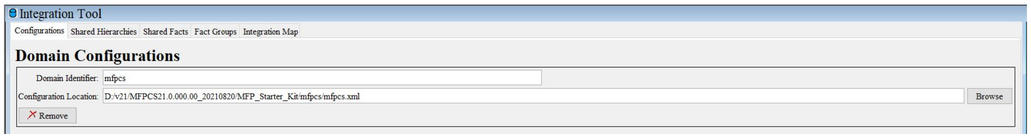
The Integration Tool can be used to create and maintain the integration configuration. This tool specifies the metadata of the PDS by creating shared hierarchies, dimensions, and facts. You can maintain the integration map that describes which application measures participate in data sharing with the PDS shared facts.

Like other tools used within the RPASCE Configuration Tools, an instance of the Integration Tool is tied to an application configuration or multiple application configurations. An XML document, called the integration configuration, is used to store all integration-related information.

Working with Application Information

The Configurations tab displays application configurations with an integration configuration.

Figure 10-1 Configuration Tab of the Integration Tool



Domain Configurations

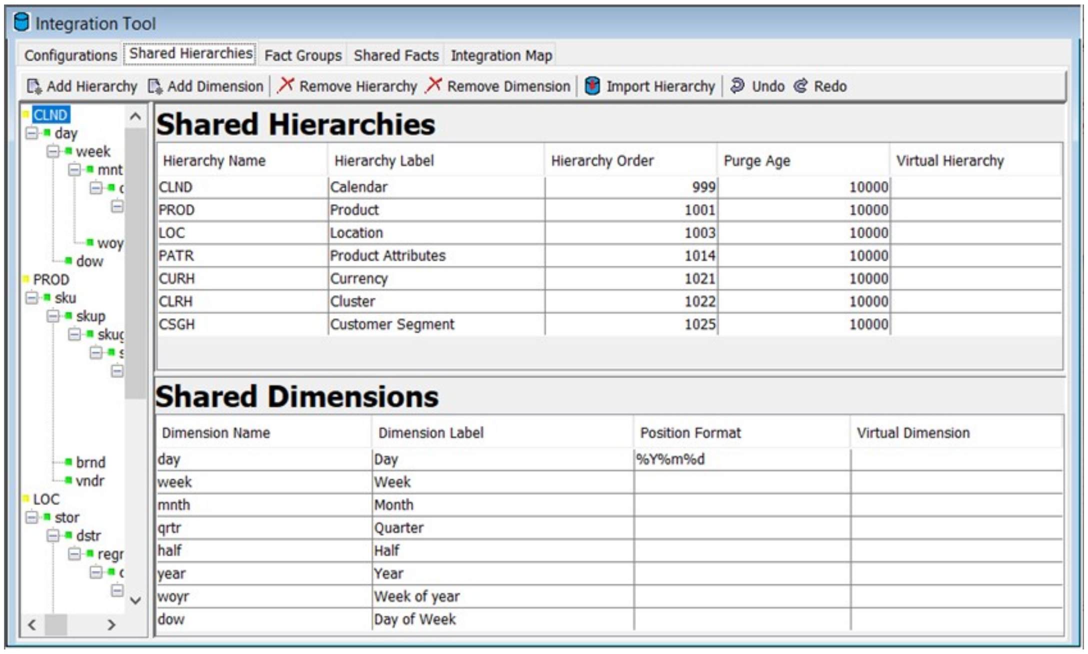
Domain Identifier: mfpcs

Configuration Location: D:/v21/MFPCS21.0.000.00_20210820/MFP_Starter_Kit/mfpcs/mfpcs.xml

Working with Shared Hierarchies

The Shared Hierarchies tab is used to display the shared hierarchies and dimensions that exist within the PDS.

Figure 10-2   Shared Hierarchies Tab of the Integration Tool



Shared Hierarchies

Hierarchy Name	Hierarchy Label	Hierarchy Order	Purge Age	Virtual Hierarchy
CLND	Calendar	999	10000	
PROD	Product	1001	10000	
LOC	Location	1003	10000	
PATR	Product Attributes	1014	10000	
CURH	Currency	1021	10000	
CLRH	Cluster	1022	10000	
CSGH	Customer Segment	1025	10000	

Shared Dimensions

Dimension Name	Dimension Label	Position Format	Virtual Dimension
day	Day	%Y%m%d	
week	Week		
mnth	Month		
qrtr	Quarter		
half	Half		
year	Year		
woyr	Week of year		
dow	Day of Week		

The Shared Hierarchies tab is similar in appearance and functionality to the Hierarchy Tool of the Configuration Tools.

Shared Hierarchies Tab Validations

The Integration Tool supports several validations of hierarchy and dimension content. As with the traditional Configuration Tools, these validations are evaluated in real-time as content is rendered by the UI. When a property of a hierarchy or a dimension violates a validity constraint, the field displaying that property will display with the Configuration Tools-standard red text color. In addition, the tool tip for the invalid cell will contain a message describing the validity constraint that has been violated.

The following validity constraints apply to shared hierarchies and dimensions:

Hierarchy Name

- Hierarchy name is a required property. It cannot be empty.
- Hierarchy names must be valid names for RPASCE hierarchies. They cannot exceed four characters in length, must begin with a letter, and can contain only letters and numerals.
- All hierarchy names must be unique. No other element in the Integration Configuration may share the name of a shared hierarchy.

Hierarchy Label

Hierarchy labels are not validated.

Hierarchy Order

- Hierarchy order values must be positive integer values between 999 and 9999.
- The Calendar hierarchy must be registered at order 999.
- No hierarchy can have a lower order value than the hierarchy that precedes it.
- If any registered application configuration is currently loaded in the Configuration Tools, the order of the shared hierarchies will be compared to the application hierarchies in order to validate that the relative ordering of hierarchies is identical.

Hierarchy Purge Age

- Purge Age is a required property. It cannot be empty.
- Purge Age must be a positive integer value.

Virtual Hierarchy

- If the Virtual Hierarchy column is left blank, it indicates that this shared hierarchy has no virtual hierarchy.
- The Integration Tool can support multiple applications, and the Shared Hierarchies table can only specify the common settings among all applications. If application configurations have different Virtual Hierarchy settings, users cannot use shared hierarchies to specify which application has which specific Virtual Hierarchy setting.
- If Virtual Hierarchy is set, it will be validated against all application configurations open in the Integration Tool. If the application configuration contains its original hierarchy, then the hierarchy must have the same virtual hierarchy setting as in the shared hierarchy.

Dimension Name

- Dimension name is a required property. It cannot be empty.
- Dimension names must be valid names for RPASCE dimensions. They cannot exceed four characters in length, must begin with a letter, and can contain only letters and numerals.
- All dimension names must be unique. No other element in the Integration Configuration may share the name of a shared dimension.

Dimension Label

Dimension labels are not validated.

Dimension Position Format

- The root dimension of the Calendar hierarchy must have a defined position format.
- No dimension that is not the root dimension of the Calendar hierarchy may have a defined position format.
- The supplied position format must be a valid RPASCE format.
- The supplied position format must be a valid Oracle date format.

Calendar Hierarchy

If an integration configuration includes the Calendar hierarchy, that hierarchy must include the day dimension.

Virtual Dimension

- If the Virtual Dimension is left blank, this shared dimension has no virtual dimension.
- The Integration Tool can support multiple applications, and the Shared Dimensions table displays the merged settings among all applications. If Virtual Dimension is set, it will be validated against all application configurations open in the Integration Tool. If the application configuration contains its original dimension, then the original dimension must have the same virtual dimension setting as in the shared dimension.

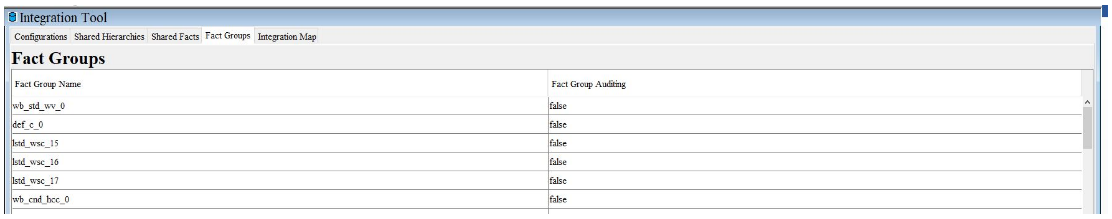
Working with Shared Facts

The Shared Facts tab is used to display the shared facts that exist within the PDS.

Fact Groups Tab

This tab is used to configure the Auditing features. Each fact group can be set to true or false. Each fact group must exist in the next tab of Shared Facts tab. If not, an error occurs, and if it is not resolved, the RPASCE server will ignore the fact group.

Figure 10-3 Fact Groups Tab in the Integration Tool

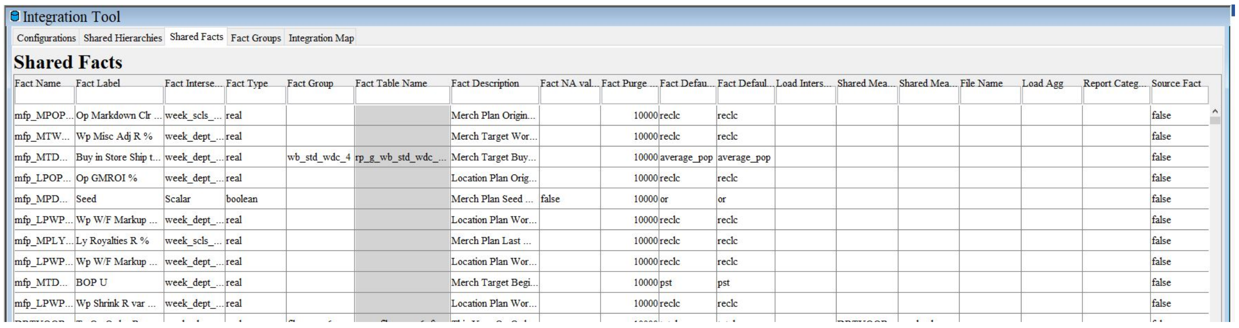


Fact Group Name	Fact Group Auditing
wb_std_wv_0	false
def_c_0	false
lstd_wsc_15	false
lstd_wsc_16	false
lstd_wsc_17	false
wb_cnd_hcc_0	false

Shared Facts Tab

The main feature of the Shared Facts tab is the fact table. This table lists the shared facts that have been defined within the integration configuration.

Figure 10-4   Shared Facts Tab in the Integration Tool



Shared Facts

Fact Name	Fact Label	Fact Interse...	Fact Type	Fact Group	Fact Table Name	Fact Description	Fact NA val...	Fact Purge ...	Fact Defau...	Fact Defaul...	Load Inters...	Shared Mea...	Shared Mea...	File Name	Load Agg	Report Categ...	Source Fact
mfp_MPOP...	Op Markdown Clr ...	week_scls_...	real			Merch Plan Origin...		10000	reclc	reclc							false
mfp_MTW...	Wp Misc Adj R %	week_dept_...	real			Merch Target Wor...		10000	reclc	reclc							false
mfp_MTD...	Buy in Store Ship t...	week_dept_...	real	wb_std_wdc_4	rp_g_wb_std_wdc_...	Merch Target Buy...		10000	average_pop	average_pop							false
mfp_LPOP...	Op GMROI %	week_dept_...	real			Location Plan Orig...		10000	reclc	reclc							false
mfp_MPD...	Seed	Scalar	boolean			Merch Plan Seed ...	false	10000	or	or							false
mfp_LPWP...	Wp W/F Markup ...	week_dept_...	real			Location Plan Wor...		10000	reclc	reclc							false
mfp_MPLY...	Ly Royalties R %	week_scls_...	real			Merch Plan Last ...		10000	reclc	reclc							false
mfp_LPWP...	Wp W/F Markup ...	week_dept_...	real			Location Plan Wor...		10000	reclc	reclc							false
mfp_MTD...	BOP U	week_dept_...	real			Merch Target Begi...		10000	pst	pst							false
mfp_LPWP...	Wp Shrink R var ...	week_dept_...	real			Location Plan Wor...		10000	reclc	reclc							false

Shared Fact Tab Validations

The Integration Tool supports several validations of fact content. As with the traditional Configuration Tools, these validations are evaluated in real-time as content is rendered by the UI. When a property of a fact violates a validity constraint, the field displaying that property is displayed with the Configuration Tools-standard red text color. In addition, the tooltip for the invalid cell contains a message describing the validity constraint that has been violated.

The following validity constraints apply to shared facts.

Fact Name

- Fact name is a required property. It cannot be empty.
- Fact names must be between one and thirty characters in length. They must begin with a letter and can contain letters, numerals, and underscores.
- All fact names must be unique. No other element in the integration configuration may share the name with a fact.
- Some names are reserved for use by RPASCE. In addition, fact names may not end with certain strings, such as _id, _stg, _ft.

Fact Label

Fact labels are not validated.

Fact Intersection

- Fact intersection is a required property. It cannot be empty, but may be scalar.
- All dimension references in a fact's intersection must resolve to dimensions that exist in the PDS.
- Intersections cannot contain references to more than one dimension in any single hierarchy within a fact intersection.

Fact Type

- Fact type is a required property. It cannot be empty.
- Fact type must conform to one of a set of valid RPASCE measure types.

Fact Group

- Fact group is a required property. It cannot be empty.
- Fact groups must be between one and twenty characters.
- Fact groups must begin with a letter and can contain letters, numerals, and underscores.
- No two facts may have the same fact group if they do not have the same intersection.
- All facts within a fact group belong to the same set of applications.

Fact Table

Fact table is a derived property and so is not itself validated.

Fact Description

Fact descriptions are not validated.

Fact Na Value

- Fact Na Value is a required property. It cannot be empty.
- The value provided for fact Na Value must be valid, based upon the type of the measure. The guidelines for what values are valid for any given type are identical to those used for measure Na Values.

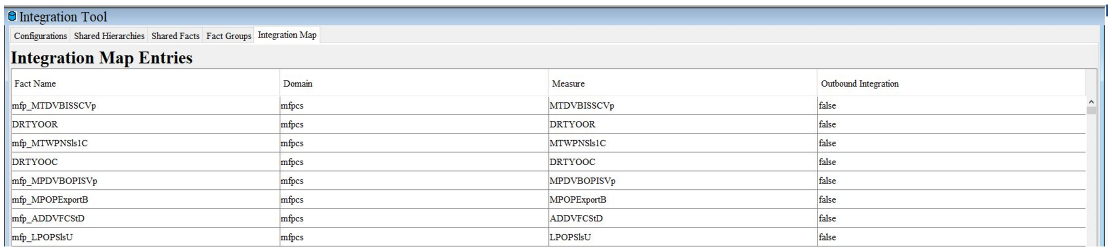
Fact Purge Age

Fact purge age is not validated.

Working with the Integration Map

Use the Integration Map tab of the Integration Tool to display configure the application measures that will share data with a PDS fact.

Figure 10-5 Integration Map Tab in the Integration Tool



Integration Map Entries

Fact Name	Domain	Measure	Outbound Integration
mfp_MTDVBISSCVp	mfpcs	MTDVBISSCVp	false
DRTYOOR	mfpcs	DRTYOOR	false
mfp_MTWPNSls1C	mfpcs	MTWPNSls1C	false
DRTYOOC	mfpcs	DRTYOOC	false
mfp_MPDVBOPISVp	mfpcs	MPDVBOPISVp	false
mfp_MPOPExportB	mfpcs	MPOPExportB	false
mfp_ADDVFCStD	mfpcs	ADDVFCStD	false
mfp_LPOPSlsU	mfpcs	LPOPSlsU	false

The primary feature of the Integration Map tab is the integration map entries table. You can view the application measures that will participate in sharing data with the PDS.

Fact

- Fact is a required property. It cannot be empty.
- The name listed for fact must correspond to a fact defined in the Shared Facts tab.

Application

- Application is a required property. It cannot be empty. It references one application.
- The name listed for the domain must correspond to a application registered in the Application Configurations tab.

Measure

- Measure is a required property. It cannot be empty.
- If the application configuration for the application specified for this entry is loaded into the Configuration Tools, the value of measure will be validated and must correspond to a measure in that application.
- If the application configuration for the application specified for this entry is loaded into the Configuration Tools, the specified measure must share the values for type, intersection, and NA Value with the fact specified for this entry.

Outbound Integration

- Outbound Integration is an optional property. When not set, this property uses the default value of false, indicating that the shared fact is not in the Retail Bulk Data Outbound Integration.
- To include the shared fact in the outbound integration, click the field and select true in the list.

11 Integration Configuration Generation

The Integration Configuration is generated automatically from the application configuration(s) using the RPASCE fact grouping algorithm. The philosophy for the fact grouping algorithm is covered in Fact Grouping Best Practice. RPASCE automatically imports all application hierarchies, dimensions, and measures to create the shared hierarchies, shared dimensions, facts, and integration map inside the integration configuration. One application hierarchy is mapped to one shared hierarchy, and one application dimension is mapped to one shared dimension. In the case of multiple applications, their hierarchies and dimensions are merged following the hierarchy merging guidelines.

Facts are created from application measures. One non-shared measure is mapped to one fact. For shared measures, multiple measures are mapped to one fact, with at most only one such measure limited from one application. Measures without a storage database are still created as facts and stored in the PDS metatable. However, such facts have no data to store and do not need fact tables and are thus not assigned to any fact group.

To change the integration configuration, users must edit the application configuration first, save it, and then regenerate the integration configuration.

There are two ways to generate the integration configuration. The GUI way is to use the Integration Tool on Windows, and the Shell way is to run rpasInstall at the command line in the UNIX-like Cygwin or the UNIX environment. The Integration Tool no longer requires extensive configuration. It is mainly used for generating and displaying the integration configuration. The individual components, Add Hierarchy, Add Dimension, Add Fact, Import Fact, Remove Fact, Add Integration Map, and so on, in prior versions, have been deprecated and removed.

Important Configuration Items for Fact and Grouping

The application configuration determines the outcome of the integration configuration. Fact grouping is an essential step in generating an optimal integration configuration that provides better RPASCE performance on PDS. The following sections summarize several configuration items that directly impact the fact creation and grouping. The fact grouping algorithm is covered in more detail later in this chapter.

Batch Control File

When a rule group is listed as “group” in the application’s batch control file, the batch_calc_list.txt file, RPASCE automatically treat this rule group as a Batch Rule Group during fact grouping.

For example, the rule group Batch_RMS_G is listed as a “group” in the batch_calc_list.txt file. The rule group Batch_RMS_G will be treated as Batch rule group even when the batch=true rule set property is not configured for the rule set containing this rule group.

```
# Calc Set for Batch Aggregation Weekly
calc_week | group | Batch_RMS_G
```

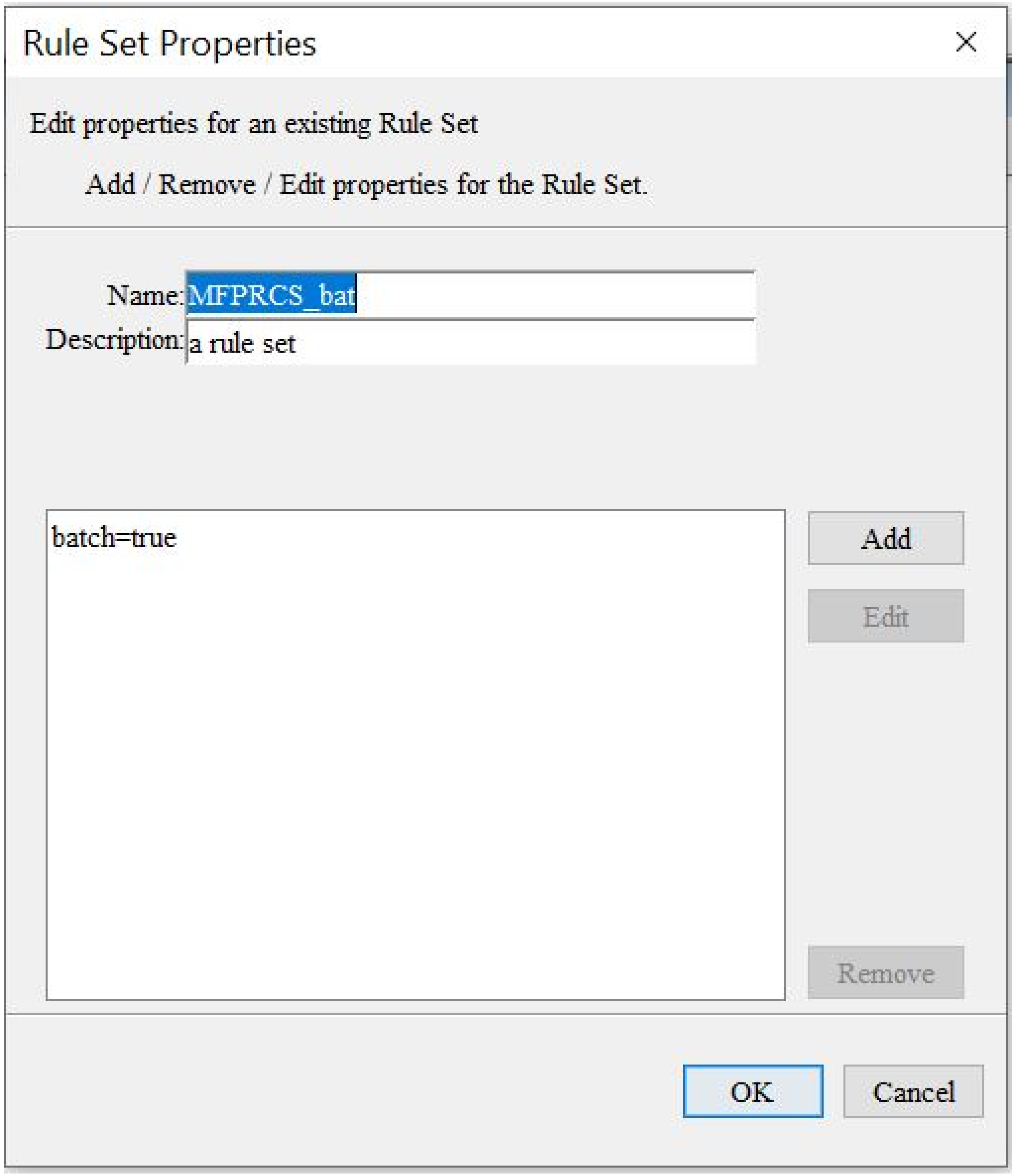
Batch Rule Set Property

Using the batch rule set property, users indicate all rule groups within a rule set are to be treated as batch rule groups during fact grouping.

The batch rule set must be identified inside the application configuration by adding the Rule Set property batch=true.

The Batch Rule Set property is still supported but no longer required. RPASCE now checks the batch control file to find out which rule groups are used as batch.

Figure 11-1 Identify Batch Rule Set by Adding Rule Set Property


Rule Set Properties

Edit properties for an existing Rule Set

Add / Remove / Edit properties for the Rule Set.

Name: MFPRCS_bat

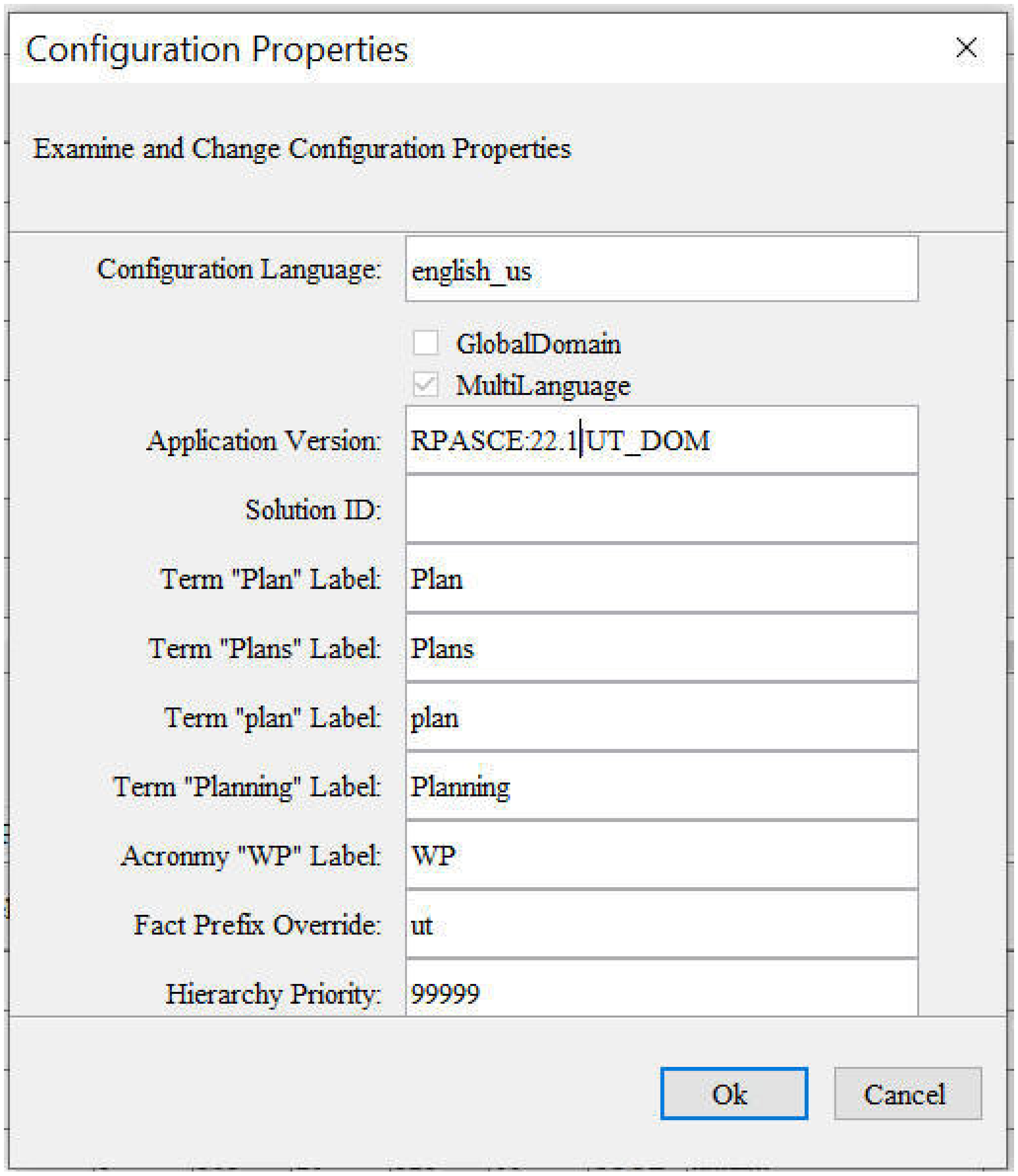
Description: a rule set

batch=true

Fact Prefix Override

In the following example, the Fact Prefix Override is customized to ut for the application RPAS_UT.

Figure 11-2 Fact Prefix Override Property In Configuration Properties



Configuration Properties

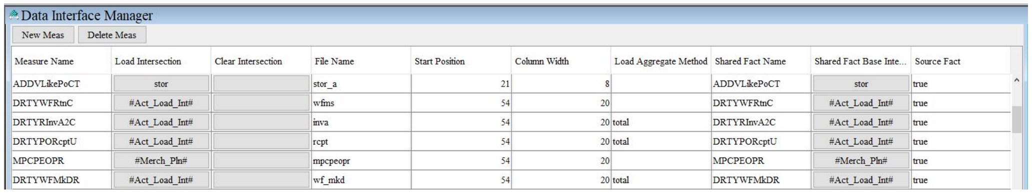
Examine and Change Configuration Properties

Configuration Language:	english_us
	☐ GlobalDomain
	☑ MultiLanguage
Application Version:	RPASCE:22.1UT_DOM
Solution ID:	
Term "Plan" Label:	Plan
Term "Plans" Label:	Plans
Term "plan" Label:	plan
Term "Planning" Label:	Planning
Acronmy "WP" Label:	WP
Fact Prefix Override:	ut
Hierarchy Priority:	99999

Shared Fact Name, Shared Fact Base Intersection, and File Name

Shared measures must have the Shared Fact Name specified within the Data Interface Manager. The Shared Fact Base Intersection value is optional. For facts whose data is loaded via flat input files, the File Name properties of the corresponding measures must be specified in the Data Interface Manager. Refer to the Data Interface Manager chapter for details.

Figure 11-3 Data Interface Manager



Measure Name	Load Intersection	Clear Intersection	File Name	Start Position	Column Width	Load Aggregate Method	Shared Fact Name	Shared Fact Base Inte...	Source Fact
ADDVLikePoCT	stor		stor_a	21	8		ADDVLikePoCT	stor	true
DRTYWFRtnC	#Act_Load_Int#		wfms	54	20		DRTYWFRtnC	#Act_Load_Int#	true
DRTYRInvA2C	#Act_Load_Int#		inva	54	20	total	DRTYRInvA2C	#Act_Load_Int#	true
DRTYPORcptU	#Act_Load_Int#		rcpt	54	20	total	DRTYPORcptU	#Act_Load_Int#	true
MPCPEOPR	#Merch_Pln#		mpcpeopr	54	20		MPCPEOPR	#Merch_Pln#	true
DRTYWFMkDR	#Act_Load_Int#		wf_mkd	54	20	total	DRTYWFMkDR	#Act_Load_Int#	true

Interface Configuration File

The interface configuration file has a default name “interface.cfg” and contains the mapping of dimensions/facts in PDS to columns mapped to external tables for each interface. Each application within the PDS may have its own interface configuration file. The presence of the file interface.cfg is optional for integration configuration generation, but if this file exists, it affects this application’s fact grouping. The facts imported via the same interface are assigned to the same fact group if not further subdivided by intersections, commits, and other fact grouping criteria.

The following example shows a block within one interface.cfg. For fact grouping, RPASCE only looks at rows that contain the “DATA” keyword, which indicates these rows are for facts. Breaking the row string into tokens separated by “:”, the first token is the interface name and is used as the interface identifier. The fourth token designates the name of the fact.

For the row `RSE_FCST_DEMAND_EXP:MPI:DATA:mfp_MPWPDmdI1U:REG_PR_SLS_QTY:`, the fact is `mfp_MPWPDmdI1U` and identifier is `RSE_FCST_DEMAND_EXP`.

```
RSE_FCST_DEMAND_EXP:MPI:DATA:mfp_MPWPDmdI1U:REG_PR_SLS_QTY:
RSE_FCST_DEMAND_EXP:MPI:DATA:mfp_MPWPDmdI1R:REG_PR_SLS_AMT:
RSE_FCST_DEMAND_EXP:MPI:DATA:mfp_MPWPDmdI2U:CLR_SLS_QTY:
RSE_FCST_DEMAND_EXP:MPI:DATA:mfp_MPWPDmdI2R:CLR_SLS_AMT:
RSE_FCST_DEMAND_EXP:MPI:DATA:mfp_mpwprtniu:RET_QTY:
RSE_FCST_DEMAND_EXP:MPI:DATA:mfp_mpwprtnir:RET_AMT:
RSE_FCST_DEMAND_EXP:MPI:FILTER::CAL_HIER_LEVEL:Fiscal Week
```

Only the facts having the same interface identifier can be assigned to the same fact group. Of course, other fact grouping criteria, such as fact intersection and so on, may subdivide this interface grouping further into more fact groups.

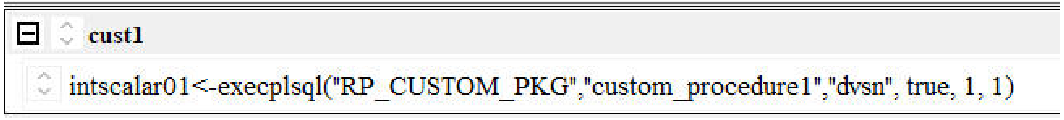
Customer-Managed Facts

The customer-managed facts must reside in their own fact groups, separate from regular RPASCE facts. To facilitate fact grouping, RPASCE historically requires two-level configuration (as outlined below) for customer-managed rule sets and rule groups. . Although such configuration is still supported, it is no longer necessary since RPASCE recognize the execplsql special expressions used in these rule groups and thus treat them accordingly during fact grouping.

- **Pure customer-managed rule set**. This is the situation where all rule groups within this rule set contain the special expression *execplsql*. Then users must not set the rule set property batch=true for this customer-managed rule set for fact grouping correctly.

- **Mixed rule set.** When a batch rule set contains rule groups using the execplsql special expression and regular RPASCE rules groups, if the batch=true property is configured at the rule set level, then users must configure the property CMF=true for each rule group containing execplsql.

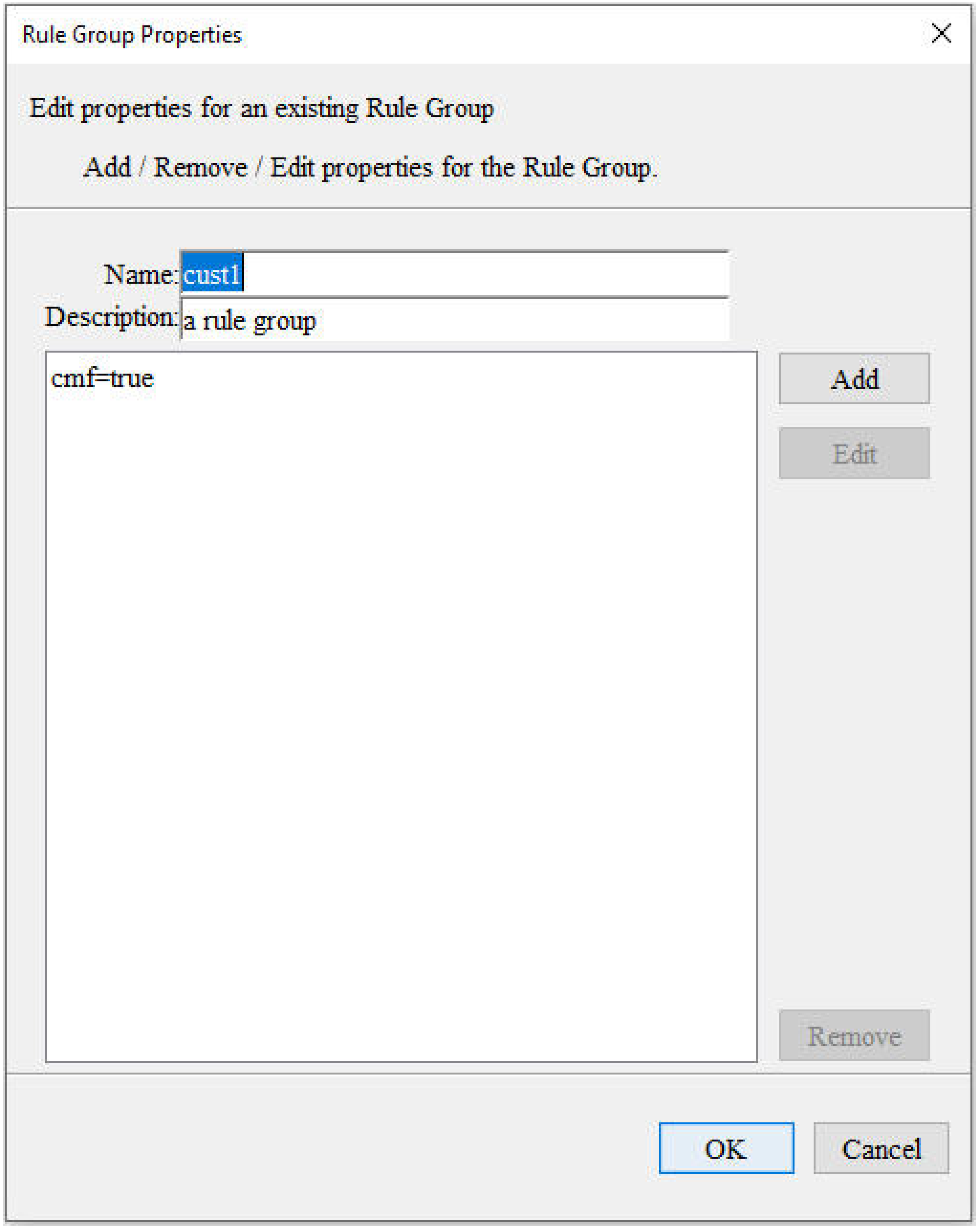
Figure 11-4 Rule Group Containing execplsql Special Expression



cust1

```
intscalar01<-execplsql("RP_CUSTOM_PKG","custom_procedure1","dvsn", true, 1, 1)
```

Figure 11-5 Rule Group Properties Window



Rule Group Properties

Edit properties for an existing Rule Group

Add / Remove / Edit properties for the Rule Group.

Name: cust1

Description: a rule group

cmf=true

Note:

The customer-managed facts should be used in rule groups that contain special expressions execplsql. Refer tothe RAP Extension Guide for details on Innovation Workbench and customer-managed facts.

Rule Property Hint

Currently the integration configuration is automatically generated by RPASCE internally using a sophisticated algorithm based on the application configurations. Sometimes, users want to have a way to hint for the fact creation or fact grouping based on their specific business logic in order to achieve optimal performance results.

RPASCE provides this customization by letting users configure the specific rule properties for the specific rules and then processing accordingly.

Two new rule properties are supported, Peer and Subgroup.

Hint for Peer Measures

In one application, one or more measures can share the same data with the source measure. For example, some large volume RDF measures are copies of other measures. They are calculated as A=B, with A and B on the same intersection.

Performance testing has demonstrated that if these measures use the same fact, instead of one fact for one measure, to store data, the performance will improve significantly.

Such measures are called *Peer Measures*. During Integration Configuration generation, RPASCE consider this hint indicated by the users and assign peer measures to the same fact if the indicated measures meet the following criteria andare allowed by the RPASCE internal batch dependency graph analyzer. Otherwise, this user hint is simply ignored by the fact grouping.

All peer measures that share the same fact must belong to the same application. The peer measures cannot be shared measures that share facts with other applications.

Qualification Requirements for Peer Measures

- In the batch rule group, the rule with the expression A=B must have the rule property peer=true configured.
- The measures A and B must also satisfy the following conditions:
  - A and B are not shared measures with other applications.
  - A is not calculated in any other batch rule group or in any expressions in the batch control file.
  - A is not committed in any commit rule group in the current application.
  - A is not writable and insertable into the measure analysis workbook - means: !(insertable==true AND (baseState==write || aggState==write))

In the case A and B can be peer measures, A is considered a peer for measure B; B is the source peer measure, while A is the non-source peer measure.

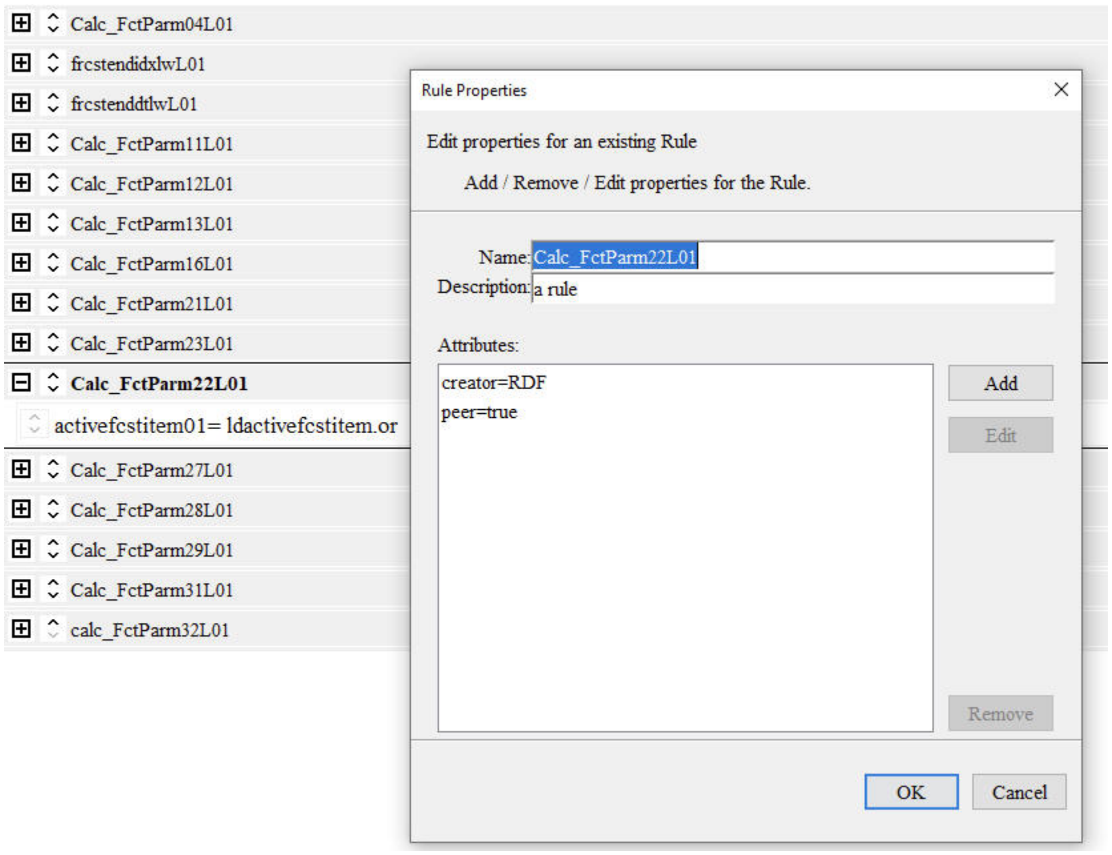
Set Rule Property "peer=true"

For example, the batch rule group RDF_batch has the rule Calc_FctParm22L01 with the expression as :

activefcstitem01= ldactivefcstitem.or

The rule Calc_FctParm22L01 has the rule property peer=true added. The RPASCE Batch Dependency Graph will take this rule property and consider it when it computes the peer measure set of the application.

Figure 11-6 Configure Rule Property Peer for Peer Measure


Calc_FctParm04L01
frcstendidxlwL01
frcstenddtlwL01
Calc_FctParm11L01
Calc_FctParm12L01
Calc_FctParm13L01
Calc_FctParm16L01
Calc_FctParm21L01
Calc_FctParm23L01
Calc_FctParm22L01
activefcstitem01= ldactivefcstitem.or
Calc_FctParm27L01
Calc_FctParm28L01
Calc_FctParm29L01
Calc_FctParm31L01
calc_FctParm32L01

Rule Properties

Edit properties for an existing Rule

Add / Remove / Edit properties for the Rule.

Name: Calc_FctParm22L01
Description: a rule

Attributes:
creator=RDF
peer=true

Hint for SubGroup

The RPASCE batch dependency graph analyzer splits expressions from one batch rule group into multiple subgroups. Based on fact grouping algorithm, the left-hand side facts of the same subgroup will be assigned to the same fact group if no other grouping criteria split them. However, in certain situations, users prefer certain rule expressions to be assigned to the same subgroup, so that their left-hand measures can be assigned to the same fact group. Users wants a way to hint to the RPASCE batch dependency graph analyzer for this subgrouping preference in order to achieve optimal performance.

RPASCE supports user customization using the rule property subgroup.

For example, the batch Rule Group RDF_batch rule group contains the following two rules. Without a customer hint, these two rule expressions are assigned to two subgroups, and thus the facts frcststartid01 and frcstendid01 are assigned to two different fact groups although their fact intersections are the same.

Rule: Calc_FctParm31L01

Expr:
frcststartid01=if(frcststartidx01>-1,if(spdfrcstind01,frcststartidxlw01+frcstclndfirstID01,frcststartidx01+prepclndfirstID01),-1)

Rule: calc_FctParm32L01 Expr:
frcstendid01=if(frcstendidx01>-1,if(spdfrcstind01,frcstendidxlw01+frcstclndfirstID01,frcstendidx01+prepclndfirstID01),-1)

Set Rule Property subgroup

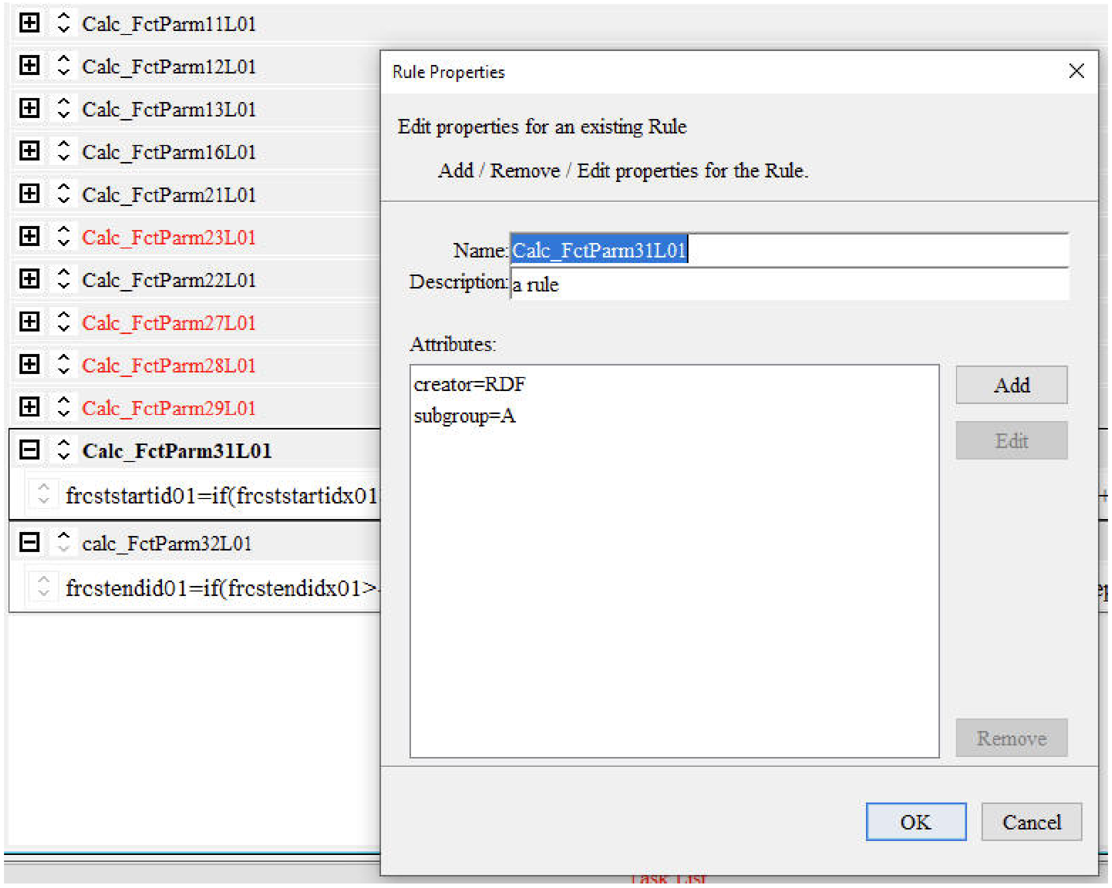
Users can add the rule property subgroup=A to both rule Calc_FctParm31L01 and calc_FctParm32L01 to indicate that these two rules must be assigned to the same subgroup.

The subgroup property A is a token to indicate that these two rules have the same subgroup value and thus must be assigned to the same subgroup, if possible.

Within one batch rule group, the rules having the same subgroup value hint to be assigned into the same subgroup.

Note: If this subgroup hint is not allowed by the RPASCE internal batch dependency graph analyzer, then the user hint will be simply ignored in the fact grouping. No error will occur.

Figure 11-7 Configure Rule Property Subgroup



Calc_FctParm11L01
Calc_FctParm12L01
Calc_FctParm13L01
Calc_FctParm16L01
Calc_FctParm21L01
Calc_FctParm23L01
Calc_FctParm22L01
Calc_FctParm27L01
Calc_FctParm28L01
Calc_FctParm29L01
Calc_FctParm31L01
frcststartid01=if(frcststartidx01
calc_FctParm32L01
frcstendid01=if(frcstendidx01>

Rule Properties

Edit properties for an existing Rule

Add / Remove / Edit properties for the Rule.

Name: Calc_FctParm31L01
Description: a rule

Attributes:

creator=RDF
subgroup=A

Consideration of Batch Control Files

The free expressions with the batch control files can impact the processing of rule property hint. To consider batch control files during integration configuration generation, the paths to the GA and Custom versions of batch control files must be given in the pdsappconfigs.xml file. Refer to the example pdsappconfigs.xml file in the later section of “Generate Integration Configuration – Integration Tools”.

Fact Grouping Best Practices

With the addition of shared facts within the PDS, a performance consideration becomes important within RPASCE applications. Operations that access the Oracle database are optimized to read or write fact data. Internal testing of PDS operations has shown that the organization of facts into fact groups can have a significant impact on the amount of time required to perform operations within the PDS.

Grouping Based on Concurrent Access

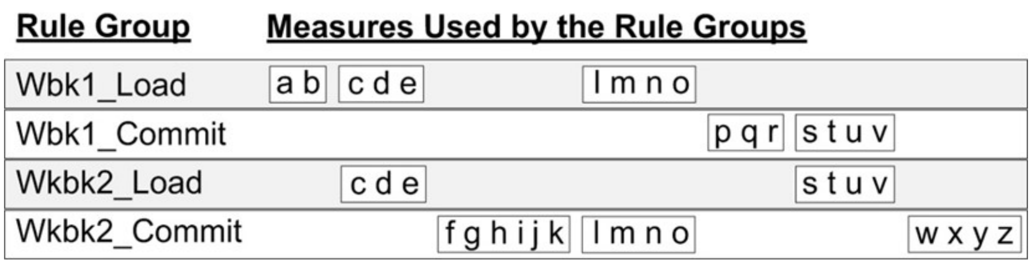
Because the tables that store the fact data within the PDS can contain multiple facts, processing operations against those tables are affected not only by the number of populated values of a single fact, but also of all other facts within the table. This becomes even more of a concern when writing data to the fact tables, as those facts being updated by the write operation must be merged with other facts within the fact table that are not affected by the write.

For this reason, it is possible to design the tables of the PDS so that every fact is contained within a separate table. While this would prevent performance issues related to having multiple facts within a single table, it would not result in optimal performance. If facts that are read and/or written together are placed within a single table, they can be processed together, greatly improving performance.

It is therefore desirable to group facts together in the fact tables, but to do so using a distribution that groups facts accessed together into common tables, but that excludes facts that would not benefit from being grouped into those tables.

To illustrate, consider the following scenario. Assume two workbooks with measures that load from and commit to a partially overlapping set of facts. The sets of facts associated with the load and commit rule groups of the two workbooks have a distribution of facts being loaded and committed (listed as a through z) such as the following:

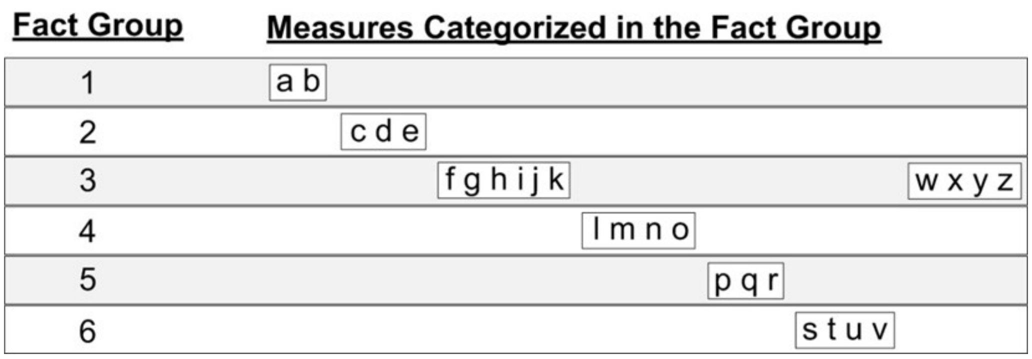
Figure 11-8 Example of the Distribution of Loaded and Committed Facts



Rule Group	Measures Used by the Rule Groups						
Wbk1_Load	a b	c d e		l m n o			
Wbk1_Commit					p q r	s t u v	
Wkbk2_Load		c d e				s t u v	
Wkbk2_Commit			f g h i j k	l m n o			w x y z

In such a case, the measures being loaded and committed can be organized into groups such as:

Figure 11-9 Sample Grouping of Facts for Rule Groups



Fact Group	Measures Categorized in the Fact Group						
1	a b						
2		c d e					
3			f g h i j k				w x y z
4				l m n o			
5					p q r		
6						s t u v	

By organizing the facts into the groups shown in Figure 11-9, the performance benefit from grouping facts accessed together into groups is maximized and the performance hit from having additional facts within the groups that are not involved in the access is avoided.

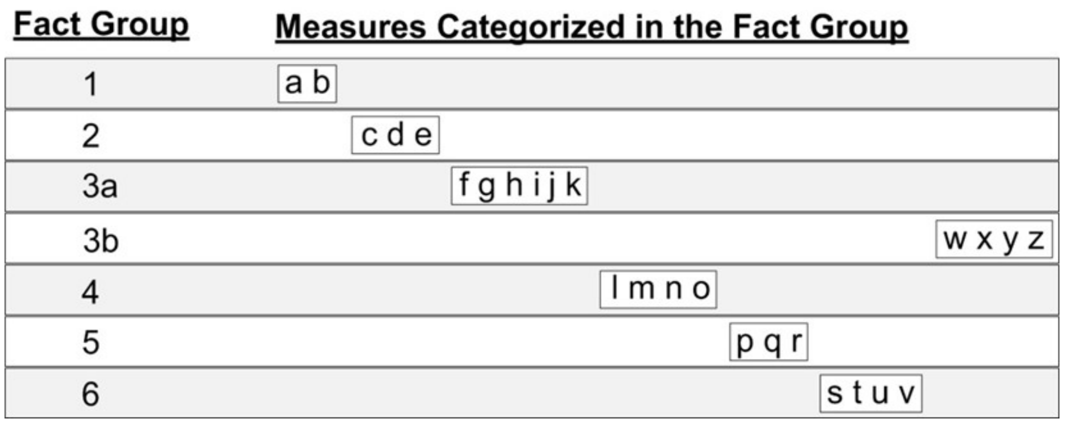
Conditional Commits of Facts

When performing fact analysis such as previously described, the presence of conditional commits must be taken into consideration. Some commit rules include a condition that acts as a mask to prevent some range of the available data from being committed. Take for example a measure that only commits values for non-elapsed periods:

Measure1.master = if (current > today, Measure1, ignore)

Because of the way RPASCE processes conditional commits, facts committed using different conditions must be separated from each other, as each condition must be merged separately. In the previous example, if it was determined that the measures associated with facts w, x, y, and z are committed using the same condition, but that condition was not used for the measures associated with facts f, g, h, I, j, and k, then it is preferable to assign w, x, y and z to a separate fact group from that used for f, g, h, i, j, and k.

Figure 11-10   Modified Grouping of Facts to Accommodate Conditional Commits



Fact Group	Measures Categorized in the Fact Group
1	a b
2	c d e
3a	f g h i j k
3b	w x y z
4	l m n o
5	p q r
6	s t u v

Fact Group Assignment Process

Internal testing suggests that merging data into the Oracle database is the process whose overall performance is impacted the most by fact grouping. For this reason, it is recommended that facts be assigned to fact groups in such a way as to optimize the writing of data to the PDS.

Facts are assigned by looping through each application and each solution within the application. For example, if multiple applications such as MFPRCS and RDF share the same PDS instance, their facts, except the facts for shared measures, are stored in their own Oracle tables, even if the fact intersections and other criteria are the same. Within one application, the facts are grouped by the solution.

The following example process determines the fact groups to use for the facts within the PDS and determines which facts should be assigned to each fact group.

1. Each position filter measure is assigned into its own fact group
2. For the remaining facts, create a fact group for each unique intersection used by facts within the PDS.

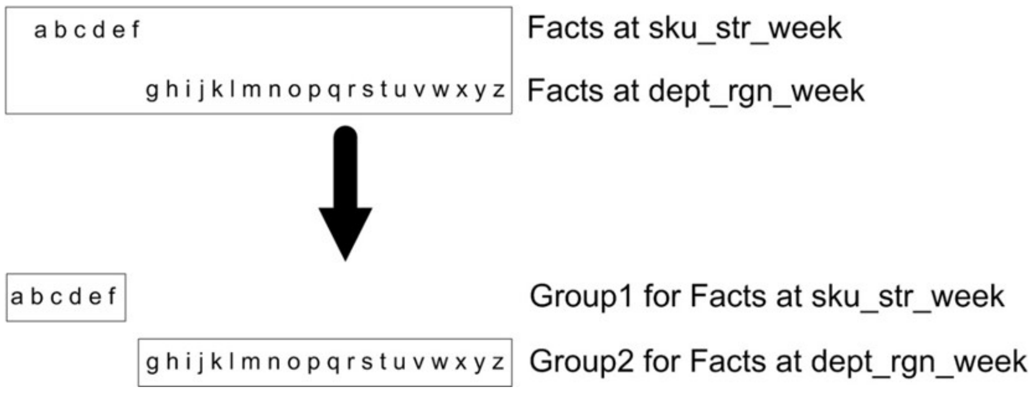
3. Divide each group from step two into subgroups such that facts loaded via the same input file (that is, with the same File name) or the same interface configuration mapping, are assigned to the same fact group.
4. Divide each group from step three into subgroups such that facts committed or loaded together are partitioned from the other facts within the group from step one.
5. Divide each group from step four into subgroups such that facts used in batch are partitioned from non-batch facts.
6. Divide each group from step five, if necessary, to accommodate the use of conditional commits.

Create Groups Based Upon Intersection

All facts are defined along an intersection. The PDS does not allow facts with different intersections to be stored together in a single table. Because the fact group is the entity that corresponds to a single table within the Oracle database, it is impossible to place facts with different intersections within a single fact group.

Once facts have been assigned to groups based upon the fact intersection, the set of facts for the PDS is divided into a number of pools. Treat each of these pools as an input to the second step of the fact grouping process.

Figure 11-11 Dividing Global Fact Pool by Fact Intersection


a b c d e f
g h i j k l m n o p q r s t u v w x y z
Facts at sku_str_week
Facts at dept_rgn_week
a b c d e f
Group1 for Facts at sku_str_week
g h i j k l m n o p q r s t u v w x y z
Group2 for Facts at dept_rgn_week

Partition Facts Based Upon Data Source

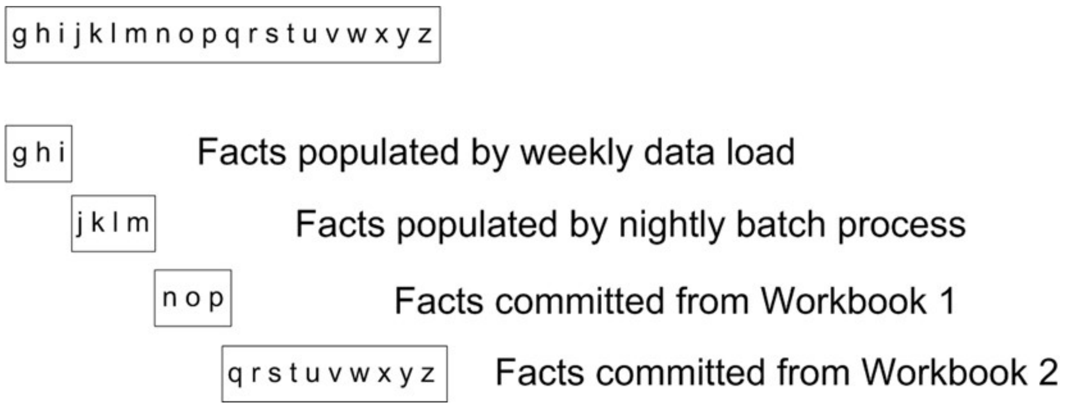
Each group created by grouping by intersection must now be subdivided in order to partition the facts based upon the process by which their information enters the PDS. Fact data is assumed to enter the PDS through one of three processes: workbook commit, batch process, or import from external system (either by a data load or through the PDS interface tables).

For each of the previous processes, it should be possible to determine the set of facts whose data enters the PDS in each operation. These sets of facts must be used to partition the initial fact groups to create individual sub-groups for the facts in each process.

Note that it is possible that some facts may be populated from more than one source. For example, a fact may initially be populated with raw data from an external source that is then transformed by a batch process or workbook prior to use by other processes within the application. This possibility is potentially more likely when dealing with applications that have never been integrated within the PDS.

When dealing with a fact with multiple sources, the sources must be prioritized based upon which process is the most time-critical. Once this prioritization has been done, the grouping suggested by the highest priority must be used. In cases in which the priority cannot be identified, or when no process's grouping provides an acceptable trade-off, the fact or facts in question must be assigned to a separate fact group.

Figure 11-12 Subdivision of Group by Data Source


g h i j k l m n o p q r s t u v w x y z
g h i
Facts populated by weekly data load
j k l m
Facts populated by nightly batch process
n o p
Facts committed from Workbook 1
q r s t u v w x y z
Facts committed from Workbook 2

Partition Groups by Batch Dependency Graph

Each group created by a data source are further subjected to batch dependency graph grouping. RPASCE use the PL/SQL translator to translate the individual Rule Group to PL/SQL procedures. To facilitate the PL/SQL translation, RPASCE CalcEngine processes each batch rule group into multiple subgroups such that each subgroup can be individually translated into PL/SQL codes. For performance reasons, the PL/SQL process for one subgroup should work on its own fact tables, instead of sharing the tables with other subgroups.

Here are the guidelines for the CalcEngine to generate subgroups:

Each subgroup must have the same type of expression. A special expression and a normal expression must not reside in the same subgroup. For Special Expressions, a subgroup can only contain the same special expression, for example, timeshift. Also, if two timeshifts run in different modes (for example, one runs in the constant timeshift mode and the other one against a CalendarMap) they are further broken down into smaller subgroups.

Facts used in the left side of a special expression must be in their own database table and cannot be further subdivided by other batch and workbook commit rules.

For normal expressions, this means expressions in a single subgroup must be similar to one another. (For example, the if ... ignore expression is put into one subgroup, while the expression "a+b..." is put into another subgroup.)

One special case is Cycle Group. Facts used in the cyclic group must be in their own Oracle table. In the following example, the facts for the measure Bop and Eop must be assigned to the same fact group.

Example 11-1   Example: one Simplest Cycle Group

```
{
    Bop = if (current == first, InitialStock, lag(Eop))
    Eop = Bop + Receipt - Sales
}
```

Another special case is Boolean measures within a subgroup. For a subgroup that is neither a cycle group or a special expression group and the left-hand facts of this subgroup are only two Boolean facts, if either or none of these two boolean facts appear on the right side of the same subgroup, assign each such Boolean fact into its own fact group.

Logical dependencies are also considered by the RPASCE Calc Engine to determine subgroups. If one expression's RHS is the LHS of another expression, then these two expressions must be separated into different sub-groups.

Batch rule sets have been identified by the rule set property batch=true within the application configuration. For each batch rule group, the RPASCE fact grouping process obtains the list of subgroups from the RPASCE Calc Engine. If the subgroup is non-cyclic, non-special expression and non-boolean (as in the special cases described above), the left-hand side facts of the same rule group are combined and are partitioned into their own group that may be further partitioned by commits..

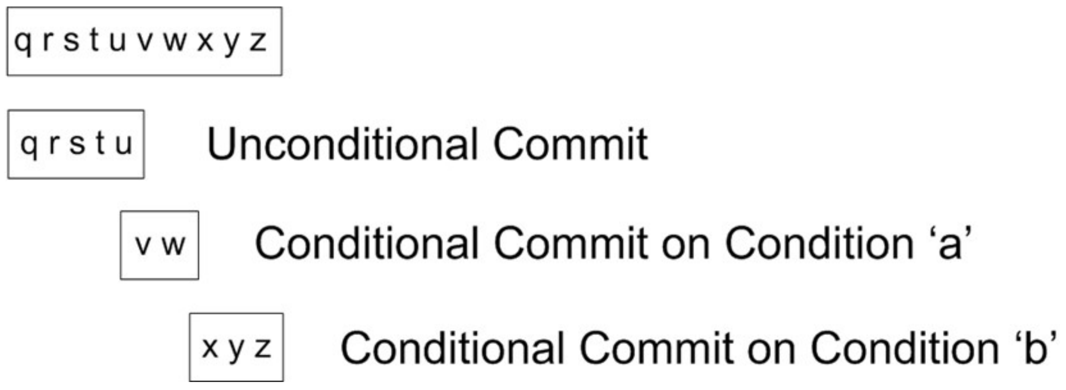
In the fact grouping example provided in this section, the listed facts are not in the batch rule sets and thus the fact groups generated so far are not subdivide by this step.

Partition Groups for Conditional Commits

The final step in determining the grouping for the facts within the PDS is to take the groups identified, as described above, and divide them into partition facts that use conditional commits. Any fact groups created based upon workbook commit processes must be examined.

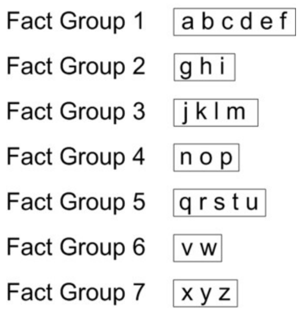
If any rules for the commit make use of a conditional commit (that is, the rule makes use of an if statement to commit only a subset of the measure used with the fact), the processing of any facts that do not make use of a conditional commit will be impacted by those facts that do use the conditional commit. In order to reduce this impact, the group containing the facts for the conditional and non-conditional commits must be divided again to isolate the conditional commit facts from the non-conditional commit measures. In cases where the commit group contains multiple conditional commits that use different conditions, each distinct condition must be assigned to its own group.

Figure 11-13   Subdivision of Group Due to Conditional Commits


q r s t u v w x y z
q r s t u
Unconditional Commit
v w
Conditional Commit on Condition 'a'
x y z
Conditional Commit on Condition 'b'

Once these steps have been completed, the initial set of facts must be partitioned into the set of fact groups that must be optimized for processes that write data into the PDS.

Figure 11-14   Subdivision of Group Due to Conditional Commits



Multi-Application Support

Multiple applications can reside on one common PDS instance. To ensure that multiple applications function properly on one PDS, some guidelines that the application configurations must be considered.

Fact Prefix Override

This property must be unique across all applications on the PDS.

Name Convention

If a hierarchy or dimension has the same RPASCE name among the multi-applications, then the name must mean the same thing. On the other hand, if different RPASCE names are used to represent the same hierarchy or dimension, then such names must be reconciled before deploying the additional application onto the PDS.

For example, LOC is a defined hierarchy in both RDF and MFP, but RDF uses STOR while MFP uses STR for store dimension. The STORE dimension must be reconciled before deploying/patching PDS.

For example, before deploying RDF onto a PDS that already contains MFP, Users must rename all STOR references in RDF configuration to STR.

Minimum Hierarchy CLND and PROD

RPASCE allows both common and non-common hierarchies across multi-applications. The hierarchies with the same name are called common hierarchies. Some hierarchies may exist in one application but not in the other(s) and are not common hierarchies. All applications must at least have the common hierarchy, Calendar (CLND) and Product (PROD).

The root dimension in all CLND hierarchies must be DAY. The DAY dimension must have the same position format, for example, “d%YEAR%MO%DAY”. A valid DAY position format is a valid Date Time Format dictated by Oracle Database.

Merge Hierarchies

In PDS deployment/patching, multi-application hierarchies and dimensions are merged into one PDS hierarchy within the integration configuration generation.

Merge Dimension Rollups

The merged hierarchy takes a union of all hierarchies and dimensions. Common hierarchies must be compatible, although they may have different alternate dimensions. They must not have conflicting dimension rollups across multi-applications.

If the hierarchies are not compatible, the integration configuration cannot be generated and the PDS build/patching will fail.

Example 1: Hierarchy Rollup Scenario in Multiple Applications

Case	App1	App2	Compatible Hierarchy	Merged Hierarchy	Notes
1	Day->mnth->qrtr->year	week->mnth→year	No		Root dim is not day Cannot decide which dim to be the new root of the merged hierarchy. Cannot decide the relationship between day and week
2	Day->mnth->qrtr->year	Day->mnth->year->qrtr	No		Qrtr and year have conflicting dimension rollup

Case	App1	App2	Compatible Hierarchy	Merged Hierarchy	Notes
3	styl->scls->clss	sku->styl->clss	Yes	Sku->styl->scls->clss	RPASCE supports patching new root dimension
4	Sku->scls	Sku->skup	Yes	Sku->scls, sku->skup	New leaf dimension is supported
5	day->mnth->qrtr→year	day->week->biweek->mnth→year	Yes	day->week->biweek->mnth→qrtr→year	New in between dims are supported
6	Sku->stco->scls	Stco->new1->new2	Yes	Sku->stco->scls Stco->new1–>new2	New althernate branch is supported

Example 2: In the Product hierarchy

App1:

sku - skup - skug - scls - clss

sku - vndr

App2:

sku - scls - clss - dept

scls - cons

App3:

sku - skup-skug-scls-clss

sku -brnd

sku -vndr

Final merged PROD hierarchy:

sku - vndr

sku - brnd

sku - skup-skug-scls-clss-dept

scls-cons

Merge Hierarchy Order

Each application configuration has order numbers assigned to its hierarchies. During integration configuration generation, RPASCE assigns new order numbers to the merged hierarchies.Although the absolute values of the order number change, the relative hierarchy order in the new PDS hierarchy are preserved as much as possible.

The following two factors can impact the merged hierarchy order:

- Hierarchy Priority Value
- Application Registration Order

The application having the larger Hierarchy Priority value dictates the merged hierarchy order. For example, if RDF has the Hierarchy Priority configured as 200 and MFP has the Hierarchy Priority configured as 100, then the merged hierarchy order will follow RDF's order

.If more than one applications have the same Hierarchy Priority value, then when each application is deployed to the PDS, that is, the Application Registration Order, matters. Whenever an application is deployed or patched to the PDS, a registration order number is assigned by RPASCE. The first deployed application will have number 1 as its registration order. The application registration order is tracked internally by RPASCE and does not require user configuration. The application with the lower registration order means it is registered before others and will dictate the final merged hierarchy order.

Example:

Note:

The RPASCE internal hierarchies such LANG, ADMU, and so on are not merged and are not shown here.

Application 0: Hierarchy Priority is 300

clnd -> atth -> conh -> rulh -> flvh -> glvh -> prod -> loc -> nhir -> pror -> grph -> patr -> locr -> offh -> rdth -> path -> loch -> null

Application 1: Hierarchy Priority is 200

clnd -> prod -> loc -> curh -> vath -> nhir -> cmsh -> satr -> lvlh -> offh -> null

Hierarchy Order after Merging

clnd -> atth -> conh -> rulh -> flvh -> glvh -> prod -> loc -> curh -> vath -> nhir -> pror -> grph -> patr -> locr -> cmsh -> satr -> lvlh -> offh -> rdth -> path -> loch -> null

Merge Virtual Hierarchy

The virtual hierarchy in multiple applications must be compatible. For the common hierarchy, it can have virtual hierarchy defined in one application but not in the other application. RPASCE takes a union of virtual hierarchy settings during merging.

For example, RDF has PROR defined as the virtual hierarchy for PROD, but MFP has no virtual hierarchy defined for PROD. In the merged PDS hierarchy, PROD has PROR as the virtual hierarchy, but not all dimensions in PROD have virtual dimensions. The PROD dimensions that only exist in MFP will not have corresponding virtual dimensions.

If the other application on the PDS instance does not have a PROR or a PROD hierarchy, it is also a valid configuration.

Note that if one application has PROR defined as virtual hierarchy, then in all other applications, if the PROR is defined, it must be a virtual hierarchy and cannot be a normal hierarchy. On PDS, the hierarchy with the same name must mean the same thing across the board.

Generate Integration Configuration - Integration Tools

In the same way as with application configurations, you can access integration configurations within the Configuration Tools via the File menu.

Complete the following steps using the Open Integration Configuration option to create a new configuration.

1. Create a dummy, starting xml file with the file name pdsappconfigs.xml.

   The required pdsappconfigs.xml file contains the name of the integration configuration to be generated, the application (that is, Application) name, the absolute path to the application configuration xml files, and the preferred fact name prefix and language setting.

   <domain_set> tags contain the list of applications included in this integration Configuration.

   <domain> tag contains information for one application.

   The tags batch_control_ga, batch_control_cust, app-reg-order, and preferred-fact-prefix such as mfp, are all optional. The tag batch_control_ga specifies the full path to the parent directory that contains the GA version batch control files. The tag batch_control_cust specifies the full path to the parent directory that contains the Custom version batch control files.

   The path to the application xml file must be a valid, full path and is always required.

   The interface_cfg element is optional. If there is no interface.cfg file, this element is not required. The tag interface_cfg specifies the full path to the application-specific interface mapping file, interface.cfg file.

   Example pdsAppConfigs.xml file.

   ```
   <?xml version="1.0" encoding="UTF-8" ?>
   <rpas_hsa_configuration name="PDS" language="English">
     <domain_set>
       <domain name="MFP">
        <app-reg-order>2</app-reg-order>
        <preferred-fact-prefix>mfp</preferred-fact-prefix>
        <config_path>
               C:\RPASExps\V23-MFP-IW\config_home\mfpcs\mfpcs.xml
        <config_path>
        <batch_control_ga>
               C:\RPASExps\V23-MFP-IW\config_home\batch_control_ga
        <batch_control_ga>
        <batch_control_cust>
               C:\RPASExps\V23-MFP-IW\config_home\batch_control_cust
        </batch_control_cust>
        <interface_cfg>
               C:\RPASExps\V23-MFP-IW\config_home\interface.cfg
        </interface_cfg>
       </domain>
   ```

```
  </domain_set>
</rpas_hsa_configuration>
```

2. Load the function libraries for parsing rules and expressions.

   The $RIDE_HOME/resources/applibs.xml file must contain the application function library names.

   Example applibs.xml.

   <?xml version="1.0" encoding="UTF-8"?>

   <librarylisting>

   <entry>dAaiFunctions</entry>

   <entry>dAaiJniFunctions</entry>

   <entry>dAppFunctions</entry>

   <entry>dClusterEngine</entry>

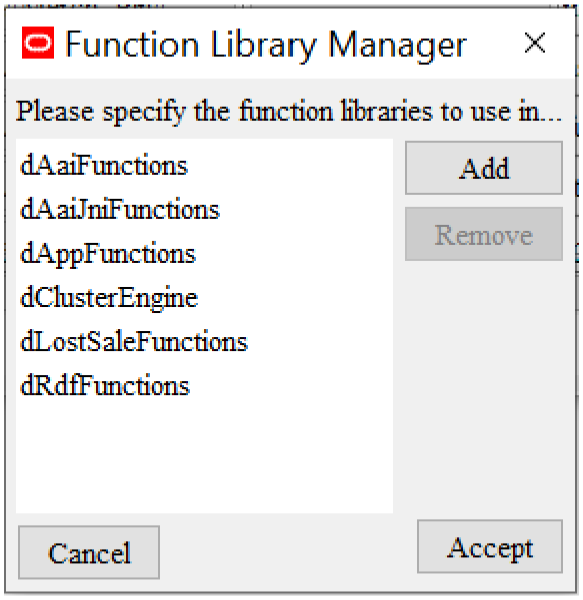
   <entry>dLostSaleFunctions</entry>

   <entry>dRdfFunctions</entry>

   </librarylisting>

3. Open the Config Tools and verify that the function libraries have been loaded.

Figure 11-15 Function Library Manager

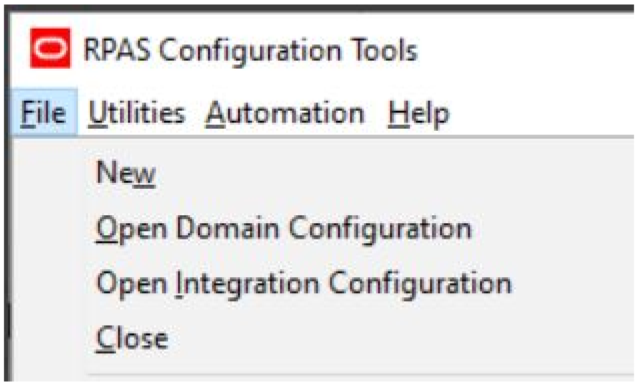


Please specify the function libraries to use in...

dAaiFunctions
dAaiJniFunctions
dAppFunctions
dClusterEngine
dLostSaleFunctions
dRdfFunctions

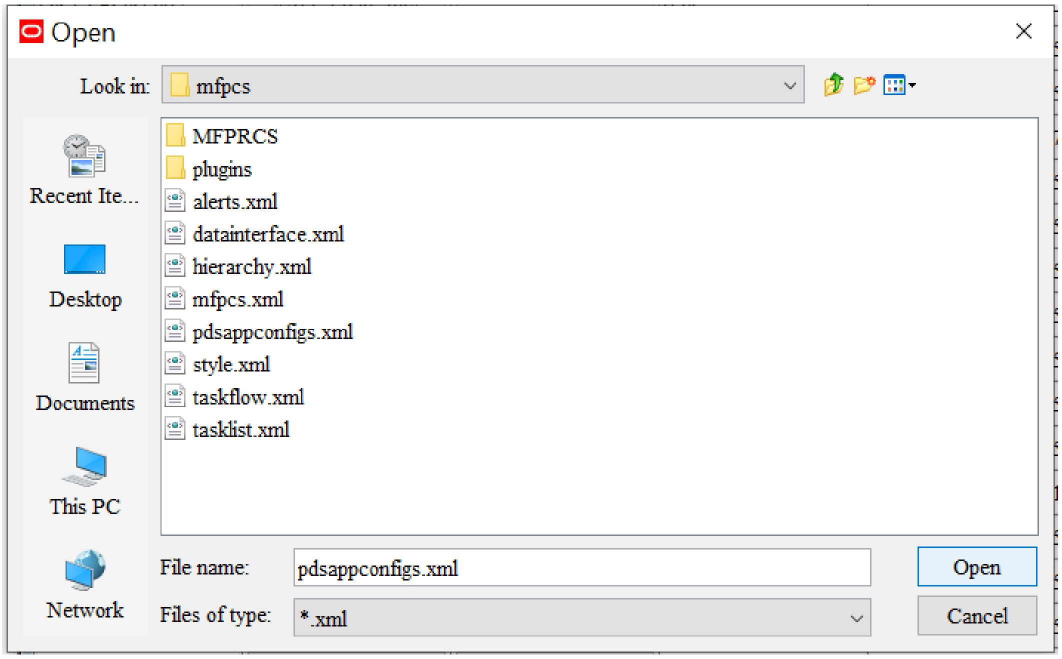
4. Open the MFPRCS application configuration in the Config Tools.
5. Select Open Integration Configuration from the File menu.

Figure 11-16   Open Integration Configuration Menu



6. A File selection dialog box is displayed. Select the dummy Integration Configuration file pdsappconfigs.xml file.

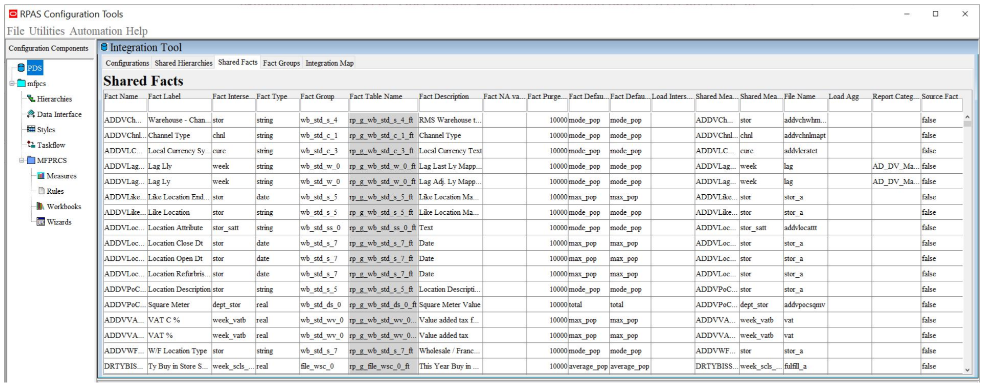
Figure 11-17   Load pdsappconfig.xml



File name: pdsappconfigs.xml
Files of type: *.xml

7. Select Open. The Integration Tool automatically populates and creates the integration configuration. Once the integration configuration has been created, the Integration Tool loads it into the Configuration Tools Workbench.

Figure 11-18 Generated Integration Configuration


RPAS Configuration Tools

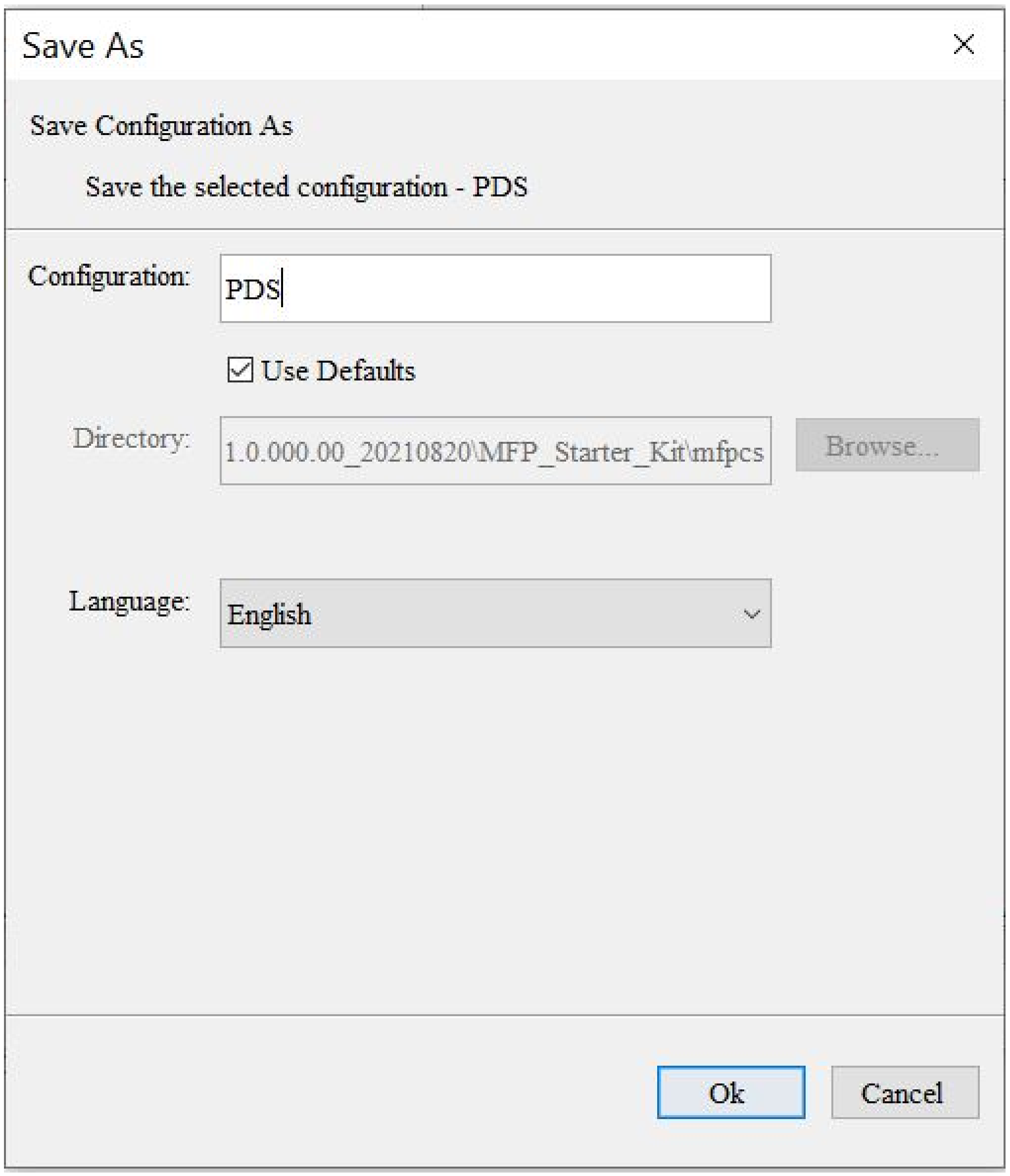
Shared Facts

Fact Name	Fact Label	Fact Interse...	Fact Type	Fact Group	Fact Table Name	Fact Description	Fact NA va...	Fact Purge...	Fact Defau...	Fact Defau...	Load Inters...	Shared Mea...	Shared Mea...	File Name	Load Agg	Report Categ...	Source Fact
ADDVCh...	Warehouse - Chan...	stor	string	wb_std_s_4	rp_g_wb_std_s_4_ft	RMS Warehouse t...		10000	mode_pop	mode_pop		ADDVCh...	stor	addvchwhm...			false
ADDVChnl...	Channel Type	chnl	string	wb_std_c_1	rp_g_wb_std_c_1_ft	Channel Type		10000	mode_pop	mode_pop		ADDVChnl...	chnl	addvchnlmapt			false
ADDVLC...	Local Currency Sy...	curc	string	wb_std_c_3	rp_g_wb_std_c_3_ft	Local Currency Text		10000	mode_pop	mode_pop		ADDVLC...	curc	addvlcratet			false
ADDVLag...	Lag Lly	week	string	wb_std_w_0	rp_g_wb_std_w_0_ft	Lag Last Ly Mapp...		10000	mode_pop	mode_pop		ADDVLag...	week	lag		AD_DV_Ma...	false
ADDVLag...	Lag Ly	week	string	wb_std_w_0	rp_g_wb_std_w_0_ft	Lag Adj. Ly Mapp...		10000	mode_pop	mode_pop		ADDVLag...	week	lag		AD_DV_Ma...	false
ADDVLike...	Like Location End...	stor	date	wb_std_s_5	rp_g_wb_std_s_5_ft	Like Location Ma...		10000	max_pop	max_pop		ADDVLike...	stor	stor_a			false
ADDVLike...	Like Location	stor	string	wb_std_s_5	rp_g_wb_std_s_5_ft	Like Location Ma...		10000	mode_pop	mode_pop		ADDVLike...	stor	stor_a			false
ADDVLoc...	Location Attribute	stor_satt	string	wb_std_ss_0	rp_g_wb_std_ss_0_ft	Text		10000	mode_pop	mode_pop		ADDVLoc...	stor_satt	addvlocattt			false
ADDVLoc...	Location Close Dt	stor	date	wb_std_s_7	rp_g_wb_std_s_7_ft	Date		10000	max_pop	max_pop		ADDVLoc...	stor	stor_a			false
ADDVLoc...	Location Open Dt	stor	date	wb_std_s_7	rp_g_wb_std_s_7_ft	Date		10000	max_pop	max_pop		ADDVLoc...	stor	stor_a			false
ADDVLoc...	Location Refurbris...	stor	date	wb_std_s_7	rp_g_wb_std_s_7_ft	Date		10000	max_pop	max_pop		ADDVLoc...	stor	stor_a			false
ADDVPoC...	Location Description	stor	string	wb_std_s_5	rp_g_wb_std_s_5_ft	Location Descripti...		10000	mode_pop	mode_pop		ADDVPoC...	stor	stor_a			false
ADDVPoC...	Square Meter	dept_stor	real	wb_std_ds_0	rp_g_wb_std_ds_0_ft	Square Meter Value		10000	total	total		ADDVPoC...	dept_stor	addvpocsqmv			false
ADDVVA...	VAT C %	week_vatb	real	wb_std_wv_0	rp_g_wb_std_wv_0...	Value added tax f...		10000	max_pop	max_pop		ADDVVA...	week_vatb	vat			false
ADDVVA...	VAT %	week_vatb	real	wb_std_wv_0	rp_g_wb_std_wv_0...	Value added tax		10000	max_pop	max_pop		ADDVVA...	week_vatb	vat			false
ADDVWF...	W/F Location Type	stor	string	wb_std_s_7	rp_g_wb_std_s_7_ft	Wholesale / Franc...		10000	mode_pop	mode_pop		ADDVWF...	stor	stor_a			false
DRTYBIS...	Ty Buy in Store S...	week_scls_...	real	file_wsc_0	rp_g_file_wsc_0_ft	This Year Buy in ...		10000	average_pop	average_pop		DRTYBISS...	week_scls_...	fulfill_a			false

8. Save the Integration Configuration XML file.

   In Configuration Tools, right click the integration configuration PDS. Select Save to save the integration configuration under the name PDS.xml. You can also select Save As to save the configuration under a different file name.

Figure 11-19 Saving the Integration Configuration



Save As

Save Configuration As

Save the selected configuration - PDS

Configuration: PDS

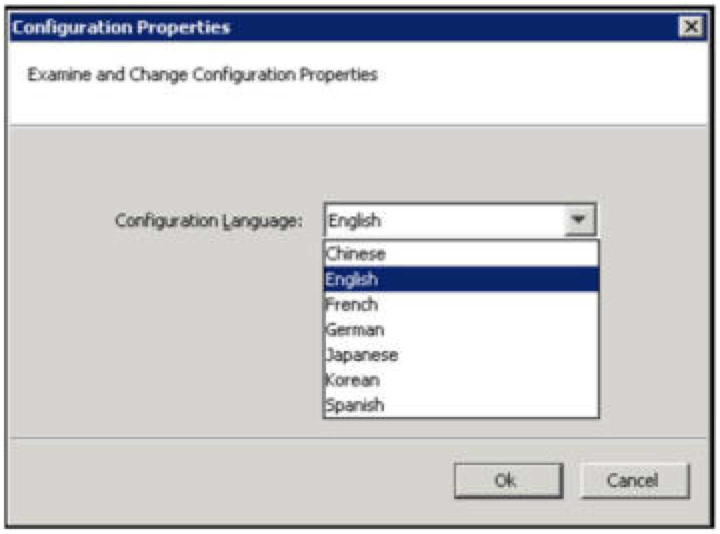
☑ Use Defaults

Directory: 1.0.000.00_20210820\MFP_Starter_Kit\mfpcs

Language: English

9. Once the integration configuration has been created, use the Configuration Properties window to inspect and modify the language setting for the configuration.

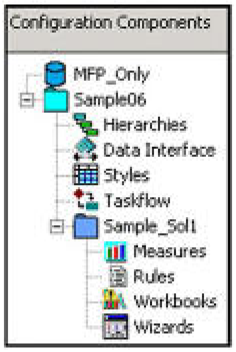
Figure 11-20 Configuration Properties



10. Select either Save or Save As in the File menu to save changes to an integration configuration. Select either Close or Close All to close an open integration configuration.

    When working with multiple integration configurations or with a mixture of integration configurations and application configurations, use the Configuration Components pane to navigate to the integration configurations, which are represented within the pane by the product database icon.

Figure 11-21 Configuration Components Pane



Generate Integration Configuration - rpasInstall

Use the rpasInstall command with the -createIntegrationCfg switch to create the integration configuration in the Cygwin shell or UNIX environment.

```
rpasInstall -createIntegrationCfg -ch <config_home> -integrationCfg
<output_integration_config_file> -log <log_file> -rf
<Application_Function_Library>
```

- -createIntegrationCfg: the command switch.

- -ch <config_home>: the <config_home> is the full path to the directory containing the application configurations, the optional interface.cfg file, and the required pdsappconfigs.xml file.
- -integrationCfg: the full path to the output integration configuration xml file.
- -log: the full path to the rpasInstall log file.
- -rf: the application function library.

Example 11-2 Example:

```
rpasInstall -createIntegrationCfg -ch /cygdrive/d/v21/mapps/config_home -
integrationCfg /cygdrive/d/v21/mapps/integration/PDS.xml -log /
cygdrive/d/v21/mapps/logs/log.txt -rf dAaiFunctions -rf dAaiJniFunctions -rf
dAppFunctions -rf dClusterEngine -rf dLostSaleFunctions -rf dRdfFunctions
```

12

Deployment Tool

Note:

This chapter contains descriptions of features that are not supported in the current version of RPASCE. However, in order to assist in the process of migration of a pre-CE solution to the CE platform, the configuration components associated with these deprecated features remain present within the RPASCE Configuration Tools and therefore continue to be described within this document. For additional information, see Appendix – Calculation Engine User Guide in this document.

In order to create and maintain an RPASCE application, administrators make use of a number of RPASCE server utilities. Some of those utilities require specially formatted input or command files. Examples of the files include the globaldomainconfig.xml used in application partition creation and maintenance and the distwbconfig.xml used to set up and maintain distributed workbook storage.

While these files can be manually created and modified using a text editor, care must be taken not only to ensure the correctness of the information contained within the resources but also to maintain the proper XML formatting of the file. In order to assist in file creation and maintenance, the RPASCE Configuration Tools has a new tool called the Deployment Tool.

Using the Deployment Tool, it is possible to generate and modify the contents of RPASCE server utilities within a graphical user interface that represents the content of the resource with user interface controls. Content can be modified through conventional tabular interfaces and through user actions, as opposed to through manual editing of an XML file resource. The Deployment Tool will then create the appropriately formatted XML schema within the file so that it may be used by RPASCE server utilities.

Because deployment resources are not a part of a configuration, the Deployment Tool is not a part of creating or loading a configuration; instead, users launch and dismiss it separately through the Configuration Tools workbench.

General Process Flow for Generating Deployment Resources

The Deployment Tool supports the creation, inspection and modification of the following RPASCE deployment resources:

- Workbook storage files (distwbconfig.xml)
- Application partitioning files (globaldomainconfig.xml)
- Online administration task description files (admintasks.xml)

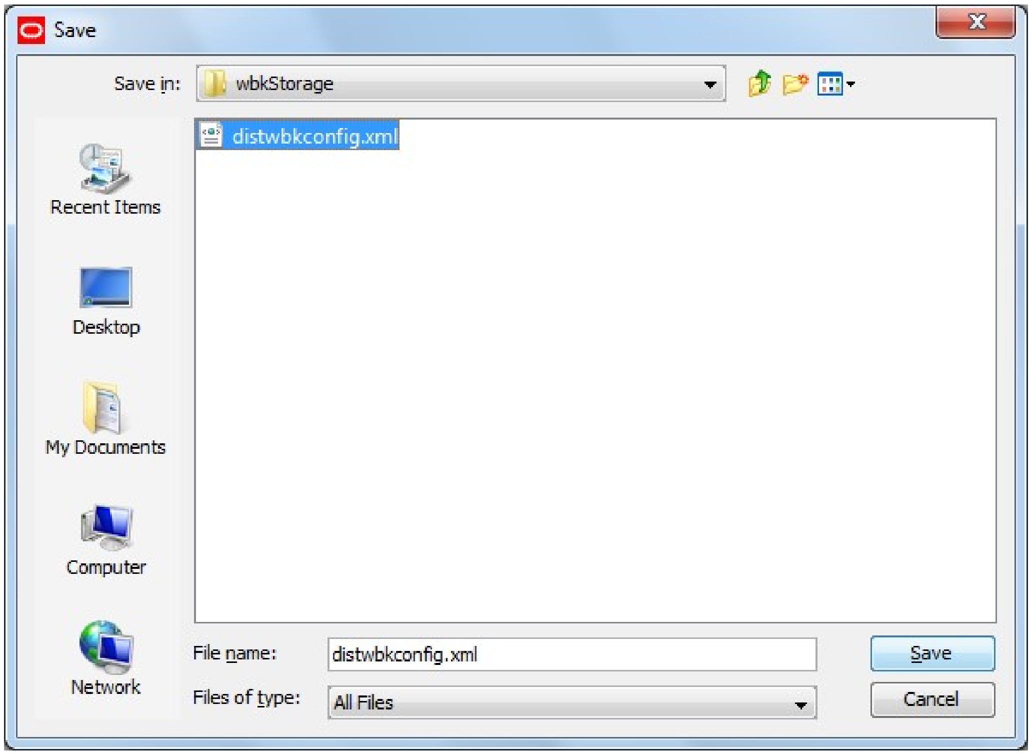
For each of these resources, distributed workbook storage, global application configuration and administrative tasks, the following functionality is available:

Creation of New Resource

Perform the following steps to create a new resource.

1. Upon launching the Deployment Tool from the Utilities menu of the Configuration Tool, the user will select to create a new resource of the desired type.
2. Using the UI controls present in the view designed for the desired resource, the user will enter the information contained within the resource.
3. The user saves the created resource as a file on the user's local system.
4. Once saved, the resource must be transferred to the system containing the RPASCE application or upon which the RPASCE application will be created as a manual process. The information can then be applied to the application using the method appropriate for the resource (for example, `rpasInstall` for initial `globaldomainconfig.xml` information).

Figure 12-1 Saving a New Resource



distwbkconfig.xml

File name: distwbkconfig.xml

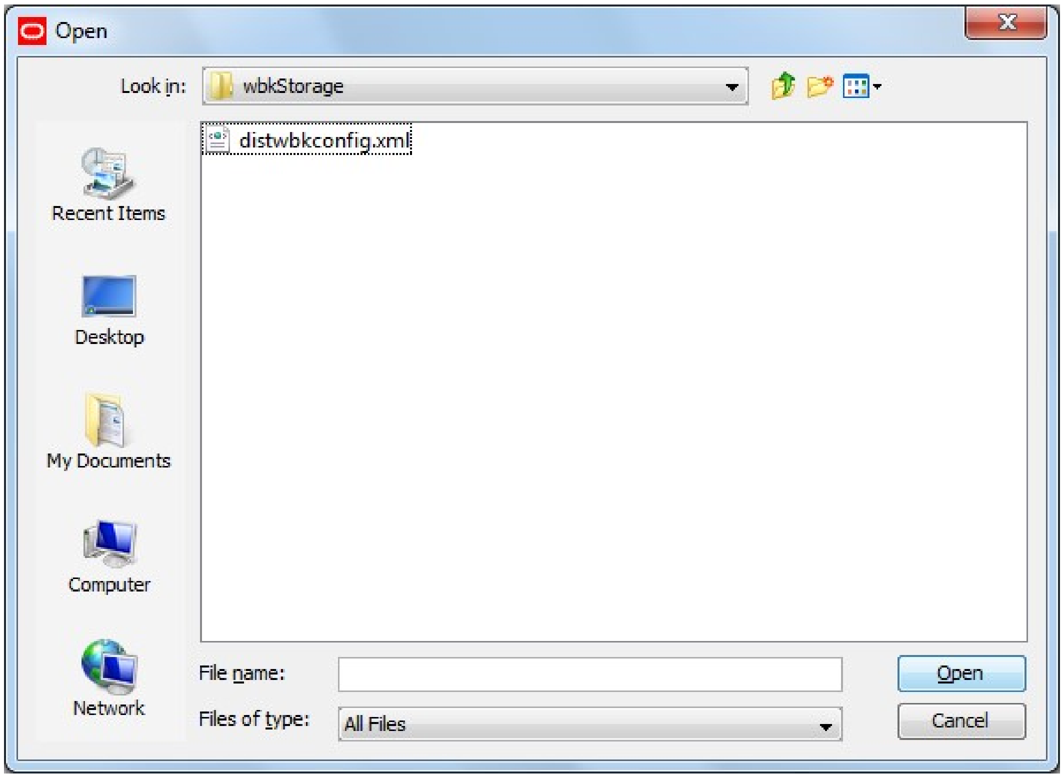
Files of type: All Files

Modification of an Existing Resource

Perform the following steps to edit an existing resource.

1. The original resource file must be transferred from the system containing the RPASCE application to the user's local system.
2. Upon launching the Deployment Tool from the Utilities menu of the Configuration Tool, the user will select to open an existing resource of the desired type. When prompted by the tool, the user will select the file on their local system.

Figure 12-2 Open Window for Opening an Existing Resource



3. Using the UI controls present in the view designed for the desired resource, the user can inspect the content of the resource and modify it as desired.
4. The user saves the modified resource to their local system.

User Interface

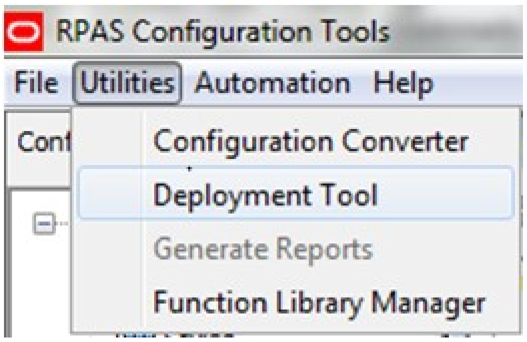
The Configuration Tools is extended with an option to open the Deployment Tool. This option is accessed as a menu item in the Utilities menu of the Configuration Tools workbench. Once opened, the Deployment Tool is represented within the Configuration Manager tree to allow users to navigate between it and any other open tools. Selecting the Deployment Tool menu item when the tool is already open will cause the tool to close; if any resource is currently being edited within the tool, the user will be prompted to save this resource before closing.

When first opened, the Deployment Tool presents the user with the set of deployment resources it supports. For each resource, the user can select between creating a new resource and opening an existing resource file.

When a new resource is created or an existing resource is opened, the Deployment Tool changes to present a resource-specific set of UI controls that can be used to specify the information for that specific resource. The user can then modify the resource and upon saving the specified information will be translated into the appropriate format. The functionality of each of these resources is described in detail in the following pages.

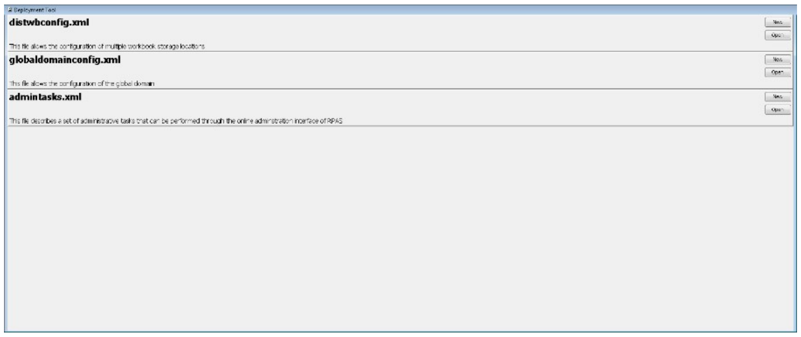
Navigate: In the Configuration Tools workbench pane, select Utilities and then Deployment Tool.

Figure 12-3   Utilities Menu



Result: The Deployment Tool window opens in the workspace.

Figure 12-4   Deployment Tool Window



distwbconfig.xml

This file allows the configuration of multiple workbook storage locations

globaldomainconfig.xml

This file allows the configuration of the global domain

admintasks.xml

This file describes a set of administrative tasks that can be performed through the online administration interface of RPAS

Deployment Tool - Distributed Workbook Storage

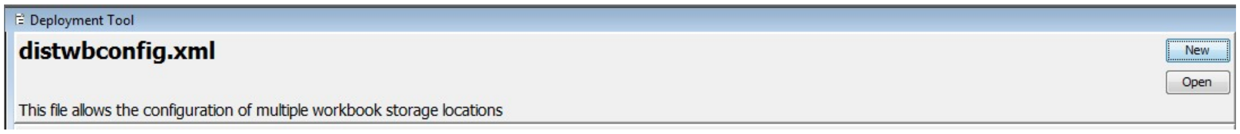
The initial resource to be implemented within the Deployment Tool is Distributed Workbook Storage or `distwbconfig.xml`. This resource is used to specify the locations of some storage locations across which workbooks created for a application can be balanced.

The initial set of workbook storage locations can be supplied at PDS creation time in the form of an XML resource. The Deployment Tool will allow users to specify the information required by this resource and will use that information to create the XML resource file on the user's local system so that it can be migrated to the RPASCE application host system and used.

Creation of a New Resource

To configure a new workbook storage location resource, click **New** located in the upper right-hand corner of the `distwbconfig.xml` resource.

Figure 12-5   Distributed Workbook Storage Resource



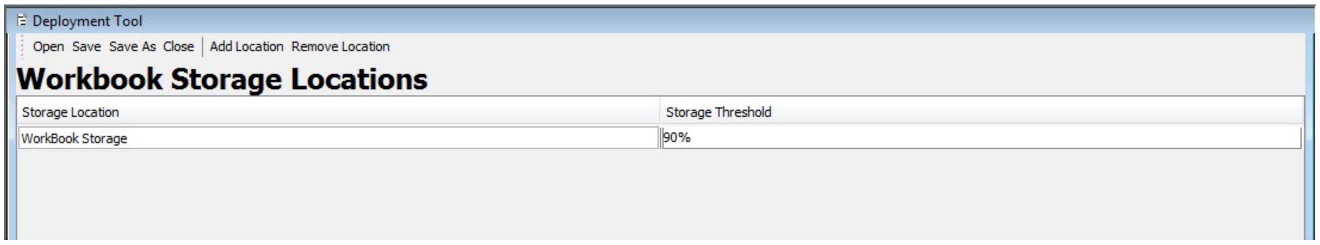
distwbconfig.xml

This file allows the configuration of multiple workbook storage locations

The Workbook Storage Locations window opens in the workspace.

The Workbook Storage Locations Window

Figure 12-6 Workbook Storage Locations Window

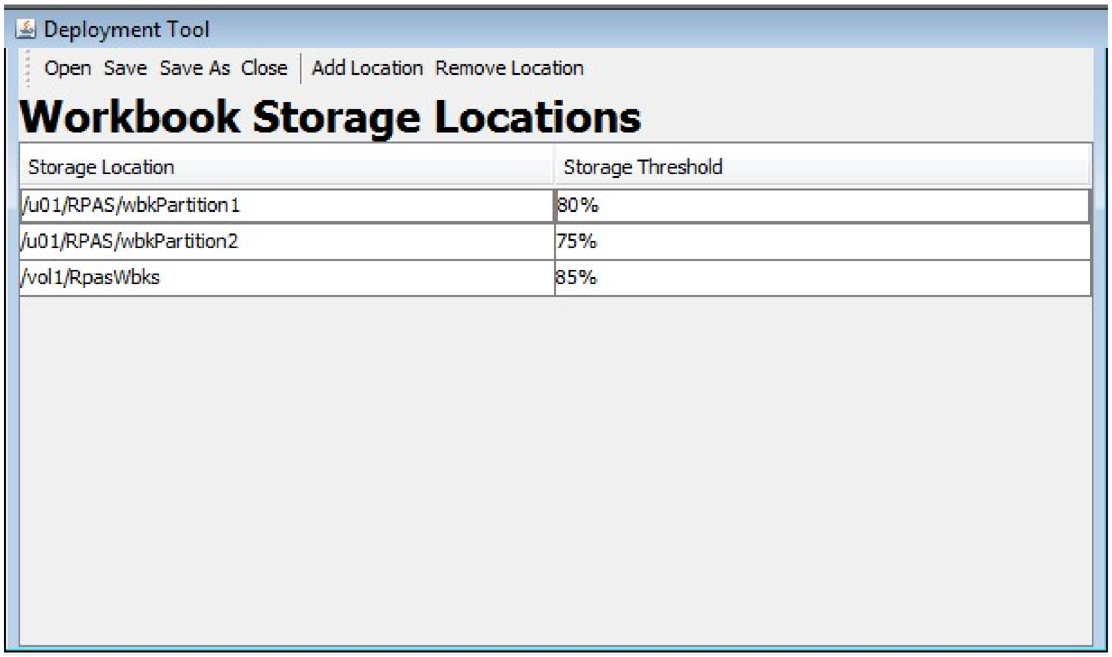

Deployment Tool
Open Save Save As Close Add Location Remove Location
Workbook Storage Locations

Storage Location	Storage Threshold
WorkBook Storage	90%

Upon creating a new storage location or upon selecting an existing partition, users will set values for the following attributes:

- Storage location root – The location on drive that will serve as the root of the workbook storage
- Storage location threshold – A threshold (in disk usage percentage) beyond which the storage location will not be used for the creation of new workbooks.

Figure 12-7 Workbook Storage Locations Thresholds Window

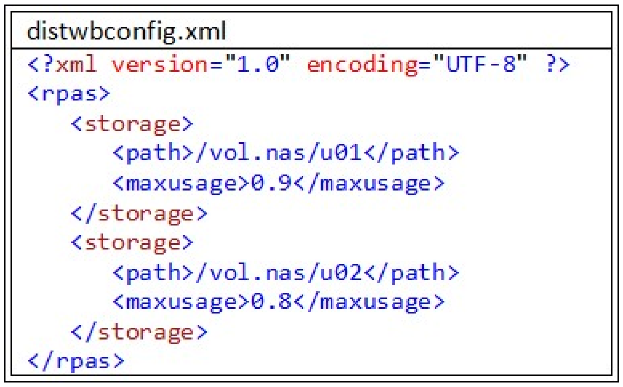

Deployment Tool
Open Save Save As Close Add Location Remove Location
Workbook Storage Locations

Storage Location	Storage Threshold
/u01/RPAS/wbkPartition1	80%
/u01/RPAS/wbkPartition2	75%
/vol1/RpasWbks	85%

Contents of distwbconfig.xml Resource

As part of the deployment information managed by Configuration Tools, design requires an extra configuration file “distwbconfig.xml" to be created. This file should contain the complete information of the distributed workbook storage, including its root directory and various parameters it may have. Meanwhile, the format of the file conforms to other similar configurations we already have.

Figure 12-8 distwbconfig.xml



distwbconfig.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpas>
   <storage>
      <path>/vol.nas/u01</path>
      <maxusage>0.9</maxusage>
   </storage>
   <storage>
      <path>/vol.nas/u02</path>
      <maxusage>0.8</maxusage>
   </storage>
</rpas>
```

Top node of <rpas> is needed to conform to other RPASCE configuration files. Within <rpas> tag, the <storage> tag is used to specify the settings of each workbook storage location. There can be multiple entries for <storage> within the <rpas> tag.

Within each <storage> tag, currently we require two tags:

<path> tag specifies the root directory of the distributed storage. This root directory must exist otherwise RPASCE considers the configuration as invalid and throws exception when it parses this file. However, it is not necessary for the directory to be empty at the application creation time since such storage device can be shared by multiple RPASCE applications.

Usually, the directory specifies the path that is outside the root of the global application, where the extra storage is mounted. Note that if the RPASCE administrator also wants to save workbooks under the application as before, a workbook storage location must be added where the parent directory of the application is specified in the <path>.

<maxusage> tag specifies the max usage level for the storage. If the workbook size stored in this storage location grows over this threshold, RPASCE does not allow new workbooks to be in order to prevent possible disk full failure. The level is represented as a percentage of the total volume of the storage. When specifying the safe level, we should keep in mind that workbook size usually doubles when opened due to the checkpoint directories. As a result, the administrator should allow extra room in the max usage level so that it is buffered against workbook open action. The actual percentage also depends on the typical workbook size of the application and the total storage capacity. If the typical workbook size is small while the storage device is larger, the max usage level can be higher in percentage, and vice versa.

Directory Structure of the Distributed Workbook Storage

In the configuration file, the RPASCE administrator specifies the root directory of the distributed storage, which must be an existing directory. RPASCE will manage the directory structure beneath the root directory such that it reflects the same directory structure as in a regular RPASCE application, including a directory for master application, all local application directories under the master application, users and user-id directory under each application. Eventually the workbooks will be stored within the user-id of the workbook owner.

These are the reasons to maintain similar directory structures:

The distributed storage may be shared by multiple RPASCE applications. So, the master application is needed as a direct subdirectory within the distributed storage root.

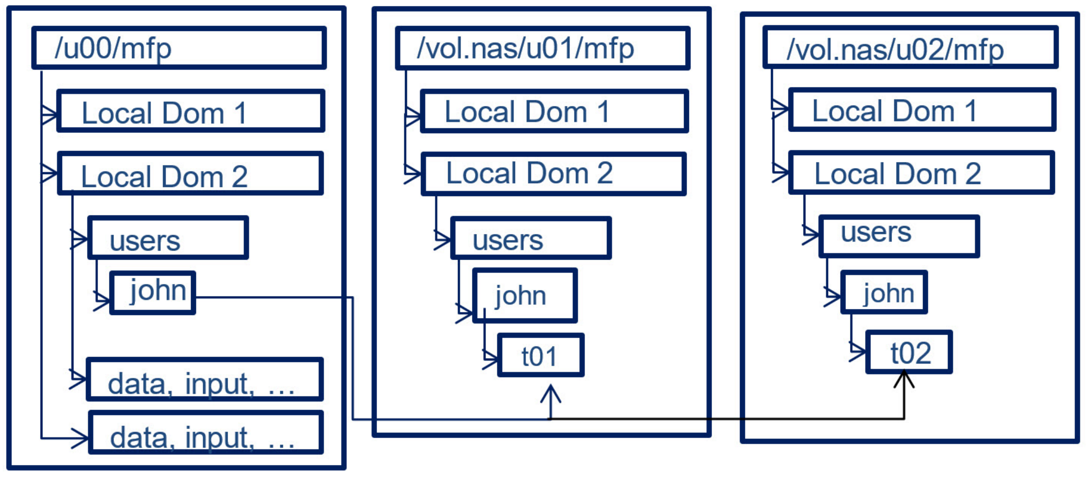
RPASCE workbook names are unique within a single application, but not unique across applications. Both ldom1 and ldom2 may have workbook t00001, though they are completely different workbooks. The local application path is put right beneath the master application directory.

In the original RPASCE application, the local application storage may be partitioned using globaldomainconfig.xml so that each local application may use a different storage. Here, since we are in the distributed storage already, no need to further complicate the local application directory. All local application subdirectories within a master application are located on the same directory level.

In other RPASCE functionality, such as workbook copy, RPASCE assumes the workbook's parent directory is the user-id and users of the application. So, the users/user-id directory must also exist in the distributed storage directory structure.

Figure 12-9 shows the directories of a Merchandise Financial Planning application with two distributed storage locations assigned to it, and a user ,***john***, who has already built two workbooks in a local application:

Figure 12-9 Example of Distributed Storage Directories



If no distributed workbook storage is specified for the application, workbooks are saved exactly as before.

If the RPASCE administrator put the root directory of the application as an entry for distributed workbook storage, together with other distributed storage locations, the existing application directory will be used to store workbooks as well. The only caveat here is that RPASCE might create ldom0 right beneath the master application's directory to store the workbook but the real local application ldom0 with its data, input, and so on, may be stored in another location as specified by globaldomainconfig.xml.

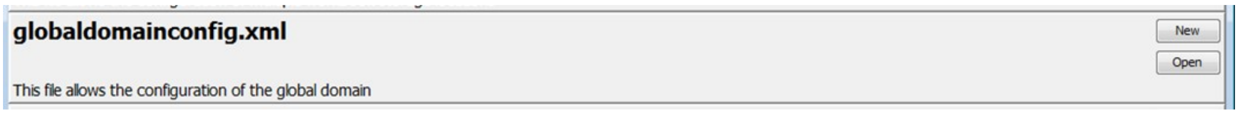
Since RPASCE only requires the root directory to exist at the time of configuration, RPASCE automatically creates any subdirectories on demand when a new workbook is to be stored in the distributed storage. As a result, actions such as adding new user, adding new local application, removing user, or removing a local application have no impact on the directory structure of the distributed storage. When a user is removed from the application, all workbooks belonging to this user are removed from the application, and from the distributed storage, but the user's directory will stay in the distributed storage, though it must be empty. When a new user is added to a application, the user's directory is created in the application, but not in the distributed storage. Only when this new user builds a new workbook, is the user's directory created in the distributed storage.

Note that the storage location should not reside inside a application or any other places which may be moved or deleted frequently or outside of the control of RPASCE. In that event, the application loses access to the workbooks in that storage.

Deployment Tool – Global Domain Configuration

This release of the Deployment Tool will include support for the creation and maintenance of the domain partitioning resource called globaldomainconfig.xml. This resource is used to specify the path to the master domain, the partition dimension in the master domain, the path to the sub-domains and the list of positions, from each sub-domains, on the partition dimension.

Figure 12-10 Global Domain Configuration Resource


globaldomainconfig.xml

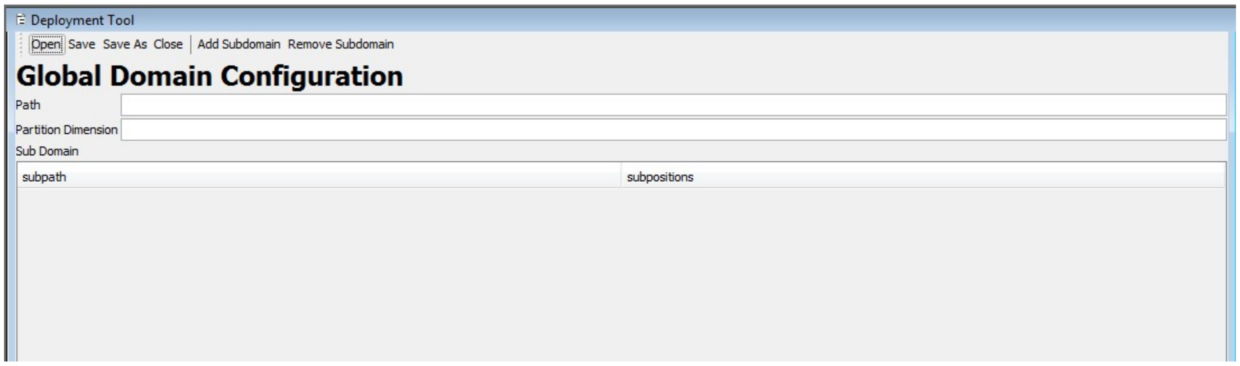
This file allows the configuration of the global domain

To configure a new workbook storage location, click **New** located in the upper right-hand corner of the `globaldomainconfig.xml` resource. You are taken to the Global Domain Configuration window.

To access an existing resource, click **Open** located in the upper right-hand corner of the `globaldomainconfig.xml` resource. You are presented with the files saved on their local system.

To add a sub-domain, click **Add Subdomain**.

Figure 12-11 Global Domain Configuration Window



Global Domain Configuration

Path

Partition Dimension

Sub Domain

subpath	subpositions

To remove a sub-domain, the user clicks the Remove Subdomain menu option.

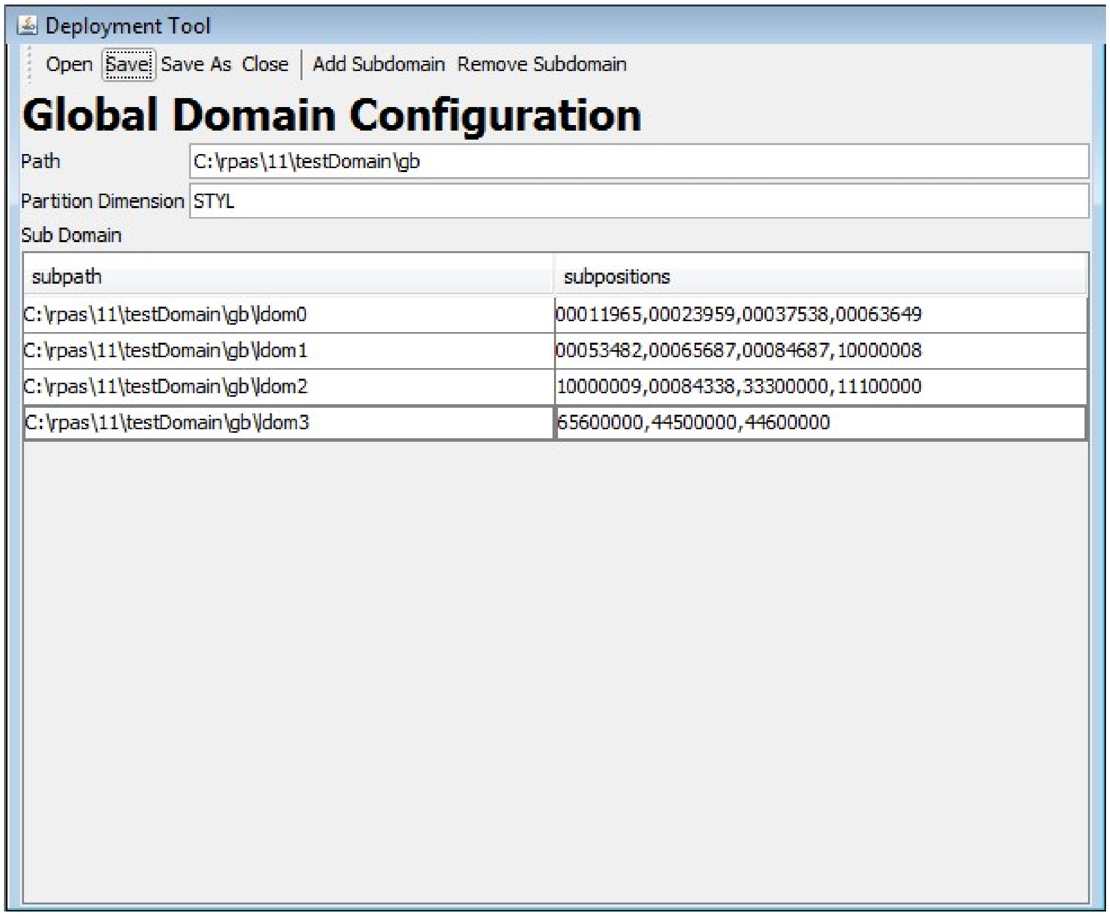
To access an existing resource the user clicks the Open button located in the upper right-hand corner of the globaldomainconfig.xml resource. The user is presented with the files saved on their local system.

The initial set of globaldomainconfig can be supplied at domain creation time in the form of an XML resource. The Deployment Tool will allow users of the Configuration Tool to specify the information required by this resource and will use that information to create the XML resource file.

When opening an instance of this file, the user will be presented with the path to the master domain, the partition dimension in the master domain, the path to the sub-domains and the list of positions, from each sub-domain, on the partition dimension. When creating a new resource, all this information will be empty. Upon creating a new global domain configuration or upon selecting an existing global domain configuration, users will be able to set values to the following attributes:

- **Path** - The path to the master domain
- **Partition Dimension** – how each of the local domain must be partitioned based on position groupings.
- **Subpath** – the path to the sub-domain
- **Subpositions** – the list of positions, in each sub-domain, on the partition dimension.

Figure 12-12 Global Domain Configuration Sub-Domains



Global Domain Configuration

Path: C:\rpas\11\testDomain\gb

Partition Dimension: STYL

Sub Domain

subpath	subpositions
C:\rpas\11\testDomain\gb\ldom0	00011965,00023959,00037538,00063649
C:\rpas\11\testDomain\gb\ldom1	00053482,00065687,00084687,10000008
C:\rpas\11\testDomain\gb\ldom2	10000009,00084338,33300000,11100000
C:\rpas\11\testDomain\gb\ldom3	65600000,44500000,44600000

Deployment Tool – Online Administrative Tasks

RPASCE adds the ability for selected users of the RPASCE Client to manage the execution of server-side operations from within the client. This functionality takes the form of some standard templates that can be used to launch and monitor traditionally offline operations.

For example, administrators of RPASCE applications often have a regular data load process in which new positions enter a hierarchy, have their roll-up information modified or are retired from a hierarchy. This process is today managed by a system administrator making a call to the loadhier utility on the host system of the RPASCE application.

Using the Online Administrative interface, this task is accomplished by connecting to the RPASCE PDS through the RPASCE Client and selecting to build the Online Administration wizard (the Online Administrative workbook is implemented as a wizard-only workbook). From within the workbook wizard, the administrator (or any properly authorized user) can select the hierarchy load process from a set of tasks supported by the Online Administration functionality. The user would then specify any information the task was configured to require at the time the wizard (for example, the hierarchy or hierarchies to load) and then submit the task to the RPASCE server which would execute the necessary call to RPASCE server utilities.

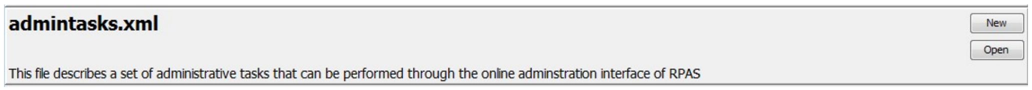
These administrative task templates require a document that describes the administrative tasks that can be performed within the application. Currently, a minimum of three resources is needed to describe administrative tasks:

- One resource to define common RPASCE administrative tasks
- One or more resources to define application-specific administrative tasks
- One resource to define customer-specific tasks

The ability of the system to support multiple application-specific resources allows support of PDS that contain multiple applications.

The Deployment Tool supports the creation and maintenance of administrative task resources. The Tool supports the creation of new resources and the loading, inspection, and modification of existing resources through a graphical user interface (GUI) and allows these resources to be saved to local storage so that they can be packaged and deployed to RPASCE servers.

Figure 12-13 Administrative Tasks Resource


admintasks.xml

This file describes a set of administrative tasks that can be performed through the online adminstration interface of RPAS

To configure a new administrative task resource the user clicks the **New** button located in the upper right-hand corner of the admintasks.xml resource. The user is taken to the Administrative Task View window.

To access an existing resource the user clicks the **Open** button located in the upper right-hand corner of the admintasks.xml resource. The user is presented with the files saved on their local system.

Administrative Task View

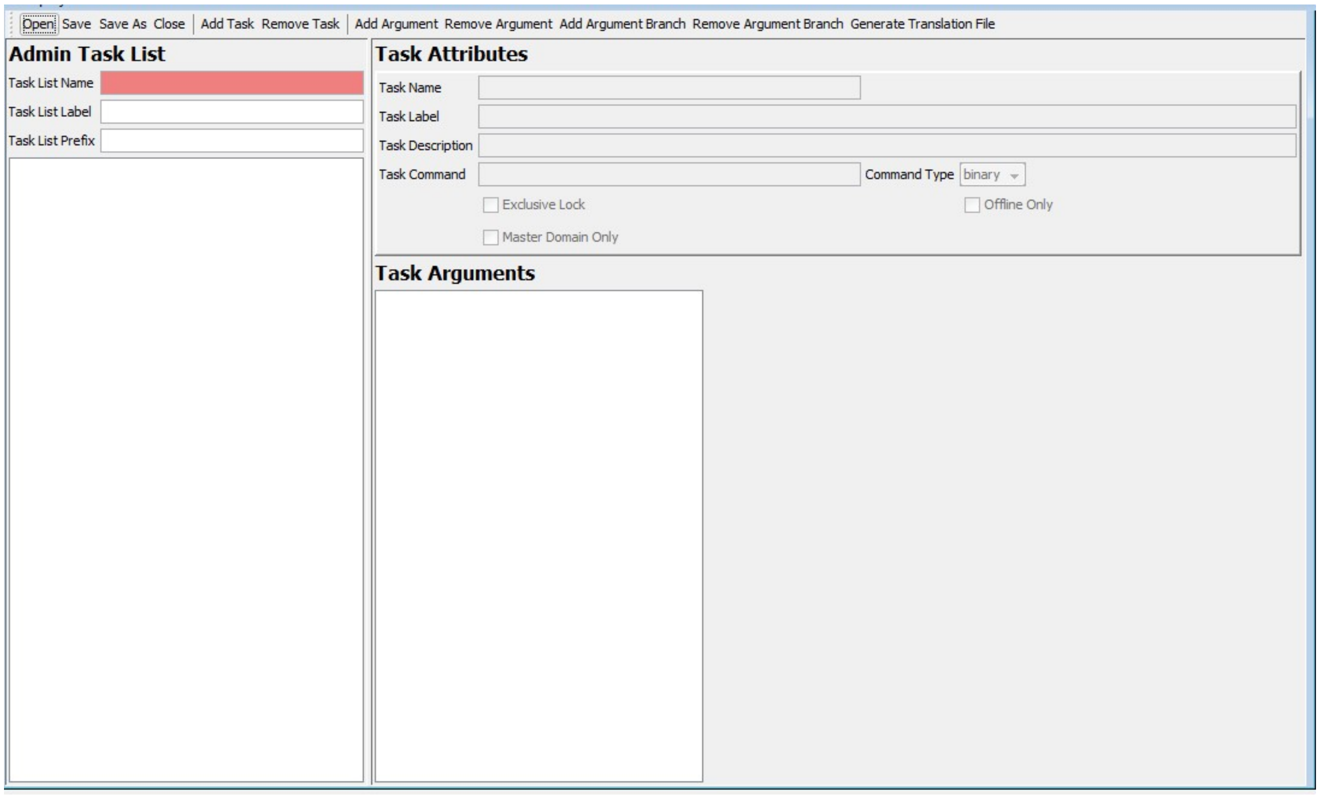
A new view has been added to the Deployment Tool to support the configuration of administrative task resources. This view is used to create a standard set of tasks packaged as a part of RPASCE, by creators of applications to create application specific tasks and by customers and/or implementers to create task resources for implementation specific tasks.

The new view supports creating new resources or opening and modifying existing resources. When a resource is created or opened, users of the Deployment Tool will be presented with a set of controls to create, modify, or remove the functional elements of the administrative task resource.

Within the view, users create and modify administrative tasks. New tasks are created through a menu bar action; existing tasks may be inspected or modified by selecting them from the administrative task list on the left side of the view. When viewing an existing task, additional menu bar actions allow the creation of arguments, argument lists and argument branches. A final set of menu bar actions will allow the removal of existing tasks, arguments, argument lists or argument branches.

Once a task is selected, the upper area of the view allows inspection and modification of task properties. Underneath the task properties is a navigation tree to allow selection of arguments, argument lists and argument branches. The center component under the task properties will be a set of controls to specify the properties of the selected argument, argument list or argument branch.

Figure 12-14 Administrative Tasks Window



Administration Task Resource Contents

Although RPASCE will support multiple resources defining administrative tasks, each of these resources must have a common format. The resources will be stored in an XML format with a rigidly defined internal structure.

Within the administrative task resource, the following container elements are supported:

- admin-task-list
- admin-task
- argument-list
- argument-branches
- argument-branch
- argument

Hierarchy of Elements in the Administrative Task Resource

In addition, there are some elements used to describe properties of the container elements. Each of these elements is described later.

Admin Task List Element

Every administrative task resource will have as its root element the <rpas> element that is common to all rpas XML resources. This root element will contain a single child element, the admin-task-list element. The admin-task-list is primarily a container for the admin-task elements contained within the resource but will does support the following properties:

- label – a user label used to describe the tasks contained within this resource

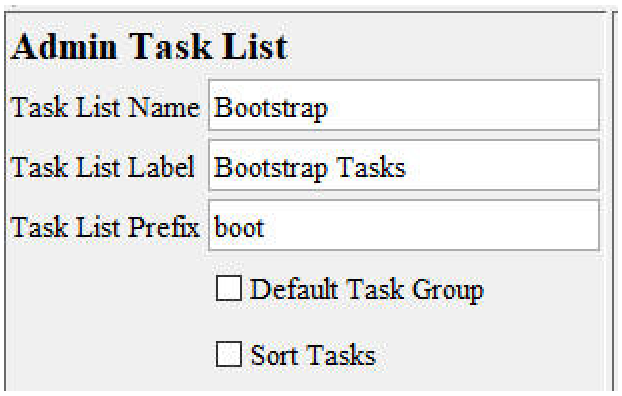
Create a New Task

Upon creating a new task, you can set values for the following values:

- Task List Name. Text field for entering the name of the Task List.
- Task List Label. Text field for entering the label of the Task List.
- Task List Prefix. Text field for entering the prefix for the Task List.
- Default Task Group. If the check box is selected, then a default task group in the Select Task page in RPASUI can be selected.
- Sort Tasks. If the check box is selected, then the tasks within the task list will be sorted by task labels.

After the values have been entered, the user clicks **Add Task** to add the new task to the Task List.

Figure 12-15 Admin Task List



Admin Task List

Task List Name	Bootstrap
Task List Label	Bootstrap Tasks
Task List Prefix	boot

☐ Default Task Group

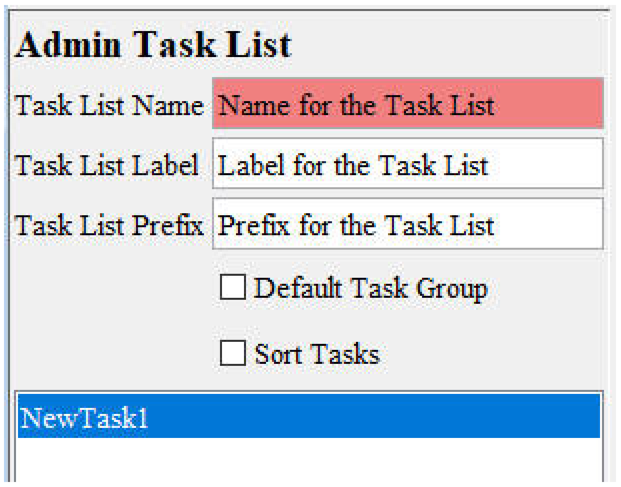
☐ Sort Tasks

Adding a new or selecting an existing task loads the properties of that task into the Task Attributes panel and populates the Task Arguments tree with the arguments (if any) defined for that task.

Remove a Task

This removes the currently selected administrative task from the list of tasks contained within this adminTasks resource.

Figure 12-16 Admin Task List with Task Selected for Removal



Admin Task List

Task List Name	Name for the Task List
Task List Label	Label for the Task List
Task List Prefix	Prefix for the Task List

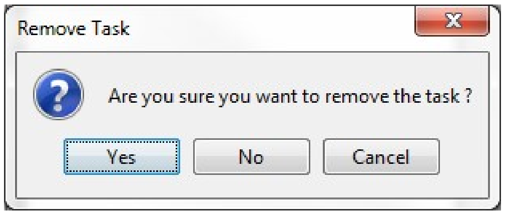
☐ Default Task Group

☐ Sort Tasks

NewTask1

Click **Remove Task** and the confirmation window opens.

Figure 12-17 Remove Task Confirmation



Remove Task

Are you sure you want to remove the task ?

Click **Yes** to remove the task and the task is then removed from the Admin.

Task Attributes Element

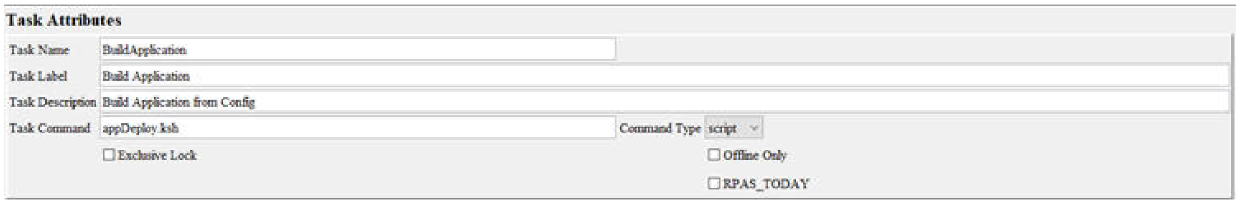
Each admin-task element represents a single administration task that can be launched through the online administration interface. These elements represent an RPASCE executable or executable script. The contents of the admin-task element define the task and its properties. Each admin-task element supports the following properties:

- **Task Name** – a user name identifying the administrative task
- **Task Label** – a user label identifying the administrative task
- **Task Description** – a longer description of the activity performed by this task
- **Task Command** – the command performed by this task
- **Command Type** –whether the command is an executable binary or script
- **Exclusive Lock** – checked if the command requires an exclusive lock on the application; unchecked otherwise
- **Offline Only** - checked if the command requires all currently active DBServer processes to be halted prior to execution of the task; unchecked otherwise

  .

The administrative task view of the Deployment Tool allows users to configure values for each of the previous attributes. It will also allow users to create the argument-list element that describes the arguments to the command represented by the admin-task element.

Figure 12-18 Task Attributes Window



Task Attributes

Task Name	BuildApplication
Task Label	Build Application
Task Description	Build Application from Config
Task Command	appDeploy.ksh
Command Type	script

☐ Exclusive Lock
☐ Offline Only
☐ RPAS_TODAY

Task Argument List Element

The argument-list element is used to describe the group of argument elements that make up the set of parameters passed to a command within an administrative task. The argument-list has no properties; it serves as a container for argument elements.

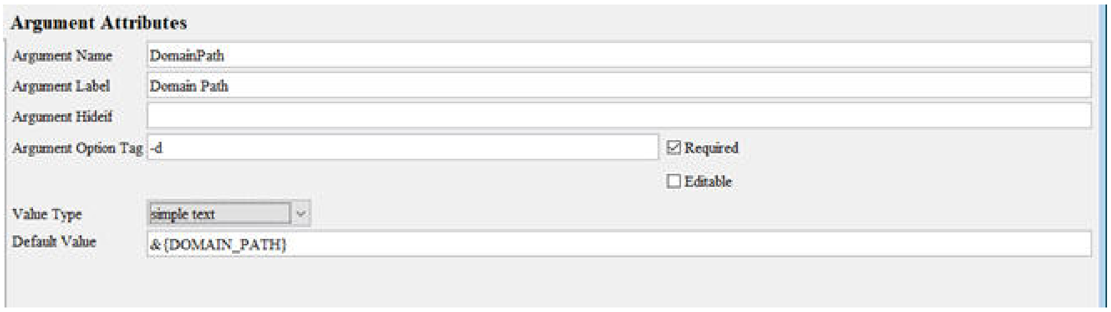
Argument Element

The argument element contains the definition of a single argument passed as a parameter to a command within an administrative task. Each argument element contains a set of properties that describe the usage and appropriate values for the argument. These properties are:

- **Argument Name** – a user name identifying the argument.
- **Argument Label** – a user label identifying the argument.
- **Argument Option Tag** – the argument flag (an example is –d).
- **Required** – checked if the argument is required, unchecked if the argument is optional.
- **Editable** – unchecked if the argument can be modified within the Administrative Task Workbook, checked if the argument is fixed.
- **Value Type** – User can select one value type from the drop-down list that contains the following terms: simple text, single select sort list, multi select sort list, switch-only, single select unsort list, multi select unsort list, text, date, datetime. For arguments that have a switch-only value of 0, this property holds a value for the additional parameter. If the value of the editable property is 1, then this value is a default that may be modified within the Administrative Task Workbook. If the value of the editable property is 0, then this value is fixed.
- Default Value – Specify the default value.

To add an argument the user clicks on Add Argument. The Argument Attributes Window is opened in the workspace.

Figure 12-19 Argument Attributes Window


Argument Attributes

Argument Name	DomainPath
Argument Label	Domain Path
Argument Hideif	
Argument Option Tag	-d
Required	✓
Editable	☐
Value Type	simple text
Default Value	&{DOMAIN_PATH}

The argument is added according to the following rules:

- If the argument tree has no selection or if the current selection is an argument not contained within an argument branch, the argument will be added to the argument list of the task.
- If the current selection is an argument contained within an argument branch or is an argument branch, the argument will be added to the argument list of the argument branch.

Argument Branch Element

In order to support the use of RPASCE utilities that have multiple valid sets of arguments, the argument list element also allows the definition of an argument-branches element. Each argument-branches element represents a choice between sets of arguments. Argument-branches elements then in turn contain an argument element and, optionally, an argument-list element representing the arguments required by the utility when that usage of the utility is required.

As an example, consider the loadhier utility. This utility can be used to perform multiple operations:

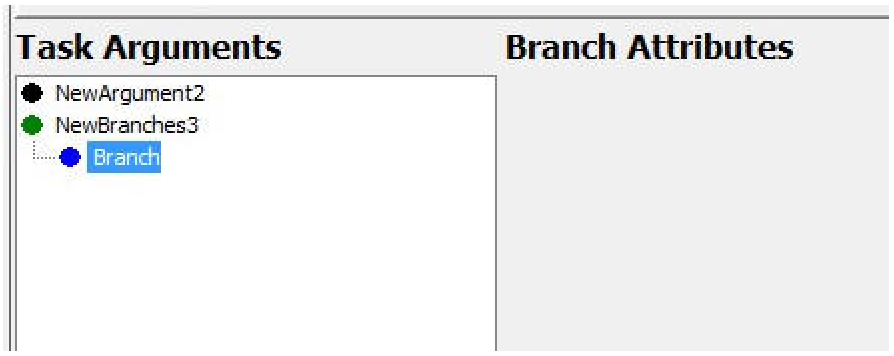
- loading a single hierarchy data file, executed by the –load operation
- loading multiple hierarchy data files, executed by the –loadAll operation
- purging hierarchy data, executed by the –purgeAll operation

In order to support the previous commands, the task definition for the loadhier utility would require an argument-branches element with three child argument-branch elements, one describing the usage for each of the three commands. When using the Administrative Task Workbook, the user would be required first to select the desired command and would then supply the information for the arguments associated with that argument list's arguments.

The argument-branch element is used to organize the arguments associated with one of the set of choices represented by an argument-branch element. Each argument-branch contains an argument element. This argument element represents the desired choice for the argument-branch. In cases where a choice in an argument-branch requires more than a single argument, an argument-branch may also optionally include an argument-list element containing the additional arguments for the branch.

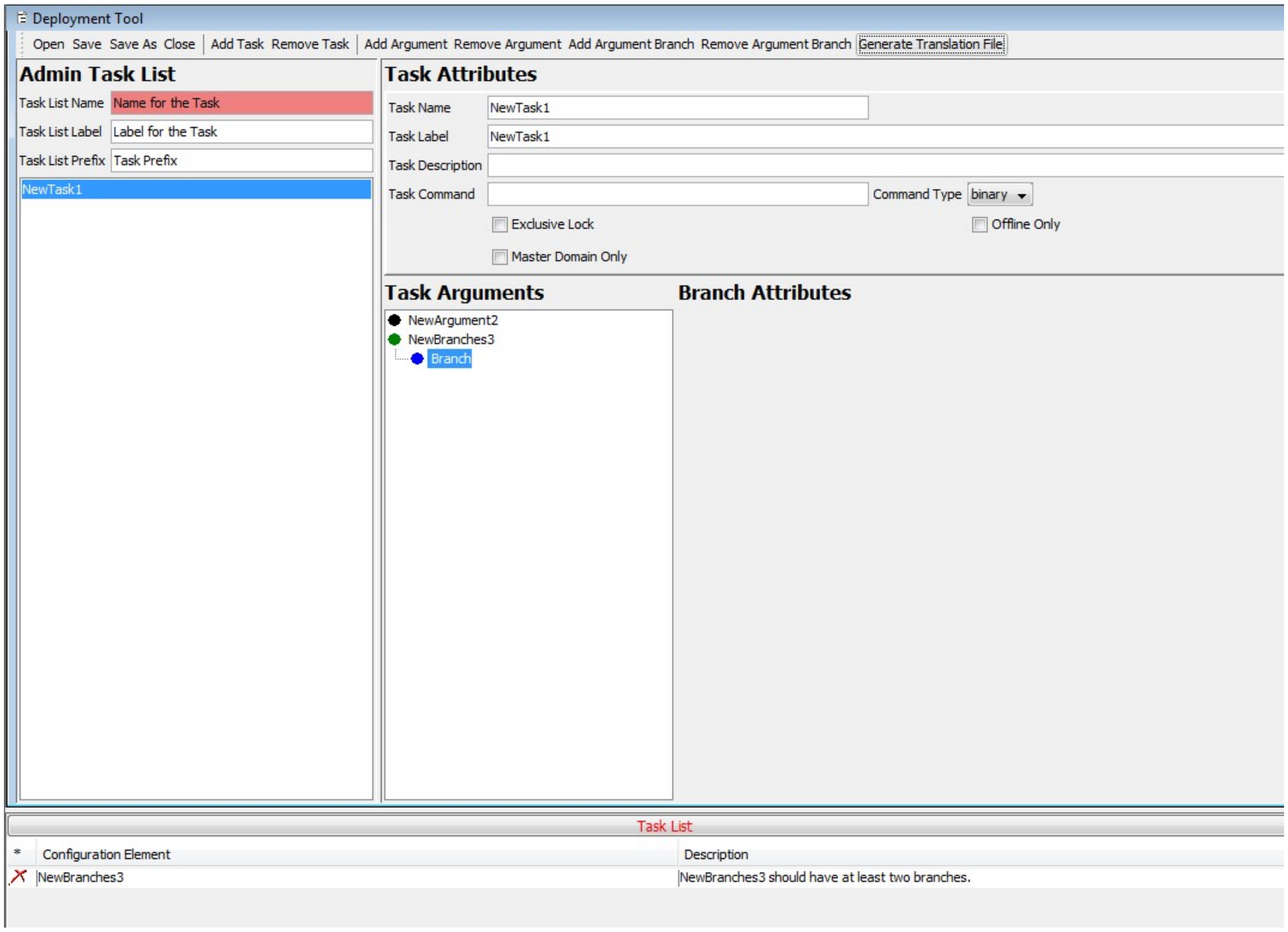
To add an argument branch, the user clicks on Add Argument Branch. This adds a new argument to the currently selected administrative task and the Branch Attributes window opens in the workspace.

Figure 12-20 Branch Attributes Window



When creating argument-branches, it is required to have at least two argument-branch options. Otherwise, a configuration element error will be generated in the Configuration Tools Task List.

Figure 12-21   Task List Error Due to Only Creating One Branch


Deployment Tool
Open Save Save As Close Add Task Remove Task Add Argument Remove Argument Add Argument Branch Remove Argument Branch Generate Translation File
Admin Task List
Task Attributes
Task Arguments
Branch Attributes
Task List

Task List Name	Name for the Task
Task List Label	Label for the Task
Task List Prefix	Task Prefix

NewTask1

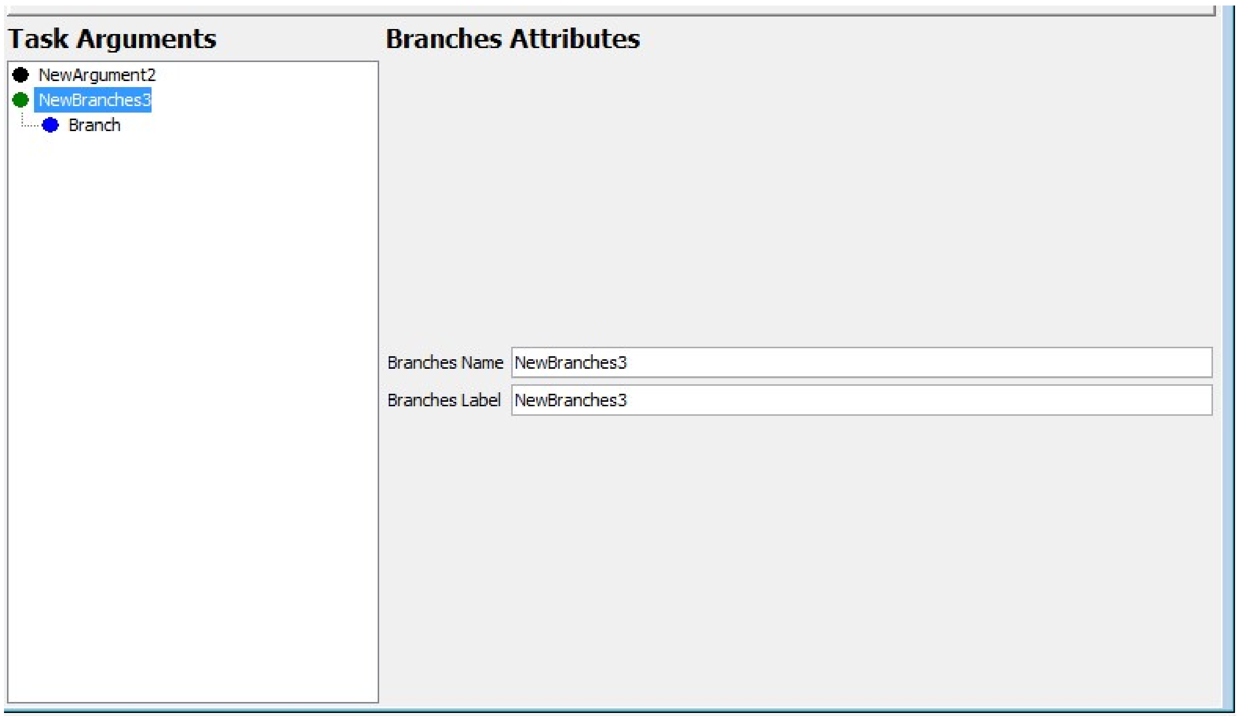
Task Name	NewTask1
Task Label	NewTask1
Task Description	
Task Command	
Command Type	binary
Exclusive Lock	☐
Offline Only	☐
Master Domain Only	☐

- NewArgument2
- NewBranches3
  - Branch

*	Configuration Element	Description
✗	NewBranches3	NewBranches3 should have at least two branches.

To see the Branch Attributes, highlight the desired argument branch. Selecting an argument or argument branches will populate the Argument Attribute panel with the properties of the argument or argument branch.

Figure 12-22 Argument Attribute Panel (Populated)



Branches Name NewBranches3
Branches Label NewBranches3

Remove Argument Branch

Select **Remove Argument Branch** to remove the currently selected argument branch from the argument list of the administrative task. If the branch to be removed is the only branch contained within its branches, the user is prompted to allow the now-empty branches to be removed as well.

Generate Translation File

Select **Generate Translation File** to create a pair of resource files within the directory containing the admintask.xml currently loaded into the UI. These files can be used to support translation of the content present within the admin task list.

Figure 12-23 Generate Translation File Button

You are prompted to save the admin task prior to generating the translation file. This ensures that the translation file is saved in the same directory as the admin task list xml.

Deployment Tool – Dashboard Settings Resource

The RPASCE platform contains a home page that users view and interact with when logging into the system. This page is called the Dashbord, and it provides summary information in the form of health-check metrics about a user's plans as well as summary information about exceptions present in that user's area of responsibility.

The Dashboard provides at-a-glance information in order to offer guidance to the user about areas of their information that require attention. This allows users to better organize the time and effort they spend in the application.

Within the system, a Dashboard is a hybrid of a traditional workbook containing the information backing the display and a new view representation consisting of information provided in a set of tiles, each of which can be selected to provide supporting information. The supporting information is presented in charts that display lower-level information about the summary information presented within the tile.

Because the dashboard is a hybrid of traditional RPASCE workbooks and a new visual representation, the process of configuring an RPASCE Dashboard consists of two components. First, a workbook must be configured in the application that defines the data available within the dashboard and how it is created and maintained within that workbook. Second, a resource file that defines the presentation layer configuration that determines how that information is made available in the form of tiles. Their detail charts must be created and deployed to the client tier in a manner like existing taskflow resources.

This file resource, often called the “Dashboard Settings" file, is a JSON encoded text document that contains information about the dashboard profiles and their tiles that is used by the client to organize and present the data contained within the dashboard workbook. JSON is an information encoding format similar to the XML encoding used in other parts of the platform that does not suffer from some of the limitations and security concerns in XML and is seeing wide adoption in the world of Cloud applications.

However, JSON is also a format that is new and not as widely known as XML. Like XML, it is meant primarily for machine consumption; that is, it can be difficult for humans to read and write JSON structures.

For this reason, the Deployment Tool contains an interface for the creation of the content of the Dashboard Settings resource. Using this tool, the person performing the configuration can interact with a graphical user interface that is like other Configuration Tools-based interfaces and the Tool will generate the appropriate JSON file content. Like with other Configuration Tools interfaces, this allows the people performing configuration to focus on what information they want present in the configuration (as defined in terms of measures, RPASCE views, and so on) and let the tool manage the generation of the appropriate file resources.

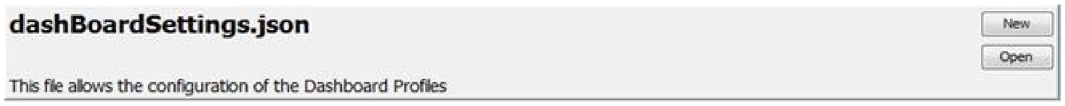
DashboardSettings.json View

The DashboardSettings.json View can be used to create, delete, and modify the contents of a Dashboard Settings file resource using a set of actions represented through buttons, menu items, and text forms. This information is organized around a set of entities that describe the view layer of the Dashboard, such as dashboard profiles and metric and exception tiles.

By representing this information in a user interface, the tool allows users to work with an organized form of the information that can be easily navigated and modified. Then, through the Save action, the information present within the interface can be written out into the appropriately formatted JSON file by the tool, allowing uers to concentrate on the content and allowing the tool to manage the encoding process.

The DashboardSettings.json View can be accessed from the main Deployment Tool screen through the DashboardSettings.json item. Users can either select New to initially create a Dashboard Setting configuration or select Open to load an existing resource.

Figure 12-24 DashboardSettings.json Option in Deployment Tool

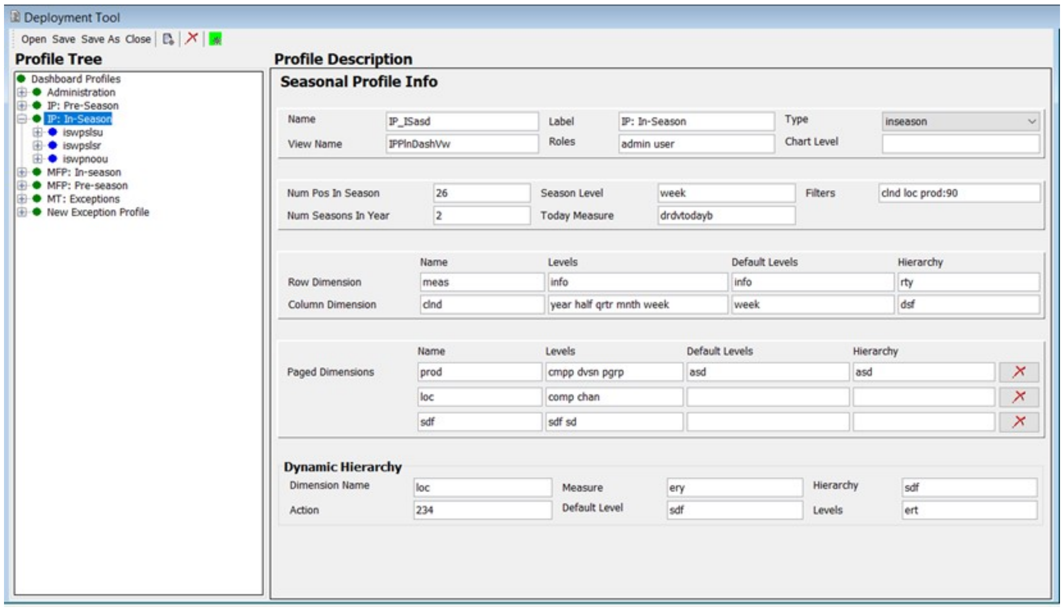


This file allows the configuration of the Dashboard Profiles

To access an existing resource, click **Open** which is located in the upper right-hand corner of the DashboardSettings.json entry. The user is presented with the files saved on their local system.

Once a dashboard settings resource is opened, its contents will be loaded into the DashboardSettings.json View.

Figure 12-25 DashboardSettings.json View with Loaded Resource



Seasonal Profile Info

Name	IP_ISasd	Label	IP: In-Season	Type	inseason
View Name	IPPlnDashVw	Roles	admin user	Chart Level	

Num Pos In Season	26	Season Level	week	Filters	clnd loc prod:90
Num Seasons In Year	2	Today Measure	drdvtodayb		

	Name	Levels	Default Levels	Hierarchy
Row Dimension	meas	info	info	rty
Column Dimension	clnd	year half qrtr mnth week	week	dsf

	Name	Levels	Default Levels	Hierarchy
Paged Dimensions	prod	cmpp dvsn pgrp	asd	asd
	loc	comp chan		
	sdf	sdf sd		

Dynamic Hierarchy

Dimension Name	loc	Measure	ery	Hierarchy	sdf
Action	234	Default Level	sdf	Levels	ert

Within the DashboardSettings.json view, users are presented with a navigation pane on the left, containing a tree structure of the entities making up a dashboard settings configuration such as dashboard profiles and the tiles they contain. When an entity is selected within the navigation pane, its information is loaded into the content pane on the right. This allows users to view and modify that entity's information. Profiles and tiles can be created and deleted through actions present on the top toolbar.

DashboardSettings.json Resource Contents

The following elements make up the contents of the `DashboardSettings.json` resource and are described in these sections:

- Metric Profile Element
- Seasonal Profile Element
- Exception Profile Element

- Metric Tile Element
- Comparison Metric Tile Element
- Dynamic Hierarchy Element
- Metric Tile Variance Element
- Exception Tile Element

Note:

For additional information on the elements of the `DashboardSettings.json` resource, their properties, and how they work together with the dashboard workbook, consult the Configuring Dashboards in RPASCE chapter.

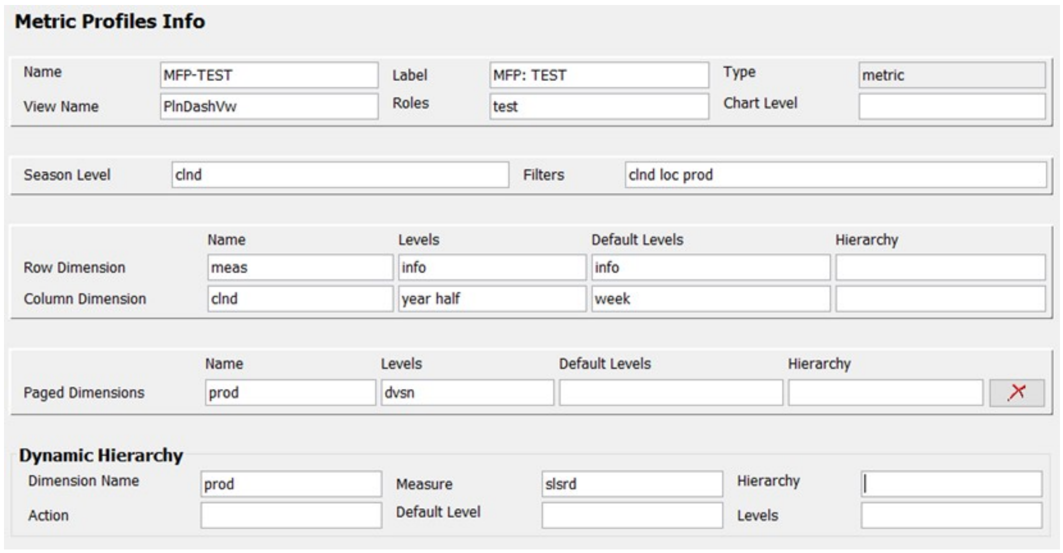
Metric Profile Element

A Metric Profile represents a collection of tiles containing metric information accessible by a user within the Dashboard. Each user typically has access to one or more profiles, depending on that user's role, and each profile aligns with an aspect of their role's responsibility. For example, users of planning applications may have access to both Pre-Season and In-Season Dashboard profiles that contain information about upcoming or current seasons, respectively. Users can select which of the profiles they wish to view through a selector control in the RPASCE Client.

Metric Profiles contain information in the form of tiles that provide summary information about their area of the business. This information often takes the form of Key Product Indicators or other health-check metrics such as actual performance versus performance in prior periods or versus the planned performance.

Metric Profiles contain not only the set of tiles that are associated with that profile but also a set of common information that defines how the Dashboard is presented when that profile is selected.

Figure 12-26 Metric Profile Information in the DashboardSettings.json View



Metric Profiles Info

Name	MFP-TEST	Label	MFP: TEST	Type	metric
View Name	PlnDashVw	Roles	test	Chart Level	

Season Level	clnd	Filters	clnd loc prod

	Name	Levels	Default Levels	Hierarchy
Row Dimension	meas	info	info	
Column Dimension	clnd	year half	week	

	Name	Levels	Default Levels	Hierarchy
Paged Dimensions	prod	dvsn		

Dynamic Hierarchy

Dimension Name	prod	Measure	slsrd	Hierarchy	
Action		Default Level		Levels	

The set of properties defined for a Metric Profile is:

- Name–an internal identifier for the profile.
- Label–the user facing label for the profile.
- Type–the type of profile will be metric. This field is read-only.
- ViewName–the name of the view configured in the dashboard workbook that supports this profile.
- ChartLevel–the default level to use in filters for detail charts in this profile.
- Filters–the whitespace-separated list of dimensions that must be presented as filters in the dashboard when this profile is being displayed. In order to accommodate positions with differing position label lengths, each dimension may optionally provide a width value in the format <dimension>:<length>.
- Dynamic Hierarchy- used to create hierarchical roll-ups based upon the attribute values of positions.

Additionally, the view contains information describing how to lay out dimensions across the row, column, and page axes of the detail chart. Each profile contains one Row Dimension, one Column Dimension, and as many additional paged dimensions as are needed, based upon the intersection of the view configured in the View Name.

The properties defined for a dimension entry are:

- Name–the name of the dimension being assigned.
- Levels–the set of levels in the assigned dimension that must be made available in the chart filter controls.
- Default Level–the initial level in the dimension that must be selected within the chart filter controls.

To add a new metric profile to the dashboard settings configuration, click **Add Metric Profile** action (located under **Add**) from the toolbar. A new profile entry is added to the navigation pane and selected in the content pane so that its information can be entered.

To remove an existing metric profile from the dashboard settings configuration, select the profile and click **Remove** from the toolbar. A confirmation window opens.

Click **Yes** to remove the current metric profile from the configuration.

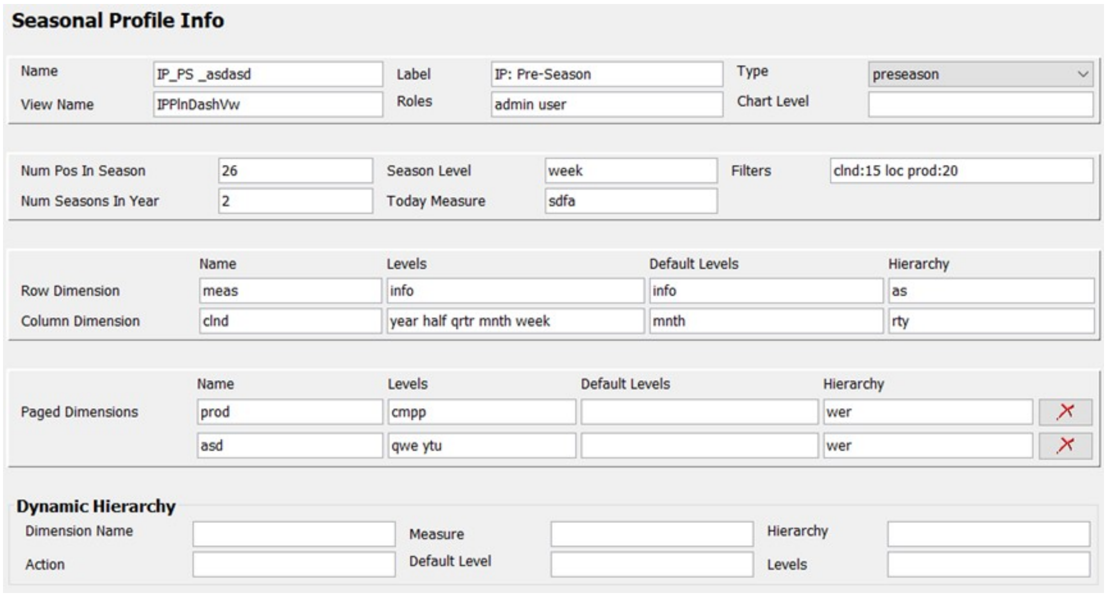
Seasonal Profile Element

The properties defined for a Seasonal Profile are:

- Name–an internal identifier for the profile.
- Label–the user facing label for the profile.
- Type–the type of profile. Supported types include preseason, inseason, and exceptions. Seasonal profiles must use either preseason or inseason from the drop-down field.
- ViewName–the name of the view configured in the dashboard workbook that supports this profile.
- ChartLevel–the default level to use in filters for detail charts in this profile.

- Num Pos In Season–the number of positions in the seasonLevel that make a season.
- Num Seasons In Year–the number of seasons in a calendar year. In cases where the product of the numPosInSeason and NumSeasonInYear values does not align with a calendar year, the dashboard will build with a logged warning.
- Filters–the whitespace-separated list of dimensions that must be presented as filters in the dashboard when this profile is being displayed. In order to accommodate positions with differing position label lengths, each dimension may optionally provide a width value in the format <dimension>:<length>.
- Dynamic Hierarchy–used to create hierarchical roll-ups based upon the attribute values of positions.

Figure 12-27 Seasonal Profile Information in the DashboardSettings.json View


Seasonal Profile Info

Name	IP_PS _asdasd	Label	IP: Pre-Season	Type	preseason
View Name	IPPlnDashVw	Roles	admin user	Chart Level	

Num Pos In Season	26	Season Level	week	Filters	clnd:15 loc prod:20
Num Seasons In Year	2	Today Measure	sdfa		

	Name	Levels	Default Levels	Hierarchy
Row Dimension	meas	info	info	as
Column Dimension	clnd	year half qrtr mnth week	mnth	rty

	Name	Levels	Default Levels	Hierarchy
Paged Dimensions	prod	cmpp		wer
	asd	qwe ytu		wer

Dynamic Hierarchy

Dimension Name		Measure		Hierarchy	
Action		Default Level		Levels	

Additionally, the view contains information describing how to lay out dimensions across the row, column, and page axes of the detail chart. Each profile contains one Row Dimension, one Column Dimension, and as many additional paged dimensions as are needed, based upon the intersection of the view configured in the View Name.

The properties defined for a dimension entry are:

- Name–the name of the dimension being assigned.
- Levels–the set of levels in the assigned dimension that must be made available in the chart filter controls.
- Default Level–the initial level in the dimension that must be selected within the chart filter controls.

To add a new metric profile to the dashboard settings configuration, click **Add Seasonal Profile** (located beneath **Add**) from the toolbar. A new profile entry is added to the navigation pane and selected in the content pane so that its information can be entered.

To remove an existing metric profile from the dashboard settings configuration, select the profile and click **Remove** from the toolbar. A confirmation window opens.

Click **Yes** to remove the current seasonal profile from the configuration.

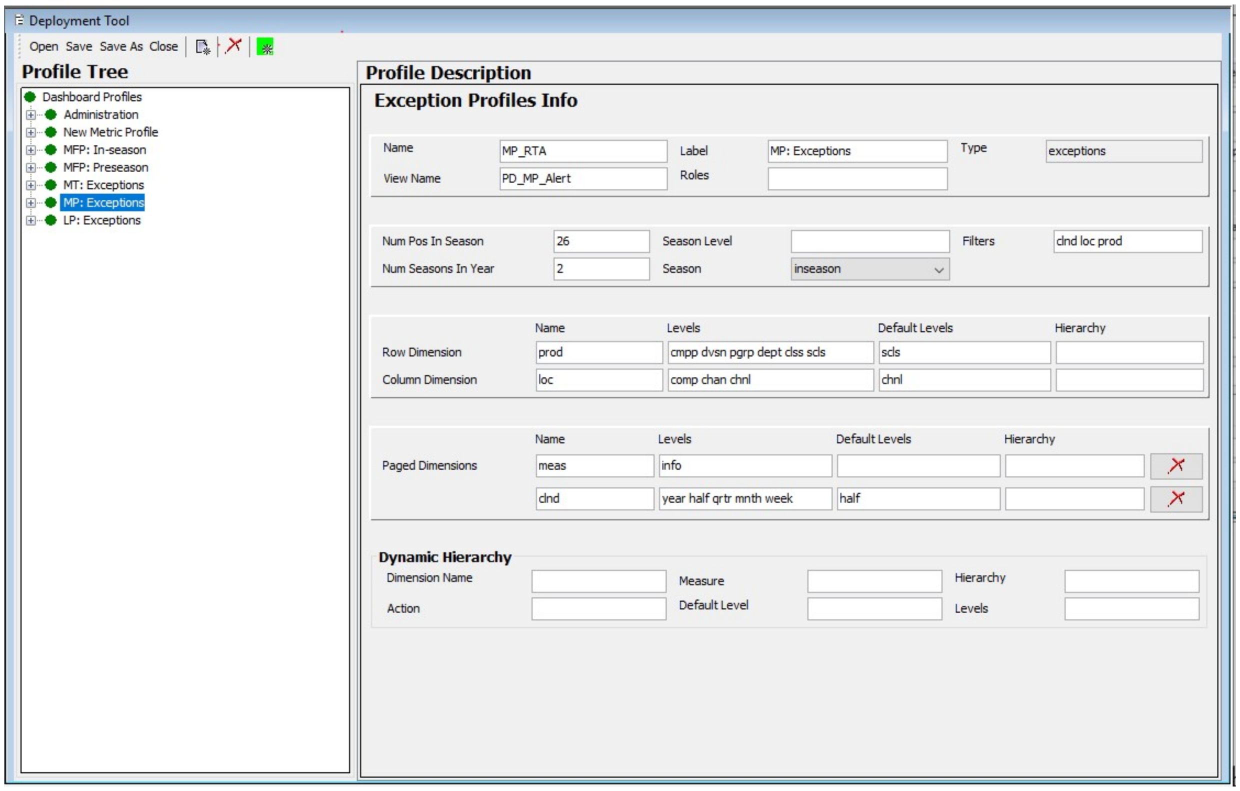
Exception Profile Element

An Exception Profile represents a collection of tiles containing exception information accessible by a user within the Dashboard. As with Metric Profiles, a configuration can contain multiple Exception Profiles, and different users may have access to one or more of the available profiles, based upon their role.

Unlike Metric Profiles, Exception Profiles containing tiles that report on the count and nature of exceptions as defined by the application are intended, in an exception-driven planning model, to provide guidance to the user about what aspects of the business require their time and attention.

Exception Profiles contain not only the set of tiles that are associated with that profile but also a set of common information that defines how the Dashboard as a whole is presented when that profile is selected.

Figure 12-28 Exception Profile Information in the DashboardSettings.jsonView



Profile Tree

- Dashboard Profiles
  - Administration
  - New Metric Profile
  - MFP: In-season
  - MFP: Preseason
  - MT: Exceptions
  - MP: Exceptions
  - LP: Exceptions

Profile Description

Exception Profiles Info

Name	MP_RTA	Label	MP: Exceptions	Type	exceptions
View Name	PD_MP_Alert	Roles			

Num Pos In Season	26	Season Level		Filters	dnd loc prod
Num Seasons In Year	2	Season	inseason		

	Name	Levels	Default Levels	Hierarchy
Row Dimension	prod	cmpp dvsn pgrp dept clss scls	scls	
Column Dimension	loc	comp chan chnl	chnl	

	Name	Levels	Default Levels	Hierarchy
Paged Dimensions	meas	info		
	dnd	year half qrtr mnth week	half	

Dynamic Hierarchy

Dimension Name		Measure		Hierarchy	
Action		Default Level		Levels	

The properties defined for an exception profile are:

- Name–an internal identifier for the profile.
- Label the user-facing label for the profile.
- Type–the type of profile is exceptions. This is a read-only field.
- ViewName–the name of the view configured in the dashboard workbook that supports this profile.

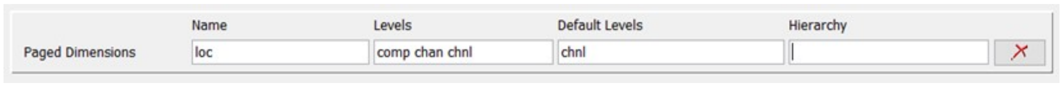
- Num Pos In Season–the number of positions in the seasonLevel that make a season.
- Num Seasons In Year–the number of seasons in a calendar year.
- seasonLevel - the level in the calendar dimension used to define a season in conjunction with the numPosInSeason and numSeasonInYear attributes (for example, week)
- season - the seasonality timeframe that this Exception dashboard profile can show. This season property can be set to either inseason, preseason, or both. If none is specified, then this property defaults to inseason.
- Filters-the whitespace-separated list of dimensions that must be presented as filters in the dashboard when this profile is being displayed. In order to accommodate positions with differing position label lengths, each dimension may optionally provide a width value in the format <dimension>:<length>.
- Dynamic Hierarchy–used to create hierarchical roll-ups based upon the attribute values of positions.

Additionally, the view contains information describing how to lay out dimensions across the row, column, and page axes of the detail chart. Each profile contains one Row Dimension, one Column Dimension, and as many additional paged dimensions as are needed, based upon the intersection of the view configured in the View Name.

When created, a Dashboard Profile (Metric/Seasonal/Exception) contains no Page Dimension entries. Click **Add Page Dimensions** (inside the asterisk button from the tool bar) to add entries.

Click **Remove** (located within the page dimension entry) to remove Page Dimension entries for the Dashboard Profile (Metric/Seasonal/Exception).

Figure 12-29 Page Dimension Entry Showing Remove


	Name	Levels	Default Levels	Hierarchy
Paged Dimensions	loc	comp chan chnl	chnl	

The properties defined for a dimension entry are:

- Name–the name of the dimension being assigned.
- Level– the set of levels in the assigned dimension that must be made available in the chart filter controls.
- Default Level–the initial level in the dimension that must be selected within the chart filter controls.
- Hierarchy–the hierarchy (for dimensions with multiple hierarchies) to be used within the filters of the dashboard.

To add a new exception profile to the dashboard settings configuration, click **Add Exception Profile** (located beneath **Add**) from the toolbar. A new profile entry is added to the navigation pane and selected in the content pane so that its information can be entered.

To remove an existing exception profile from the dashboard settings configuration, select the profile and click **Remove** in the toolbar. A confirmation window opens.

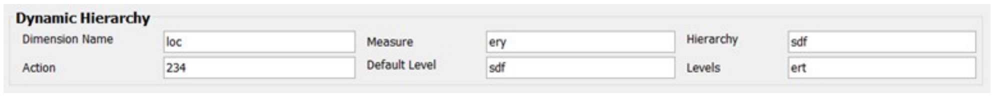
Click **Yes** to remove the current exception profile from the configuration.

Dynamic Hierarchy Element

Dynamic Hierarchy represents the following information:

- Dimension Name–this name must match with either of the row, column or paged dimensions. This indicates to which dimension the dynamic hierarchy is linked.
- Measure–the name of the measure that is meant to hold the user's selection of the attribute to be used in the attribute roll-up.
- Hierarchy–the hierarchy that contains the attribute roll-up. The dashboard only supports the use of a single hierarchy (for dimensions with multiple hierarchies) within the chart filters.
- Action–the menuId property of the menu item that performs the dynamic hierarchy operation in the workbook. This value is accessible in both the tmpl.cfg file generated for the dashboard workbook as well as in the .xml file that contains the configuration of the dashboard workbook.
- Default Level–the default level to display in the filters of the dashboard when the dynamic hierarchy is active.
- Levels–a JSON array of the levels present within the hierarchy created by the dynamic hierarchy operation.

Figure 12-30 Dynamic Hierarchy Entry


Dynamic Hierarchy

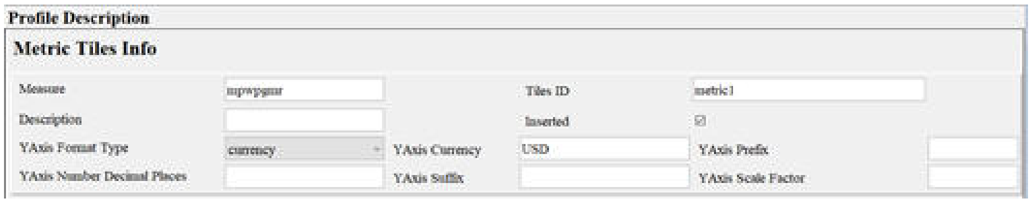
Dimension Name	loc	Measure	ery	Hierarchy	sdf
Action	234	Default Level	sdf	Levels	ert

Metric Tile Element

A Metric Tile provides information about a specify KPI or other health-check information. Each tile provides the value of the corresponding metric scoped to the set of positions defined by the filters of the dashboard profile containing the tile. Each tile can support zero, one, or two supporting metrics to provide additional information about the main metric.

The configuration of a Metric Tile consists of supplying information about the metrics displayed in that tile as well as information used to create the detail chart that is displayed when that tile is selected in the Dashboard.

Figure 12-31 Metric Tile Information in the DashboardSettings.json View


Profile Description

Metric Tiles Info

Measure	mpwpgmr		Tiles ID	metric1	
Description			Inserted	☑	
YAxis Format Type	currency	YAxis Currency	USD	YAxis Prefix	
YAxis Number Decimal Places		YAxis Suffix		YAxis Scale Factor	

The set of properties defined for a Metric Tile is:

- Measure–the measure containing the information to be displayed in the metric tile and its supporting chart.
- Inserted–true if the tile must be initially visible in the profile. Users may show and hide available tiles from within the Client.
- Tiles ID - a string that serves as a unique identifier for the tile. This property can be used when more than one metric tile contains information for a single measure (for example, the same base value with different comparison metrics in each tile). This property is optional; if a value is not provided, the value for measure will be used. However, when multiple tiles use a single measure, the UI may not be able to provide the correct information for things such as tile description.
- Description - a short description of the information provided by the tile that can be displayed in the tile info Window.

The Metric Tile element can contain yAxisFormat. For yAxisFormat properties, see "Configuring the Unit of Measurement Display in Dashboards".

To add a new metric tile to the current metric profile, select the metric and click **Add Metric Tile** action (located beneath **Add**) in the toolbar. A new tile entry is added to the navigation pane and selected in the content pane so that its information can be entered.

To remove an existing metric tile from the dashboard settings configuration, select the tile and click Remove in the toolbar.

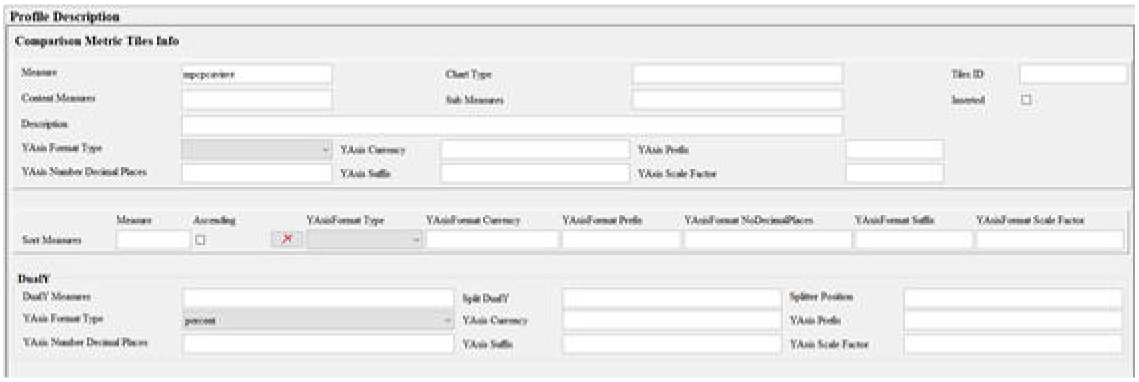
The confirmation window opens.

Click **Yes** to remove the current metric tile from the configuration.

Comparison Metric Tile Element

The properties defined for a Comparison Metric Tile are:

Figure 12-32   Comparison Metric Tile Information in the Dashboard Settings JSON View



Profile Description

Comparison Metric Tiles Info

Measure: mpcpcaview
Chart Type:
Tiles ID:
Content Measures:
Sub Measures:
Inserted: ☐
Description:
YAxis Format Type:
YAxis Currency:
YAxis Prefix:
YAxis Number Decimal Places:
YAxis Suffix:
YAxis Scale Factor:

	Measure	Ascending	YAxisFormat Type	YAxisFormat Currency	YAxisFormat Prefix	YAxisFormat NoDecimalPlaces	YAxisFormat Suffix	YAxisFormat Scale Factor
Sort Measures		☐						

DualY

DualY Measures:
Split DualY:
Splitter Position:
YAxis Format Type: percent
YAxis Currency:
YAxis Prefix:
YAxis Number Decimal Places:
YAxis Suffix:
YAxis Scale Factor:

- Measure–the name of the measure that is the primary metric displayed in the tile.
- Content Measures –an optional list of measures to be used within the supporting chart for this tile. If no value is given, then the measures used in the tile will be displayed within the chart.
- Chart Type–the type of chart to display for this metric tile. Supported values are bar and line.

- Sub Measures–an optional list of measures to be displayed underneath the main metric within the tile. Space on the tile constrains the maximum number of subMeasures to three.
- Inserted–true if the tile must be initially visible in the profile. Users may show and hide available tiles from within the Client.
- Tiles ID - a string that serves as a unique identifier for the tile. This property can be used when more than one metric tile contains information for a single measure (for example, the same base value with different comparison metrics in each tile). This property is optional; if a value is not provided, the value for measure will be used. However, when multiple tiles use a single measure, the UI may not be able to provide the correct information for things such as tile description.
- Description - a short description of the information provided by the tile that can be displayed in the tile info Window.
- Sort Measures–an optional list of measures that can be used to sort the information displayed within the chart. The contents of the sortMeasures array are a list of JSON objects with the following properties:
  - Measure–The name of the measure that is used to order the chart.
  - Ascending–True or false, depending on whether the ordering must be ascending (true) or descending (false).
- DualY–a JSON object used to support the use of dual-Y axis scales. By specifying the dualY property, it is possible to display a secondary set of values with independent scaling, either within the same chart as other measures or in a secondary chart displaying underneath the primary chart. The properties supported by DualY are:
  - measure – A list of measures to display in a splitY chart.
  - splitDualY– Either on or off. When on, the dualY measures displays in a separate chart. When off, a single chart with two y-axis scales will be displayed.
  - splitterPosition–When splitDualY is set to on, this value will provide the ratio of space on the y axis to be given to each of the displayed charts. Valid values are 0 to 1.

The Comparison Metric Tile element can contain yAxisFormat. For yAxisFormat properties, see "Configuring the Unit of Measurement Display in Dashboards".

To add a new comparison metric tile to the current metric profile, click **Add Comparison Metric Tile** (located beneath **Add**) in the toolbar. A new tile entry is added to the navigation pane and selected in the content pane so that its information can be entered.

To remove an existing comparison metric tile from the dashboard settings configuration, select the tile and click **Remove** in the toolbar.

The confirmation window opens.

Click **Yes** to remove the comparison metric tile from the configuration.

Metric Tile Variance Element

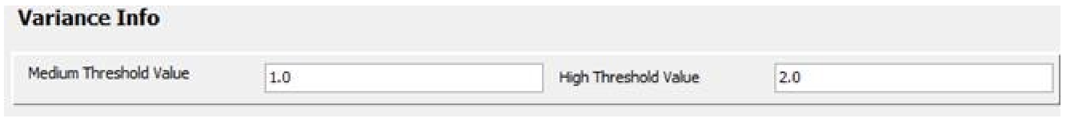
Metric Tiles can be configured to supplement the metric they display with up to two supporting metrics. The supporting metrics can provide additional information about the metric in question, such as variance from performance to performance in a prior period or to planned performance.

These supporting metrics are often configured as variances, and the Dashboard supports the ability to specify threshholds for those variances. Should the variance fall higher or lower than a certain threshold, the Metric Tile will present represent that information through a color-coding scheme:

- Blue–tile contains no supporting metrics or no thresholds defined on those supporting metrics.
- Green–tile contains at least one supporting metric with a defined threshold but the value is in the defined acceptable range.
- Yellow–tile contains at least one supporting metric with a Severe threshold defined and the value is out of the acceptable range of the Severe threshold but supporting metric does not contain a Critical threshold or value is in the acceptable range for that Critical threshold.
- Red–tile contains at least one supporting metric with a Critical threshold defined, and the value is out of the acceptable range for that Critical threshold.

Information about supporting metrics for a metric tile as well as threshold information for those tiles that require it is configured through the Metric Tile Variance element and its child elements, the Left Variance and Right Variance elements (for the left and right slots in the tile, respectively. Each of these elements is covered later.

Figure 12-33 Variance information in the DashboardSettings.json View


Variance Info

Medium Threshold Value 1.0 High Threshold Value 2.0

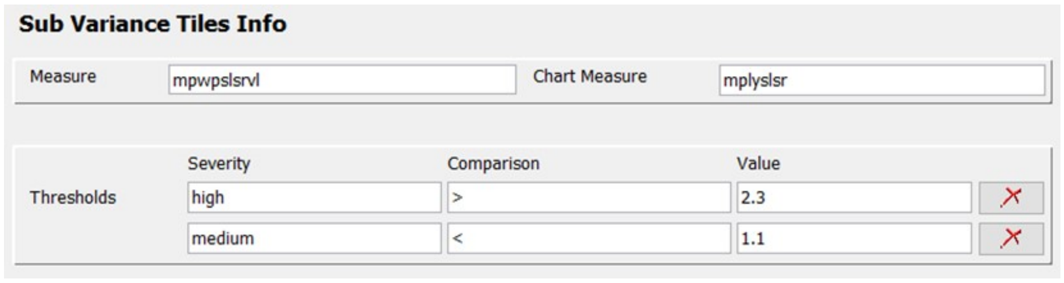
Variance elements are automatically created for every metric tile added to the configuration. However, they only need to be configured for those metric tiles that make use of supporting metrics. Single-metric tiles need not provide values for the variance element.

The set of properties defined for a variance element is:

- Medium Threshold Value–the value that sets the severity threshold for evaluation of variances.
- High Threshold Value–the value that sets the Critical threshold for evaluation of variances.

Additionally, variance elements must contain either of a Left Variance, a Right Variance, or both a Left Variance and Right Variance element. These elements contain information about the supporting metrics and how they should be presented within the Tile and within the detail chart displayed when the tile is selected.

Figure 12-34 Supporting Metric Information in the DashboardSettings.jsonView



Sub Variance Tiles Info

Measure	mpwpslsrvl	Chart Measure	mplyslsr

Thresholds	Severity	Comparison	Value
	high	>	2.3
	medium	<	1.1

Within the Sub Variance Tiles Info panel is located the Thresholds entity. It has the following three fields:

- Severity–sets the severity level to high, medium or low based on the severity
- Comparison–sets the comparison operators ('>', '<', '=')
- Value–sets the severity value

Thresholds entry can be removed by clicking the Remove button () located within each threshold's entry.

The properties defined for a Left Variance or Right Variance element are:

- Measure–the measure containing the value to display in the tile and use for the evaluation of thresholds.
- Plan–a short label to identify the supporting value in the tile.
- Chart Measure–the measure whose data should be displayed within the detail chart for this tile when it is selected.

To add a supporting metric to a tile, select the tile's variance element and click either **Add Left Tile** or **Add Right Tile** (located beneath **Add**) in the toolbar (to create an entry for the left or right slots, respectively). A new left or right variance element is added to the navigation pane and selected in the content pane so that its information can be entered.

To remove an existing left or right variance element from the configuration, select the element and click **Remove** in the toolbar. A confirmation window opens.

Click **Yes** to remove the selected left or right variance element from the configuration.

Exception Tile Element

An Exception Tile contains information about a specific defined exception condition defined within the application. Each tile provides information to identify the exception and to provide a count of the number of violations of the exceptions condition, both in total for the set of positions currently selected within the profile filters but also broken down to provide the number of unique positions in one or more dimensions that contain exceptions for example, the number of sub-classes containing at least one exception).

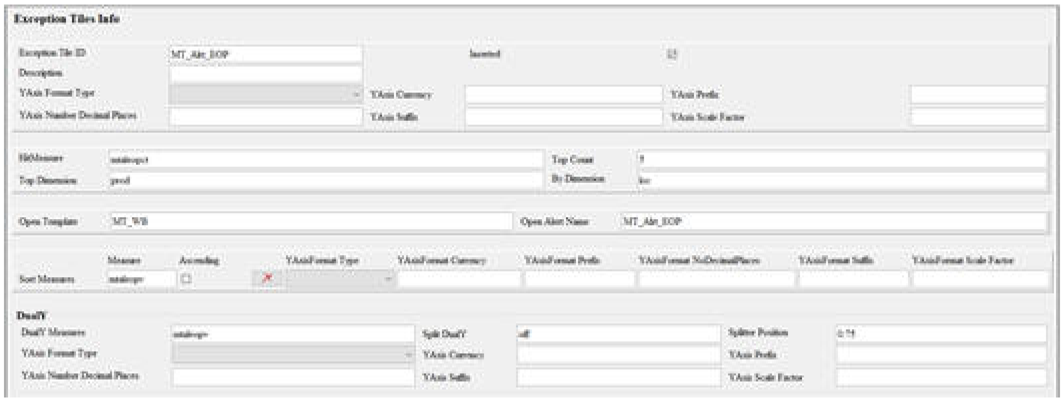
Additionally, many Exception Tiles contain information to allow them to participate in the workflow designed to resolve the exception. These tiles provide the options to perform a launch from the Dashboard into a new or existing workspace for a step defined as the appropriate place to resolve the exception and to initiate Alert Navigation Mode for the appropriate Real Time Alert defined within that step.

Note:

For more information on Alert Navigation Mode in the RPASCE client, see the *Real Time Alerts* chapter in *Oracle Retail Predictive Application Server Cloud Edition User Guide*.

Configuring an Exception Tile consists of providing information about the exception to be displayed and optionally information to support the ability to launch from the Dashboard into the appropriate step and Real Time Alert to resolve the exception.

Figure 12-35 Exception Tile Information in the DashboardSettings.jsonView



Exception Tiles Info

Exception Tile ID	MT_Alt_EOP	Inserted	✓		
Description					
YAxis Format Type		YAxis Currency		YAxis Prefix	
YAxis Number Decimal Places		YAxis Suffix		YAxis Scale Factor	

HitMeasure	totalexpct	Top Count	5
Top Dimension	prod	By Dimension	loc

Open Template	MT_WB	Open Alert Name	MT_Alt_EOP

	Measure	Ascending	YAxisFormat Type	YAxisFormat Currency	YAxisFormat Prefix	YAxisFormat NoDecimalPlaces	YAxisFormat Suffix	YAxisFormat Scale Factor
Sort Measures	totalexpt	☐						

DualY

DualY Measures	totalexpt	Split DualY	off	Splitter Position	0.75
YAxis Format Type		YAxis Currency		YAxis Prefix	
YAxis Number Decimal Places		YAxis Suffix		YAxis Scale Factor	

The set of properties defined for an Exception Tile is:

- Exception Tile ID–an internal name to identify the tile.
- Inserted–true if the tile must be initially visible in the profile. Users may show and hide available tiles from within the Client.
- Hit Measure–the measure containing exception hit count information for the tile.
- Top Count–the number of positions in Top Dimension to display within the detail chart for the tile.
- Top Dimension–the dimension used to organize hit count information in the detail chart. In the chart, the information is displayed as a series of charts, one for each position in Top Dimension being displayed.
- By Dimension–the dimension forming the series displayed in the charts created for the positions of Top Dimension.
- Open Template–the template into which the user can launch to resolve the exception displayed in the tile.
- Open Alert Name–the Real Time Alert in the configured template that is used to drive the exception resolution in Alert Navigation Mode. The value entered for this property must match the case of the name of the Real Time Alert.

- Sort Measures–an optional list of measures that can be used to sort the information displayed within the chart. The contents of the sortMeasures array are a list of JSON objects with the following properties:
- DualY–a JSON object used to support the use of dual-Y axis scales. By specifying the dualY property, it is possible to display a secondary set of values with independent scaling, either within the same chart as other measures or in a secondary chart displays underneath the primary chart. The properties supported by DualY are:
  - measure – A list of measures to display in a splitY chart.
  - splitDualY– Either on or off. When on, the dualY measures display in a separate chart. When off, a single chart with two y-axis scales will be displayed.
  - splitterPosition–When splitDualY is set to on, this value will provide the ratio of space on the y axis to be given to each of the displayed charts. Valid values are 0 to 1.

The Exception Tile element can contain yAxisFormat. For yAxisFormat properties, see "Configuring the Unit of Measurement Display in Dashboards".

To add a new exception tile to the configuration, select an exception profile and click **Add Exception Tile** (located beneath **Add**) in the toolbar. A new exception tile is added to the navigation pane and selected in the content pane so that its information can be entered.

To remove an existing tile from the configuration, select the tile and click **Remove** in the toolbar. A confirmation window opens.

Click **Yes** to remove the tile from the configuration.

Creating Dashboard Profiles: Two Examples

This section provides two examples that describe how to initially create Dashboard Profiles using the Deployment Tool. Refer to previous sections of this chapter for questions about the GUI components within the Deployment Tool.This section focuses on the business logic in configuring the profiles. The unit of measurement formatting, that is, the yAxisFormat, can be applied to the Dashboard Profile Tiles such as Metric Tile, Comparison Metric Tile, or Exception Tile, Sort Measures, or Dual Y-Axis. See Configuring Dashboards in RPASCE for detailed examples of the Y-Axis Format.

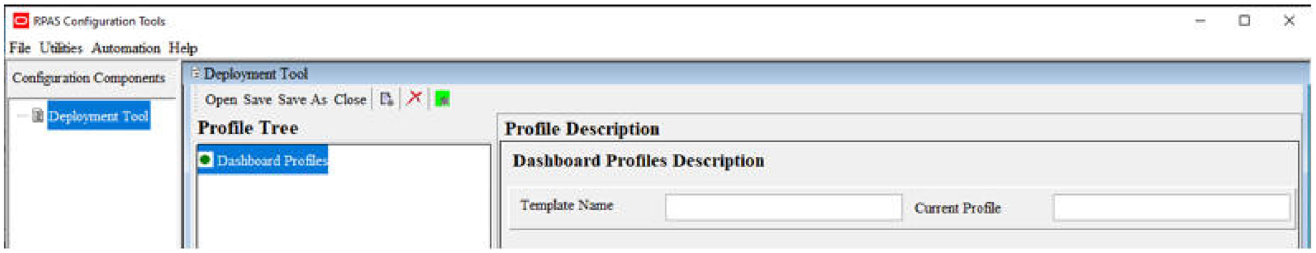
Example 1: Creating a Dashboard Seasonal Profile

The process for creating Metric and Seasonal Profiles are very similar. This example describes the creation of a seasonal profile, including the definition of one metric tile within that profile. Starting from an (nearly) empty file, each step of the profile creation process using the Configuration Deployment Tool is described.

Creating a Seasonal Profile

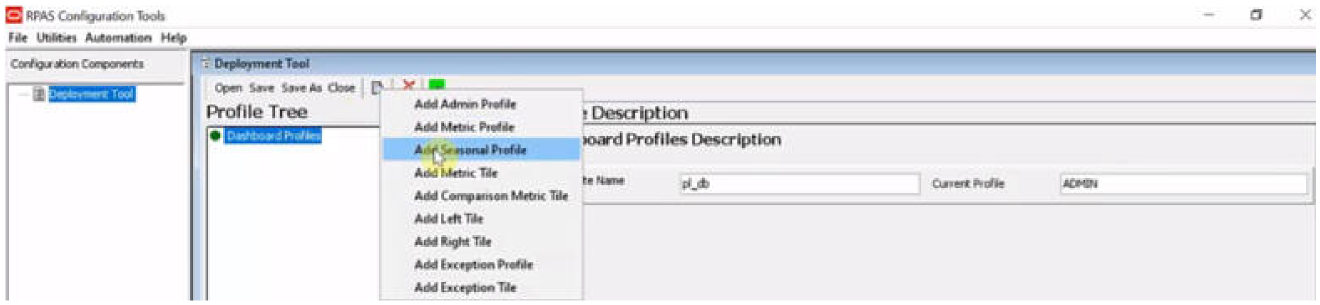
From the Configurator, select **New** to initially create a Dashboard Setting Configuration or select **Open** to load an existing resource and then delete the content of that resource in order to create the blank one. Figure 12-36 shows a blank Dashboard Setting Configuration.

Figure 12-36 Blank Dashboard Setting Configuration



Next, create a new Seasonal Profile, shown in Figure 12-37.

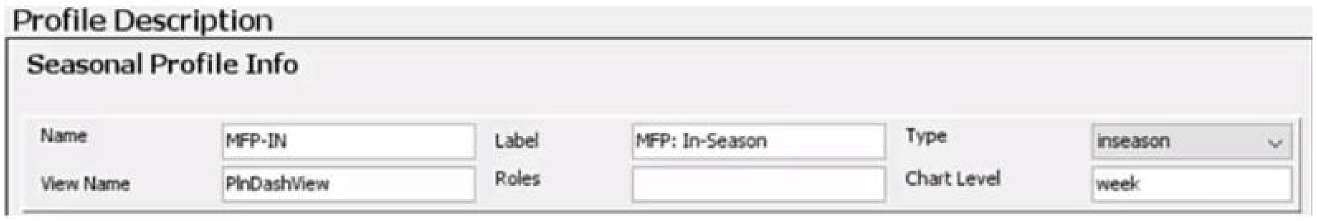
Figure 12-37 Seasonal Profile



Template Name: pl_db
Current Profile: ADMIN

Figure 12-38 shows an In-Season profile created with the name MFP-IN and the label MFP: In-Season to match. The view to be displayed in the Dashboard workbook has a View Name of PlnDashView. This profile is available to all MFP application users and is not restricted to a particular single role, so the Roles field is left blank. In order to restrict the profile view to MFP planners only, the Roles field must be filled with the role of MFP planners' user group. The level to be displayed in the charts for the time series is the week level, so the value of week is entered into the Chart Level field. Data is most commonly viewed as a time series at week level.

Figure 12-38 Profile Description



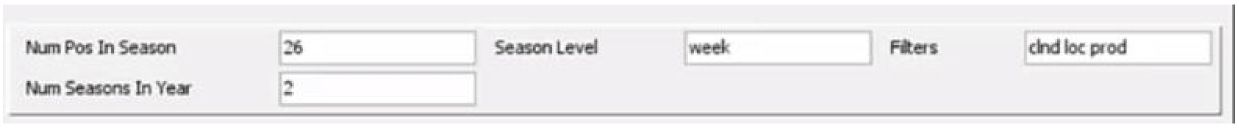
Profile Description

Seasonal Profile Info

Name	MFP-IN	Label	MFP: In-Season	Type	inseason
View Name	PlnDashView	Roles		Chart Level	week

Because the Seasonal Profile is selected, the dashboard will draw a time horizon based on seasonality. Two fields must be configured. Seasonal type can be either the current period (InSeason type) or in the coming period (PreSeason type). In Figure 12-38, the Configurator selected the Inseason type. The season must be defined, as shown in Figure 12-39 The configuration divides a fiscal year into two seasons in which each season has 26 weeks. The filters available at the top of the Dashboard UI are clnd loc prod for a standard three-dimensional Dashboard workbook.

Figure 12-39 Season Definition



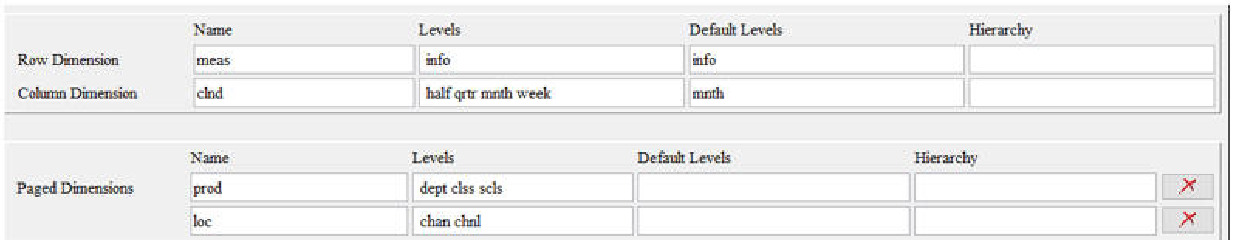
Num Pos In Season	26	Season Level	week	Filters	clnd loc prod
Num Seasons In Year	2				

Next, define how to lay out dimensions across the row, column, and page axis of the detail chart in the Dashboard view. Each profile contains one row dimension, one column dimension, and as many additional paged dimensions as needed, based upon the intersection of the view configured in the View Name field. In Figure 12-40, Row Dimension is at meas dimension and info is the default level. If no levels are specified, the lowest level will be used as the default for the display. The Hierarchy field was originally designed to tell the Dashboard which branch the branch hierarchy to use; however, the Dashboard can now determine this automatically, and this field is often left blank.

Since this is a seasonal profile, the Column Dimension field is set to clnd to display calendar on the x-Axis in order to see time series. The Levels field is used to determine at which levels the dashboard displays the data inside the chart. In Figure 12-40, there are four level options available: half, qrtr, mnth, and week. To see aggregated data for the default level, enter mnth.

Next, configure the dimensions Product and Location. They are configured as page dimensions. Since by default there is only one Paged Dimension object in the blank profile, a second Paged Dimension object is added. The Default Levels fields are left blank, which means the lowest level specified in the Levels field will be used as the default.

Figure 12-40 Configure Dimensions



	Name	Levels	Default Levels	Hierarchy
Row Dimension	meas	info	info	
Column Dimension	clnd	half qrtr mnth week	mnth	

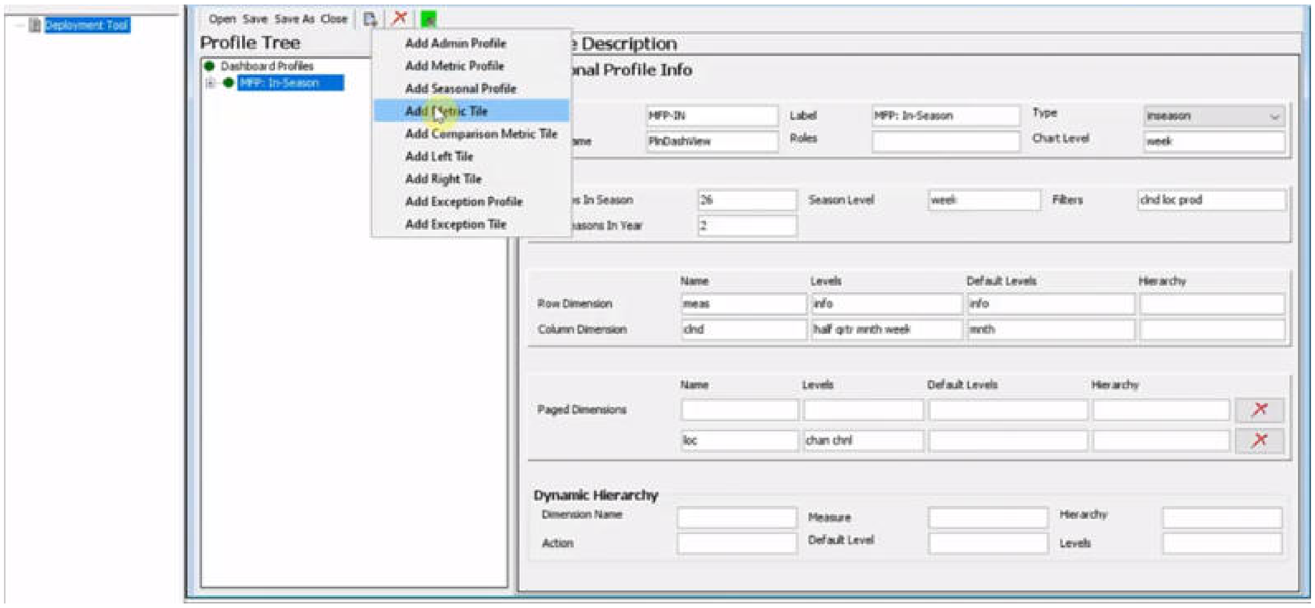
	Name	Levels	Default Levels	Hierarchy
Paged Dimensions	prod	dept clss scls		
	loc	chan chnl		

This seasonal profile does not use dynamic hierarchy, so the Dynamic Hierarchy fields are left blank. If dynamic hierarchy is used, values must be entered into the Dynamic Hierarchy fields. Users must enter the name of dimension affected by the dynamic hierarchy into the Dimension Name field, the measure containing the information for setting up the dynamic hierarchy mapping into the Measure field, the hierarchy that representing the branch inside the dimension that uses the dynamic hierarchy, the menu ID of the menu item into the Action field, the Default Levels, and Levels. See Seasonal Profile Element for details.

Creating a Metric Tile

Next, right click the MFP: In-Season profile and select Add Metric Tile to add a new metric tile, as shown in Figure 12-41.

Figure 12-41 Create Metric Tile



Description
nal Profile Info

	MFP-IN	Label	MFP: In-Season	Type	inseason
me	FinDashView	Roles		Chart Level	week
s In Season	26	Season Level	week	Filters	clnd loc prod
asons In Year	2				

	Name	Levels	Default Levels	Hierarchy
Row Dimension	meas	info	info	
Column Dimension	clnd	half qtr mnth week	mnth	

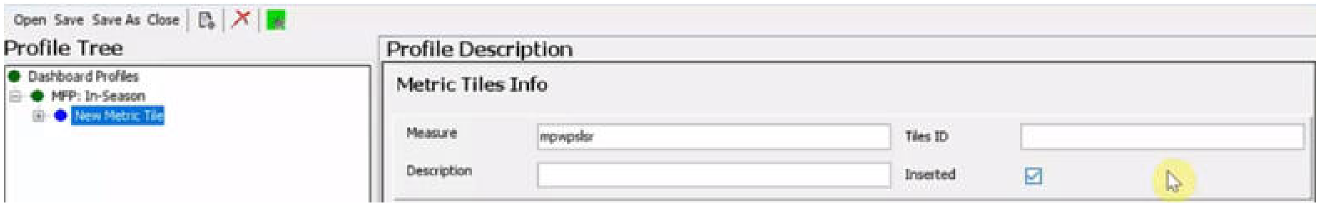
	Name	Levels	Default Levels	Hierarchy
Paged Dimensions				
	loc	chan chnl		

Dynamic Hierarchy

Dimension Name		Measure		Hierarchy	
Action		Default Level		Levels	

In the Metric Tile Info interface shown in Figure 12-42, enter the name of measure for the tile. If the dashboard is to be used in a Retail Home implementation, the tile must be uniquely identified and the Tiles ID field requires a unique name, since more than one tile may reference the same measure. However, since this is not the case in the current example, the Tiles ID field is left blank. The Inserted check box is selected, indicating that this tile is visible by default in the Dashboard UI and no editing for setting visible tiles is required.

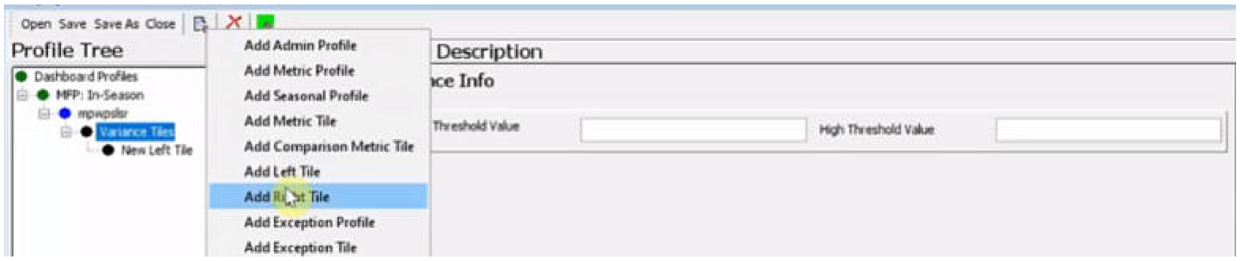
Figure 12-42 Measure Name



Measure	mpwpslsr	Tiles ID	
Description		Inserted	☑

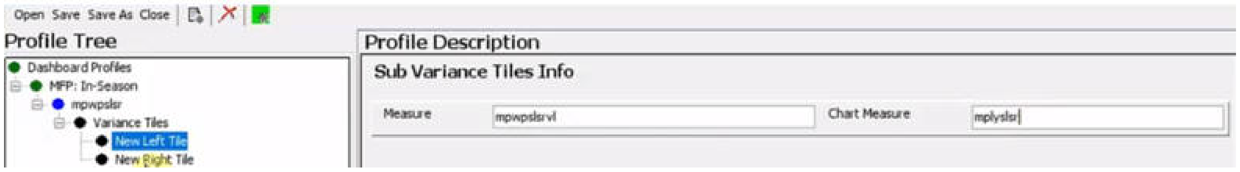
Next, add additional metrics for the tile. Click the **+** sign next to the new metric tile to display the Variance Tiles menu item and select it to add the variance tile. Supplement the main metric tile by adding a left tile and a right tile.

Figure 12-43 Add Variance Tile



The left tile is used to compare the current year sales with the last year sales. The measure name mpwpslsvl shown in Figure 12-44 is the variance of the current year sales in comparison to last year sales; that is the value shown in the tile itself. The measure that holds the historical sales data will be visible inside the chart.

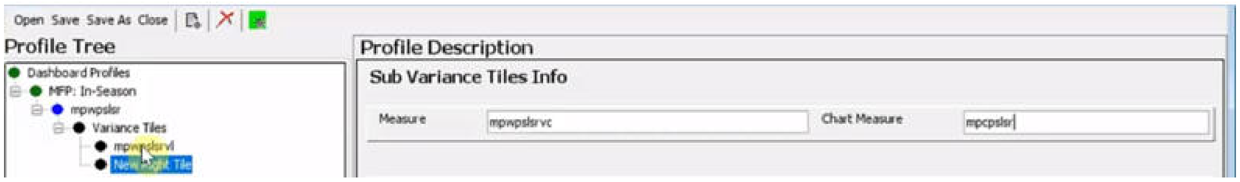
Figure 12-44 Compare Sales



Measure: mpwpslsrvl
Chart Measure: mplyslsr

The right tile, shown in Figure 12-45, displays the comparison to the current plan. As before, the measure appearing in the tile is the actual variance itself (mpwpslsrvs), but inside the chart, the metric that holds current sales data mpcpslsr is displayed.

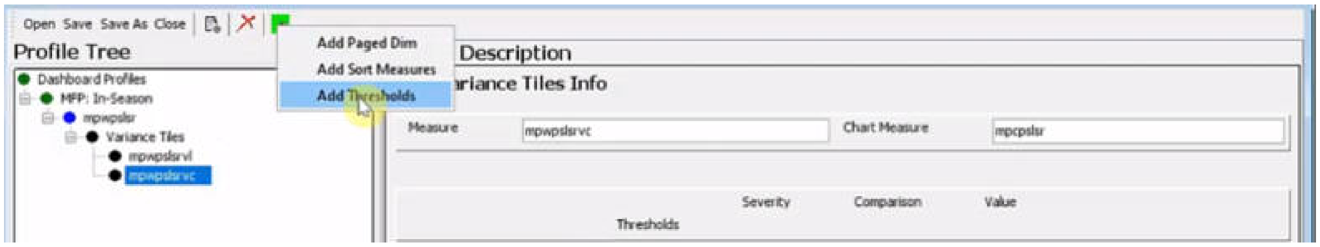
Figure 12-45 Compare Plans



Measure: mpwpslsrvc
Chart Measure: mpcpslsr

The final step is specifying the thresholds. Thresholds can be specified for either the left tile or the right tile. In Figure 12-46, the right tile has two thresholds, one medium and the other severe or high, for the variance to the current plan. Right click the right tile (mpwpslsrvc tile) shown in Figure 12-46, then click Add Thresholds to add one threshold. Repeat this step to add the second threshold.

Figure 12-46 Specify Thresholds



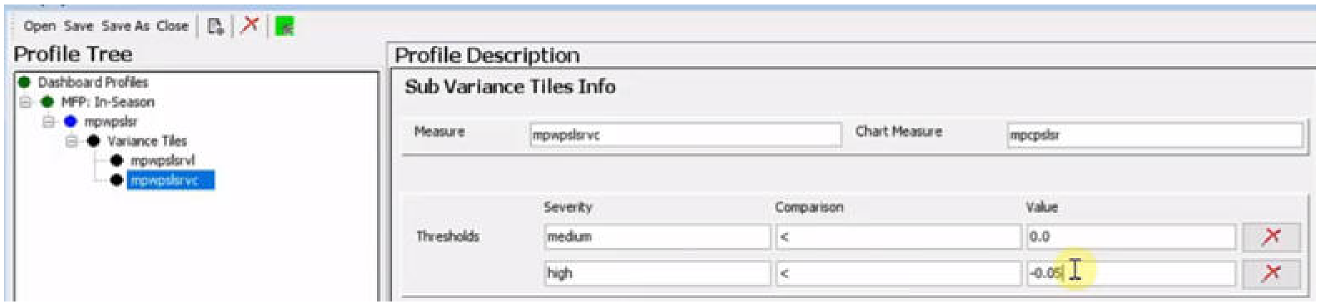
Measure: mpwpslsrvc
Chart Measure: mpcpslsr

Thresholds: Severity, Comparison, Value

The first threshold is a medium threshold. If sales do not perform to the plan, the variance is less than 0, and the Dashboard UI flags it as medium error. The badge inside the tile on Dashboard UI will show that it is not performing up to the current plan.

The second threshold is a high threshold. If the variance is more than five percent less than the plan, then it is flagged as a high error. The red badge in the UI indicates this, as shown in Figure 12-47.

Figure 12-47 Flag Errors



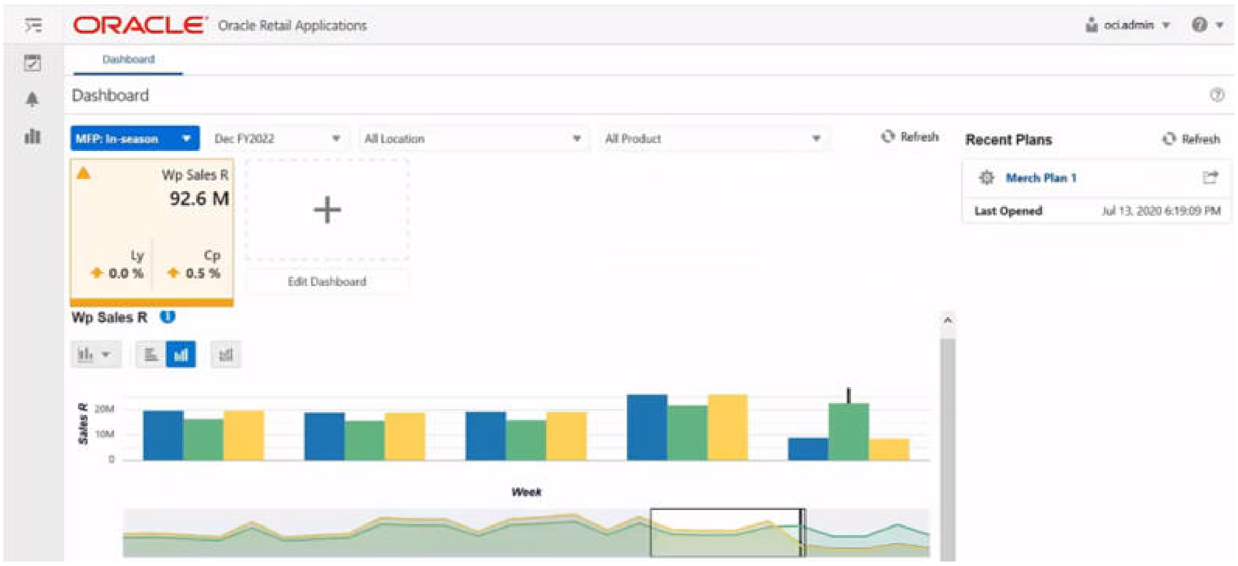
Measure: mpwpslsrvc
Chart Measure: mpcpslsr

Thresholds

Severity	Comparison	Value
medium	<	0.0
high	<	-0.05

The results of the Dashboard Seasonal Profile configuration are shown in Figure 12-48.

Figure 12-48 Dashboard UI for the MFP In-Season Profile



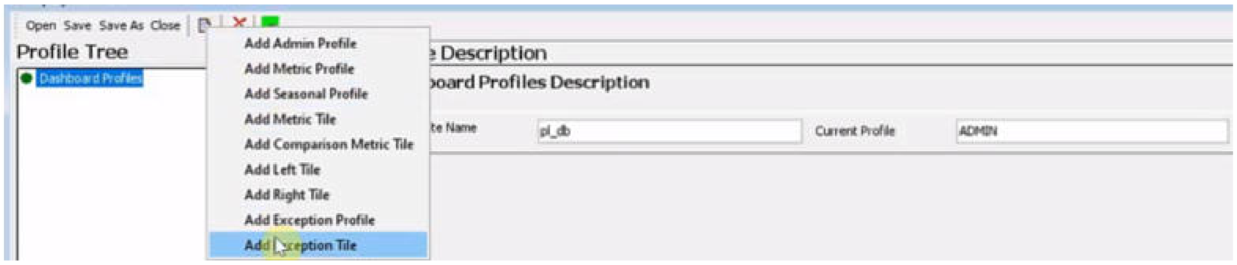
Example 2: Creating a Dashboard Exception Profile

This section describes the steps to create a dashboard exception profile.

Creating an Exception Profile

As in Example 1: Creating a Dashboard Seasonal Profile, starting from an (nearly) empty file, this example describes the steps to create the exception profile. For greater detail regarding the components used, see Example 1.

Figure 12-49 Add Exception Tile



Complete the fields in the Exception Profiles Info interface as described in this section. The Type field is pre-populated with exceptions. A view name, profile name, and profile label are provided. The Roles field is left blank so that the Dashboard profile view is available for all users of the application.

The standard, two half seasons with 26 weeks within each season, is configured. A fiscal year contains two seasons. The filters used to range data in the Dashboard UI contain clnd loc prod. The Exception Profile supports performing exception-based processes in pre-season planning. Users can select the inseason, pre-season, or both as the season type. As in the Seasonal Metric Profile, if inseason is selected, the Dashboard will center itself in the time period containing today. If preseason is selected, it will determine which period today is in and will center on the upcoming time period so that users can resolve exceptions in the preseason planning.

The Product dimension is configured as the Row Dimension and Location as the Column Dimension. That means that the exceptions are first grouped by product, then further grouped by location. The Levels contain levels available in Filters that users can range down. As in the Metric Profile, the hierarchy field is usually left blank since the Dashboard can determine the value.

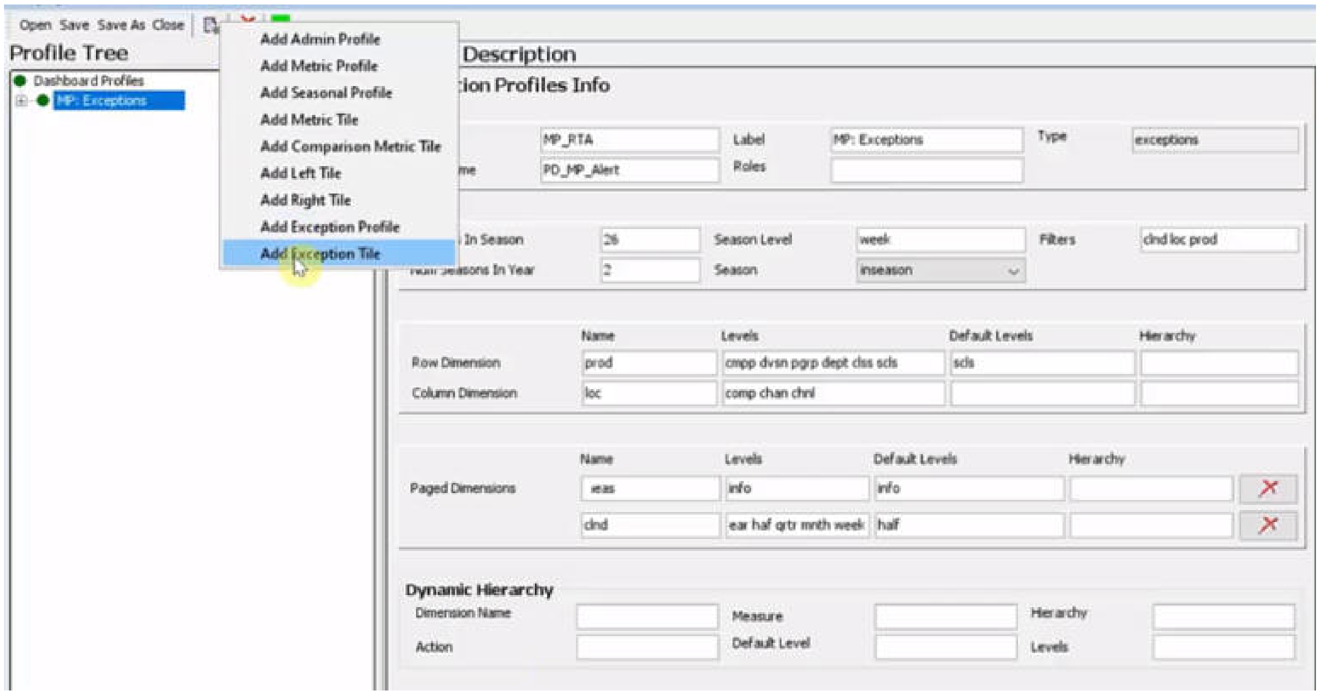
The paged dimensions have two dimensions, one at meas and the other at clnd. The default level is at dimension half, a higher level, so that users can drill down the data to lower levels defined within the Levels field.

This configured Exception profile does not use the Dynamic Hierarchy feature. If the Dynamic Hierarchy is used, then the corresponding information must be entered.

Creating an Exception Tile

Right click the MP:Exception profile and select Add Exception Tile from the drop-down list, as shown in Figure 12-50.

Figure 12-50 Adding an Exception Tile



ion Profiles Info

	MP_RTA	Label	MP: Exceptions	Type	exceptions
me	PD_MP_Alert	Roles			

In Season	26	Season Level	week	Filters	clnd loc prod
Num Seasons In Year	2	Season	inseason		

	Name	Levels	Default Levels	Hierarchy
Row Dimension	prod	cmpp dvsn pgrp dept clss scls	scls	
Column Dimension	loc	comp chan chnl		

	Name	Levels	Default Levels	Hierarchy
Paged Dimensions	meas	info	info	
	clnd	ear hlf qtr mnth week	half	

Dynamic Hierarchy

Dimension Name		Measure		Hierarchy	
Action		Default Level		Levels	

The alert of interest in is the Min Sales count. Enter the corresponding information is entered into the Exception Tiles Info interface. See Exception Profile Element for a detailed explanation of each component.

To begin, an explicit Tile ID is required for every Exception Tile (but not necessarily for a Metric Tile). The Inserted check box is selected by default so that this tile is visible in the Dashboard UI.

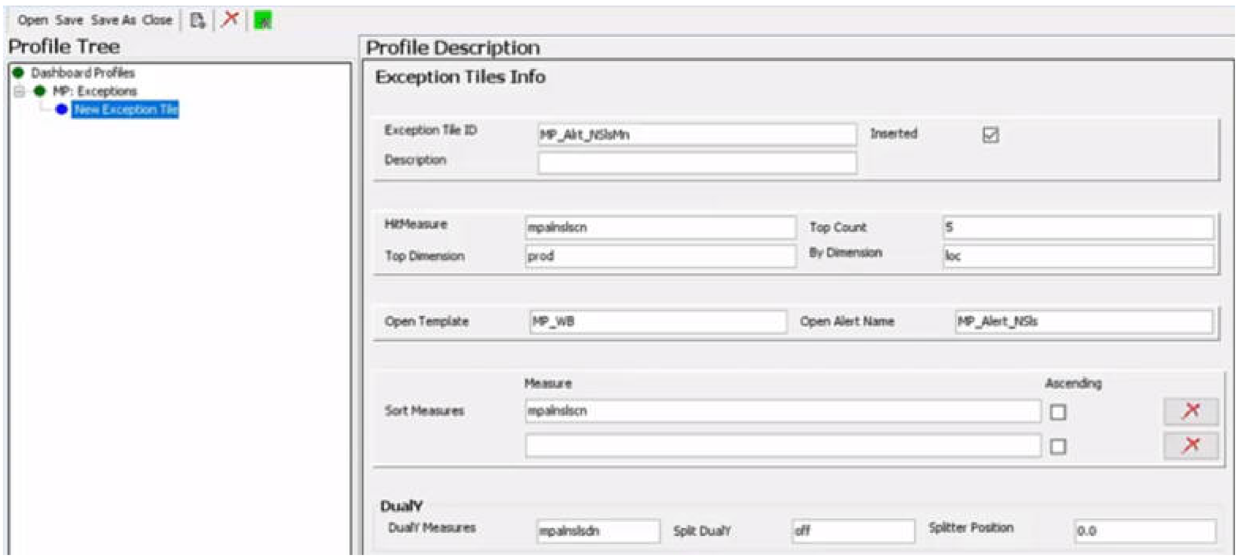
The HitMeasure field has the measure name mpalnslscn, indicating the measure containing the exception hit count information for the tile. The Top Dimension entered is the Product dimension, and By Dimension is Location dimension. The Top Dimension and By Dimension override the settings in the profiles if they are different. Note that in the example, these two settings are the same. The TopCount is set to 5 so that when users view the alert hits by product and location dimensions, they will start out with the top five products in the RPASCE UI display.

Next, configure the ability to launch out of the Dashboard into the workbook in order to resolve exceptions displayed in the tile. Enter the workbook template to be launched into the Open Template. Enter the real time alert name into the Open Alert Name.

In addition to simply ranking the display based on those products that have the most alert hits, users can also use Sort Measures to group data based on the metric values. In this example, the check box for Ascending is deselected for the first sort measure, so the data is sorted in descending order.

The DualY section is optional. Once configured, it is used to display a secondary metric inside the chart when the user is looking at the information. The DualY section contains an optional configuration to display alert hits on a different scale via a split DualY. With Split DualY, instead of one chart, users see a distinct, secondary chart that stacks on the other in the RPASCE UI display. In this example, the Split DualY is not used, and the field contains off and Splitter Position 0.0.

Figure 12-51   Exception Tile Info



Exception Tiles Info

Exception Tile ID	MP_Alrt_NSlsMn	Inserted	✓
Description			
HitMeasure	mpalnslscn	Top Count	5
Top Dimension	prod	By Dimension	loc
Open Template	MP_WB	Open Alert Name	MP_Alert_NSls

	Measure	Ascending
Sort Measures	mpalnslscn	☐
		☐

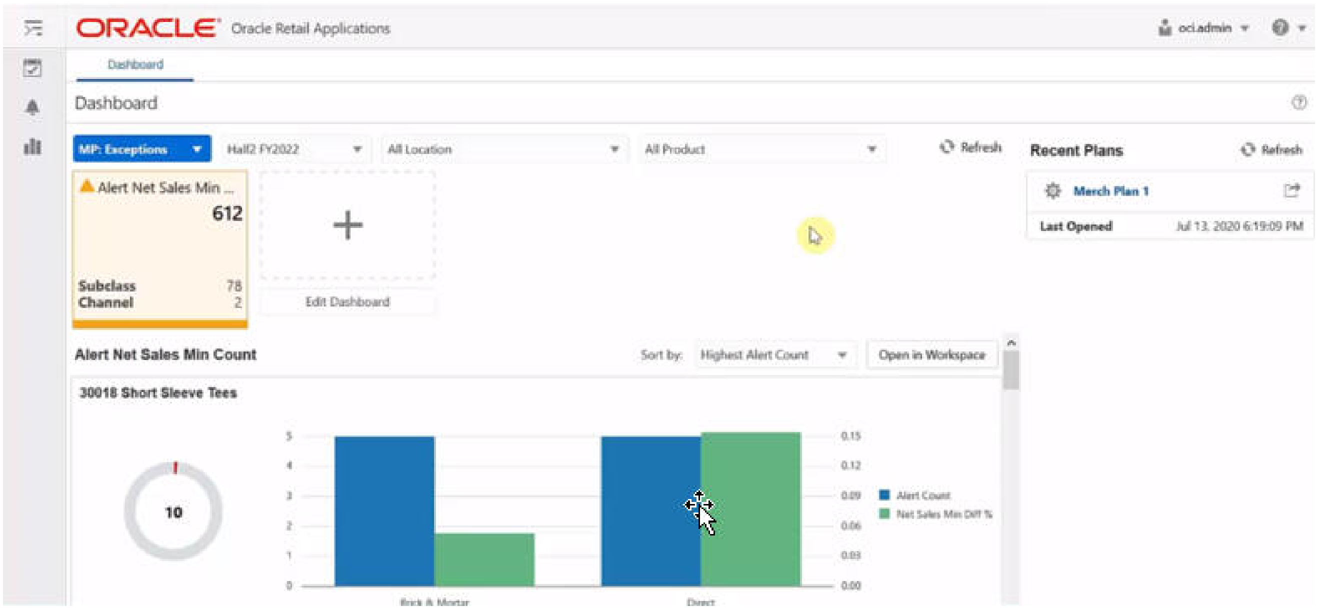
DualY

DualY Measures	mpalnslsdn	Split DualY	off	Splitter Position	0.0

Dashboard Exception Profile Instance in the RPASCE UI

Figure 12-52 shows the Dashboard instance in the RPASCE UI. In the MP: Exceptions profile, users can view the exception Tile, the Location, Product and the optional levels within each hierarchy. Users can launch the workbook from the Open in Workspace option. The Sort By option contains an additional sorting metric specified by the Sort Measure configuration. Alert hits are broken out first by product, then by location dimension. The second metric Net Sales Min Diff % plugged into the DualY configuration is also in the chart. There is one range for the alert hit and a second range for the diff percentage.

Figure 12-52 Dashboard Instance



Deployment Tool – Retail Home Dashboard Settings

Retail Home Portal contains a home page that users view and interact with when logging into the system. This page is called the Retail Home Dashboard and it provides summary information in the form of health-check metric tiles of the retail application.

The Cloud engineering team deploys and starts up the customer's retail planning applications and the Retail Home application to various provisioned servers. When the user logs into the Retail Home Portal, the metric tiles display on the Dashboard authorized by the user's currently effective enterprise role. For each tile, the Retail Home application invokes a web service (in this example, called Edge web service) on the deployed instance of the planning application that contributed the tile. The Edge web service retrieves the tile's measure data from the dashboard workbook and returns the data to Retail Home, which renders the measure data in the tile.

How Metric Tiles are Populated in the Retail Home Portal Dashboard Page

Conceptually, a Dashboard is a hybrid of a traditional workbook containing the information backing the display and a new view representation consisting of information provided in a set of tiles, each of which can be selected to provide supporting information. The supporting information is presented in various UI formats that display lower-level information about the summary information presented within the tile.

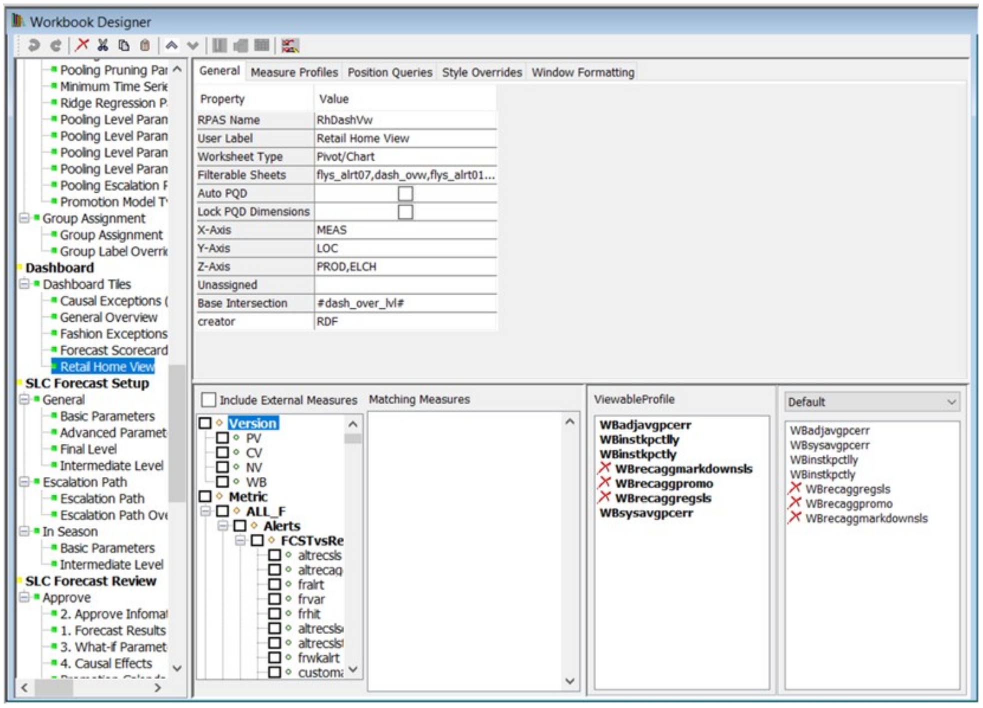
Because the dashboard is a hybrid of traditional RPASCE workbooks and a new visual representation, the process of configuring a Retail Home Dashboard consists of two components. First, a workbook must be configured in the application that defines the data available within the dashboard and how it is created and maintained within that workbook. Second, a resource file that defines the presentation layer configuration that determines how that information is made available in the form of tiles. Their detail charts must be created and deployed to the client tier.

In detail, the retail planning application must perform the following two tasks for Retail Home uptake:

1. Create a Retail Home-specific worksheet in a Dashboard workbook.

   The Dashboard Workbook can be an existing regular Dashboard Workbook or a new Dashboard Workbook dedicated to Retail Home. The Retail Home worksheet must be populated with all the Retail Home tile measures, calculation rules, and any special expression libraries that are required to produce measure data. This separate retail-home-specific worksheet in the Dashboard workbook can start life as a copy of the existing application-level dashboard worksheet (for example, `PlnDashVw`), renamed to `RhDashVw` and modified as required. This worksheet must have all the tile measures. Any special rules to produce measure data (such as a rollup over a certain number of weeks up to the current date) can be used in this worksheet. See Configuring Dashboards in RPASCE for how to configure a Dashboard workbook and its worksheets.

   Figure 12-53   Retail Home Worksheet Added to the Regular Dashboard Workbook



Property	Value
RPAS Name	RhDashVw
User Label	Retail Home View
Worksheet Type	Pivot/Chart
Filterable Sheets	flys_alrt07,dash_ovw,flys_alrt01...
Auto PQD	☐
Lock PQD Dimensions	☐
X-Axis	MEAS
Y-Axis	LOC
Z-Axis	PROD,ELCH
Unassigned	
Base Intersection	#dash_over_lvl#
creator	RDF

2. Create the Retail Home configuration file resource.

   This file resource, often called the **Retail Home Dashboard Settings** file, is a JSON encoded text document that contains information about the dashboard tile states and their tiles that is used by the client to organize and present the data contained within the dashboard workbook. The Deployment Tool contains an interface for the creation of the content of the Retail Home Dashboard Settings resource. Using this tool, the person performing the configuration can interact with a graphical user interface that is similar to other Configuration Tools-based interfaces and the tool will generate the appropriate JSON file content. Like other Configuration Tools interfaces, this allows the people performing the configuration to focus on what information they want to present in the configuration (as defined in terms of measures, RPASCE views, and so on) and let the tool manage the generation of the appropriate file resources.

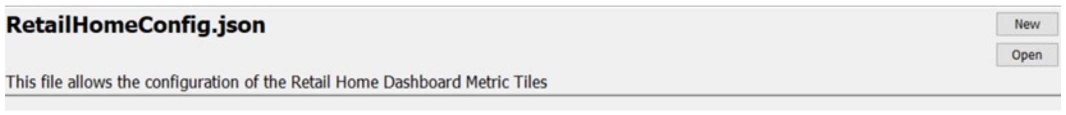
RetailHomeConfig.json View

The RetailHomeConifgs.json View can be used to create, delete, and modify the contents of a Retail Home Dashboard Settings file resource using a set of actions represented through buttons, menu items, and text forms. This information is organized around a set of entities that describe the view layer of the Retail Home Dashboard, such as dashboard tile states and metric tiles.

By representing this information in a user interface, the tool allows users to work with an organized form of the information that can be easily navigated and modified. Then, through the Save action, the information present within the interface can be written into the appropriately formatted JSON file by the tool, allowing users to concentrate on the content and allowing the tool to manage the encoding process.

The RetailHomeConfig.json View can be accessed from the main Deployment Tool screen through the RetailHomeConfig.json item.

Figure 12-54 RetailHomeConfig.json View

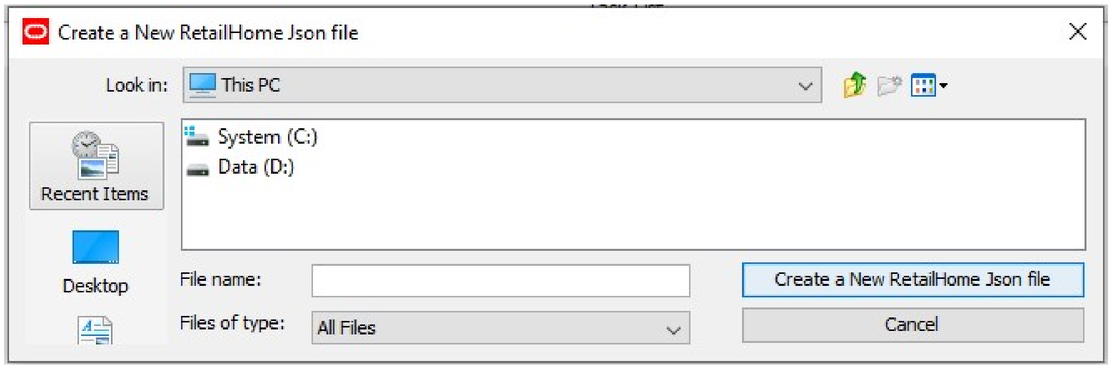

RetailHomeConfig.json

This file allows the configuration of the Retail Home Dashboard Metric Tiles

RetailHomeConfig.json Option in Deployment Tool

To access an existing resource, the user selects Open, located in the upper right-hand corner of the RetailHomeConfig.json entry. The user is presented with the files saved on the local system. The user can also use the New action to create a new Retail Home Dashboard Settings resource. Select New and the following window displays. The user can specify the name of the Retail Home JSON file. If the file name does not end with a .json suffix, Tools will add it automatically.

Figure 12-55 Create a New RetailHome JSON File Window


Create a New RetailHome Json file

Look in: This PC

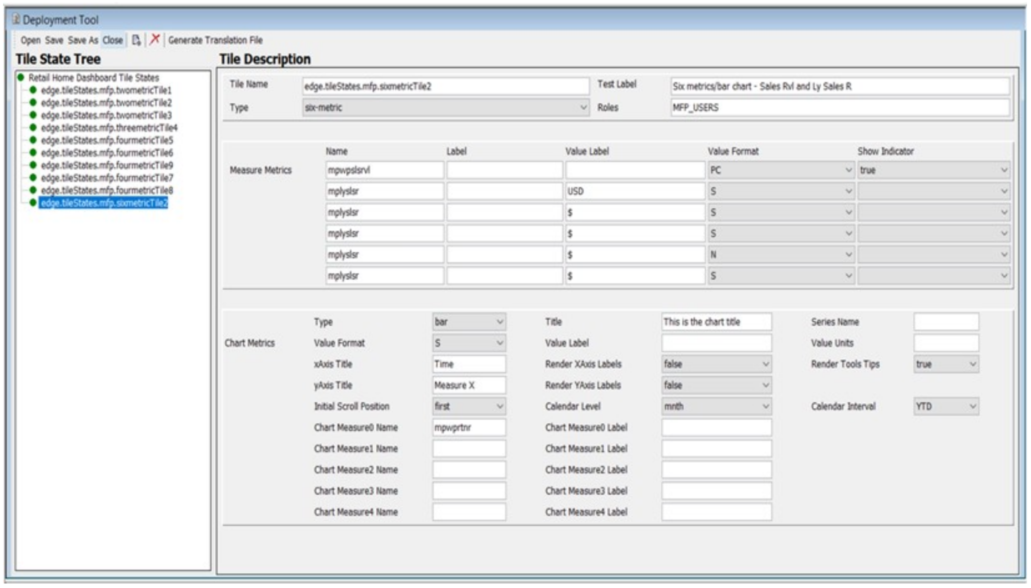
System (C:)
Data (D:)

File name:

Files of type: All Files

Once a dashboard settings resource opens, its contents are loaded into the RetailHomeConfig.json View.

Figure 12-56 RetailHomeConfig.json View with Loaded Resource



Tile State Tree

- Retail Home Dashboard Tile States
  - edge.tileStates.mfp.twometricTile1
  - edge.tileStates.mfp.twometricTile2
  - edge.tileStates.mfp.twometricTile3
  - edge.tileStates.mfp.threemetricTile4
  - edge.tileStates.mfp.fourmetricTile5
  - edge.tileStates.mfp.fourmetricTile6
  - edge.tileStates.mfp.fourmetricTile9
  - edge.tileStates.mfp.fourmetricTile7
  - edge.tileStates.mfp.fourmetricTile8
  - edge.tileStates.mfp.sixmetricTile2

Tile Description

Tile Name	edge.tileStates.mfp.sixmetricTile2	Test Label	Six metrics/bar chart - Sales Rvl and Ly Sales R
Type	six-metric	Roles	MFP_USERS

	Name	Label	Value Label	Value Format	Show Indicator
Measure Metrics	mpwpslsrvl			PC	true
	mplyslsr		USD	S	
	mplyslsr		$	S	
	mplyslsr		$	S	
	mplyslsr		$	N	
	mplyslsr		$	S	

Chart Metrics	Type	bar	Title	This is the chart title	Series Name	
	Value Format	S	Value Label		Value Units	
	xAxis Title	Time	Render XAxis Labels	false	Render Tools Tips	true
	yAxis Title	Measure X	Render YAxis Labels	false		
	Initial Scroll Position	first	Calendar Level	mnth	Calendar Interval	YTD
	Chart Measure0 Name	mpwprtnr	Chart Measure0 Label			
	Chart Measure1 Name		Chart Measure1 Label			
	Chart Measure2 Name		Chart Measure2 Label			
	Chart Measure3 Name		Chart Measure3 Label			
	Chart Measure4 Name		Chart Measure4 Label			

Within the RetailHomeConfig.json view, users are presented with a navigation pane on the left, containing a tree structure of the entities making up a Retail Home dashboard settings configuration such as dashboard tilestates and the tiles they contain. When an entity is selected within the navigation pane, its information is loaded into the content pane on the right. This allows users to view and modify that entity's information. Metric tiles can be created and deleted through actions present on the top toolbar.

RetailHomeConfig.json Resource Contents

The following elements make up the contents of the RetailHomeConfig.json resource:

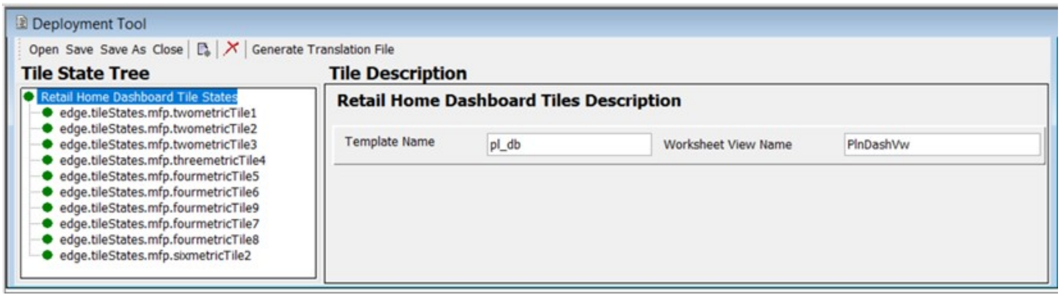
- Tile States
- Metric Tile

Template, Worksheet, and TileStates Definition

The tree root on the left side of the Navigation pane defines the template name, worksheet view name, and the tileStates.

Example Template and Worksheet Configuration for Retail Home Dashboard

Figure 12-57 Example Template and Worksheet Configuration for Retail Home Dashboard



Retail Home Dashboard Tile States
- edge.tileStates.mfp.twometricTile1
- edge.tileStates.mfp.twometricTile2
- edge.tileStates.mfp.twometricTile3
- edge.tileStates.mfp.threemetricTile4
- edge.tileStates.mfp.fourmetricTile5
- edge.tileStates.mfp.fourmetricTile6
- edge.tileStates.mfp.fourmetricTile9
- edge.tileStates.mfp.fourmetricTile7
- edge.tileStates.mfp.fourmetricTile8
- edge.tileStates.mfp.sixmetricTile2

Retail Home Dashboard Tiles Description

Template Name: pl_db Worksheet View Name: PlnDashVw

Table 12-1 provides more detailed descriptions for each of these three parameters.

Table 12-1 Retail Home Dashboard Parameters

Parameter	Type	Required	Description
templateName	String	Y	The template name for the dashboard workbook. For example, in MFPRCS it is pl_db.
worksheetViewName	String	Y	The name of the worksheet view in the dashboard workbook that has the dashboard data. It must be same as configured within the dashboard workbook. For example, in MFPRCS it is called `PlnDashVw`.
tileStates	Object	Y	An array of tile state objects. Choose a unique name field for each tile state. Only one tilestates object is supported currently.

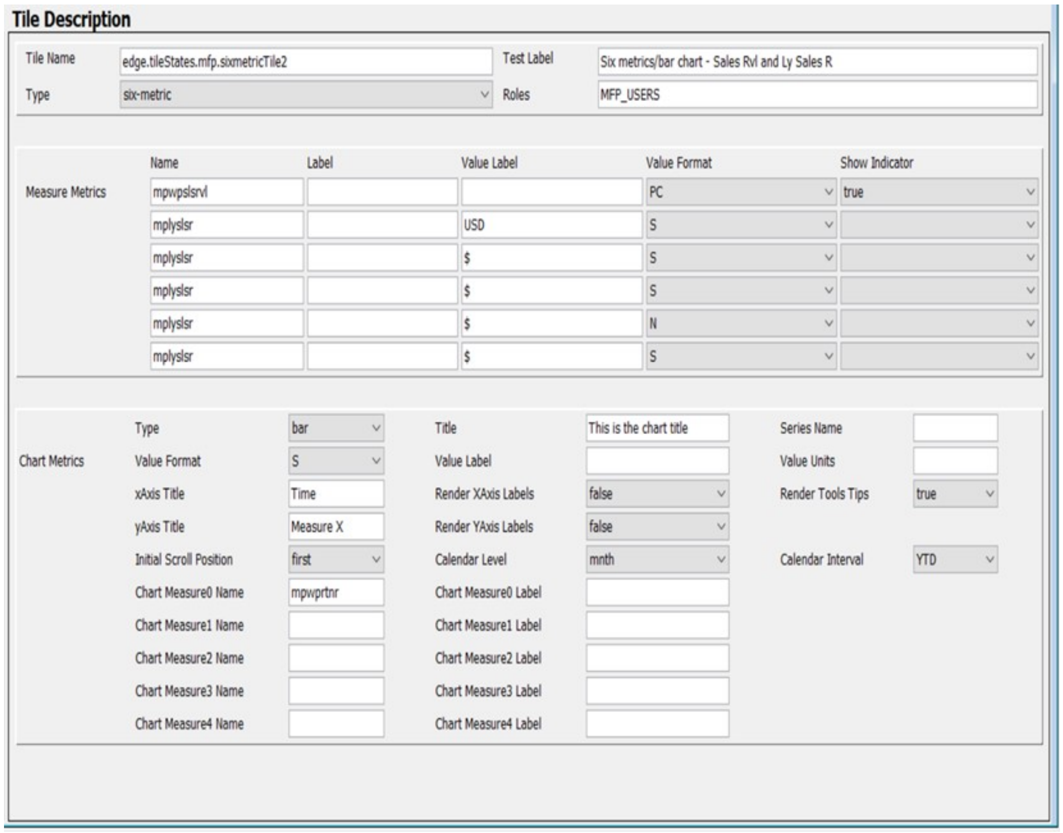
Here is an example of the JSON content at the top level of RetailHomeConfig.json file. Note that the `templateName` and `worksheetViewName` must be the same as configured in the Dashboard Workbook.

```
{
    "templateName": "pl_db",
    "worksheetViewName": "PlnDashVw",
    "tileStates":  [ ... ]
}
```

Metric Tile Definition

The Tilestates can contain one or more metric tile objects that are displayed in the right-side Tile Description pane.

Figure 12-58 Tile Description



Tile Description

Tile Name	edge.tileStates.mfp.sixmetricTile2	Test Label	Six metrics/bar chart - Sales Rvl and Ly Sales R
Type	six-metric	Roles	MFP_USERS

Measure Metrics	Name	Label	Value Label	Value Format	Show Indicator
	mpwpslsrvl			PC	true
	mplyslsr		USD	S	
	mplyslsr		$	S	
	mplyslsr		$	S	
	mplyslsr		$	N	
	mplyslsr		$	S	

Chart Metrics

Type	bar	Title	This is the chart title	Series Name	
Value Format	S	Value Label		Value Units	
xAxis Title	Time	Render XAxis Labels	false	Render Tools Tips	true
yAxis Title	Measure X	Render YAxis Labels	false		
Initial Scroll Position	first	Calendar Level	mnth	Calendar Interval	YTD
Chart Measure0 Name	mpwprtnr	Chart Measure0 Label			
Chart Measure1 Name		Chart Measure1 Label			
Chart Measure2 Name		Chart Measure2 Label			
Chart Measure3 Name		Chart Measure3 Label			
Chart Measure4 Name		Chart Measure4 Label			

The set of properties defined for a Metric Tile is:

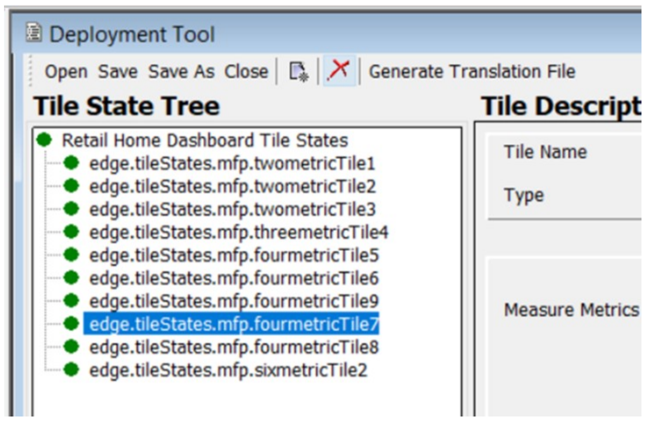
- name – A required property. It is the tile ID, string identifier. This must be unique across all the metric tiles.
- type – A required property, indicating metric tile type. It must be two-metric, four-metric, or six-metric. Two-metric means one to two measures, and no chart. Four-metric means one to four measures and an optional chart. Six-metric means one to six measures and an optional chart. In brief, the only difference between the 4-metric type and the 6-metric type is that 6-metric displays up to six measures and 4-metric shows only the first four.
- testLabel – A required property. Title of the tile. Used only in the test harness. In Retail Home it is the application name (same for all tiles in the application).
- roles – A required property. A list of role names separated with a comma whose users are authorized to see this tile in Retail Home. Not used in the test harness.
- measures – A required property. A list of JSON array objects. A list of one or more measure metrics to be displayed in the tile. Specify one to two measures for two-metric, one to four measures for four-metric, and one to six measures for six-metric.
- chart – An optional property. An object defining a chart to be displayed underneath the metrics. The chart is only displayed in four- and six-metric tiles.

Here is an example of the JSON format illustration of a metric tile.

```
{
"name": "tile1",
            "type": "four-metric",
            "testLabel": "Four metrics/area chart - Sales Rvl and Ly Sales R",
            "roles":"MFP_PLANNERS, MFP_USERS",
            "measures": [ ... ],
            "chart": { ... }
},
```

Users can add or delete a metric tile using the top menu item.

Figure 12-59   Add or Delete a Metric Tile



Retail Home Dashboard Tile States
- edge.tileStates.mfp.twometricTile1
- edge.tileStates.mfp.twometricTile2
- edge.tileStates.mfp.twometricTile3
- edge.tileStates.mfp.threemetricTile4
- edge.tileStates.mfp.fourmetricTile5
- edge.tileStates.mfp.fourmetricTile6
- edge.tileStates.mfp.fourmetricTile9
- edge.tileStates.mfp.fourmetricTile7
- edge.tileStates.mfp.fourmetricTile8
- edge.tileStates.mfp.sixmetricTile2

To add a new metric tile to the Retail Home dashboard settings configuration, highlight a Metric Tile in the left navigation pane, click **Add New Node** (located beneath **Add**) from the toolbar. A new tile entry is added to the navigation pane and selected in the content pane so that its information can be entered.

To remove an existing metric pane from the Retail Home dashboard settings configuration, select the tile and click **Remove** from the toolbar. The confirmation window opens. Click **Yes** to confirm.

Measure Metrics Definition

The Measure Metrics section can contain multiple Measure objects. The properties defined for a Measure object are:

- name - A required property. The RPASCE name of the measure. No extended measure.
- Label - An optional property for the measure label override. If not specified, the RPASCE measure label will be used. If MultiSolutionBundle.properties has a resource key entry called `<tile-id>.<measure-name>.retailhome.measure.label` then that is used. In other words, measure labels can be localized for retail home purposes. See “Generate Translation File" section for details about the translation of labels.
- valueLabel - An optional property. An additional label to display underneath the measure value.
- valueFormat - An optional property for measure value Format. The drop-down list contains blank, PC, S, and N. These represent: PC - Percent (.01 = 1%), S - Short (1000 = 1K), N - Number (1000 = 1,000).

- showIndicator - An optional property. True or False. By default, this property is false. This flag controls whether the up or the down icon is displayed in the dashboard.

Here is the JSON format for one Measure object.

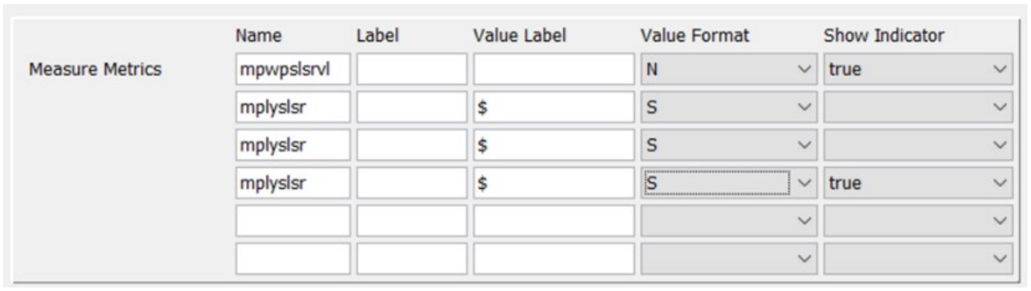
```
{
                "name": "mplyslsr",
                "label": "Measure Label Override",
                "valueLabel": "$",
                "valueFormat": "S",
                "showIndicator": "true"
            }
```

Here is an example of an array of tile measure objects. Note that some optional properties are not specified for certain tile measures.

```
"measures": [
        {
          "name": "mpwpslsrvl",
          "valueFormat": "N"
        },
        {
          "name": "mplyslsr",
          "valueLabel": "$",
          "valueFormat": "S"
        },
        {
          "name": "mplyslsr",
          "valueLabel": "$",
          "valueFormat": "S"
        }
      ]
```

Figure 12-60 shows the measure definition section in the content pane.

Figure 12-60 Measure Metrics Section Example



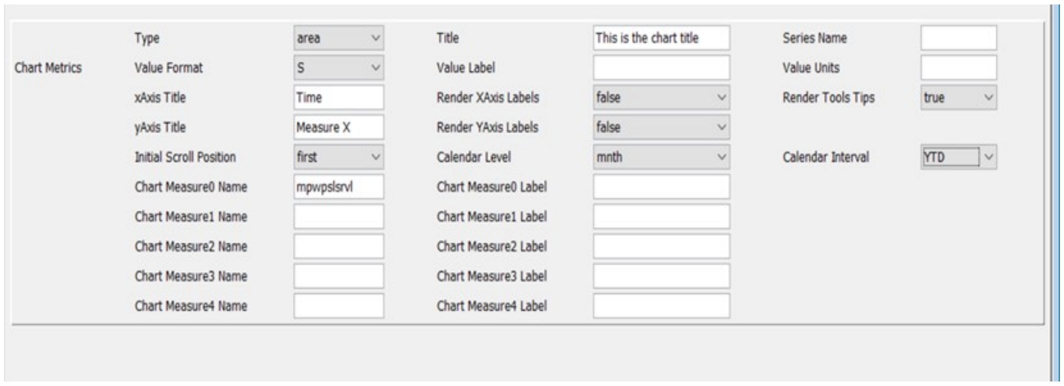
	Name	Label	Value Label	Value Format	Show Indicator
Measure Metrics	mpwpslsrvl			N	true
	mplyslsr		$	S	
	mplyslsr		$	S	
	mplyslsr		$	S	true

You can select the optional Value Format and Show Indicator from their lists.

Chart

The Chart metrics is an optional JSON object used with four-metric and six-metric types only. If no chart is configured for the Retail Home Dashboard Settings, all fields in the Chart section must be left blank.

Figure 12-61 Chart Metrics Section



	Type	area	Title	This is the chart title	Series Name	
Chart Metrics	Value Format	S	Value Label		Value Units	
	xAxis Title	Time	Render XAxis Labels	false	Render Tools Tips	true
	yAxis Title	Measure X	Render YAxis Labels	false		
	Initial Scroll Position	first	Calendar Level	mnth	Calendar Interval	YTD
	Chart Measure0 Name	mpwpslsrvl	Chart Measure0 Label			
	Chart Measure1 Name		Chart Measure1 Label			
	Chart Measure2 Name		Chart Measure2 Label			
	Chart Measure3 Name		Chart Measure3 Label			
	Chart Measure4 Name		Chart Measure4 Label			

The properties defined for a Chart object are:

- chartType - A required property. Values include line, bar, area, or total.
- measures - A required property. An array of chart measure objects. One or more measures to plot in the chart. Only one measure is supported for line/bar/area charts. These charts display the measure's values over the weeks or months before the current date, from the current year.
- Totals chart is unique. You must specify at least two measures. The totals chart shows the values of the measures rolled up across all time, side by side in the horizontal bar. The maximum number of chart measures that can be specified is five. See next section for a discussion of the measure object format.
- title - An optional property. The title displays above the chart.
- renderTooltips - Optional. Hide or show tooltips on the chart. Default is true.
- seriesName - Optional. True or False. By default, this property is false. This flag controls whether the up or the down icon is displayed in the Label for the tooltip. Applies to line, bar, and area charts.
- valueFormat - Optional. Formats the output of values in the chart. Takes the same values as metrics items.
- valueLabel - Optional. Prefix for values. Applies to total charts only.
- valueUnits - Optional. Suffix for values. Applies to total charts only.
- xAxisTitle - Optional. Label for the x-axis. Applies to line, bar, and area charts.
- renderXAxisLabels - Optional. Default is false. Show or hide tick labels on the x-axis. Applies to line, bar, and area charts.
- yAxisTitle - Optional. Label for the y-axis. Applies to line, bar, and area charts.
- renderYAxisLabels - Optional. Default is false. Show or hide tick labels on the y-axis. Applies to line, bar, and area charts.
- initialScrollPosition - Optional. The end of the chart to start at if there are too many items to show at once. Options are first and last. Default is first. Applies to line, bar, and area charts.
- calendarLevel - Required if the Chart metric is configured. Defaults to the level name corresponding to month. (In MFP it is MNTH, for example). Users can select week or mnth from the drop-down list.

- calendarInterval - Required if the Chart metric is configured. Currently, only YTD is supported.

Here is an example of the JSON format for the chart object.

```
"chart": {
        "chartType": "line",
        "measures": [
          {
            "name": "mpwpslsr",
            "label": "Override lbl"
          }
        ],
        "title": "Sample chart title",
        "renderToolTips": "true",
        "seriesName": "Working Plan Sales Receipts over time",
        "valueFormat": "S",
        "valueLabel": "USD",
        "valueUnits": "cents",
        "xAxisTitle": "Time",
        "renderXAxisLabels": "false",
        "yAxisTitle": "Measure X",
        "renderYAxisLabels": "false",
        "initialScrollPosition": "first",
        "calendarLevel": "mnth",
        "calendarInterval": "YTD"
      }
```

Measure (in chart)

The chart object can contain one or more chart measures. The properties defined for a chart measure object are:

- name - A required property. The RPASCE name of the measure. No extended measure.
- Label - An optional property for measure label override. If not specified, the RPASCE measure label is used. See Generate Translation File for details about the translation of labels.

Here is an example of the JSON format for one measure object.

```
{
                "name": "mplyslsr",
                "label": "Measure Label Override"
            }
```

Validation against the Application Configuration

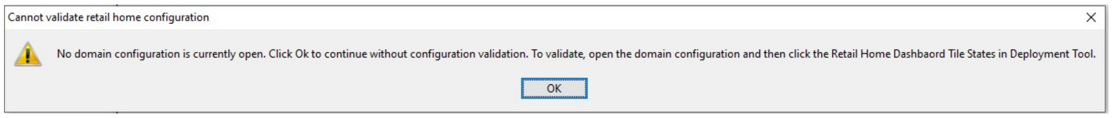
As explained in the Deployment Tool Limitation section, the deployment tool does not automatically validate the deployment configuration with the corresponding application configuration. However, the situation for Retail Home is different. The GA planning applications come with a configuration for Retail Home out of the box. In order to support the customization of the application content in the form of the EE products, solution implementors may need to modify the GA content or to create new custom content for use in Retail Home using the Retail Home Deployment Tool.

The process of updating Retail Home is expensive, requiring the upload of the deployment configuration, sending changes to the Retail Home server, and potentially patching the PDS. As such, the more errors that can be caught within the Tool, the less time and energy implementors will waste trying to diagnose and correct problems when deploying their configurations.

Open the Application Configuration First

To validate Retail Home Dashboard Settings with its application configuration, open the application configuration first in the Configuration Tools before clicking the Open or New button in the Retail Home View of the Deployment Tool. Otherwise, a window will display to warn the User.

Figure 12-62   Cannot Validate Retail Home Configuration Window


Cannot validate retail home configuration

No domain configuration is currently open. Click Ok to continue without configuration validation. To validate, open the domain configuration and then click the Retail Home Dashbaord Tile States in Deployment Tool.

Once the user selects OK, the File Selection Window displays. The user can either select Cancel to go back to the Configuration Tools to open an application configuration for validation or select Open or Create a Retail Home JSON File to proceed without validation.

Figure 12-63   Open File Selection Window

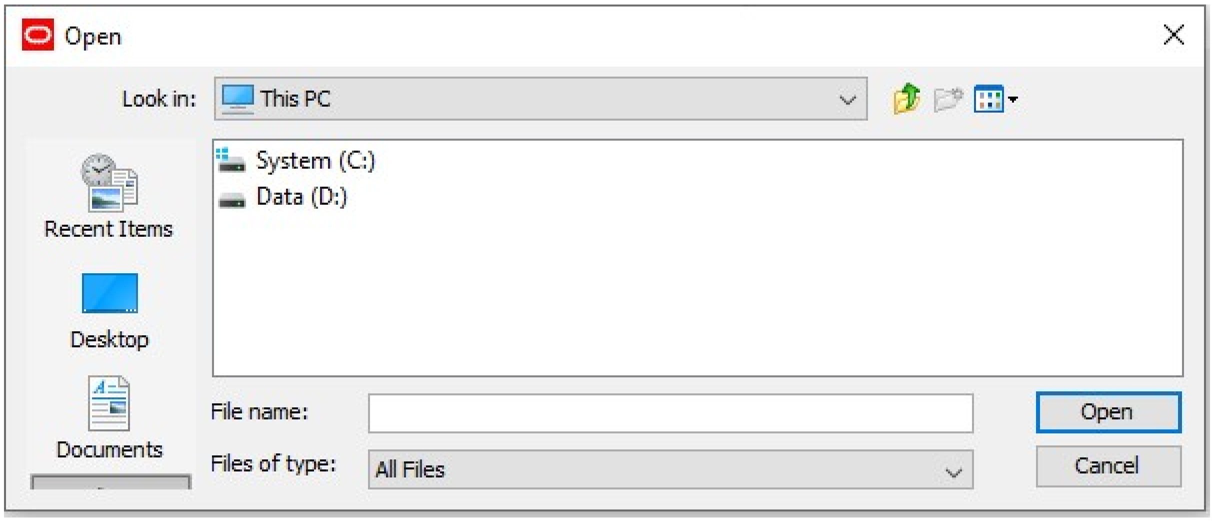
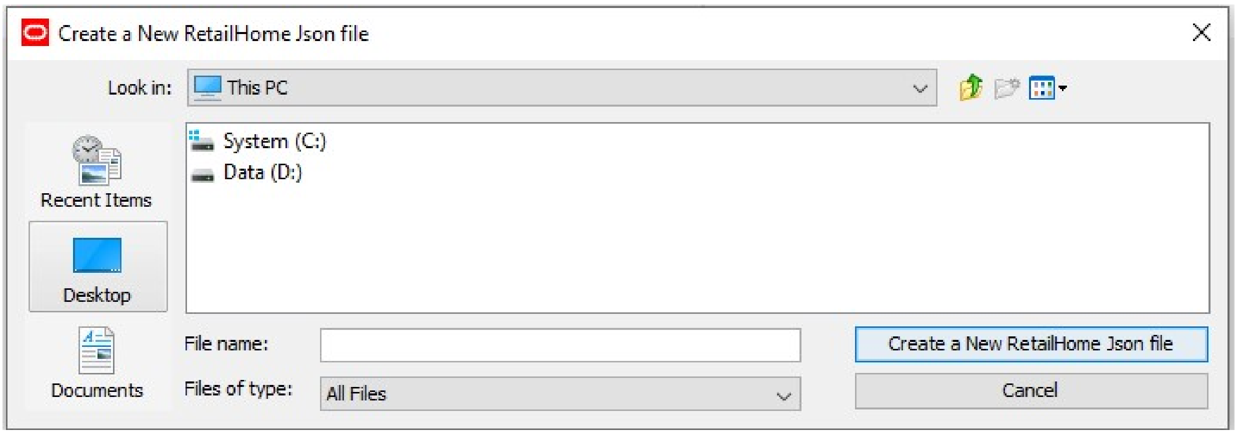


Figure 12-64 Create a New Retail Home JSON File Window



The Retail Home Dashboard configuration contains the dashboard workbook template name. Using that information, Retail Home Deployment Tool searches open application configurations to determine which application contains the dashboard workbook and then obtain the reference to that application object. If the Retail Home dashboard workbook is configured with the same workbook template name in more than one application configurations, then the deployment tool will use the reference to the first encountered application configuration in the search path.

Validation Criteria

For both Metric Measures and Chart Measures, if they do not meet following criteria, the behavior will be flagged as an error in the TaskList pane in the RH deployment tool. The measure:

- Does not exist within the dashboard workbook template.
- Is hidden within the dashboard workbook template.
- Is neither Integer nor Real type.

For both Metric Measures and Chart Measures, if they do not meet following criteria, the behavior will be flagged as a warning in the TaskList pane in the RH deployment tool. The measure:

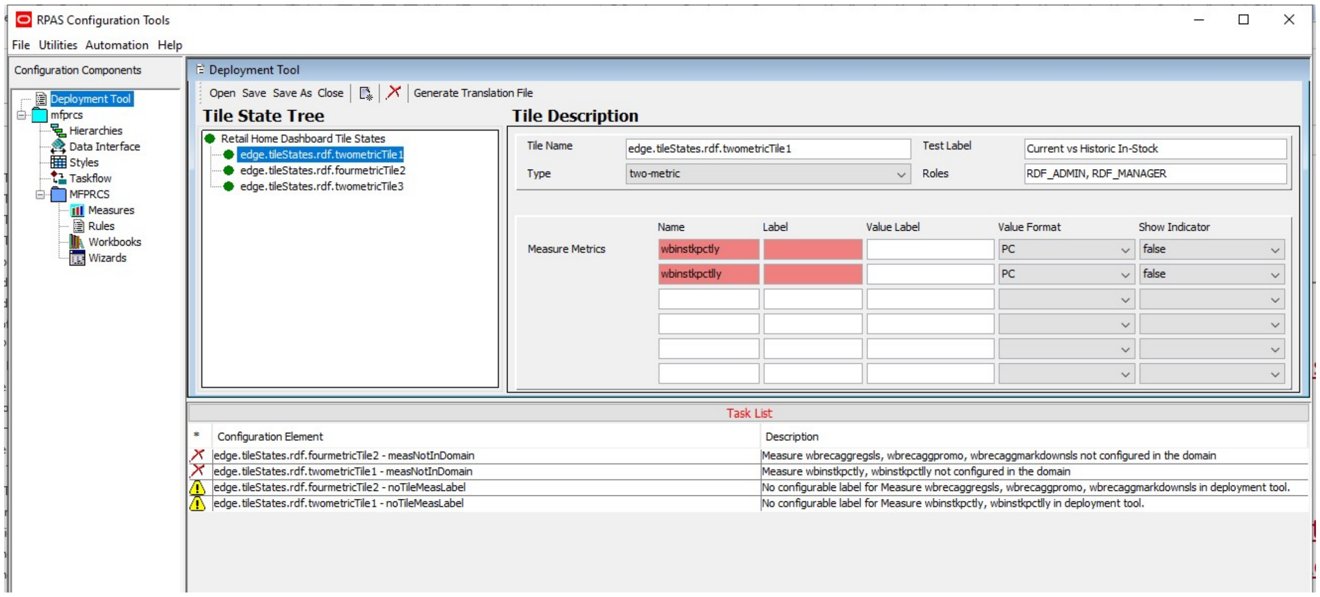
- Does not have a configured label in deployment tool.

For Chart Measures, if they do not meet following criteria, the behavior will be flagged as an error in the TaskList pane in the RH deployment tool. The measure:

- Does not contain a level of the calendar dimension within its base intersection.

Note that, as for all other tools TaskList error/warning messages in general, users can still save the Retail Home Dashboard Settings before the validation Tasklist error/ warnings are completely resolved.

Figure 12-65 Retail Home Dashboard Settings Validation Errors and Warnings



Tile Name	edge.tileStates.rdf.twometricTile1	Test Label	Current vs Historic In-Stock
Type	two-metric	Roles	RDF_ADMIN, RDF_MANAGER

	Name	Label	Value Label	Value Format	Show Indicator
Measure Metrics	wbinstkpctly			PC	false
	wbinstkpctlly			PC	false

Task List

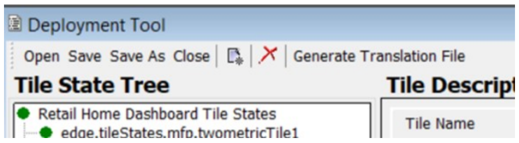
*	Configuration Element	Description
✗	edge.tileStates.rdf.fourmetricTile2 - measNotInDomain	Measure wbrecaggregsls, wbrecaggpromo, wbrecaggmarkdownsls not configured in the domain
✗	edge.tileStates.rdf.twometricTile1 - measNotInDomain	Measure wbinstkpctly, wbinstkpctlly not configured in the domain
⚠	edge.tileStates.rdf.fourmetricTile2 - noTileMeasLabel	No configurable label for Measure wbrecaggregsls, wbrecaggpromo, wbrecaggmarkdownsls in deployment tool.
⚠	edge.tileStates.rdf.twometricTile1 - noTileMeasLabel	No configurable label for Measure wbinstkpctly, wbinstkpctlly in deployment tool.

Retail Home Translation Resources

In order to support the localization of the strings provided to Retail Home, RPASCE must provide an extracted internationalization of the information passed by the Retail Home web service. This resource, when translated, allows the web service to retrieve strings localized into the appropriate language when serving requests and, in conjunction with any strings translated by the RPASCE Server (for strings representing metadata pulled from the PDS), allows users to view their Retail Home data in the language of their choice.

The Retail Home Dashboard Configuration tool provides a menu button Generate Translation File in the upper right menu to pull translation labels from the Retail Home Dashboard Configuration. This button must only be used when any change to the RetailHomeConfig.json has been made and saved. Click this button to generate a base resource file called RHResources.properties. This file is generated within the same directory where the RetailHomeConfig.json file is stored. The resource file RHResources.properties can be translated into various languages/locales and accessed at run-time by the Retail Home web service through the Java ResourceBundle APIs.

Figure 12-66 Generate Translation File Menu Button


Deployment Tool
Open Save Save As Close
Generate Translation File
Tile State Tree
Tile Descript
Retail Home Dashboard Tile States
edge.tileStates.mfp.twometricTile1
Tile Name

The following table lists those strings that are set in the Retail Home JSON configuration file that require internationalization, along with the algorithmic formula to generate the key for that string in the RHResources.properties.

Table 12-2 Strings that Are Set in the Retail Home JSON Configuration File

Property	Description	Translation Key
metricLabel	A label to be used in place of the default measure label.	tileStates.<tile_name>.<measure_name>.label
measureValueLabel	A label provided for metrics to give UoM information.	tileStates.<tile_name>.<measure_name>.valueLabel
chartTitle	A header for a chart in a metric tile.	tileStates.<tile_name>.chart.title
chartMeasureLabel	A label to be used in place of the default chart measure label.	tileStates.<tile_name>.chart.<measure_name>.label
chartSeriesName	A label shown as tooltip for chart values.	tileStates.<tile_name>.chart.seriesName
chartValueLabel	A label provided as prefix for values in chart.	tileStates.<tile_name>.chart.valueLabel
chartValueUnits	A label provided as suffix for values in chart.	tileStates.<tile_name>.chart.valueUnits
xAxisTitle	A label provided to describe the x-axis of the chart.	tileStates.<tile_name>.chart.xAxisTitle
yAxisTitle	A label provided to describe the y-axis of the chart.	tileStates.<tile_name>.chart.yAxisTitle

Here is an example of content within the RHResources.properties file.

```
tileStates.edge.tileStates.mfp.fourmetricTile5.mplyslsr.valueLabel=$
tileStates.edge.tileStates.mfp.fourmetricTile6.chart.mpwpslsr.label=Override lbl
tileStates.edge.tileStates.mfp.fourmetricTile6.chart.seriesName=Working Plan
Sales Receipts over time
tileStates.edge.tileStates.mfp.fourmetricTile6.chart.title=Sample chart title
tileStates.edge.tileStates.mfp.fourmetricTile6.chart.valueLabel=USD
tileStates.edge.tileStates.mfp.fourmetricTile6.chart.valueUnits=cents
tileStates.edge.tileStates.mfp.fourmetricTile6.chart.xAxisTitle=Time
tileStates.edge.tileStates.mfp.fourmetricTile6.chart.yAxisTitle=Measure X
```

Deployment Tool Limitations

The following section describes a limitation of the Deployment Tool.

Validation of Resource Contents

Note that the Deployment Tool provides only the most basic validation over a deployment resource and that the validations do not reflect issues in the resource itself, but merely detect errors that could interfere with the functioning of the Deployment Tool.

The Tool only ensures that the information contained within the resource file is formatted properly for the use of the resource (for example, the Tool ensures that the tag structure of an XML resource is correct but does not prevent incorrect or incomplete information within the tags of the resource).

The reasons for this are twofold.

First, the Deployment Tool is designed to provide lightweight interfaces for the creating of resource files. It translates the structure of the information contained in the resources into a set of user interface controls that remove the need to manually edit resources in a text editor.

The Tool does not connect with a RPASCE PDS nor is it equipped to automatically read the contents of an application configuration representing an application and so does not have access to the information necessary to provide such validation. Retail Home Dashboard Configuration has been enhanced to validate certain content against the application configuration if the application configuration is open in the Tools.

Second, because the Tool operates on a user's desktop and not necessarily on the host of the RPASCE application, many categories of information are simply not available for validation, such as the legitimacy of file path locations on the RPASCE Server.

For these reasons, the content of a resource generated through the Deployment Tool must always be manually inspected to ensure the correctness prior to use of the resource in an RPASCE PDS.

A

Appendix – Note on Deprecated Functionality

The RPASCE platform is substantially different from the traditional RPAS platform. The RPASCE platform provides a number of new capabilities that were not present in non-CE RPAS, but also deprecates some features present in non-CE RPAS.

In some cases, these features will be added to the CE platform as it continues to support more solutions. In some cases, these features will be replaced with more flexible, powerful or intuitive features that provide improved solutions to challenges the pre-CE feature was intended to address.

However, in order to assist in the process of migration of a pre-Version 21.0 solution to the Version 21.0 CE platform, the configuration components associated with these deprecated features remain present within the RPASCE Configuration Tools and therefore continue to be described within this document.

The next section provides a list with a summary description of features not currently supported within the RPASCE platform, organized by which section of the document describes their pre-CE configuration process. Use of these features in configuration can result in non-functional content, unexpected behavior, or a configuration that cannot be deployed in the CE environment.

Deprecated Features in the RPASCE Platform

This section describes the features that are not currently supported within the RPASCE platform. The features are organized according to the section of the Configuration Tools User Guide documentation where they are described.

Create a Project

The configuration settings for Global Application and Multilanguage are deprecated. The value for global application property is always false, and the value for Multilanguage property is always true.

Working with Styles

The RPASCE platform contains a new client interface implemented in JET, a Javascript based Web application development framework. Due to limitations in the configurability of JET applications and due to the need to provide support for accessibility, many of the style properties defined within the Style Manager cannot be applied within the RPASCE Client.

Working with Taskflows

Within the RPASCE platform, it is not possible to create a deployment of multiple solutions that share a single client instance. As such, the need for preparation of a multi-solution taskflow is negated.

The RPASCE platform does not support the use of hyperdynamic tasks and steps or the use of the Dynamic Task property for non-hyperdynamic tasks.

Working with Measures

While most measure behaviors configured through measure properties are fully supported within the RPASCE Platform, batch alerts are not and so have been deprecated.

Measures are loaded into the PDS instance using CSV-formatted input files with a header using the utility loadFactData. The measure properties, Start Position, Column Width, and Clear Intersection, no longer apply and have been deprecated.

Working with Workbooks

Many elements of the Workbook Designer are not currently supported within the RPASCE platform. These elements include:

- Workbook Transitions, defined within the Workbook Properties Panel, are not supported in RPASCE.
- The use of the Library, Use Custom Wizard, Custom Wizard in the General tab of the Workbook Properties Panel is not supported in the RPASCE platform.
- While Style Overrides, configured within the Worksheet Properties Panel, are still supported in RPASCE, many properties of a configured style cannot be honored within the RPASCE platform.
- The RPASCE platform does not support the use of worksheets with a view type of Detail Popup or Tiled View.
- The use of the Auto PQD or Lock PQD Dimensions properties within the General tab of the Worksheet Properties Panel is not supported in the RPASCE platform.

Working with Wizards

The RPASCE platform does not support the use of custom template libraries to extend the workbook build process. As such, the wizard configuration process performed within the Wizard Designer has been deprecated and workbooks that attempt to make use of custom wizards will fail to be registered at PDS build time.

Deployment Tool

The deployment aspects of an application instance are managed entirely by Oracle in the RPASCE platform. As such, the views, except the dashboardSettings.json view, in the Deployment Tool are not required by the configuration process.

B

Appendix – RPASCE Cloud Partitioning Guide

Note:

This chapter contains descriptions of features that are not supported in the current version of RPASCE. However, in order to assist in the process of migrating to the RPASCE platform, the configuration components associated with these deprecated features remain present within the RPASCE Configuration Tools and therefore continue to be described within this document.

The domain partitioning is deprecated. The domain is always created as a simple domain and only used in the PDS building/patching process. However, in the data mart schema of PDS, fact tables are partitioned. Proper partitioning is crucial for optimal performance. The concept and method of partitioning PDS is the same as that of partitioning global domain in older versions of RPASCE: the configurator first chooses a partitioning dimension level, and then assigns positions to partitions. The partition configuration file partitioninfo.xml is one of the required input files to the deployment process. Refer to Planning Data Schema Administration in *Oracle Retail Predictive Application Server Cloud Service Administration Guide* for PDS partitioning.

This chapter provides guidelines regarding Oracle Retail Predictive Application Server Cloud Edition (RPASCE) partitioning for the application. It is intended to help the RPASCE application configurator and system integrator properly partition the data storage for optimal performance.

Overview

Traditionally, there are two ways to configure the partitioning of a global domain. One is using only a command line parameter -p to specify the partition dimension for the RPAS Installer command (rpasInstall), in which case each position of the partition dimension will be used to create a local domain. Another way is providing a configuration file called globaldomainconfig.xml, which specifies the partition dimension, the absolute path to the master domain, and the absolute path and grouping of the positions of each local domain. The first approach is very straightforward and allows the users to control the partitioning through the hierarchy structure. Some users even create an artificial dimension solely for this purpose (for example, a dimension called ldom). The second approach provides flexibility and more control over the locations of the master domain and the local domains, and how the partition positions are grouped into the local domains.

In RPASCE, both options are supported. The first option is usually called internally by RPASCE since the customer may not have direct access to the rpasInstall command. The second option requires modification to the partition configuration in order to be cloud compliant.

Domain Partitioning

The purpose of using a global domain and partitioning data across multiple domains is to help reduce contention, provide smaller domains for most users to interact with, and allow for parallel processing during batch. If the partitioning is not done correctly, it can lead to unnecessary contention or poor performance. Here are some key considerations to understand when determining how to partition a global domain environment:

- The hierarchy that you partition on must allow the users the ability to work in a single local domain. If users require access to all positions within a hierarchy, that is not a good candidate for partitioning. For example, it does not make sense to partition on the location hierarchy if your business process requires all users to include all locations in each workbook.
- The partition level must also be above the level at which most of the data is stored. If most data is stored at the division level or below in the product hierarchy, the partition level must be at the division level or above. When data is based above the partition level of the domain, the data will be stored in the single master domain. All users across the local domains that require this data will have contention from all of the users and not only the users of the local domain that they are working in.
- The partitioning must be set such that the business requirements do not require high usage of the master domain. The performance of a workbook built from the master domain will never match that of a local domain workbook. The heavy usage of workbooks must take place across the local domains. For example, if most of the users only need to see data within a division, the partitioning must not be done below that level.
- The number of users that are in a single local domain must be evenly distributed across all the domains in a global domain environment. If there are a larger number of users in a single local domain than others, it will not matter how many partitions you create. The domain with the largest user group will always have the potential to experience more contention issues and poor performance. If possible, create more domains and separate more users across those domains.

Note that the domain partitioning is not visible from the front end or User Interface (UI). It is solely used for better organization of data storage on the server.

Performance Considerations

When deciding on the partitioning dimension and how many of its positions are in each local domain, one should consider two main items: batch performance and front-end usability. From a batch standpoint, local domains must be as balanced as possible. This will allow for the most efficient usage of system resources. Balancing local domains refers to the idea that each local domain must be performing a similar amount of work with a similar number of positions. Front-end usability refers to the preference of keeping the slice of the hierarchy that a user needs to access regularly, contained within a single local domain.

Domain Partition Configuration

The original globaldomainconfig.xml in RPAS is broken down into two files for RPASCE:

- globaldomainpartition.xml, which is modified from the original file to remove the path and add a partition name for each grouping
- globaldomainpaths.xml, which contains all the path information

Original content:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- Initial creation of domain -->
<rpas>
<globaldomain>
<path>C:/work/rpasce_btree/domains/RPAS_UT_G</path>
<partitiondim>dept</partitiondim>
<subdomain>
<subpath>C:/work/rpasce_btree/domains/RPAS_UT_G/ldom0</subpath>
<subpositions>dept01,dept02</subpositions>
</subdomain>
<subdomain>
<subpath>C:/work/rpasce_btree/domains/RPAS_UT_G/ldom1</subpath>
<subpositions>dept03,dept04</subpositions>
</subdomain>
<subdomain>
<subpath>C:/work/rpasce_btree/domains/RPAS_UT_G/ldom2</subpath>
<subpositions>dept05,dept06,dept07</subpositions>
</subdomain>
<subdomain>
<subpath>C:/work/rpasce_btree/domains/RPAS_UT_G/ldom3</subpath>
<subpositions>dept08</subpositions>
</subdomain>
</globaldomain>
</rpas>
```

New globaldomainpartition.xml modified from the original globaldomainconfig.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- Initial creation of domain -->
<rpas>
<globaldomain>
<partitiondim>dept</partitiondim>
<subdomain name="ldom0">
<subpositions>dept01,dept02</subpositions>
</subdomain>
<subdomain name="ldom1">
<subpositions>dept03,dept04</subpositions>
</subdomain>
<subdomain name="ldom2">
<subpositions>dept05,dept06,dept07</subpositions>
</subdomain>
<subdomain name="ldom3">
<subpositions>dept08</subpositions>
</subdomain>
</globaldomain>
</rpas>
```

These names for the sub-domain must be unique. They will be used as the directory name of the local domain.

Optionally, a second configuration file globaldomainpaths.xml can be used to specify the root paths of the master and local domains. This file must be managed by the OCI or Oracle Cloud Engineering/AMS. It contains system information (file or directory paths) that must be kept confidential.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- Initial creation of domain -->
<rpas>
<globaldomain>
<path>C:/work/rpasce_btree/domains/RPAS_UT_G</path>
<subdomain name="ldom0">
<path>C:/work/rpasce_btree/domains/RPAS_UT_G</path>
</subdomain>
<subdomain name="ldom1">
<path>C:/work/rpasce_btree/domains/RPAS_UT_G</path>
</subdomain >
<subdomain name="ldom2">
<path>C:/work/rpasce_btree/domains/RPAS_UT_G</path>
</subdomain>
<subdomain name="ldom3">
<path>C:/work/rpasce_btree/domains/RPAS_UT_G</path>
</subdomain>
</globaldomain>
</rpas>
```

Note that the path under <subdomain> is the parent directory of the local domain. It is appended with the sub—domain name to form the final path of the local domain. The globaldomainpartition.xml file is required if the globaldomainpaths.xml is provided. They must have matching local domain lists.

If the globaldomainpaths.xml file is not present, all local domains will be created under the global domain directory with the sub-domain name as its directory name.

Note:

The paths specified in the xml file must not exist before the domain is built. Otherwise, an Domain Already Exist exception will be thrown during the domain build.

Building a Domain Using the RPASCE Installer

If globaldomainpartition.xml is provided, it must be put under the directory specified by the rpasInstall command line parameter -ch <config_home>. If globaldomainpaths.xml is provided, it must be put under the directory specified by the environment variable $RPAS_CUST_ROOT.

The globaldomainpartition.xml is required if the globaldomainpaths.xml is provided. They must have matching local domain lists.

The partition dimension can be specified by the -p parameter or by the <partitiondim>xxxx</partitiondim>. If both -p and partition dimension name must match.

Both globaldomainpartition.xml and globaldomainpaths.xml are only required during the domain initial build. They are not required for domain patching since the domain partitioning cannot be changed after domain creation.

C Appendix – Calculation Engine User Guide

The RPASCE calculation engine is a powerful and flexible engine that is built to support OLAP type calculations against a multi-dimensional model. At first sight, the engine is very complex. However, when the building blocks of the calculation engine are properly understood, much of this apparent complexity goes away. This overview of the calculation engine processes will therefore start by describing the three fundamental processes of aggregation, spreading, and expression evaluation before explaining how the various processes integrate into a comprehensive whole.

RPASCE supports an OLAP-type model. In this model, individual pieces of data, called cells, apply to a single position in one or more hierarchies or dimensions. These will typically include a measures dimension, a calendar or time hierarchy, and other hierarchies such as for products and locations. The measures dimension is fundamentally different to the other hierarchies because measures (in other systems measures may be referred to as facts, performance indicators, or variables) represent the fundamental events or measurements that are being recorded, whereas the positions in the other hierarchies provide a context for the measurement (for instance, where, when, or what). Measures relate to one another through rules and expressions. Positions in all the other hierarchies relate to each other through hierarchical relationships.

RPASCE supports two different forms of relationships between cells:

- Hierarchical relationships that require aggregation and spreading
- Measure relationships that require rules and expressions

Hierarchical relationships, such as weeks rolling up to months or stores rolling up to regions, require the aggregation of data values from lower levels in a hierarchy to higher levels. This is performed using a variety of methods as appropriate to the measure. To enable such data to be manipulated at higher levels, RPASCE supports spreading the changes, which are performed using a variety of methods. Aggregation and spreading are basic capabilities of the engine that require no coding by implementation personnel, other than the selection of aggregation and spreading types to use for a measure.

The inherent relationships between measures can be modeled through rule and expression syntax. Most of the effort in configuring an application model is in modeling these relationships.

The RPASCE calculation engine is designed to be robust and extensible, while in complete control of the calculation process. It enforces data integrity by ensuring that all known relationships between cells are always enforced whenever possible. Much of the logic of the processing of rules and rule groups depends on this basic principle.

Measure Definition and Base Intersections

Certain characteristics of a measure determine how the calculation engine must handle it with regard to calculation, aggregation and spreading, and the dimensions in the hierarchies at which the measure is calculated. Since this information applies across all rules and rule groups, it is set up as part of the definition of a measure.

Data Types

RPASCE supports the following data types:

- Real
- Floating point numeric values. Most measures are of this type.
- Integer
- Numeric integer values. There are no special spreading algorithms for integer measures, which must normally be used only for measures that are calculated bottom up.
- Date
- Date and time. Can easily be converted to position names by standard functions.
- String
- Variable length strings. Typically used for notes and names.
- Boolean
- True or false values. Typically used for flags and indicators.

Note the following about data types:

- Integer measures have a range of 2,147,483,648 to 2,147,483,647 which is four bytes.
- [-2147483647:2147483647]
- Real measures have a range of 1.7E +/- 308 (15 digits) which is eight bytes.
- [-1.7976931348623e+308:1.7976931348623e+308]
- When running printMeasure, it gives the range of the measure. However, internally in the arrays, the integer and real data are stored as Numeric type which is eight bytes long.
- The calculations always happen as double. Internally, the calculation always happens on an eight-byte long number.
- The scientific representation of numbers is only for display and is not involved during calculation. There should not be any loss in data.

When used in the Client and with exportMeasure and loadMeasure, the following was observed:

- You can enter a number with more than 15 digits. But once you finish editing that cell, the number displays as 7.777778e+036.
- For such large numbers, the position of the decimal cannot be changed to point to any other position. It always displays as previously shown.
- For printArray, it also displays in the previously shown format.
- When you load a measure with large values, loadMeasure stores the data in the previously shown format.
- When you export the measure using exportMeasure, you can specify the format in which you want to export the data. By default, it exports in the previously shown format. For example, the following exports in the format you specify.

- exportMeasure -d . -intx str_sku_week -out MyOut1.dat -meas `R_EX_DEMOA.format("%13.2f")`
- 77777777777777781888888888888888888.00
- In all the previous cases, there is no truncation directly, but it will be rounded off to the correct precision

Base Intersection

The base intersection for a measure is a list of dimensions (such as Class/Store/Week), one per appropriate hierarchy, which defines the lowest level at which data is held for the measure. Data is assumed to apply to the All position in any hierarchy, which is not explicitly referenced in the base intersection (see “Non-Conforming Expressions" for more information). Through aggregation, data will logically exist (though there may not be a value) for all levels higher than the base intersection up all alternative rollups.

Aggregation and Spreading Types

The aggregation type defines the aggregation method to be used for the measure (refer to Aggregation for more information) to produce values at higher levels from values at the base intersection. There is a normal spreading method associated with an aggregation type, which defines the method to be used to spread changes from higher levels to the base intersection (see Spreading for more information). Depending upon the desired characteristics of the measure, there may be several valid allowed spreading types.

Aggregation

By definition, the OLAP-type model has hierarchical relationships between positions in hierarchies. The values of measures higher than their base intersections for these hierarchical relationships are automatically maintained through a process referred to as aggregation.

Different types of measures need to be aggregated in different ways. Many measures, such as sales, receipts and markdowns, record the events that occurred or are planned to occur during a time period. Simple totaling can produce aggregate values for these: the value for a region is the sum of the stores in the region; the value for a month is the sum of the weeks in the month; and so on. But this technique does not work for all types of measures. For example, with stock, the values record a snapshot at a point in time rather than a total of events over a time period. The value of stock for a region is the sum of the stock in the stores in the region, but the value of stock for a month is certainly not the sum of the stocks for the weeks in the month. It is usually either the value for the first week or the last week in the month. Similarly, there are measures where the appropriate aggregation type may be to calculate an average, or a minimum, and so on. For some calculation purposes, only cells that are populated (have a value other than their default value, which is typically zero) should participate in aggregations. RPASCE supports a wide variety of aggregation types to support all these requirements.

There is also another class of measures where no aggregation technique would produce the correct result. These measures are typically prices, ratios, variances, and similar performance indicators. The average price of sales for a class cannot be calculated by summing the prices of items in the class. Averaging the prices of items in the class produces a better result, but it is still not accurate because it fails to take account of the weighting of the sales of the items in the class. One item with a very large volume of sales at a low price would pull down the average price attained for the entire class, but this would not be reflected in an average aggregation. The way to get a correct result is to redo the price calculation at the required level. By dividing the sales value for the class by the sales units for the class (both of which will have been aggregated by summing), a correctly weighted result will be produced. The type of measure that requires this type of aggregation is referred to as a recalc measure, as aggregation is by recalculation of the expression used to calculate the measure. In planning applications, it is not unusual for 40% or more of the measures to be of recalc type.

A complete list of the aggregation types supported by the RPASCE calculation engine can be found in Appendix – Aggregation and Spread Types.

Spreading

By definition, an OLAP-type model has hierarchical relationships between positions in hierarchies. Measures are calculated in dimensions higher than the base intersection by aggregation by using the parent-child relationships between the positions. RPASCE allows such measures to be manipulated not only at the bottom levels, but also at aggregated levels. In order to preserve the integrity of the data with such a change, RPASCE needs to change the underlying data values at the base intersection for the measure, so that when they are aggregated again, they result in the changed value at the aggregated level. The method of changing the base intersection values to achieve this is known as spreading.

Spreading always applies to cells at the base intersection of the measure. At all aggregated levels higher than the base intersection, the effect of any change is applied by considering all cells at the base intersection that are descended from the changed cell (for instance, children and grandchildren). These calls are described as ‘child cells' in this description. Spreading does not operate from level to level to level down a hierarchical roll-up, which would not only be less efficient, but would also generate different (and generally less acceptable) results when there are changes or locks at levels between the change being spread and the base intersection.

The RPASCE engine allows changes to be made to a measure for positions at multiple levels, and the effect of all such changes are performed in a single calculation step. The basic technique for managing this spreading is the same for all spreading methods, and it is described in Multi-Level Spreading.

For calculation purposes, a lock to a cell for a spreadable measure is treated as a change to that cell that re-imposes the previous value. If none of the child cells of the locked cell have changed, the lock has no effect, and all child cell values remain unchanged.

Locks and Spreading Around Locked and Changed Cells

Other than in the special case where there are no cells that are free to be changed, spreading only affects cells that are free to be changed. All child cells are free to be changed except for those that are elapsed (see Chapter 8), locked by the user, explicitly changed by the user, or that have already been recalculated as the result of spreading another (lower level) change. Spreading always attempts to spread around locked or changed cells without changing their values. Where none of the child cells are free to be changed, spreading applies to all child cells that are not elapsed by using the changed or recalculated values as the base values to spread upon. For spreading purposes, when something must give, elapsed cells are considered to be more important than locked or changed cells.

Locked cells for recalc type measures are treated in an analogous manner: the mapping expression (see Spreading of Recalc Type Measures) is reimposed (using recalculated values of other measures on the right-hand side of the mapping expression if necessary) to recalculate the mapped measure. It is then spread normally.

Note:

The effect of spreading where there are no child cells free to be changed is that the result for some lower level locked or changed cells will be different to the locked value or the change made. Effectively, higher level locks or changes are deemed to be more important than lower level ones. Causing the circumstance where there are no free child cells can be a very useful technique when initializing data. For example, in a single calculation, a shape can be applied to child cells, and then a total to the parent cell. The result is that the parent total is spread across the children using the appropriate spreading technique, but according to the supplied shape. This is because the higher-level change takes precedence.

Spreading Methods

Just as different types of measures require different aggregation techniques, different types of measures require different spreading techniques. Measures that cannot be aggregated, such as recalc type measures, are not usually spread at all (see Spreading of Recalc Type Measures), but they may employ the replicate spreading technique bu using rapid entry. The default spreading method for a measure is set up as part of the definition of the measure. This is the spreading technique that is used for all changes to the measure unless explicitly overridden on edit by the user.

The spreading methods that are supported by RPASCE are listed here and described in the following sections:

- Proportional Spreading
- Replicate Spreading
- Even Spreading
- Delta Spreading
- PET and PST Spreading
- Multi-Level Spreading

Proportional Spreading

Proportional spreading is the most commonly used spreading technique once data has been initialized, and it is the default spreading method for most spreadable measures. In proportional spreading, all children that are free to be changed are changed in the same proportion so that their existing ratios to each other are maintained, and the required value for the parent is achieved. If proportional spreading is used for a measure that is not initialized (that is, its children all have the naval), the children are assumed to all have the same weight, so the effect of the spreading is the same as the even spreading method.

Example:

- **Starting values** – ChildA 10, ChildB 20, ChildC 30, ChildD 40, Parent 100.

- **Changes** – Parent changed to 145, ChildA changed to 20, ChildB locked.
- **Resulting values** – ChildA 20, ChildB 20, ChildC 45, ChildD 60, Parent 145.
- **Spreading Process** – ChildA and ChildB are not free to be changed by spreading because ChildA was explicitly changed and ChildB was locked. The required parent value of 145 must include 40 from ChildA and ChildB, and thus ChildC and ChildD must total 105. The previous total for ChildC and ChildD was 70, so their values must be changed by applying the multiplier of 105/70. The new value for ChildC is 45 and for ChildD is 60.

After aggregation, the result is as follows:

- The parent has the value 145, as required
- ChildA has the required 20
- ChildB did not change
- The ratio of ChildC being 75% of ChildD is maintained

This spreading method is not allowed for measures with a recalc aggregation type.

Replicate Spreading

Replicate spreading is sometimes used when initializing data, especially for recalc type measures, and for measures with aggregation type such as average, minimum, and maximum. It is unusual for it to be the default spreading method for any measure but may be used by overriding the spread method on data entry. In replicate spreading, all child cells that are free to be changed are changed to the value of the parent cell. With replicate spreading, there is no guarantee that after aggregation the value of the parent cell will be the value that was replicated. In fact, it usually will not be. Replicate spreading should be considered an indirect way of entering the same value into multiple child cells.

Example:

- **Starting values** – ChildA 10, ChildB 20, ChildC 30, ChildD 40, Parent 100.
- **Changes** – Parent changed to 145, ChildA changed to 20, ChildB locked.
- **Resulting values** – ChildA 20, ChildB 20, ChildC 145, ChildD 145, Parent 330.
- **Spreading Process** – ChildA and ChildB are not free to be changed by spreading because ChildA was explicitly changed and ChildB was locked. The parent value of 145 is replicated to ChildC and ChildD. After aggregation, the result is that the parent has the value 330.

Even Spreading

Even spreading is sometimes used when initializing data. It is unusual for it to be the default spreading method for any measure, but it may be used by overriding the spread method on data entry. In even spreading, all child cells that are free to be changed are changed to the same value, which is the total for the parent cell for the free child cells divided by the number of free child cells.

Example:

- Starting values – ChildA 10, ChildB 20, ChildC 30, ChildD 40, Parent 100.
- Changes – Parent changed to 145, ChildA changed to 20, ChildB locked.

- Resulting values – ChildA 20, ChildB 20, ChildC 52.5, ChildD 52.5, Parent 145.
- Spreading Process – ChildA and ChildB are not free to be changed by spreading because ChildA was explicitly changed and ChildB was locked. The required parent value of 145 must include 40 from ChildA and ChildB, and thus ChildC and ChildD must total 105. This is spread evenly, thus the new values for ChildC and ChildD are both 52.5.

After aggregation, the result is:

- The parent has the value 145, as required
- ChildA has the required 20
- ChildB did not change
- The remainder has been spread to ChildC and ChildD evenly

This spreading method is not allowed for measures with a recalc aggregation type.

Delta Spreading

Delta spreading is sometimes used when data is fully initialized. If it is used when the measure is not initialized, the effect will be the same as even spreading. It is unusual for it to be the default spreading method for any measure, but it may be used by overriding the spread method on data entry. In delta spreading, all child cells that are free to be changed are changed such that the delta to the parent cell is spread evenly across those child cells.

Example:

- Starting values – ChildA 10, ChildB 20, ChildC 30, ChildD 40, Parent 100.
- Changes – Parent changed to 145, ChildA changed to 20, ChildB locked.
- Resulting values – ChildA 20, ChildB 20, ChildC 47.5, ChildD 57.5, Parent 145.
- Spreading Process – ChildA and ChildB are not free to be changed by spreading because ChildA was explicitly changed and ChildB was locked. The required parent value of 145 must include 40 from ChildA and ChildB, and thus ChildC and ChildD must total 105. The previous total for ChildC and ChildD was 70, so the delta to the parent is 35. This delta is spread evenly across the children, so ChildC and ChildD are both increased by 17.5. The new value for ChildC is 47.5 and for ChildD is 57.5.

After aggregation, the result is as follows:

- The parent has the value 145, as required
- ChildA has the required 20
- ChildB did not change
- The increase to the parent has been evenly divided between ChildC and ChildD

This spreading method is not allowed for measures with a recalc aggregation type.

PET and PST Spreading

PET (period end total) and PST (period start total) are special spreading types to support measures with the PET or PST aggregation types where the values of cells represent snapshots at a time period, rather than a total of events. Opening and closing stock (inventory) are typical examples of such measures, where the value for a month will be the value for the first (opening stock) or last (closing stock) week in the month, but values up non-time hierarchies will be produced by total aggregation.

PET and PST measures require special spreading. We anticipate a future enhancement to support spreading changes to such measures at aggregated time positions by spreading the effect of the change across all children of the time period. At present, the PET and PST spread types change the first or last child only. At present, a change to closing stock for a month has the same effect as a change to closing stock for the last week in the month.

Multi-Level Spreading

The RPASCE engine allows changes to be made to a measure at multiple levels, all of which are dealt with in a single calculation. Because spreading requires parent-child relationships, and spreading is affected between the intersection that is changed and the base intersection for the measure, there is a requirement that all changes to be effected by a single calculation must fall on a single hierarchical roll-up. This is controlled by Hierarchical Protection Processing, which is described in the next section.

When there are changes at multiple levels, the spreading process fundamentally works bottoms-up. That means lower level changes are implemented before higher-level changes. The spreading algorithm starts with the lowest level in the hierarchical roll-up that has changes, and it spreads each change at that level in turn.

The result of this process is that every child cell of a changed cell is no longer free to be changed. If it was previously free to be changed, it has now been recalculated by spreading. When all changes at a level have been performed, the algorithm moves on to the next lowest level in the hierarchical roll-up that has changes, and it continues in this manner until all changes have been performed. If a higher-level change overlaps a lower-level change, the lower-level changes are unaffected because all child cells of the lower-level change will not be free to be changed.

Example (using proportional spreading):

- **Starting values** – jan 10, feb 15, mar 20, apr 25, may 30, jun 35, jul 40, aug 45, sep 50, oct 55, nov 60, dec 65. firsthalf 135, secondhalf 315, year 450.
- **Changes** – year changed to 500, firsthalf changed to 150, jan changed to 15, feb changed to 20, mar locked, jul and aug changed to 50, sep locked.
- **Resulting values** – jan 15, feb 20, mar 20, apr 26.39, may 31.67, jun 36.94, jul 50, aug 50, sep 50, oct 61.11, nov 66.67, dec 72.22. firsthalf 150, secondhalf 350, year 500.
- **Spreading process** – The first change to be spread is the change to the first half to be 150. jan, feb and mar now total 55, so apr, may and jun must total 95. By proportional spreading the results are 26.39, 31.67 and 36.94. The second change to be spread is the 500 for the year. Only the months oct through dec are now free to be changed. The other months total 300, so oct through dec must total 200. By proportional spreading, the results are 61.11, 66.67, and 72.22.

Hierarchical Protection Processing

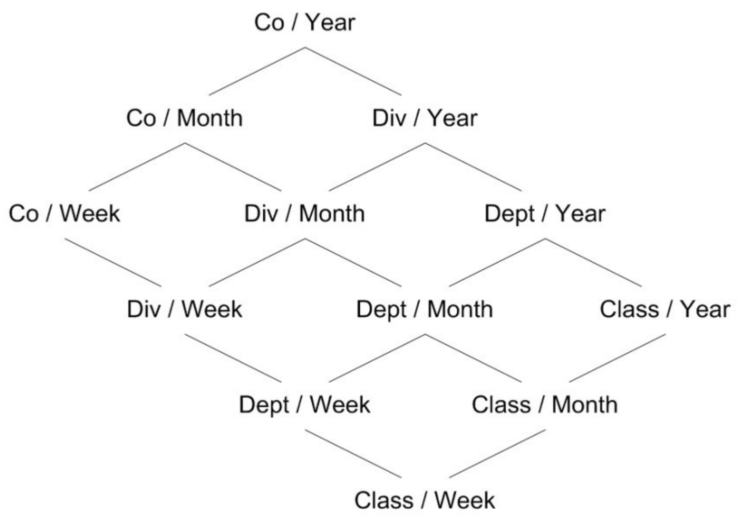
Hierarchical protection processing is a process that ensures that all changes made at aggregated levels fall on a single hierarchical roll-up, which is a prerequisite for the spreading process to function correctly. Hierarchical protection processing operates by protecting (preventing direct manipulation) cells for intersections for combinations of dimensions that cannot reside on a single hierarchical roll-up with the changes already made.

In theory, since hierarchical protection processing is necessary to ensure the integrity of the spreading process and each measure is individually spread, hierarchical protection processing could operate independently for each measure. Having the manipulated measures varying from intersection to intersection would probably cause considerable confusion to the users and would make implementing a consistent methodology difficult. For simplicity, hierarchical protection processing operates on all measures.

An OLAP-type model has multiple hierarchies and spreading operates from the cell that has been changed to all child cells at the base intersection, so hierarchical protection processing must operate across multiple hierarchies. A single hierarchy may have multiple roll-ups, which are also considered. Whenever a change or a lock is made to an intersection for a new combination of dimensions, the calculation engine checks all other combinations of dimensions, and it protects those that cannot be on the same hierarchical roll-up as changes already made. It does this by considering a cross multiplication of hierarchical roll-ups across all the hierarchies.

A simple example will clarify the process. Consider the matrix of cross-multiplied dimension combinations that result when there is a 2-dimensional product by time model with the dimensions Co/Div/Dept/Class and Year/Month/Week. Figure C-1 shows the schematically with parent-child relationships.

Figure C-1 Hierarchical Protection Processing Parent-Child Relationships



Other than at the top of a hierarchical roll-up, each combination of dimensions has two parent combinations: one per hierarchy with the next highest dimension, so class/week has parents of class/month and department/week.

Note:

For the sake of simplicity, the picture does display the roll-up of the all dimension (all products, all time periods).

All spreading is from the changed level to the base intersection (class/week in this example). Consider a change at an intersection of Div/Month. We know that the spreading hierarchical roll-up must be a path from the top (Co/Year) to the bottom (Class/Week) that passes through Div/Month. There are six such paths, none of which go through the combinations Co/Week, Dept/Year or Class/Year, so those combinations of dimensions are all protected. If the next change is at an intersection of Class/Month, then Div/Week and Dept/Week are similarly protected.

Note:

Hierarchical protection processing always reflects the current set of locks and changes.

RPASCE allows cells that have been changed or locked to be unchanged or unlocked before the calculation is initiated. If unchange or unlock removes the last change or lock for a combination of dimensions, some other combinations of dimensions that were previously protected could become unprotected. In our previous example, if the first change to a Div/Month were now unchanged so that the only change outstanding is at Class/Month, then Dept/Year and Class/Year is manipulated again, but Co/Week, Div/Week and Dept/Week would still be protected.

Note:

Non-conforming measures (see Handling of Non-Conforming Expressions) may lead to hierarchical protection processing that may seem to be over protective.

When considering hierarchical protection processing, all measures have their scope expanded to include the all level of all hierarchies that they are not dimensioned on. For example, the implications of this are that a change to a measure with a base intersection of class, which is interpreted as meaning class/all, would prevent the manipulation of a measure with a base intersection of year, which is interpreted as all/year.

Spreading of Recalc Type Measures

Measures that are of recalc type are not usually spread by any spreading technique. Spreading techniques typically rely on the existing relationships between a parent and its children in the spreading process. In a recalc measure, those relationships cannot be relied upon because they are not weighted. Spreading of changes to a recalc measure is therefore indirect and applying a mapping rule effects the change.

A mapping rule is a rule (with two or more expressions) that calculate a spreadable measure from the changed value of a recalc measure and other measures. The selected expression for the rule is evaluated at the level of the change to the recalc measure. It results in a changed value of a spreadable measure, and if this is higher than the base intersection for that measure, it is spread normally using its default spreading method. Therefore, a change to a recalc measure must be considered an indirect change to the spreadable measure that it is mapped to.

The only constraint on the manipulability of a normally spreadable measure is through protection processing, which prevents the manipulation of measures that will be calculated. For a recalc measure, the measure must have a mapping rule. Without a mapping rule, the measure cannot be manipulated.

Note:

A recalc measures can only display in a single rule in a rule group. The RPASCE calculation engine therefore knows that rule contains the recalc expression for the recalc measure. If there are other expressions in the rule, they may be used as mapping expressions, which allows the recalc measure to be manipulated. If there is just a single expression in the rule that calculates the recalc measures, the recalc measure is non-manipulated through normal protection processing.

Non-Conforming Recalc Measures

Having a recalc measure on the right-hand side of an expression that calculates a measure whose base intersection is higher than that of the recalc measure or using the level modifier on a recalc measure on the right-hand side of an expression can cause incorrect values to be calculated. These incorrect values can then have a knock-on effect onto other measures. Therefore, in these circumstances, expressions must be written such that the right-hand side of the expression should have a recalc expression rather than a recalc measure. See Handling of Non-Conforming Expressions for more information.

Expressions, Rules, and Rule Groups

Measures are related together through algorithmic relationships. For example, the sales value may be the sales units multiplied by the selling price. In RPASCE, these relationships are specified through expressions, which are grouped for usage into rules and rule groups.

It is a fundamental principle in RPASCE that the calculation engine always maintains and guarantees the integrity of all active relationships between cells. Hierarchical relationships are maintained through the processes of spreading (see “Spreading") and aggregation (see “Aggregation"). Relationships between measures are maintained by the evaluation of expressions. One of the great strengths of RPASCE is that both types of relationships are automatically maintained in a non-procedural manner. You do not have to write code to determine what is calculated, how it is calculated, or in what sequence it is calculated. All that is required is the definition of the relationships themselves, although you do provide prioritization information to guide the calculation engine when there is a choice of calculation paths.

In an RPASCE model, all cell values at aggregated level can be determined by aggregation from the cells at the base intersection. Although the description that follows is a simplification, a basic understanding of the working of the calculation process, and the importance of expressions, can be gained by understanding the interconnection between three fundamental processes: spreading, bottom level expression evaluation, and aggregation. A more detailed and precise description refers to The Calculation Cycle. Changes to measures at aggregated levels are spread down to their base intersections. Here, the calculation engine enforces all measure relationships that are no longer guaranteed to be true by evaluating an expression. This is because the cell values of one or more of the measures in the relationship have changed directly, through spreading, or by prior evaluation of an expression When base intersection calculation is complete, all measures at the base intersection that have been changed are aggregated to re-impose cell integrity.

Expressions

Expressions are the basis of all calculations of the relationships between measures. They are evaluated by the calculation engine during a calculation. Expressions are written in a syntax that allows for the calculation of one or more measures from other measures, constants and parameters, using standard arithmetical functions and a rich set of mathematical, technical, and business functions. Expressions are therefore an algorithmic statement of a relationship between measures. Details of the allowable syntax for expressions are provided in a separate document.

Rules

An expression describes the relationship between measures in a way that causes a measure to be calculated through the expression. An expression may be said to ‘solve' the relationship for the measure that is calculated through the expression. In some cases, there may be business methodology reasons for wanting more than one of the measures in a relationship to be calculable or solvable through that relationship.

To support this requirement, RPASCE has the concept of a rule, which consists of one or more expressions that describe the same relationship between measures, but that solve for different measures. All of the expressions in a rule must use the same measures and must have a different target measure. The target measure is the measure on the left-hand side (LHS) of the expression that is calculated by the expression.

Where a rule has multiple expressions, those expressions are given a priority sequence to help the calculation engine select a calculation path that follows business priorities. Consider the rule that relates together sales value, sales units, and sales price. Assume that there are three expressions in this rule. Each of the measures involved in the rule may be ‘solved' through the rule. For instance, if there is a change to a sales value, it should be clear that the calculation engine could enforce the mutual integrity of all the cells by holding the sales price constant and recalculating a new sales unit. It could also achieve the same end by keeping the sales units constant and recalculating the sales price. Both approaches are mathematically valid and produce a consistent result with complete data integrity. However, it is likely that one approach makes more ‘business sense' than the other. In this case, most businesses in most circumstances would want the price to remain constant and have the units recalculated. The prioritization of the expressions in the rule provides this information to the calculation engine. Considerable care must be taken in the design of models to ensure that appropriate expression priorities are established.

When given a choice, the calculation engine will always select the highest priority expression in the rule that is available to be selected. In this example, the expression that calculates sales units would have a higher priority than the expression that calculates sales price. Similar consideration of the desired effect of a change to sales units will probably lead to a conclusion that the expression that calculates sales value would also have a higher priority than the expression that calculates sales price.

What of the relative priority of the expressions to calculate sales value and sales units, and the "business priority" for those expressions? That may vary from implementation to implementation. It may even vary from one type of plan to another in the same implementation. For a financial merchandise plan, the preferred behavior may be that a change to the sales price only causes a recalculation of the sales units, whereas in a unit-oriented lower level plan, the preferred behavior may be that a change to the sales price causes a recalculation of sales value.

The same measure may appear in multiple rules. This is often necessary because the same measure can be involved in many different relationships with other measures. For example, there may be a relationship between sales value, sales units, and sales price. Sales value may also be involved in another relationship with closing stock and a cover value, and yet another with opening stock, receipts, markdowns and closing stock.

Rule Groups

It is most unusual for a model to only require a single rule. In most cases, there will be a collection of relationships between measures that must be maintained. In RPASCE, a Rule Group is a collection of rules that are treated as a unit by the calculation engine with the integrity of all the rules in the rule group being maintained together. The calculation engine always has one (and only one) active rule group. Even if all that is required is a single expression, that single expression will be in a rule, and that single rule will be in a rule group. The process by which the integrity of all the rules in a rule group is maintained is quite complex. It is described in detail in The Calculation Cycle topic.

Rules within a rule group are given a priority. The calculation engine uses this to select a calculation path that follows business priorities by using rule priorities to determine which rule to enforce when there is a choice to be made. This is described in more detail in The Calculation Cycle topic.

There may be many rules defined within a complete system. The validation of rules is performed in isolation, but rules within a rule group are also validated in the context of all the other rules in the rule group. This can mean that a rule that is perfectly valid syntactically, but it is not valid within a particular rule group. Rule group validations include:

- Each rule in a rule group must represent a completely different measure relationship. Therefore, no two rules in a rule group may use exactly the same collection of measures, and neither may one rule group use a collection of measures that is a sub-set of the collection of measures in another rule.
- There must be an expression that calculates each recalc measure.
- Any measure that is on the LHS of the only expression in a rule may not be on the LHS of any other expression.

Although there may only be one active rule group at any time, RPASCE allows for the definition of multiple rule groups to satisfy different calculation requirements. Rule groups may be one of four different types:

- **load** – The RPASCE application automatically uses the load rule group when loading data into the workbook.
- **calculate** – RPASCE supports multiple calculation rule groups. Menu options may be configured to allow the user to select a different calculation rule group. RPASCE ensures a smooth transition from one calc rule group to another.
- **refresh** – The RPASCE application automatically uses the refresh rule group to refresh data.
- **commit** – The RPASCE application automatically uses the commit rule group when committing data to the PDS.

These rule groups are perfectly normal so although they will typically include many rules that use the master modifier to load or commit data, they may also have other rules. For example, it is perfectly possible to commit data to the domain for a measure that does not exist in the workbook merely by including the appropriate rule to calculate the measure (with the master modifier) in the commit rule group. Similarly, a measure may be loaded into a workbook that does not exist in the PDS by including an appropriate rule to calculate the measure in the load rule group.

Rule Group Transitions

Although only a single rule group may be active at any time, RPASCE supports the transition from one rule group to another. The calculation engine always ensures the integrity of measure relationships so this process is not merely a case of switching from one rule group to another. There is no guarantee that the integrity of the rules in the rule group that is being transitioned has been maintained.

RPASCE makes a worst-case assumption when transitioning rule groups. Any rule that is in both the old and new rule groups is assumed to have its integrity maintained. Any other rule is assumed to be potentially wrong, and so is flagged as affected. A normal calculation is then initiated. Expressions to be evaluated are determined by the usual process (see The Calculation Cycle). All affected rules will therefore have their integrity imposed by the evaluation of an expression, and knock-on effects may cause some rules that occur in both the old and new rule groups to also be evaluated. Since all base intersections must be calculated during rule group transition, a large or complex rule group transition is likely to take longer than a normal calculate.

There are circumstances when automatic rule group transitions occur:

- **On Data Loading**

  Data is loaded using the load rule group. This will typically load measures by calculating them from the data values held on the PDS using the master modifier, but it may also calculate other measures that are not explicitly loaded. When the load is complete, the system will automatically transition to the calculate rule group.

- **On Data Refreshing**

  Data refreshing causes some measures to be updated from values held on the domain. Refreshing uses the refresh rule group, but there is no real transition.

  The measures that are affected by the refreshed measures are treated as affected in the calculate rule group, and a normal calculate of that rule group follows. Effectively, data refreshing causes a calculation by using the calculate rule group as if the cells that were refreshed were directly changed by the user.

- **On Data Committing**

  There is a normal transition from the current calculate rule group to the commit rule group. This will typically commit measures by calculating them on the PDS by using the master modifier. When transitioning back from the commit rule group to the calculate rule group, there is an assumption that only measures with a master modifier have changed and therefore no transition is required.

The Calculation Cycle

The calculation cycle always uses the current active rule group. It is a comprehensive process that uses non-procedural hierarchical cell relationships and expression-driven measure relationships from the rule group. These relationships are used together with details of the locks and changes to individual cells to determine and then execute the required actions to apply the effect of the changes and locks. This section describes how the calculation engine determines what to calculate, how to calculate it, and in what order to perform the calculations. Refer to the *Oracle Retail Predictive Application Server User Guide* and the *Oracle Retail Predictive Application Server Administrator Guide* for details of processes, such as spreading, aggregation, and the evaluation of expressions.

There are four distinct stages of the calculation cycle.

1. In the first stage, protection processing occurs while the user is making changes to cell values, and it protects those measures that the user cannot change either because they are never changeable or because changes already made force them to be calculated.
2. In the second stage, the engine decides what expressions will be evaluated.
3. In the third stage, the sequence of calculation is determined.
4. The final stage is the physical process of doing the calculation.

Note:

The calculation cycle can operate in one of two modes: full and incremental. In full mode, it is assumed that all of the cells for the measures being evaluated need to be calculated. This mode is used when calculating in batch, and in all rule group transitions. Incremental mode is used when manipulating cells in an online session, and only those cells that are directly or indirectly affected by user edits are calculated.

Protection Processing

Other than in exceptional circumstances, the calculation engine guarantees the integrity of all relationships and ensures that the value for a cell changed by a user after calculation is the value entered by the user. In order to ensure this, the calculation engine must prevent the user from making changes to any cells where it would be unable to guarantee that integrity. The process that achieves this is called protection processing.

A measure may only be manipulated when the calculation engine is able to change other cells by spreading and/or evaluation of an expression to enforce the integrity of relationships. A measure that is not used in any rules may only be manipulated if it has a spreading technique other than recalc.

It is a basic principle of the calculation engine that a measure that is changed (or locked) cannot also be recalculated by evaluating an expression. It will be aggregated, which in the case of a recalc measure, does involve the evaluation of an expression. A measure that is to be evaluated can only be evaluated using one expression because there is no guarantee that the same result would be produced from two expressions that represent different measure relationships. It is also a basic principle that any measure relationship (rule) must be evaluated when one or more of the measures in that relationship have been changed because this is the only way to enforce the integrity of the rule relationship. Therefore, a rule where there is just a single expression means that the measure calculated by that expression cannot be changed by the user because there is no expression to evaluate to effect that change for that measure relationship. Such measures can never be manipulated in any rule group that uses the rule and are protected.

Where a rule has two expressions, the two measures that are calculated by those expressions are available to be manipulated. However, as soon as one measure is manipulated by the user, we know that the expression that calculates the other measure must be evaluated, as one of the expressions in the rule has to be evaluated, and we cannot evaluate the expression that calculates the measure that was changed. The expression that must be calculated is said to be forced, and the measure that it calculates is protected to prevent the user from changing it. That measure may be involved in more than one rule, and in the other rules in which it is used it must be treated as if the user changed it. This so-called knock-on effect may force further measures to be forced and protected. Evaluating these effects is the basic technique of protection processing.

Protection processing occurs continuously while the user is editing cells. Each time the changed state of a measure changes, protection processing evaluates the measures that should now be protected. The changed state of a measure means the measure goes from not having changes or locks to having them. Protection processing always reflects the current set of locks and changes. RPASCE allows cells that have been changed or locked to be unchanged or unlocked before the calculation is initiated. If unchange or unlock removes the last change or lock for the measure so that the measure is no longer affected, protection processing is quite likely to find the other measures that were previously forced but are no longer forced. These measures are free to be manipulated, so they must be unprotected.

Protection Processing Details

The following terms are used in this description:

- Affected measure is a measure that has been changed by the user, is locked by the user, or is forced.
- Affected rule is a rule that contains one or more affected measures.
- Free measure is a measure that is not affected.
- Free expression is an expression for an affected rule that calculates a free measure.
- Forced rule is an affected rule that has only one free expression.
- Forced measure is the measure calculated by the free expression in a forced rule.

Any measure that is the measure on the LHS of the only expression in a rule is protected.

Protection processing considers each affected rule in turn. Each affected rule will be in one of three conditions:

- Affected rules that have previously been forced are ignored
- If the affected rule has two or more free expressions, it is ignored because nothing is forced.
- If the affected rule has just a single free expression, it becomes a forced rule, and the measure calculated by the free expression is forced and becomes an affected measure. The forced measure is protected. All rules that use the forced measure become affected.

When a new measure becomes forced, checking of affected rules begins again. When all affected rules have been considered without any further measures becoming forced, the first stage of protection processing is complete.

The second stage of protection processing is to perform look ahead protection processing. Look ahead protection processing ensures that all measures that are visible in windows (and still unprotected) can be manipulated. It does this by performing the protection processing that would occur if the measure were changed. This includes ensuring that there is a solution to the processes of determining what to calculate and ordering the calculation. If these processes fail to find a solution, the process that determines what to calculate will repeatedly back up the decision tree and select a different expression that is looking for a solution. If there is no such solution, the measure that was being checked is protected. In this manner, the calculation engine ensures that there will always be a method to calculate the effects of all changes that it allows the user to make.

Note:

This is a somewhat simplified description of protection processing, as it ignores the implications of cycle groups (see Cycle Groups) and synchronized measures (see Synchronized Measures).

Protection Processing Example

The following example illustrates the evaluation of protection processing. For purposes of this example, consider the following set of rules:

Rule 1:

1. A = B + C
2. B = A - C

Rule 2:

1. D = E + A
2. E = D – A

Rule 3:

1. H = F + G

For this set of rules, assume a user edited or locked measure B. Upon evaluation of the protection processing process, the following would occur:

1. B becomes an affected measure.
2. Rule 1 becomes an affected rule.
3. The expression A = B + C in Rule 1 is a free expression that calculates the free measure A.
4. Because Rule 1 has only one free expression, it becomes a forced rule.
5. A becomes a forced measure and therefore an affected measure.
6. Rule 2 becomes an affected rule because it contains the affected measure A.
7. Because Rule 2 contains two free expressions, it does not at this time become a forced rule.
8. Because Rule 3 contains a single expression and because the measure calculated by that expression cannot be mapped back to the right-hand side measures, measure H is protected by the calc engine and cannot be edited.

And so, at the conclusion of evaluating protection processing for the given set of rules, the states of the measures is as follows:

- B is edited.
- A is forced and therefore, protected.
- C is not protected and can be edited.
- D and E are free measures of an affected rule. Either can be edited but editing one will cause the other to be forced and, therefore, to be protected.
- F and G unaffected measures and therefore can be edited.
- H is protected as the calc engine cannot resolve changes to the measure's values.

At this point, the calc engine would begin to calculate knock-on effects based on the protections of B, A and H. These knock-on effects could result in the forcing and protection of additional measures. The process will be evaluated iteratively until all the knock-on effects of the original edit have been processed.

To provide an example of how protection processing can force measures outside the scope of the triggering rule, consider the case where Rule 1 and Rule 2 are unchanged, but the expression of Rule 3 is instead:

Rule 3:

A = F + G

In this scenario, protection processing causes measure A to be protected because changes to A cannot be resolved against measures F and G. Furthermore, measure B also becomes protected, as changes to it would cause A to be an affected measure. In this case the state of the measures, before any edits by the user, will be:

- A is protected.
- B is protected as changes to it would force an update to A.
- C is not protected and can be edited.
- D and E are both unaffected and can be edited (the presence of the protected measure A on the right-hand side of an expression does not cause them to be protected).
- F and G are both unaffected and can be edited. Because measure B is protected, the calc engine can resolve changes to F or G by making B a forced measure.

Determining What to Calculate

The protection processing process has established which measures are forced given the current set of changes and locks. When a calculate is issued, those forced measures will be calculated (using the forced expressions). However, there may be affected rules that are not forced. For those affected rules, we know that an expression must be evaluated, and the calculation engine must select one of the expressions. Otherwise the integrity of the rule is compromised.

When there are one or more affected rules that are not forced, the highest priority affected rule is selected. From this selected rule, the highest priority free expression is selected, and it will be evaluated. These are the only uses to which the rule and expression priorities are put. The measure that is calculated by the selected expression is then treated as forced, and knock-on effects considered, which are likely to cause other rules and measures to become forced. At the end of this process, if there are still affected rules that are not forced, the process is repeated until there are no affected rules that are not forced. At this point, any rule that is not affected does not need to be evaluated, and an expression has been forced or selected for all rules that need to be evaluated to ensure the integrity of all measures.

Determining the Calculation Sequence

Determining What to Calculate has established which expressions to evaluate, but not the sequence in which they are evaluated. The sequence of evaluation of expressions is driven by the status of the right-hand side (RHS) measures. All normally spreadable (not of recalc type) measures that are changed can be spread and then aggregated at the start of the calculation cycle. Normally, spreadable measures that have been changed and those measures that will not change during the calculation are considered complete. Any expression whose RHS measures are all complete may be evaluated. If the expression is a mapping rule for a recalc measure, the changed values for the mapped spreadable measure will be calculated for all changed cells. That measure may then be spread and aggregated normally. If the expression is for normal base intersection evaluation, the measure will be calculated, and may then be aggregated. In either case, the calculated measure is now ‘complete,' which may make further expressions available to be evaluated. The process continues until all expressions have been sequenced.

When determining the sequence of calculation, the evaluation of expressions is intermingled with spreading and aggregation. In very trivial cases, where all changed measures are spreadable, there will be:

- a phase where some measures are spread.
- a second phase where some measures are calculated at the base intersection.
- a third phase where some measures are aggregated.

However, if any recalc measures have been changed at aggregated levels, the mapping rule cannot be applied until any affected measures on the RHS of the expression have been spread or calculated and then aggregated.

Note:

This is a simplified description of the calculation sequence. For efficiency purposes, groups of measures that must be spread, aggregated, or evaluated are batched together, so that an individual measure is not necessarily spread, aggregated, or evaluated as soon as it is available for that action. However, it is always spread, aggregated, or evaluated before the results of that action are required for another step. Also, expressions are not evaluated for all cells, but only for those cells where one or more of the measures on the RHS of the expression have changed. There are similar efficiencies in aggregation to avoid the redundant re-aggregation of cells that will not have changed.

Cycle Groups

This section describes the cycle group feature of the RPASCE calculation engine. This feature enables relationships between measures that have cyclic dependencies from the measure perspective (there seems to be a deadly embrace where each measure depends upon the other) but are acyclic when the time dimension of these measures is considered.

Without this feature, such relationships could not be set up because the calculation engine would be unable to find a calculation sequence that enabled both measures to be calculated.

A common application of cycle groups can be found in inventory calculations that involve measures, such as beginning of period (BOP) and end of period (EOP). It is typical that EOP is calculated in some way from BOP for the same period. Other than in the very first period, the BOP of a period is equal to the EOP of the previous period. Since BOP is dependent on EOP and EOP is dependent on BOP, a cycle exists from a measure perspective. However, when the time dimension is considered, calculations can be performed in an acyclic fashion. In this example, if EOP for the first period is calculated first, then BOP for the second period can be calculated. This allows EOP for the second period to be calculated, and so on.

Cycle Breaking Functions

Some of the functions supported by the calculation engine have special cycle breaking logic associated with them. These include functions that reference previous time periods and functions that reference future time periods. When these functions are used, the calculation engine automatically determines when measure dependencies that seem to be cyclic are in fact acyclic when the calculations are performed one period at a time. The lag and lead functions are examples of cycle breaking functions.

Cycle Group Evaluation

A cycle group is a group of expressions that the calculation engine must calculate together in order to avoid cyclic dependencies. If the apparent cycle is broken by a function that looks backward in the time dimension (such as lag), the calculation proceeds with the first time period of each expression in sequence. This is followed by the second time period of each expression in sequence, and it continues until all time periods have been calculated. If the apparent cycle is broken by a function that looks forwards in the time dimension (such as lead), calculation proceeds in reverse order starting with the last time period.

Note:

Since the acyclic calculation of expressions in a cycle group is a base level calculation, all measures being calculated in the cycle group must share the same base intersection. That is, the cycle group evaluation process cannot aggregate measures calculated in the cycle group during the cycle group evaluation.

Cycle Group Example

Consider the following measures:

BOP: beginning of period inventory

EOP: end of period inventory

OS: opening stock (that is, the opening inventory for the first period in the plan horizon)

SLS: sales

RCP: receipts

And consider the following rules:

R1: BOP = if(current == first, OS, lag(EOP))

R2: EOP = BOP – SLS + RCP

RCP = EOP – BOP + SLS

When the measure RCP is edited, R2 is affected and the EOP expression in this rule is forced. Then rule R1 is affected and the BOP expression in this rule is forced.

Since the calculation of EOP requires BOP and the calculation of BOP requires EOP, a cycle is detected that contains both of the selected expressions. This is a valid cycle group because the calculation of BOP is dependent on the lag of EOP. Therefore, the cycle can be broken and the intra-cycle ordering results in the BOP expression being evaluated first and EOP expression second.

The evaluation of this cycle group involves the calculation of the first time period of BOP, followed by the first time period of EOP, followed by the second time period of BOP, followed by the second time period of EOP, and continues until all time periods have been calculated.

Synchronized Measures

Measure synchronization is an RPASCE user interface and calculation engine feature. It enables measures that are very closely related to be represented in the user interface (UI) in a more intuitive manner. It can give the appearance of a cell edit or lock affecting two different measures. From a calculation perspective, the cell edit or lock is only applied to one of these measures. A common application of synchronized measures is to allow BOP and EOP to be synchronized. From a business logic perspective, the BOP in one period and the EOP in the previous period are the same thing, and measure synchronization means that even before calculation, an edit or lock of BOP in one period also displays on the UI as an edit or lock of EOP for the previous period, and vice versa.

To accomplish measure synchronization, a measure is defined with a synchronized view type and a list of synchronized source measures. The measure defined with these attributes is called the synchronized target measure. Synchronized target measures may be edited, but any such edits are treated as edits to the underlying synchronized source measures. Protection processing is performed on the synchronized source measures. The protection state of the target measure is then derived from that of the source measures. An edit to one of the source measures is also reflected in the display of the target measure.

The synchronized view types that can be used to define synchronized target measures are as follows:

1. sync_first_lag: The first period of the target measure is synchronized with the first source measure, and periods 2...N of the target measure are synchronized with periods 1...N-1 of the second source measure, where N represents the last period. The first source measure will not have a time dimension. This view type is particularly useful for defining BOP target measures. Here the first source measure would be an opening inventory, and the second source measure would be the EOP.
2. sync_lead_last: Periods 1...N-1 of the target measure are synchronized with periods 2...N of the first source measure and period N of the target measure is synchronized with the second source measure, where N represents the last period. The second source measure will not have a time dimension. This view type is particularly useful for defining

EOP target measures. Here the first source measure would be BOP, and the second source measure would be a closing inventory.

3. sync_first: The target measure is synchronized with the first period of source measure. The target measure will not have a time dimension. This view type is particularly useful when defining OS target measures.
4. sync_last: The target measure is synchronized with last period of the source measure. The target measure will not have a time dimension. This view type is particularly useful when defining CS target measures.

> **Note:**
>
> In order for a synchronized measure to be editable, all of the measures that it is synchronized with must be viewable on the worksheet, but they do not need to be visible.

Synchronized Inventory Examples:

Consider the following measures:

BOP: beginning of period inventory

EOP: end of period inventory

OS: opening stock (that is, the opening inventory for the first period in the plan horizon)

CS: closing stock (that is, the closing inventory for the last period in the plan horizon)

SLS: sales

RCP: receipts

And consider the following rules:

R1: BOP = if(current == first, OS, lag(EOP))

R2: EOP = BOP – SLS + RCP

RCP = EOP – BOP + SLS

BOP can be defined as a synchronized measure constructed from the OS and EOP measures with the sync_first_lag type. Only one expression in the rule group may have BOP on the LHS. This expression is used to construct views of BOP, and it is merged with expressions that require BOP on the RHS.

When edits or locks are made to BOP, it is the underlying values of OS or EOP that are changed or locked. Thus, even though rule R1 has only one expression and this expression calculates BOP, the BOP measure is not protected by protection processing because of the measure synchronization. The BOP measure is only protected when the underlying OS or EOP measures are protected, so the first period is protected when OS is protected, and the remaining periods are protected when EOP is protected.

In this example, a CS measure is not required for calculation purposes, but it may be desired for viewing and editing purposes. For example, a window that contains only OS and CS but not BOP nor EOP may be wanted. In this case, the CS measure should be defined as a synchronized measure with type sync_last and the synchronized source measure would be EOP. As a result, an edit to CS becomes an edit to the last period of EOP.

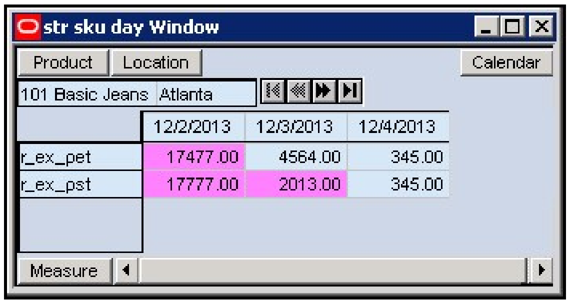
Elapsed Period Locking

Many planning and prediction applications will cover a time horizon where some of the time periods are in the past (that is, have elapsed), and others are in the future. RPAS CE assumes that time periods that have elapsed contain actuals, and that these actuals should not be editable. Therefore, all measures are rendered un-editable during elapsed periods. For positions at aggregated levels in a time hierarchy, the position is considered elapsed when the last lowest level time period descended from it has elapsed.

The RPAS CE Calculation Engine has special logic for handling elapsed time. Apart from being un-editable in the user interface, spreading never spreads a value to an elapsed cell (for more information, see the section "Locks and Spreading Around Locks and Changed Cells").

Measures that represent beginning of period (BOP) data have special handling. From a business perspective, the BOP in a period is the same as the end of period (EOP) in the previous period. Therefore, when an EOP value is elapsed, the following BOP value must also be elapsed. In RPAS CE, all measures with a default spread method of Period Start Total (PST) (for more information, see the previous “Spreading Methods" section), or with their measure property Period Start Value set to TRUE are assumed to be "BOP type" measures, and are protected for all elapsed periods, and for the first non-elapsed period. The following example worksheet demonstrates a situation in which the elapsed threshold has been set to 12/2/2013. The pink-colored cells have been set to read-only by RPAS CE in order to honor elapsed period locking. In this example, the measure e_ex_pet is a regular measure, whereas, r_es_pst is a BOP type measure.

Figure C-2 Elapsed Period Locks for BOP and Non-BOP Measures



101 Basic Jeans Atlanta

	12/2/2013	12/3/2013	12/4/2013
r_ex_pet	17477.00	4564.00	345.00
r_ex_pst	17777.00	2013.00	345.00

There is also special handling of BOP type measures for aggregated time positions. These are treated as elapsed and are therefore protected when the first bottom level time period descended from it is elapsed.

To set up elapsed period locking in a workbook, workbook designers should set the elapsed time threshold in the load rule of the workbook using the elapsed keyword (for more information, see the Functional Keywords section. If the elapsed time threshold is not set, elapsed period locking will not be available in the workbook.

Example:

To setup the elapsed threshold to today, you would first create a one-dimensional measure, pDay for example, with its intersection at the Day level of the Calendar hierarchy. Then, you would set up a rule like the one shown later to initialize this measure with the index of today.

```
pDay = prefer (today-1, if (now>end, last, -1))
```

You would then aggregate this measure using the PST aggregation method to set the elapsed time threshold as shown in the following rule.

```
elapsed = pDay.pst
```

Setting up the elapsed threshold in the load rule fixes the threshold for the life of the workbook; however, in-season planning applications may require the elapsed threshold to change during the lifetime of a workbook. To achieve this, you can reset the elapsed threshold in a Refresh rule-group or in the Calc rule-group using rules exemplified in the preceding discussion. RPAS CE inspects the value of threshold after execution of these rule-groups and immediately adjusts the elapsed period locks in the UI. Note that since elapsed threshold is evaluated and executed after the execution of these rule-groups, any spreading performed in the Calc cycle itself would use the state of elapsed threshold before the rule-group was invoked.

Non-Conforming Expressions

One of the strengths of the RPASCE calculation engine is that a workbook may contain measures with different scopes. The size and shape of the multidimensional cube of data may vary by measure. Any two given measures in a workbook may have scopes that align exactly (for example, both measures have a base intersection of SKU/Store/Week), or where one is a subset of the other (for example, one has a base intersection of SKU/Store/Week and the other is at Class/Week). There can also be circumstances where each measure includes a hierarchy in its base intersection that the other dimension does not use (for example, one has a base intersection of Class/Week and the other is Store/Week). In extreme circumstances, the scopes of two measures may have no point of overlap at all (for example, one has a base intersection of Class and the other Store).

It is the scope of the measure on the LHS of an expression that determines the cells that must be calculated by the expression, even though that scope may be changed by the use of a modifier such as level. Where one or more measures on the RHS of an expression have a scope that is different (in any way) to the scope of the LHS measure, the expression is deemed to be non-conforming. There is special logic to handle the calculation of non-conforming expressions, which depends on the type of nonconformity.

Although not explicitly declared, there is a single logical All position at the top of every hierarchy. When considering non-conformity, any measure that is not explicitly dimensioned on a hierarchy is implicitly assumed to be dimensioned on the All dimension of that hierarchy, so all data values are assumed to be for the All position. This concept is the key to understanding the handling of non-conforming expressions.

Handling of Non-Conforming Expressions

When the concept of the All position is understood, all expressions can be considered to contain measures that use the same hierarchies. The only potential differences between them are the bottom levels (dimensions in the base intersection). For handling non-conformity, only three cases need to be considered, for each hierarchy:

- **RHS same:**

  In this case the RHS measure has the same bottom level as the LHS measure. The RHS measure is conforming for that hierarchy, and values for the RHS measure are taken from the same position as the position being calculated for the LHS measure.

- **RHS higher:**

  In this case the RHS measure has a higher bottom level than the LHS measure. The RHS measure is non-conforming for that hierarchy. The values for the RHS measure for the position being calculated are assumed to be the same as the value of the RHS measure for the position in its bottom dimension that is the parent (ancestor) of the position being calculated. Effectively, it can be considered that the value of the measure has been replicated down the hierarchy to the required level.

- **RHS lower:**

  In this case the RHS measure has a lower bottom level than the LHS measure. The RHS measure is non-conforming for that hierarchy, but because the scope of the RHS measure includes the bottom level for the LHS measure, values for the RHS measure are taken from the same position as the position being calculated for the LHS measure.

The conceptual case where the measures have scopes that do not overlap, because they have base intersections in a hierarchy that are for dimensions that are up different branches of the hierarchy, fails rule validation.

Examples

These examples all use the simple expression a = b + c

Example 1:

Consider the following values:

- a has a base intersection of SKU/Store/Week
- b has a base intersection of SKU/Week
- c has a base intersection of SKU/Region/Week

For each SKU/Store/Week, a is calculated from the value of b at SKU/Week (that is, it is assumed that the value of b is the same for all positions in the location hierarchy) and the value of c at SKU/Region/Week, for the Region the Store belongs in. If replication of c from the Region level is not appropriate, the rule writer can simulate other spreading techniques using functions and modifiers such as count and level. For example, the count function may be used to determine the number of Stores in the Region, and so dividing the measure c by that count will simulate even spreading.

Example 2:

Consider the following values:

- a has a base intersection of SKU/Store/Week

- b has a base intersection of SKU/Week
- c has a base intersection of SKU

For each SKU/Store/Week, a is calculated from the value of b at SKU/Week (that is, it is assumed that the value of b is the same for all positions in the location hierarchy) and the value of c at SKU (that is, it is assumed that the value of b is the same for all positions in the location hierarchy and time hierarchy). Note that an alternative approach, if required, would be to use a level modifier on the measure a, so that it is calculated at, say, SKU/Week, and then spread down to SKU/Store/Week, using the existing store participations to the measure a.

Example 3:

Consider the following values:

- a has a base intersection of SKU/Week
- b has a base intersection of SKU/Store/Week
- c has a base intersection of SKU/Region/Week

For each SKU/Week, a is calculated from the value of b and c at SKU/All/Week

D

Appendix – Rules Function Reference Guide

This appendix provides references for rule functions, including:

- Syntax and Design
- Special Handling for Functions
- Non-Conforming Measures
- Functional Keywords
- Modifiers
- Description of Functions

Syntax and Design

This section provides the syntax and design of functions, procedures, modifiers, and keywords that are used in expressions in the RPASCE calculation engine. There are important distinctions between each of these definitions.

Functions

Functions are separated into two types: single result functions and multiple result functions.

Single Result Functions

Mechanisms for performing operations within an expression that are controlled and executed by the calculation engine.

- Functions are most commonly used in RPASCE.
- Most functions in base RPASCE return only a single measure.
- Calculation engine controls and runs the evaluation of a function.
- Functions may be used in expressions with other functions and keywords.

Multiple Result Functions

Similar to the features and behavior of single result functions, but with semantic and syntactic differences.

- There can be more than one left-hand side (result) measure that can be specified implicitly by position in the expression or explicitly by label.
- Left-hand side measures must be at same intersection; however, the calendar hierarchy can be dropped or added.
- The results from a multiple result function cannot be used as arguments to another function, nor can the results be chained with other operations to form long expression.
- Expressions can be used as arguments to multiple result functions.
- Multiple result functions cannot be part of a cycle group.

Procedures

Procedures are mechanisms for performing operations in an expression where the calculation engine controls the execution, which is performed by the procedure itself.

- Procedures can only use measures or scalars
- Procedure runs the evaluation (instead of RPASCE/calculation engine), but the calculation engine still controls protection processing, sequence of calculation, when the procedure is called, and so on.
- Procedures can have multiple arguments on the left-hand and right-hand sides.
- Procedures cannot be used with functions, other procedures, keywords, and certain modifiers.
- Because of their flexibility and the control available to the developer, procedures can be used for a wide variety of special calculations and activities.
- Procedures require a different syntax. The syntax uses “<-“ instead of “=" in the expression.

Modifiers

Modifiers directly modify the source or destination of measures, to override the level, aggregation type, position, and so on.

Modifier Syntax

```
<measure>.<modifier>
```

Keywords

Keywords appear in expressions or as arguments inside functions to return specific data values.

Syntax Conventions

The syntax is as quick and straightforward to implement as possible. Function names, keywords, and so on are currently in lowercase.

Keywords are allowed, but they are kept to a minimum. Function parameters are comma separated and may be optional; however, they are positional, so that the absence of a parameter needs to be specified by commas if a subsequent parameter is supplied.

Table D-1 displays the syntax conventions used in this procedure.

Table D-1   Syntax Conventions

Indicator	Definition
[…]	All options listed in brackets are optional.
{…|…}	Options listed in “{}" with “|" separators are mutually exclusive (either/or).

Table D-1 (Cont.) Syntax Conventions

Indicator	Definition
{…,…}	Options listed in “{}" with “," separators way are a complete set.
Bold	Bold indicate labels.
Italics	Italics indicate a temporary placeholder for a constant or a measure.
Italics/meas	This indicates that the placeholder can be either a constant or a measure.
Bold Italics	This indicates a numeric placeholder for the dynamic portion of a label. Usually a number from 1 to N.
Normal	Normal text signifies required information.
Underlined	This convention is used to identify the function or procedure name.

The following is the functional syntax used in this guide:

- Large square brackets [ ] are used to indicate an optional parameter.
- Small square brackets [ ] are part of the expression syntax and are used to specify a hierarchy, dimension, and/or position.
- Large braces { } indicate a choice where one of the items (which will be separated by a pipe sign "|") must be selected.
- Small braces {} are part of the expression syntax, and are used to specify a measure set for functions that accept a variable number of arguments (that is, {<measureset>}).
- Parameters of a specific type (such as expressions or dimension names) are shown in angle brackets <>.
- A plus “+" sign is used to specify an intersection, which is done by connecting two or more dimension specifications.
- Keywords, modifiers, function names, and procedure names are shown in bold.

Specification of Hierarchy, Dimension, or Position

Many functions in RPASCE require the specification of a hierarchy, dimension, or a combination thereof, to define the level at which an expression is evaluated. When defining the hierarchy and dimension names in expressions square brackets [ ] must be used.

In the document, the following syntax is used to designate a hierarchy and dimension:

Hierarchy, Dimension, and Position Syntax

```
[<hierarchy>].[<dimension>].[[<position>]]
```

Note:

Position is noted as optional because it can only be specified in a limited number of functions.

For simplicity of parsing and clarity of rule writing, the `<hierarchy>` must be supplied in all cases, even when, as in calendar index functions, it might be implied from the context. Functions that require a hierarchy and dimension specification have standard validation rules whereby `[<hierarchy>]` must be a valid hierarchy name, `[<dimension>]` must be a valid dimension in `[<hierarchy>]`, and `[<position>]` must be a valid position name in `[<dimension>]`. If the position name starts with a number, the position name must be nested in a pair of double quotes. In some functions or procedures, one of the hierarchical keywords top, bottom, or current (used conditionally based on context) can be used to specify the dimension. Should this validation fail, an error is generated.

Function Inverses

Some functions (such as cover) have what are referred to as "inverse" functions. This is required, as all expressions in a rule group must be algorithmic inverses of each other. Each function states whether it has an inverse, and, if so, what the syntax of the inverse is.

An inverse function is only relevant when the function encompasses the whole of the expression. Functions embedded in longer expressions do not have inverses, though the expression itself may have an inverse if the measure being "solved" for is not an input into the function. Functions that have inverses usually have enough scope in their syntax to cover the eventualities that would typically cause them to be embedded in longer expressions (such as code to prevent an error result).

Functions with Multiple Results

The following special syntax should be used for functions with multiple results.

The left-hand side measures in a multiple result expression are comma-separated and can be identified by a labeling mechanism.

Label Syntax

```
<measure>:<label>
```

Valid label names are specified by the multiple result function syntax. If a multiple result function specifies valid labels, the function can be used in an expression without specifying all possible results. The multiple result function itself is aware of which results are being stored and may be able to run faster by skipping the computation of unneeded results.

Special Handling for Functions

There are several keywords and functions that have special control flow over the evaluation of the expression.

Error Handling

RPASCE has no facility for holding an error value for a cell. Should the evaluation of any expression, or clause in an expression, result in an error, the value for the cell or clause will be the naval.

Note:

It is good programming practice to check for any clauses that may return an error, and the prefer function provides a way to specify the behavior under these circumstances. Some functions have their own implicit error handling.

Keyword: if

Used for handling conditional logic and masking updates within expressions.

Syntax

```
if(<condition>, <use-expression>, <else-expression>)
```

where `<condition>` is any valid Boolean expression. `<use-expression>` and `<else-expression>` are any valid expressions that are evaluated based on the result of `<condition>`; one (and only one) of these expressions can contain the keyword ignore. `<use-expression>` is evaluated when the result of `<condition>` is true; `<else-expression>` is evaluated when the result of `<condition>` is not true.

`<expression>` is any valid expression. ignore is a keyword that is used to indicate that the entire expression is not to be evaluated (that is, masking the update to the entire expression).

Note:

The use of ***ignore*** can only be used in either the `<use-expression>` or `<else-expression>`, but not both.

The use of ***ignore*** can only be used in either the `<use-expression>` or `<else-expression>`, but not both.

The use of **ignore** always flags the expression as a masked update – this will always prevent the expression from being evaluated or involved with aggregations when the condition is not met. To reiterate, note that the entire expression is not evaluated, not just the sub-expression that uses the if clause. When ignore is used in expression where the LHS measure is modified with the master keyword (typically in a commit rule group), then the `<condition>` must be a Boolean measure (in other words, not an expression). This syntactical restriction is validated when the expression is parsed.

if clauses can be nested without restrictions but must be enclosed with parentheses when used more than once within an expression.

Examples:

Conditional logic:

- BOP = `if(current == first, SeasOP, lag(EOP))`
- OTB = `if(ProjEOP > PlanEOP, 0, PlanRecpt - OnOrder)`

Masked update with a single expression:

- `SalesOP = if(Approved, SalesWP, ignore)`

Updates Sales for the Original Plan version to the value in the Working Plan version when the Boolean measure Approved is set to true. ignore designates that no update is made to `SalesOP` if the Approved measure is false. This is functionally equivalent to the next example.

- `SalesOP = if(NotApproved, ignore, SalesWP)`

  Does not update the measure `SalesOP` with the values from the measure `SalesWP` when the Boolean measure `NotApproved` is true.

- Note the distinctly different behavior between the following similar expressions:
  - `a = b + (if(<condition>, c, ignore))`

    This is an example of a masked update where no update is made to measure a if the condition is not met (that is, the entire expression is not evaluated).
  - `a = b + (if(<condition>,c, 0))`

    This is an example of conditional logic where an else clause is provided and the expression is always evaluated, thus `a` is always updated to either `b` or `b+c`.

Keyword: prefer

Returns the first non-error value from a series of expressions.

The primary use is to enable the capture and appropriate calculation of error conditions.

Syntax

```
prefer(<expression1>, <expression2> [, <expression3> … <expressionn>])
```

Where `< expression1-n>` are expressions which return values of the appropriate data type. The function returns the value of the first of the expressions that does not generate an error when it is evaluated. It is good coding practice to use a prefer function around any clause of an expression, which could potentially generate an error.

Inverse

The prefer function does not have an inverse.

Examples:

- `prefer(A/B, 100)`

  This example returns the value of A divided by B, unless that generates an error (as it would if B is zero), when it returns 100.

- `prefer(lag(A), B)`

  This example returns the value the lag of A, unless that generates an error (as it would when evaluating the first period of the plan horizon), when it returns the value of B. The `prefer` function in this example is thus the functional equivalent of the expression:

  ```
  if(current == first, B, lag(A))
  ```

Non-Conforming Measures

One of the strengths of the RPASCE engine is that a workbook may contain measures with different scopes: the size and shape of the multidimensional cube of data may vary by measure. Any two given measures in a workbook may have scopes that align exactly (for instance, both measures have a base intersection of SKU/Store/Week), or where one is a subset of the other (for instance, one has a base intersection of SKU/Store/Week, and the other is at Class/Week). There can also be circumstances where each measure includes a hierarchy in its base intersection that the other dimension does not use (for instance, one has a base intersection of Class/Week and the other is Store/Week). In extreme circumstances, the scopes of two measures may have no point of overlap at all (for instance, one has a base intersection of Class and the other Store).

It is the scope of the measure on the left-hand side of an expression (referred to as the LHS measure) that determines the cells that must be calculated by the expression, though that scope may be modified using a modifier such as level. Where one or more measures on the right-hand side (RHS) of an expression have a scope that is different (in any way) to the left-hand side (LHS) measure, the expression is deemed to be non-conforming. There is special logic to handle the calculation of non-conforming expressions, which depends on the type of nonconformity.

Although not explicitly declared, there is a single logical All position at the top of every hierarchy. When considering non-conformity, any measure that is not dimensioned on a hierarchy, is implicitly assumed to be dimensioned on the All level of that hierarchy, and thus all data values are assumed to be for the All position. This concept is the key to understanding the handling of non-conforming expressions.

When the concept of the All position is understood, all expressions can be considered to contain measures that use the same hierarchies. The only potential differences between them are the bottom levels (dimensions in the base intersection). Thus, for handling non-conformity, only three cases need to be considered, for each hierarchy:

- **RHS same**

  In this case, the RHS measure has the same bottom level as the LHS measure. The RHS measure is conforming for that hierarchy, and values for the RHS measure are taken from the same position as the position being calculated for the LHS measure.

- **RHS higher**

  In this case, the RHS measure has a higher bottom level than the LHS measure. The RHS measure is non-conforming for that hierarchy. The values for the RHS measure for the position being calculated are assumed to be the same as the value of the RHS measure for the position in its bottom dimension that is the parent (ancestor) of the position being calculated. Effectively, it can be considered that the value of the measure has been replicated down the hierarchy to the required level.

- **RHS lower**

  In this case, the RHS measure has a lower bottom level than the LHS measure. The RHS measure is non-conforming for that hierarchy, but because the scope of the RHS measure includes the bottom level for the LHS measure, values for the RHS measure are taken from the same position as the position being calculated for the LHS measure. RHS measure is aggregated using the default aggregation method.

  The conceptual case where the measures have scopes that do not overlap, because they have base intersections in a hierarchy that are for dimensions that are up different branches of the hierarchy, fails rule validation.

Non-Conforming Measure Examples

The following examples all use the simple expression `a = b + c`.

Example 1

Consider the following scenario:

- a has a base intersection of SKU/Store/Week
- b has a base intersection of SKU/Week
- c has a base intersection of SKU/Region/Week

For each SKU/Store/Week, a is calculated from the value of b at SKU/Week (it is assumed that the value of b is the same for all positions in the location hierarchy) and the value of c at SKU/Region/Week, for the Region the Store belongs in. If replication from the Region level is not appropriate, the rule writer can simulate other spreading techniques using functions and modifiers such as count and level.

For example, the count function may be used to determine the number of Stores in the Region, and so dividing the measure c by that count will simulate even spreading. Additionally level could be used to force the calculation of a at Region instead of as base intersection, Store (a.level([loc].[reg])=b+c). In this scenario edits to b or c would calculate a at Region and would then spread those values down to Store for measure a using the default spread method.

Example 2

Consider the following scenario:

- a has a base intersection of SKU/Store/Week
- b has a base intersection of SKU/Week
- c has a base intersection of SKU

For each SKU/Store/Week, a is calculated from the value of b at SKU/Week (it is assumed that the value of b is the same for all positions in the location hierarchy) and the value of c at SKU (it is assumed that the value of b is the same for all positions in the location hierarchy and time hierarchy).

Note:

An alternative approach, if required, would be to use a level modifier on the measure a, so that it is calculated at SKU/Week, and then spread down to SKU/Store/Week, using the existing store participations to the measure a.

Example 3

Consider the following scenario:

- a has a base intersection of SKU/Week
- b has a base intersection of SKU/Store/Week
- c has a base intersection of SKU/Region/Week

For each SKU/Week combination, a is calculated from the value of b and c at SKU/All/Week. Otherwise stated b and c are aggregated up the location hierarchy, and then added to a for each position in SKU/Week.

Functional Keywords

Functional keywords are keywords that may be used in expressions that return specific data values. There are a group of keywords that provide information (in the form of index numbers) about the calendar hierarchy, and a further group of keywords that provide information of the current session.

Calendar Index Functional Keywords

Certain calendar index functional keywords are supported in the syntax, as described in this section. In this context, a calendar index number is an ordinal position counter of the position in a dimension within the scope of the calendar horizon, where the dimension is as for the cell being evaluated. For example, in a plan whose scope is a year, the first week will have an index of 0, week 26 will have an index of 25, and week 52 will have an index of 51. Similarly, if an expression is being evaluated at the quarter level, the first quarter will have an index of 0, and the last one an index of 3. Calendar index functional keywords may be included in any numeric expression.

first

This returns the index number of the first calendar position.

This keyword is provided for completeness and clarity of rule function writing, since the value will always be zero.

last

This returns the index number of the last calendar position.

last + 1 will therefore always be the number of positions in the calendar horizon in the current dimension.

current

This returns the index number of the period being evaluated.

current can be used as a standalone keyword only under the context of time.

Note:

It can also be used in the syntax of a function as a hierarchical keyword (for specifying the current level in a hierarchy) and is allowed for any hierarchy (but must follow the syntax `<hierarchy>.current`).

today

This returns the index number of the period that contains the current time as given by the system clock.

The index number that today returns is determined by the base intersection of the measure that is being evaluated (on the left-hand side of the expression). For example, if the base intersection of the measure being evaluated is week, today will return the index number of the current week.

Note:

The effect of this keyword may be overridden by providing the environment variable RPAS_TODAY. If this is present, the time in the RPAS_TODAY environment variable is used instead of the system clock time.

Note:

The difference between the keywords today and now is that today returns an index number; now returns the value of the current date and time. An error is generated when the current period is not included in the workbook.

elapsed

This returns the index number of the period that is the last elapsed period.

elapsed is interpreted as the last period for which actuals have been posted. When used on the RHS of an expression, it returns the index number of the period that is the last elapsed period for the level of calendar hierarchy at which the calculation takes place. When used on the LHS, this sets the last elapsed period along the base dimension of the Calendar hierarchy to the given index number, that is, it is assumed that the index number used to set elapsed is along the base dimension of the hierarchy. If there is no elapsed period, this keyword returns –1. Furthermore, when used on the RHS of an expression whose LHS does not have the Calendar hierarchy, this keyword returns -1.

elapsed must be assigned a value corresponding to the index of the last elapsed period before it can be used in calculations (on the right-hand side of other rule groups). This assignment can happen in load, refresh, or calc rule-groups but not in commit rule groups. Use the following syntax for assigning the index number in the base calendar dimension as the elapsed value.

Syntax

```
elapsed = <expression>
```

Where `<expression>` is any valid expression that returns a numeric value, of which only the integer portion is used.

Example

For example, to update elapsed to always correspond to today:

1. Add a new single dimensional measure with base intersection at day.

   Call this measure `pDay`.

2. In the Calc rule group or in the Load and Refresh rule group.

a. Add rules that initialize the new measure with the calendar index of today, as shown in this example:

```
pDay=prefer(today-1,if(now>end,last,-1))
```

b. Add a new rule that sets the elapsed measure:

```
elapsed=pDay.pst
```

> **Note:**
>
> In the previous example, the intersection of `pDay` must match the lowest dimension in the workbook. If the lowest calendar dim of the workbook is at the day level, then the elapsed index calculated must correspond to the number of days elapsed. If the lowest calendar dim of the workbook is at the week level, then the elapsed index calculated must correspond to the number of weeks elapsed.

The following examples illustrate the behavior of the keyword. For these examples, assume that RPASCE, using the previous mechanism, has calculated that the index for the last period along the base dimension (Day) to be 30 and that in the Calendar hierarchy, this day rolls up to week with index 4 and month with index 2. In this scenario, the following would occur:

- `LHS_week = elapsed` puts a value of 4 in the LHS_week measure which is along the Week dimension.
- `LHS_day = elapsed` puts a value of 30 in the LHS_day measure which is along the day dimension.
- `LHS_sku_str_day = elapsed` puts a value of 30 in the LHS_sku_str_day measure whose calendar hierarchy is along the Day dimension.
- `LHS = elapsed` changes the navalue of LHS to -1, where the LHS measure is a scalar. Since the measure does not have a Calendar hierarchy, elapsed cannot determine the period index that this measure needs. In order to get a scalar populated, declare a measure along the level at which the elapsed index is required, assign it the elapsed keyword, and then assign this measure to the scalar with an aggregate of PST, PET, or AMBIG.

Session Keywords

This section describes session keywords.

now

This returns the current date and time from the system clock.

`now` is stored with date and time information.

> **Note:**
>
> The difference between the keywords `today` and `now` is that `today` returns an index number; `now` returns the value of the current date and time.

The displayed format of `now` is based on the measure type.

This keyword can be used to hold information about when data was changed (for instance, the beginning date and time of a batch run). The value returned by `now` can be overridden by RPAS_TODAY environment variable.

userID

This returns a string that contains the id of the current user.

This keyword can be used to hold information about the user who made a specific change.

username

This returns a string that contains the account name of the current user.

This keyword can be used to hold information about the user who made a specific change.

Beginning with 19.x, the userid is ID (unique id) and Username is the username set by the customer while retrieving it from the session keywords. This means that, from now on, the customer must use the username as the session key word in all their rules. Currently, if the customer uses userid as the session keyword in the workbook template, the customer will see a unique ID uid000xx instead of the name. RPASCE applies strict rules when handling the personally identifiable information of a customer. In the past, customer have used the user email ID as the username, which cannot be logged into the log files.Anything displayed in the browser, such as OAT wizards/User Admin Wizards all show the username except for the log files, which are logged into the OAT tasks. Now and in the future, they cannot contain any personally identifiable information. These log files are shared within Oracle for analysis and it is regulated that all customer personally identifiable information must be removed. Only the userid is logged in the log files. In this way, a clear demarcation has been created between userid and username.

Calendar Hierarchical Date Keywords

This section describes calendar hierarchical date keywords.

begin

This returns a date type value for the first index in the MeasureStore calendar dimension.

Because it returns a date type value, this keyword is not context sensitive (meaning it does not depend on where it is being used) and can be compared with the now keyword.

Note:

The root calendar dimension is defined as the unique dimension that is at the root of the calendar hierarchy.

end

This returns a date type value for the last index in the MeasureStore calendar dimension.

Because it returns a date type value, this keyword is not context sensitive (meaning it does not depend on where it is being used) and can be compared with the `now` keyword.

Modifiers

Modifiers are used to directly modify the source or destination of measures. Modifiers must be used in conjunction with a measure in the manner displayed under Syntax:

Syntax

```
<measure>[.<modifier>[.<modifier>]…]
```

The following modifiers can be used with measures in a variety of ways. Note the acceptable uses for each modifier as there are restrictions regarding use on the left-hand side and if they can be used in conjunction with other modifiers.

master

This references the PDS version of a measure.

master is used as a modifier to a measure to reference the version of the measure that resides in the PDS. It can only be used in load and commit rule groups. It cannot be used in calculation rule groups.

Syntax

```
<measure>.master
```

Where `<measure>` is any valid measure. master can be used on both the left-hand side and right hand side of expressions and can be used with functions. When used with other modifiers, master must be the first modifier.

On the right-hand side of an expression, master can be used with both level and aggtype. On the left-hand side, master must be used by itself.

Examples

- `Sales=Sales.master`

  Used in load rule group to retrieve Sales from the PDS into a workbook.

- `Sales.master=Sales`

  Used in commit rule group to commit the updated Sales measure to the PDS from the version in the workbook.

aggtype

This references to alternative aggregation types.

When a measure is referenced just by name in an expression, or as a parameter in a rule function, the value used is for the default aggregation type for the measure. Values from alternative aggregation types are also available by using the syntax:

Syntax

```
<measure>.<aggtype>
```

Where `aggtype` is a supported aggregation type as listed in an appendix of this guide. Every function parameter that requires a measure will also accept this extended form.

> **Note:**
>
> If alternate aggregation types are required for a measure in rules, this approach is more efficient that defining another measure with the alternate aggregation type, as data values at the base intersection are not duplicated.

The `aggtype` modifier can only be used on the right-hand side of an expression, but it can be used with functions and other modifiers. When used with level and/or master modifiers, `aggtype` must be the specified last.

level

This returns the value of an expression for a specific intersection of parent positions or forces the calculation at a specific intersection.

The parents specified may be in one or more hierarchies.

Syntax

```
<measure>.level(<dimspec1>[+<dimspec2>… +<dimspecn>])
```

Where `<dimspec1-n>` is `[<hierarchy>].{[<dimension>] | top | current}` and each dimension specification is separated by a plus sign.

`<measure>` is the measure to be specified. `<hierarchy>` is the name of a valid hierarchy. top and current are keywords referring to the highest, and current (that is, being evaluated if on the RHS, or base intersection in the hierarchy if on the LHS) dimensions in the hierarchy. If a hierarchy is not specified, the <dimension> for that hierarchy is assumed to be current. If the `<dimension>` for a hierarchy is lower than the base intersection for the measure (when used on the LHS), or the `<dimension>` is not a valid dimension in the specified hierarchy, an error is generated.

This modifier can be used on both the LHS and RHS of a rule expression. It can only be used by itself on the LHS, but it can be combined with other functions and modifiers on the RHS.

When this modifier is on the LHS of a rule expression, the rule is evaluated at the specified intersection. The newly calculated value at an aggregated intersection is then spread down the hierarchies to the base intersection for the measure, using the default spread-type for the measure. A typical usage of this modifier on the LHS of a rule expression is to calculate a non-conforming measure where the scope of the measure includes hierarchies not present in the measures on the RHS of the expression. The calculation would usually be at the base intersection of the common hierarchies, but at the top of the additional hierarchies and spread to their base intersections.

When this modifier is on the RHS of a rule expression, the measure being modified is evaluated at the specified intersection.

Note:

Just the measure, not the rule, is evaluated at the designated level.

Examples

- `sales.level([loc].top)`

  This returns the value for the measure sales for the position at the top of the location hierarchy and for the current position in all other hierarchies.
- `sales.level([loc].[area])`

  This returns the value for the measure sales for the position in the area dimension that is the parent of the position being evaluated and the current position in all other hierarchies (that is, the total sales in my area).
- `sales.level([loc].[area]+[prod].[div])`

  This returns the value for the measure sales for the position in the area and division dimensions that is the parent of the position being evaluated and current position in all other hierarchies (that is, the total sales in my area for my division).
- `recpts.level([rec].top) = <expression>`

  The measure recpts is calculated at the base intersection of all hierarchies except the rec hierarchy, where it is calculated at the top. This value is spread down to the base intersection for the measure.

old

This references the value of a measure as of the previous calculate.

Syntax

```
<measure>.old
```

Any measure modified with old will use the value that was available at the start of the calculation process, which means that these modified measures can be ignored for such things as protection processing. Most importantly, this means that a measure can effectively be calculated from itself, as the .old modifier breaks the cycle.

Assumptions/Restrictions

The following assumptions/restrictions apply to `old`:

- Can only be used in a rule group of type calculation.
- Can only be used on the right-hand side of an expression.
- Cannot be used in combination with `.master`, `.level`, or `.aggtype` modifiers.
- Cannot be used with (cannot modify) non-materialized measures.

Use of the old modifier has no effect on calculation sequence or protection processing, as the values of measures modified with old are known before the calculation starts.

Note:

The old modifier is not designed to operate with measures whose aggregation type is recalc. In particular, expressions that attempt to use the old modifier on a measure with an aggregation type of recalc, such as

```
a=b + c.old
```

where c is a measure with an aggregation type of recalc, are not allowed. Similarly, expressions that attempt to calculate a measure with an aggregation type of recalc, but which use the old modifier, such as

```
c=a + b.old
```

where `c` is a measure with an aggregation type of recalc, are also not allowed.

Example

The old modifier can be used in conjunction with the propspread function to implement a hierarchical relationship among measures. In the following example, Total sales (TotalSls) is the parent measure and regular sales (RegSls), promotional sales (PromSls), and markdown sales (MkdSales) are the child measures. Using `old` and `propspread` to configure this relationship allows the manipulation of any combination of these measures before calculating, except for all of them.

In the following example and in other such hierarchical measure relationships, the order of the expressions within a rule is critical for the measures to be correctly calculated.

- `TotalSls = RegSls + PromoSls + MkdSls`
- `RegSls, PromoSls, MkdSls = propspread(TotalSls, RegSls.old, PromoSls.old, MkdSls.old)`
- `PromoSls, MkdSls = propspread(TotalSls - RegSls, PromoSls.old, MkdSls.old)`
- `RegSls, MkdSls = propspread(TotalSls - PromoSls, RegSls.old, MkdSls.old)`
- `RegSls, PromoSls = propspread(TotalSls - MkdSls, RegSls.old, PromoSls.old)`
- `RegSls = TotalSls - PromoSls - MkdSls`
- `PromoSls = TotalSls - RegSls - MkdSls`
- `MkdSls = TotalSls - RegSls - PromoSls`

Description of Functions

This section describes:

- Calendar Index Functions
- Calendar Calculation Functions

Calendar Index Functions

These are functions that return the calendar index numbers of positions that are specified relative to the current position through hierarchical relationships, or by date. Support is in place for functions to find the first and last children of a parent at a given dimension (for instance, the first week of the current quarter, the last week of the current month). These are to support relative time series functions, such as month to date totals. These may be constrained by setting a condition under which the expression is evaluated.

indexfirst

This returns the calendar index number of the first position in the current dimension that is descended from the parent of the current position at the specified dimension.

See the tssum function for an example of typical usage. The function may be constrained by setting a condition for the evaluation.

Syntax

```
indexfirst([<clndhierarchy>].{[<dimension>] | top}[, <boolexpr>])
```

Where `<clndhierarchy>` is the name of the calendar (time) hierarchy, and <dimension> is the name of a dimension in the calendar hierarchy. top is a keyword that implies the top dimension in the calendar hierarchy. If `<dimension>` is not a valid dimension in the calendar hierarchy or it is not a dimension that is equal to or higher than the current (being evaluated) dimension in any alternate hierarchy, an error is generated.

`<boolexpr>` is optional and is any valid Boolean expression used to set a condition for the evaluation of the function. If `<boolexpr>` is not specified, the function returns the index number of the first position of the dimension descended from the parent of the current position of the specified dimension. When `<boolexpr>` is specified, the function returns the index number of the first position of the dimension descended from the parent of the current position at the specified dimension where the <boolexpr> evaluates to true.

Inverse

The `indexfirst` function does not have an inverse.

Examples

- indexfirst([clnd].[qtr])
- If the cell being evaluated is a week, this returns the calendar index number of the first week in the quarter that the week of the cell being evaluated belongs to (that is, the first week in the current quarter).
- indexfirst([clnd].[week], Receipts != 0)

- If the cell being evaluated is a day, this returns the calendar index number of the first day of the current week when that has a value for Receipts that is not equal zero (that is, the first day in the current week with recorded Receipts).
- indexfirst([clnd].top)
- If the cell being evaluated is a week, this returns the calendar index number of the first week in the calendar horizon. This keyword is included for consistency with other functions, as it will always return the value first (that is, zero).

indexlast

This returns the calendar index number of the last position in the current dimension that is descended from the parent of the current position at the specified dimension.

The function may be constrained by setting a condition for the evaluation.

Syntax

```
indexlast([<clndhierarchy>].{[<dimension>] | top}[, <boolexpr>])
```

Where `<clndhierarchy>` is the name of the calendar (time) hierarchy. <dimension> is the name of a dimension in the calendar hierarchy. top is a keyword that implies the top dimension in the calendar hierarchy. If <dimension> is not a valid dimension in the calendar hierarchy or is not a dimension that is equal to or higher than the current (being evaluated) dimension (in any alternate hierarchy), an error is generated.

`<boolexpr>` is optional and is any valid Boolean expression used to set a condition for the evaluation of the function. If `<boolexpr>` is not specified, the function returns the index number of the last position of the dimension descended from the parent of the current position at the specified dimension. When`<boolexpr>`is specified, the function returns the index number of the last position of the dimension descended from the parent of the current position at the specified dimension where the `<boolexpr>` evaluates to true.

Inverse

The indexlast function does not have an inverse.

Examples

- indexlast([clnd].[qtr])
- If the cell being evaluated is a week, this returns the calendar index number of the last week in the quarter that the week for the cell being evaluated belongs to (that is, the last week in the current quarter).
- indexlast([clnd].[week], Receipts != 0)
- If the cell being evaluated is a day, this returns the calendar index number of the last day of the current week that has a value for Receipts that is not equal to zero (that is, the last day of the current week with recorded Receipts).
- indexlast([clnd].top)
- If the cell being evaluated is a week, this returns the calendar index number of the last week in the calendar horizon. This keyword is included for consistency with other functions, as it will always return the value last.

indextostartdate

This returns the start date of the period whose index number is supplied.

Syntax

```
indextostartdate(<index>[ ,[<clndhierarchy>].{[<dimension>] | current}])
```

Where `<clndhierarchy>` is the name of the calendar (time) hierarchy, and <dimension> is the name of a dimension in the calendar hierarchy. current is a keyword that implies the current dimension in the calendar hierarchy. If `<dimension>` is not a valid dimension in the calendar hierarchy, an error is generated. If the calendar hierarchy and dimension are not supplied, the default is the current calendar dimension.

> **Note:**
>
> This function requires that the day dimension of the calendar hierarchy be included in the workbook. If the lowest dimension of the calendar hierarchy is higher than the day dimension, the function will not be able to return a valid date.

`<index>` is an expression that returns an index number in the indicated calendar dimension. If `<index>` is non-integer, only the integer portion is used. If `<index>` is not a valid index number for the specified dimension, an error is generated. If the measure being evaluated does not have a base intersection in the calendar hierarchy, and the current option is used, an error is generated.

The function returns a date that is the start date of the period indicated by the dimension and index number. If the period being evaluated is at or lower than the day level, the start date is the date of the whole of the period. If the period being evaluated is higher than the day level, the start date is the date of the first child position at the day level of the period being evaluated.

Inverse

The `indextostartdate` function does not have an inverse.

Examples

- indextostartdate(current)
- Returns the start date of the current time period.
- indextostartdate (indexfirst([clnd].[qtr]))
- Returns the start date of the first period in the current time dimension in the current quarter.
- indextostartdate (index([clnd].[week], openweek), [clnd].[week])
- Returns the start date of the period at the week level whose name is held in the openweek measure.

indextoenddate

indextoenddate

This returns the end date of the period whose index number is supplied.

Syntax

```
indextoenddate(<index>[ ,[<clndhierarchy>].{[<dimension>] | current}])
```

Where `<clndhierarchy>` is the name of the calendar (time) hierarchy, and <dimension> is the name of a dimension in the calendar hierarchy. current is a keyword that implies the current dimension in the calendar hierarchy. If `<dimension>` is not a valid dimension in the calendar hierarchy, an error is generated. If the calendar hierarchy and dimension are not supplied, the default is the current calendar dimension.

Note:

This function requires that the day dimension of the calendar hierarchy be included in the workbook. If the lowest dimension of the calendar hierarchy is higher than the day dimension, the function will not be able to return a valid date.

`<index>` is an expression that returns an index number in the indicated calendar dimension. If `<index>` is non-integer, only the integer portion is used. If `<index>` is not a valid index number for the specified dimension, an error is generated. If the measure being evaluated does not have a base intersection in the calendar hierarchy, and the current option is used, an error is generated.

The function returns a date that is the end date of the period indicated by the dimension and index number. If the period being evaluated is at or lower than the day level, the end date is the date of the whole of the period. If the period being evaluated is higher than the day level, the end date is the date of the last child position at the day level of the period being evaluated.

Inverse

The indextoenddate function does not have an inverse.

Examples

- indextoenddate(current)
- Returns the end date of the current time period.
- indextoenddate (indexfirst([clnd].[qtr]))
- Returns the end date of the last period in the current time dimension in the current quarter.
- indextoenddate (index([clnd].[week], openweek), [clnd].[week])
- Returns the end date of the period at the week level whose name is held in the openweek measure.

Calendar Calculation Functions

These are functions that return calendar calculations.

addPeriods

This function requires three inputs and generates one output. It produces a Date value output by adding some periods specified by a dimension name to an input Date value.

Input

- Date: Input Date value.
- Integer: Number of periods to be added to the input Date specified by 1.
- String: The Dimension Name of the period, such as DAY, MNTH, and so on.

Output

Date: The input Date plus number of periods.

Example

```
targetDate = addPeriods(srcDate, 1, "DAY")
```

> **Note:**
>
> If srcDate evaluates to Jan/01/2012, targetDate should be Jan/02/2012.

calendarStart

This function has no input and produces one output. The output is a Date type value specifying the starting date of the current application's calendar hierarchy. If called in a workbook, it still returns the starting date of the application's calendar hierarchy, not the first date included in the workbook.

Output

Date: First date in the current application's calendar hierarchy.

Example

```
targetDate = calendarStart()
```

dateDiff

This function requires three inputs and generates one output. It calculates the difference of two date values. It returns the difference as an integer value as number of days, months, and so on, depending on a third string type input that specifies the scale of the output as a dimension name.

The dateDiff() function has a restriction that it can only calculate using dates loaded in the CLND hierarchy. If either the start or the end date is outside of CLND range, the function returns 0 (the same as if the start date = the end date).

Input

- Date: First date.
- Date: Second date.
- String: Dimension name for the scale of the diff to be calculated, such as DAY, MNTH, and so on.

Output

Integer: Number of periods calculated by firstDate - secondDate, in the scale of the dimension name provided.

Example

```
targetDate = dateDiff(date1, date2, "MNTH")
```

Note:

If date1 is Jan/01/2012, and date2 is Jan/01/2011, the resulting targetDate is 12, since the two dates are 12 months apart.

Date

This function requires two inputs and produces one output. It produces a date value based on an input date as a string, and a formatting string.

Input

- String: Input date string
- String: Date formatting string

The date formatting string follows the format of %[variable]%{variable]%[variable]. The specific options are as follows:

- B: month, full name
- h: month, 3-character abbreviation, such as JAN, FEB, MAR
- Y: 4-digit year
- y: 2-digit year
- m: 2-digit month
- d: 2-digit day
- H: 2-digit hour
- M: 2-digit minute
- S: 2-digit second
- s: 3-digit millisecond

For instance, if the format string evaluates to %Y%m%d and the date string is 20120102, the date is January 02, 2012. If the format string is %Y%h%d%H%M%S and the date string is 2012JAN02073030, the time and date is 7:30:30am on January 02, 2012.

Output

Date: Date value by parsing the input date string using the input date format.

Example

```
targetDate = date(dateStr, formatStr)
```

Index and Position Functions

This is a class of general functions that may be used for any hierarchy that enables reference to positions in a generic manner. In most cases, the functions do not generate results that are useful in themselves, but they are typically used as parameters that are passed into other functions.

An index is an internal reference to a position in a dimension. For dimensions in the calendar hierarchy, the index reflects an ordering of positions because there is a well-defined sequence (oldest to newest, based on the start and end dates) of periods. There are special calendar index functions that exploit this property. For other dimensions, there is no such ordering, and the index number can be considered random.

Note:

Index numbers (including calendar index numbers) should not be saved and reused between planning sessions, as there is no guarantee that the same index numbers will apply in subsequent sessions since the positions or relationships in a hierarchy may change.

These general index functions may be used for any hierarchy, including the calendar hierarchy.

index

This returns the index number of the specified position in the specified dimension of the specified hierarchy.

Syntax

```
index([<hierarchy>].{[<dimension>] | current}[,{ <stringexpr> | <dateexpr>}])
```

Where `<hierarchy>` is the name of a valid hierarchy, and <dimension> is the name of a valid dimension in that hierarchy. current is a keyword that returns the current dimension in `<hierarchy>`. If `<hierarchy>` is not a valid hierarchy or `<dimension>` is not a valid dimension in that hierarchy, an error is generated.

`<stringexpr>` and `<dateexpr>` are optional expressions that can be used to specify a position. If neither `<stringexpr>` nor `<dateexpr>` are specified, the function returns the index number of the current position of the dimension being evaluated. `<stringexpr>` is a string expression that results in a position name. If the result of `<stringexpr>` is not a valid position name in the dimension being evaluated, an error is generated. `<dateexpr>` is a numeric expression that results in a date type value and can only be used if `<hierarchy>` is the calendar hierarchy. If the result of `<dateexpr>` is not a date type value, or the result is returned when evaluating a dimension that is not in the calendar hierarchy, an error is generated.

The function returns the index number of the indicated position in the specified dimension of the specified hierarchy. When used with dates, the indicated position is the position that contains the date specified.

Inverse

The `index` function does not have an inverse.

Examples

- index([prod].[item], likeitem)
- This returns the index number of the string position in the item dimension referenced in the likeitem measure.
- index([prod].[cls], cls123)

- This returns the index number of the class cls123.
- index([clnd].[mnth], opendate)
- This returns the index number of the month that contains the date that results from the opendate measure.

position

This returns the position name of the position in the specified dimension of the specified hierarchy with the supplied index number. The returned string is in upper case.

Syntax

```
position([<hierarchy>].{[<dimension>] | current}[, <indexexpression>])
```

`<hierarchy>` must be the name of a valid hierarchy. If specified, <dimension> must be the name of a valid dimension in that hierarchy. current is a keyword that returns the current dimension in `<hierarchy>`. If `<hierarchy>` is not a valid hierarchy or <dimension> is not a valid dimension in that hierarchy, an error is generated.

`<indexexpression>` is an optional parameter to specify the index of the position to be evaluated. If `<indexexpression>` is not specified, the current position is assumed. The expression must be a valid expression that results in a numeric measure. The integers of the resulting values of the expression are used as the index numbers to determine the position to be evaluated. If `<indexexpression>` does not return a valid index number for the specified dimension an error is generated.

The function returns an uppercase string that is the position name of the position with the specified index number for the specified dimension of the specified hierarchy.

Inverse

The position function does not have an inverse.

Examples

- position([prod].[item], 3)
- This returns the position name of the item with index number =3.
- position([prod].[item], likeindex)
- This returns the position name of the item with the index number in the measure likeindex.
- position([prod].current)
- This returns the position name of the current position of the current dimension in the product hierarchy.

attribute

This returns the value of the specified attribute for the current position, or the position with the supplied index number.

Syntax

```
attribute(<attribute>, [<hierarchy>].{[<dimension>] | current}[,
<indexexpression>])
```

Where `<attribute>` is a valid attribute for the dimension to be used, otherwise an error is generated. `<hierarchy>` must be the name of a valid hierarchy. If specified, <dimension> must be the name of a valid dimension in that hierarchy. current is a keyword that returns the current dimension in `<hierarchy>`. If `<hierarchy>` is not a valid hierarchy or `<dimension>` is not a valid dimension in that hierarchy, an error is generated.

`<indexexpression>` is an optional parameter to specify the index of the position to be evaluated. If `<indexexpression>` is not specified, the current position is assumed. The expression must be a valid expression that results in a numeric measure. The integers of the resulting values of the expression are used as the index numbers to determine the position to be evaluated. If `<expression>` does not return a valid index number for the specified dimension an error is generated.

Valid values for `<attribute>` for all non-measure dimensions include the following, which must be specified using quotes:

- "label" – The label (description) for the position. This value must be specified using quotes. The attribute function requires left-hand side measure to be a string measure. All keywords which need to be passed to a function must be wrapped in double quotes. Any other syntax will throw an error.
- "dpmstatus" – The DPM status of the position. This attribute function required left-hand side measure to be a Boolean measure. True value corresponds to an informal status. A False value corresponds to a formal status.

The function returns the value of the specified attribute for the specified position.

Inverse

The `attribute` function does not have an inverse.

Examples

- attribute("label", [prod].current)
- This returns the value of the label attribute for the current position of the current dimension in the product hierarchy.
- attribute("dpmstatus", [prod].[item], likeindex)
- This returns the value of the dpmstatus attribute for the item with the index number in the measure likeindex (that is, the label for my like item).

Forecast Procedure

Using the RPASCE Configuration Tools, a time-series demand forecast may be configured as part of a planning workflow or business process. The Forecast procedure provides only a small subset of the functionality that is available through RDF. The differences between these solution extensions are as follows:

- The forecast produced by the Forecast procedure is a single-level forecast.
- RDF allows for forecasts to be generated at aggregate levels in the data (to remove sparsity), and then this forecast is spread down to the execution level by using a profile.
- The Forecast procedure allows for a single forecasting method to be specified in the calculation of the forecast.
- RDF allows for forecasting methods and forecasting parameters to be modified as needed at all levels in your data.

- No standard approval process of the resulting forecasts is included as part of the Forecast procedure.
- RDF allows for forecast adjustments and approvals to be made at the lowest level necessary in your data.

The Forecast Procedure Syntax section contains the specifications and syntax for configuring the Forecast procedure.

Forecast Requirements

The following libraries must be registered in any PDS that will use the Forecast solution extension:

- AppFunctions
- RdfFunctions

Using the Forecast Procedure

The following notes are intended to serve as a guide for configuring the Forecast procedure within the RPASCE Configuration Tools.

- Refer to the appropriate input parameters and output measures when using the Forecast procedure.
- The resultant measure (that is, the forecast output) should be at the same intersection as your history measure (that is, pos). This will be the base intersection of the final level.
- The Forecast procedure is a multiple result procedure, meaning that it can return multiple results with one procedure call within a rule. In order to get multiple results, the resultant measures must be configured in the Measure Tool and the specific measure label must be used on the left-hand side (LHS) of the procedure call. The resultant measure parameters must be comma-separated in the procedural call.

Syntax Conventions

Table D-2 displays the syntax conventions used in this procedure.

Table D-2 Syntax Conventions

Indicator	Definition
[…]	All options listed in brackets are optional.
{…|…}	Options listed in “{}" with “|" separators are mutually exclusive (either/or).
{…,…}	Options listed in “{}" with “," separators way are a complete set.
Bold	Bold indicates Labels.
Italics	Italics indicate a temporary placeholder for a constant or a measure.
Italics/meas	This indicates that the placeholder can be either a constant or a measure.
BoldItailics	This indicates a numeric placeholder for the dynamic portion of a label. Usually a number from 1 to N.

Table D-2 (Cont.) Syntax Conventions

Indicator	Definition
Normal	Normal text signifies required information.
Underlined	This convention is used to identify the function or procedure name.

Forecast Procedure Syntax

This shows the syntax for using the Forecast procedure with a simplified syntax version of the Forecast procedure. For the complete syntax version, refer to the *Oracle Retail Demand Forecasting Cloud Service Administration Guide*. The input and output parameter tables explain the specific usage of the parameter's names use in the procedure.

Simplified Syntax Version of the Forecast Procedure

Generic Example:

```
FORECAST: FORMEAS ,
CHMETHOD:METHMEAS<-BaselineForecast(MASK:MEASKMEAS, {STARTDATE:STARTDATE |
STARTDATEMEAS:STARTDATEMEAS}, HISTORY: HISTORYMEAS,
FORECASTLENGTH:FORECASTLENGTH, PERIOD:PERIOD ,{FRCSTSTARTMEAS:FRCSTSTARTMEAS
| FRCSTSTART:FRCSTSTART}, PLAN:PLAN, PROFILE:PROFILE,
BAYESIAN_HORIZ:BAYESIAN_HORIZ, {VALID_DD:VALID_DD, DDPROFILE:DDPROFILE })
```

Sample:

```
forecast:frcstout,cumint:cumintout,int:intout<-BaselineForecast(forecastlength
:12,history:pos,mask:frcstmask,period:26,startdatemeas:todaymeas)
```

Configuration Parameters and Rules

Table D-3 provides the input parameters for the Forecast procedure.

Table D-3 Forecast Procedure Input Parameters

Parameter Name	Description
FORECASTLENGTH	The length of the forecast. Data Type: Integer Multiple Allowed: No Required: Yes
HISTORY	The input measure the forecast is based on. Data Type: Real Multiple Allowed: No Required: Yes
MASK	Array that identifies what forecast method is used for each time series. Refer to Forecast Model/Model List table. Data Type: Boolean Multiple Allowed: No Required: Yes

Table D-3 (Cont.) Forecast Procedure Input Parameters

Parameter Name	Description
MAXALPHA	The maximum alpha value. Data Type: Real Multiple Allowed: No Required: No
PERIOD	The forecasting period for calculating seasonal coefficients. Data Type: Integer Multiple Allowed: No Required: Yes
PLAN	The Plan measure. Data Type: Real Multiple Allowed: No Required: No
PROFILE	The Seasonal Profile measure. Data Type: Real Multiple Allowed: No Required: No
STARTDATE/ STARTDATEMEAS	The forecast start date. Either STARTDATE or STARTDATEMEAS is required. STARTDATEMEAS, if used, must be a scalar for AutoES method. Data Type: STARTDATE - Date as a string. Data Type: STARTDATEMEAS – Date as measure. Multiple Allowed: No Required: Yes
VALID_DD	The maximum non-zero history to use de-seasonalized demand value for seasonal profile-based forecasting. Data Type: Integer Multiple Allowed: No Required: No
DDPROFILE	De-seasonalized demand measure. Used only for profile-based forecasting. Data Type: Double Multiple Allowed: No Required: No

Table D-4 provides the output parameters for the Forecast procedure.

Table D-4 Forecast Procedure Output Parameters

Parameter Name	Description
CHMETHOD	Selected method. Refer to Forecast Model/Model List table. Data Type: Integer Multiple Allowed: No Required: No

Table D-4 (Cont.) Forecast Procedure Output Parameters

Parameter Name	Description
FORECAST	Forecast output. Data Type: Real Multiple Allowed: No Required: Yes
PEAKS	Peaks, which are used for calculating baseline of the forecast. Data Type: Real Multiple Allowed: No Required: No

Forecast Method/Model List

Table D-5 provides the numeric value assigned to the forecast model/model list.

Table D-5 Forecast Procedure Output Parameters

Model	Numeric Value
AUTO ES	1
SIMPLE	2
HOLT	3
WINTERS	4
CASUAL	5
AVERAGE	6
NO FORECAST	7
COPY	8
CROSTON	9
M. WINTERS	10
A. WINTERS	11
SIMPLE CROSTON	12
BAYESIAN	13
LOADPLAN	14
PROFILE	15

Tensorflow Procedure

This section details the AppTensorflowExpr (Tensorflow procedure for generating results by loading and calling trained Tensorflow model with input features measure).

Tensorflow Parameter/Model Dependencies

The models should be trained at right version of Tensorflow version and should be already uploaded to the PDS.

Using the Tensorflow Procedure

The following bulleted items are intended to serve as a guide for configuring the Tensorflow procedure within the RPASCE Configuration Tools:

- Refer to the appropriate input parameters and output measures when using the Tensorflow procedure.
- The resultant measure (ML_OUTPUT) should be at the same intersection as RUN_MASK measure or with extra calendar dimension.
- If the output is a time series, then the ML_PREDICT_BEGIN and ML_PREDICT_END measures which specify the start date index and end date index, should be periodically updated (every week or so) by configuring rules. Otherwise, these two measures are not needed.
- The indicator to run the algorithm or not is specified using the RUN_MASK measure. This is a boolean measure and on its intersection, the Tensorflow models related information will be retrieved.

Forecast Procedure Syntax

The syntax for using the Tensorflow procedure is as follows:

```
ML_OUTPUT: FORMEAS {,ML_OUTPUT_MSG:RTNMSG}
<-AppTensorflowExpr(RUN_MASK:MEASKMEAS
ML_OPERATOR:OPERATORMEAS, ML_INPUT_PLACEHOLDER:INPUTMEAS,ML_CHKPT_PATH: PATHMEAS,
{, ML_PREDICT_BEGIN:STARTINDEX, ML_PREDICT_END:ENDINDEX,DATA_HIER:"HIERNAME"},
ML_FEATURE1:FEATURE1MEAS,{ ML_FEATUREN:FEATURENMEAS })
```

Configuration Parameters and Rules

Table D-6 and Table D-7 explain the specific usage of the parameters names used in the procedure.

Table D-6 Input Parameters for the Tensorflow Procedure

Parameter Name	Description
RUN_MASK	The run mask Data Type: Boolean Multiple Allowed: No Required: Yes
ML_OPERATOR	The model output tensor name Data Type: String Multiple Allowed: No Required: Yes
ML_INPUT_PLACEHOLDE R	The model input tensor name. Data Type: String Multiple Allowed: No Required: Yes

Table D-6 (Cont.) Input Parameters for the Tensorflow Procedure

Parameter Name	Description
ML_CHKPT_PATH	The Tensorflow model path Data Type: String Multiple Allowed: No Required: Yes
DATA_HIER	The hierarchy name along which the output will be. Usually it is CLND for the time series. Data Type: String Multiple Allowed: No Required: No
ML_PREDICT_BEGIN	The start date index. Only needed if the output is an array. Data Type: Integer Multiple Allowed: No Required: no
ML_PREDICT_END	The start date index. Only needed if the output is an array. Data Type: Integer Multiple Allowed: No Required: No
ML_FEATURE	The data feature input Data Type: Real Multiple Allowed: Yes Required: Yes

Table D-7 Output Parameters for the Tensorflow Procedure

Parameter Name	Description
ML_OUTPUT	Tensorflow procedure output. Data Type: Real Multiple Allowed: No Required: Yes
ML_OUTPUT_MSG	Tensorflow procedure return message. Data Type: String Multiple Allowed: No Required: no

Tensorflow Models Upload

Retailers should train their models with their own data and algorithm on the corresponding Tensorflow version that RPASCE supports. Once the models are trained and saved into binary files. Retailer should upload their models using the Manage Tensorflow OAT task or during the PDS build or patch.

Tensorflow Tensor Name

After training the models and saving the model into the Tensorflow binary files, retailer could run the utility **`saved_model_cli`** to find out the input and output name of the tensor node in the saved models. For the details on this command, refer to this website: `https://www.tensorflow.org/guide/saved_model`.

Tensorflow Feature Data

Most machine learning algorithms are expecting the feature data are within same range in order to product robust results. There are different methods to normalize the features data to some specific range. No matter which method retailer choose, retailer should make sure the same normalization methods should apply to feature data in both training and forecasting. In other words, the ML_FEATURE measures should be normalized with same method as the feature data which is used to train the uploaded models. Also, the order of the feature in this special expression should be the same during the training phase. Example, if price is the second feature in training phase, the price will be the second feature (ML_FEATURE2) in this special expression.

Time Series Functions

This is a collection of very similar functions to perform typical calculation tasks over a range of cells in one or more time series. The `<start>` and `<end>` positions, defined using calendar index numbers, specifies the range of cells to be used. Typically, there may be some arithmetic performed to calculate the start and/or end positions.

Note:

By using the `indexdate` or `index` functions to provide calendar index numbers, the `<start>` and `<end>` positions to be used in the time series can effectively be specified by position name or by date.

Note:

If the level modifier is used, the current keyword only has a value when the level used is higher than the level being evaluated (since, for example, the concept of the current week is ambiguous when evaluating a month, so an error is generated).

Single Time Series Functions

The following sections detail the Single Time Series functions.

tssum

Produces a sum of the cells in the time series for the measure defined by the start and end positions.

`tssum` is used for the following types of calculations:

- Season to date
- Balance to achieve
- 4 week moving sum

The function produces a sum of the cells in the time series for the positions implied by the `<start>` and `<end>` for the specified dimension.

Syntax

```
tssum(<expression>[, <start>[, <end>]])
```

Where `<expression>` is an expression or measure whose time series is to be used, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than `<start>`, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<expression>` is mandatory, the other parameters are optional. If `<start>` is not specified, the default value is first (that is, 0). If `<end>` is not specified, the default value is current.

Use the level modifier to specify a dimension in the calendar hierarchy when calculating higher or lower than the current calendar dimension.

Inverse

The tssum function does not have an inverse.

Examples

- tssum(PlanSales)
- This is a plan-to-date or running total value for sales.
- tssum(PlanSales, current, last)
- This provides a balance to achieve (that is, a sum from the current period to the end of the horizon).
- tssum(PlanSales.level([clnd].[week]), current – 3, current)
- This provides a 4-week moving total for sales.
- tssum(PlanSales, indexfirst([clnd].[qtr]))
- This provides a quarter to date running total (see the indexfirst function).

tsavg

The average (mean) value of the cells in the range.

The function produces an average of the cells in the time series for the positions implied by the `<start>` and `<end>` for the specified dimension.

Syntax

```
tsavg(<expression>[, <start>[, <end>]])
```

Where `<expression>` is an expression or measure whose time series is to be used, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than `<start>`, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<expression>` is mandatory; the other parameters are optional. If `<start>` is not specified, the default value is first (for instance, 0). If `<end>` is not specified, the default value is current.

Use the level modifier to specify a dimension in the calendar hierarchy when calculating higher or lower than the current calendar dimension.

Inverse

The `tsavg` function does not have an inverse.

tsmax

The maximum value of any cell in the range.

The function returns the maximum value of the cells in the time series for the positions implied by the `<start>` and `<end>` for the specified dimension.

Syntax

```
tsmax(<expression>[, <start>[, <end>]])
```

Where `<expression>` is an expression or measure whose time series is to be used, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than `<start>`, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<expression>` is mandatory; the other parameters are optional. If `<start>` is not specified, the default value is first (for instance, 0). If `<end>` is not specified, the default value is current.

Use the level modifier to specify a dimension in the calendar hierarchy when calculating higher or lower than the current calendar dimension.

Inverse

The tsmax function does not have an inverse.

tsmin

The minimum value of any cell in the range.

The function returns the minimum value of the cells in the time series for the positions implied by the `<start>` and `<end>` for the specified dimension.

Syntax

```
tsmin(<expression>[, <start>[, <end>]])
```

Where `<expression>` is an expression or measure whose time series is to be used, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than `<start>`, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<expression>` is mandatory; the other parameters are optional. If `<start>` is not specified, the default value is first (for instance, 0). If `<end>` is not specified, the default value is current.

Use the level modifier to specify a dimension in the calendar hierarchy when calculating higher or lower than the current calendar dimension.

Inverse

The `tsmin` function does not have an inverse.

tsmode

The modal value of the cells in the range.

The function returns the modal value of the cells in the time series for the positions implied by the `<start>` and `<end>` for the specified dimension.

Syntax

```
tsmode(<expression>[, <start>[, <end>]])
```

Where `<expression>` is an expression or measure whose time series is to be used, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than `<start>`, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<expression>` is mandatory; the other parameters are optional. If `<start>` is not specified, the default value is first (for instance, 0). If `<end>` is not specified, the default value is current.

If there is more than one value for the mode, then the function returns the first value that is calculated.

Use the level modifier to specify a dimension in the calendar hierarchy when calculating higher or lower than the current calendar dimension.

Inverse

The `tsmode` function does not have an inverse.

tsmedian

The median value of the cells in the range.

The function returns the median value of the cells in the time series for the positions implied by the `<start>` and `<end>` for the specified dimension.

Syntax

```
tsmedian(<expression>[, <start>[, <end>]])
```

Where `<expression>` is an expression or measure whose time series is to be used, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than `<start>`, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<expression>` is mandatory; the other parameters are optional. If `<start>` is not specified, the default value is first (for instance, 0). If `<end>` is not specified, the default value is current.

If there is no middle number, the function returns the average of the middle two numbers.

Use the level modifier to specify a dimension in the calendar hierarchy when calculating higher or lower than the current calendar dimension.

Inverse

The `tsmedian` function does not have an inverse.

tsstd

The standard deviation of the cells in the range.

The function returns the standard deviation of the cells in the time series for the positions implied by the `<start>` and `<end>` for the specified dimension.

Syntax

```
tsstd(<expression>[, <start>[, <end>]])
```

Where `<expression>` is an expression or measure whose time series is to be used, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than `<start>`, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<expression>` is mandatory; the other parameters are optional. If `<start>` is not specified, the default value is first (for instance, 0). If `<end>` is not specified, the default value is current.

Use the level modifier to specify a dimension in the calendar hierarchy when calculating higher or lower than the current calendar dimension.

Inverse

The `tsstd` function does not have an inverse.

tsvar

The variance of the cells in the range.

The function returns the variance of the cells in the time series for the positions implied by the `<start>` and `<end>` for the specified dimension.

Syntax

```
tsvar(<expression>[, <start>[, <end>]])
```

Where `<expression>` is an expression or measure whose time series is to be used, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than `<start>`, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<expression>` is mandatory; the other parameters are optional. If `<start>` is not specified, the default value is first (for instance, 0). If `<end>` is not specified, the default value is current.

Use the level modifier to specify a dimension in the calendar hierarchy when calculating higher or lower than the current calendar dimension.

Inverse

The `tsvar` function does not have an inverse.

Double Time Series (Statistical Error) Functions

The following sections detail the Double Time Series (Statistical Error) functions.

tsme

Produces the Mean Error of an estimate time series compared to an actuals time series.

Syntax

```
tsme(<x>, <y>[, <start>[, <end>]])
```

Where `<x>` is an expression or measure that represents the estimate and `<y>` is an expression or measure that represents the actuals, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than <start>, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated

`<x>` and `<y>` are mandatory, and the other parameters are optional. If `<start>` is not specified, the default value is first (that is, 0). If `<end>` is not specified, the default value is current. The Mean error is calculated using the following formula.

Figure D-1 Mean Error Formula

$$\frac{\sum_{i-0}^{n-1} \left| x_i - y_i \right|}{n}$$

Inverse

The tsme function does not have an inverse.

Examples

- tsme(FcstSales, ActSales)
- This calculates the Mean Error of the FcstSales measure from the start of the calendar horizon until the current time period.
- tsme(FcstSales, ActSales, first, elapsed)
- This calculates the Mean Error of the FcstSales measure from the start of the calendar horizon until the last time period with actuals loaded.
- tsme(FcstSales, ActSales, first, min(elapsed, current))
- This calculates the Mean Error of the FcstSales measure from the start of the calendar horizon until the first of the period being evaluated or the last time period with actuals loaded.

tsmae

Mean Absolute Error.

Syntax

```
tsmae(<x>, <y>[, <start>[, <end>]])
```

Where `<x>` is an expression or measure that represents the estimate and `<y>` is an expression or measure that represents the actuals, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than <start>, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<x>` and `<y>` are mandatory, the other parameters are optional. If `<start>` is not specified, the default value is first (that is, 0). If `<end>` is not specified, the default value is current.

The Mean Absolute error is calculated using the following formula.

Figure D-2   Mean Absolute Error Formula

$$\boxed{\frac{\sum_{i-0}^{n-1} \left| x_i - y_i \right|}{n}}$$

Inverse

The tsmae function does not have an inverse.

tsmape

Mean Absolute Percentage Error.

Syntax

```
tsmape(<x>, <y>[, <start>[, <end>]])
```

Where `<x>` is an expression or measure that represents the estimate and `<y>` is an expression or measure that represents the actuals, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than <start>, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<x>` and `<y>` are mandatory, and the other parameters are optional. If `<start>` is not specified, the default value is first (for instance, 0). If `<end>` is not specified, the default value is current. The Mean Absolute Percentage error is calculated using the following formula.

Figure D-3   Mean Absolute Percentage Error Formula

$$\frac{\displaystyle\sum_{i\in\{0\leq i<n\cap y_i\neq 0\}} \left|\frac{x_i - y_i}{y_i}\right|}{\displaystyle\sum_{i\in\{0\leq i<n\cap y_i\neq 0\}} 1}$$

Inverse

The `tsmape` function does not have an inverse.

tsrmse

Root Mean Square Error.

Syntax

```
tsrmse(<x>, <y>[, <start>[, <end>]])
```

Where <x> is an expression or measure that represents the estimate and <y> is an expression or measure that represents the actuals, and <start> and <end> are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of <start> or <end> are numeric, but non-integer, only the integer portion will be used. If <end> is less than <start>, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

<x> and <y> are mandatory, and the other parameters are optional. If <start> is not specified, the default value is first (that is, 0). If <end> is not specified, the default value is current. The Root Mean Square error is calculated using the following formula.

Figure D-4   Root Mean Square Error Formula

$$\sqrt{\frac{\sum_{i-0}^{n-1}(x_i - y_i)^2}{n}}$$

Inverse

The tsrmse function does not have an inverse.

tspae

Percentage Absolute Error.

Syntax

```
tspae(<x>, <y>[, <start>[, <end>]])
```

Where `<x>` is an expression or measure that represents the estimate and `<y>` is an expression or measure that represents the actuals, and `<start>` and `<end>` are expressions that calculate numbers. The current calendar dimension is assumed, and if the cell being evaluated does not have a calendar dimension, the bottom calendar dimension is assumed. If the values of `<start>` or `<end>` are numeric, but non-integer, only the integer portion will be used. If `<end>` is less than <start>, or either parameter is non-numeric or outside the scope of the calendar index numbers for the specified dimension, an error is generated.

`<x>` and `<y>` are mandatory, and the other parameters are optional. If `<start>` is not specified, the default value is first (that is, 0). If `<end>` is not specified, the default value is current. The Percentage Absolute error is calculated using the following formula.

Figure D-5   Percentage Absolute Error Formula

$$\frac{\sum_{i-0}^{n-1}\left|x_i - y_i\right|}{\sum_{i-0}^{n-1}\left|y_i\right|}$$

Inverse

The `tspae` function does not have an inverse.

Hierarchical Functions and Procedures

This is a collection of functions and procedures that provide some knowledge of hierarchical structures, and the how the current position fits in, or uses knowledge of hierarchical structures.

count

This returns the count of children at a specified level that belong to a parent at a higher level.

Syntax

```
count([<hierarchy>][.<childdimspec>[,<parentdimspec>]])
```

Where <childdimspec> is {[<childdimension>] | bottom | current}

and <parentdimspec> is [<hierarchy>].{[<parentdimension>] | top | current}

<hierarchy> is the name of a valid hierarchy (same hierarchy must be referenced throughout the function). <childdimension> and <parentdimension> must be valid dimensions in the specified hierarchy. If both are specified, then <childdimension> must be lower than <parentdimension> in a roll-up, or an error is generated. bottom, top, and current are keywords referring to the lowest, highest, and current (being evaluated) dimensions in the hierarchy. If <childdimspec> is not specified, the default is bottom. If <parentdimension> is not specified, the default is current.

The function returns the number of children in the dimension <childdimension> that are descended from the implied position (the current position or the ancestor of the current position at the specified level) in the dimension <parentdimension>. If the <childdimension> and the <parentdimension> are the same, the function returns the value of 1.

Inverse

The count function does not have an inverse.

Examples

- count([loc].bottom)
- Returns the number of children in the bottom dimension in the location hierarchy for the current position in the location hierarchy
- count([loc].[str])
- Returns the number of children in the store dimension (str) in the location hierarchy for the current position in the location hierarchy (that is, how many stores do I own)
- count([loc].[str], [loc].[area])
- Returns the number of children in the store dimension in the location hierarchy for the position in the dimension area that is the ancestor of the current position in the location hierarchy (that is, how many stores in my area).

lookup

This procedure returns the value of an expression for a specific intersection.

The positions to be looked up may be in one or more hierarchies. This procedure has the following special uses and restrictions:

lookup is a procedure and thus cannot be combined with functions and other procedures in any manner.

Used for history mapping and like SKU/sister store functionality.

The base intersection of the output measure must be the same as the input measure and/or one or more of the mapping measures.

Syntax

```
<output> <- lookup(<input>, <dimspec1> [, <dimspec2> ... , <dimspecn>])
```

Where <dimspec1-n> is [<hierarchy>].{[<dimension>] | bottom | current | top}, <map>

<output> is the measure being updated. <input> is the measure to be evaluated. Each <dimspec> is used to specify the hierarchy and dimension to be used in the mapping process and the measure that contains the mapping values.

For each <dimspec> that is specified, the <hierarchy> must be the name of a valid hierarchy and the <dimension> must be the name of a valid dimension in that hierarchy. top is a keyword that refers to the highest dimension in the hierarchy, bottom is a keyword that refers to the lowest dimension in the hierarchy, and current is a keyword that refers to the current dimension (that is, the dimension of the cell being evaluated).

<map> is either a measure or an explicitly stated position used to designate how positions in <input> are mapped to determine the resulting values in <output>. The output, input, and mapping measures used in the lookup procedure must conform in a certain manner. Specifically, the resulting measure <output> must have the same base intersection as <input>, <map>, or both measures so that all conform.

When <map> is a measure, it must result in either index numbers or position names, either of which must be valid index numbers or position names from the related dimension specification. When <map> contains index numbers that do not map to valid positions, an error is generated and the na value for <output> is returned. There is no special cycle breaking logic for the lookup procedure. This means that a measure may never be calculated from the lookup of the same measure.

Note:

`lookup` is a procedure so it cannot be combined with functions, modifiers, or other procedures in any manner. As a procedure, it requires a different syntax: “<-“ instead of “= when being assigned.

Inverse

The lookup procedure does not have an inverse.

Examples

- output <- lookup(input, [presentationstyle].[presentationstyle], map)
- Where output is at sku-week, input is at sku-presentationstyle, and map is at sku-week with position names or index numbers from the presentation style dimension. The output and mapping measures have the same base intersection. This expression calculates the output measure from the mapping of presentation styles that vary for each sku-week combination.

- output <- lookup(input, [prod].[sku], map)
- Where output and input are at sku-week and map is at sku with position names or index numbers from the sku dimension. This expression calculates the output measure from the like sku of the input measure.

Tablelookup

This procedure returns the value (interpolated if necessary) from the entry in a table of information held in measures that matches with supplied keys.

Syntax

```
tablelookup(<expression>, <matching technique>, <keymeasure> [,<resultmeasure>])
```

Where <matching technique> is {exactmatch, <nomatchvalue> | average | nearest | high | low | interpolate}

The tablelookup procedure requires that a table be available that may be the target of the lookup. This table will be formed from normal measures with a base intersection of normal dimensions. Nevertheless, the most usual usage will be where the table measures are dimensioned on a dimension built for the purpose, plus other dimensions as required.

Note:

The usage of the tablelookup special expression requires the arguments to be conformed in certain way. For example, in the tablelookup expression (valueMeasure, nearest, keyMeasure), it is required to address the following:

1. The keyMeasures baseint must have a table entry dimension, and this table entry dimension must be the innermost. Let's assume the keyMeasure is at clss/rgn/te level where dimension te is the table entry dimension, and it is the innermost. The subspace of the key entry is defined as the baseint of the key measure minus the table entry dimension; in this case, the subspace will be clss/rgn.
2. The valueMeasures baseint must be able to map to the subspace (clss/rgn) with a many to one mapping, that is, the baseint of the valueMeasure must be shows the clss/rgn. So it is a valid expression if the valueMeasures baseint is sku/str, clss/str, sku/rgn. If the valueMeasures baseint is the same or higher than clss/rgn, or cannot create a valid map to clss/rgn, the previous error message is triggered. For example, lower than baseint for valueMeasure triggers the error: clss/rgn, dept/rgn, clss/chn, dept/chn, dept/str, sku/chn.

   It is suggested to examine the baseint of the input measure with the previous criteria in mind.

Example

Imagine a requirement to look up valid price points that may be applied as prices for an item. The collection of valid price points will be different for each class. To satisfy this requirement, a table is built. A ‘table' hierarchy is defined with a tableentry dimension with a number of positions, which are named e01, e02, e03, … e99 to allow for 99 entries in the table, with the order of the positions being the same as their natural sort sequence, and the order of the hierarchy being the highest (innermost) non-time hierarchy. A measure named pp is defined with a base intersection of tableentry/class. The pp measure is populated with valid price points for each class, with the lowest valid price point in position e01, the next lowest in e02, and so on. This table can now be used to look up valid price points. The procedure call (indirectly) provides a class and (directly) provides a target price as arguments, and a valid price point is returned based on the selected matching technique. See the following examples for an example that uses this table.

<expression> is any valid expression that results in a value of the same data type as the <keymeasure>. In the description that follows, this value is referred to as the key value. <keymeasure> is the name of the measure to be used as a key when matching the key value against the table. <resultmeasure> is an optional measure that holds the return value. If <resultmeasure> is not specified, <keymeasure> is used for the values of the result as in the price point example.

The procedure attempts to match the key value against an entry in the table. The innermost non-time dimension in the base intersection of the <keymeasure> is assumed to be the dimension along which entries in the table are indexed. For all other dimensions in the base intersection of the <keymeasure>, the procedure will match against the parent at that dimension of the cell being evaluated.

Note:

The values in <keymeasure> must be in ascending order and must not contain any repeated values. A value that is either out of sequence or repeated designates that the previous value is the last entry in the table. In other words, only the sorted elements in the key measure will be considered in the lookup process.

The <matching technique> specifies the matching technique to be used when an exact match of the <keymeasure> against the key value is not found. If the matching technique is exactmatch, <nomatchvalue> is a numeric value that must be specified to indicate the value to use in cells when there is no exact match. Otherwise, if the key value is higher than the highest value in the “table," or lower than the lowest value in the table, it is assumed to match against the highest or lowest value accordingly. If the matching technique is high and no match against the key value is found, the procedure returns the value of the <resultmeasure> for the entry immediately higher than the key value. If the matching technique is low and no match against the key value is found, the procedure returns the value of the <resultmeasure> for the entry immediately lower than the key value. If the matching technique is nearest and no match against the key value is found, the procedure returns the value of the <resultmeasure> for the entry immediately lower than the key value or immediately higher than the key value, depending upon which entry is nearest (this is like rounding to the nearest value). If the matching technique is average and no match against the key value is found, the procedure returns the numeric average of the value of the <resultmeasure> for the entry immediately lower and immediately higher than the key value, or it generates an error if the <resultmeasure> is not of numeric data type.

If the matching technique is interpolate and no match against the key value is found, the procedure returns an interpolated value between the value of the <resultmeasure> for the entry immediately lower and immediately higher than the key value, or it generates an error if the <resultmeasure> is not of numeric data type. The interpolation is calculated as shown in Figure D-6.

Figure D-6 Interpolation Calculation

$$lowresult + \frac{(\,highresult - lowresult\,) * (\,keyvalue - lowvalue)}{(\,highvalue - lowvalue\,)}$$

Inverse

The tablelookup procedure does not have an inverse.

Examples

- tablelookup(tgtpr, nearest, pp)
- Returns the nearest valid value of the pp measure to the supplied target price (tgtpr).
- tablelookup(perc, interpolate, epct, elast)
- Looks up the percentage markdown (perc) of the current position against a percentage change elasticity table (epct). Returns the matching elasticity value (elast). If the percentage markdown is not found in the table, the procedure will interpolate the elasticity value from the nearest values higher or lower than the percentage markdown.

flookup

The fixed look up function returns the value of a measure for an explicitly named fixed intersection.

Syntax

```
flookup(<measure>, <posspec1> [, <posspec2> … , <posspecn>])
```

Where <posspec1-n> is: [<hierarchy>].[<dimension>].[<positionname>]

<measure> is the measure to be looked up. This <measure> must conform to the measure being calculated as follows. Some hierarchies may be present in the base intersection of both measures, and these are handled by normal non-conforming logic. For any hierarchies that are only in the base intersection of the measure being calculated (the output measure), all the positions will look up the same value. For any hierarchies that are only in the base intersection of the <measure> (input measure), the position to be used must be explicitly named through a position specification (<posspec>).

Note:

If the position to be used can only be specified indirectly (for example, if it is held in a measure), the flookup function cannot be used, and the more powerful lookup procedure should be used instead.

flookup can be used to return a constant or a slice. In case of a constant, the NA value of the flookup function will be the value of the constant. In case of a slice, the NA value of the flookup function will be the NA value of <measure>.

For each <posspec> that is specified, the <hierarchy> must be the name of a valid hierarchy, the <dimension> must be the name of a valid dimension in that hierarchy, and the <positionname> must be the name of a valid position in that dimension, If the position name includes special characters, it can be enclosed in quotes (" ") in addition to the standard requirement for square brackets ([ ]). If <hierarchy> is not a valid hierarchy or <dimension> is not a valid dimension in that hierarchy, or <positionname> is not a valid position in that dimension, an error is generated.

Additionally, <dimension> must be a dimension in the base intersection of <measure>. To use dimensions not in the base intersection, the <measure> must have a level modifier to explicitly raise it to the desired dimension.

There is no special cycle breaking logic for the flookup function. This means that a measure may never be calculated from the flookup of the same measure. The flookup function returns the value of the expression from the specified fixed intersection.

Inverse

The flookup function does not have an inverse.

Examples

- flookup(perc, [flvl].[flvl].[flvla])
- Returns the value for the measure perc for the position flvla in the flvl dimension of the flvl hierarchy.
- flookup(leadtime, [prod].[cls].[class1], [loc].[whse].[whseA])
- Returns the value for the measure leadtime for the class class1 for the warehouse whseA.

aggregate

The aggregate procedure provides similar functionality to the hybrid aggregation type. Measures that use the hybrid aggregation type cannot be manipulated higher than their base intersection (as there is no mechanism to spread changes), but since the aggregate procedure is used on recalc measures, they can be changed with the change being applied through normal mapping rules. In addition, the aggregate procedure has a recalc aggregation type that is not available in the hybrid aggregation method.

This procedure returns the value of a measure aggregated from the base intersection to the current level using the supplied aggregation type.

Syntax

aggregate (<cachemeasure>, <hierspec1> [, <hierspec2> … , <hierspecn>])

where <hierspec1-n> is [<hierarchy>].<aggtype>

The rule writer specifies a <cachemeasure> that holds the base intersection values to be aggregated, and it is also the source of values for recalc aggregation.

The rule writer also specifies the aggregation type for each hierarchy and the priority sequence to be used. The priority sequence is required because at levels that are aggregated in more than one hierarchy (for instance, Department/Region/Month for a measure with a base intersection of Class/Store/Week), different results would usually be obtained by aggregating up each of the hierarchies. For example, if the requirement is to aggregate up the product hierarchy by using the total aggregation type, up the location hierarchy by using the average aggregation type, and up the calendar hierarchy by using the first aggregation type. There are three potential ways to calculate a value at Department/Region/Month. We could total from Class/Region/Month, average from Department/Store/Month, or first from Department/Region/Week. These would almost certainly generate three completely different values. By providing a priority sequence, the rule writer explicitly determines which of these values are required. See the worked example.

Note:

The effect of a series of aggregations of the same type up a single hierarchy may return different results from those of a measure with the same aggregation type. ‘Normal' aggregation for a measure driven by its aggregation type is performed from all base intersection cells descended from the cell being evaluated. For example, for a measure with a base intersection of Class/Week and an ‘average' aggregation type, the value calculated for a cell at Department/Month is the average of all values for all Class/Week cells for the Department/Month. If the measure is a recalc measure, calculated at aggregated levels from a rule with an aggregate function, such as aggregate(x, [prod].average, [clnd].average), the value for the Department/Month will be the average of all the Class/Months (not Class/Weeks) that belong to the Department/Month. Other than coincidentally, this would generate a different value.

<cachemeasure> is the measure to be aggregated, and the value of <cachemeasure> is also the value used for cells that are at the base intersection (bottom levels), and at aggregated levels when the required aggregation type is recalc. <hierarchy> is the name of a valid hierarchy. Each hierarchy may only be specified once in the procedure, but hierarchies may appear in any order. The sequence that the hierarchies are specified in is used to determine which hierarchy to aggregate up if the cell being evaluated is at an aggregated level in more than one hierarchy. In this circumstance, aggregation is performed up the first specified hierarchy that the cell is at an aggregated level in, and the other hierarchies are ignored. <aggtype> specifies the aggregation type to be used. The <aggtype> must be one of the standard aggregation types. If any hierarchy that is in the scope of the measure being calculated is not explicitly specified, the aggregation type of that hierarchy is assumed to be total. Such hierarchies are assumed to be sequenced after all hierarchies that are explicitly referenced, and they are ordered from innermost to outermost.

Note:

The value of the <cachemeasure> is used at the base intersection of the measure being calculated. If, for a given cell, the aggregation type to be used is recalc, the value is also obtained directly from the <cachemeasure> at that level, which will normally have an aggregation type of recalc.

Note:

aggregate is a procedure so it cannot be combined with functions, modifiers, or other procedures in any manner. As a procedure it requires a different syntax: “<-“ instead of “=" when being assigned.

Inverse

The aggregate procedure does not have an inverse.

Examples

- result <- aggregate(x, [clnd].recalc)
- For cells at the base intersection, the value is calculated from the measure x. For cells at an aggregated level in the calendar hierarchy, the value is also obtained from the measure x, which we can assume has an aggregation type of recalc, and thus the result of the procedure is as if the aggregation type were recalc, using the usual expression to calculate measure x. If the cell is not at an aggregated level in the time hierarchy, and assuming in this example that the other hierarchies are product and location; in that priority, the value for a cell at an aggregated product level is calculated as the total of all cells for products descended from that product for the same location and time. Otherwise, the value for the cell is calculated as the total of all cells for locations descended from the cell's location for the same product and time.
- result <- aggregate(x, [loc].average)
- In a similar manner to the previous example, cells at aggregated levels in the location hierarchy will be calculated by averaging the values of cells for all descendent locations. Otherwise, the value will be totaled up the product or time hierarchy as appropriate.
- result <- aggregate(x, [clnd].average)
- Totals up all hierarchies except time, which uses an average aggregation type.
- result <- aggregate(x, [prod].average, [clnd].first)
- Averages up the product hierarchy if possible. Otherwise, takes the first child value up the calendar hierarchy. Otherwise, totals up the other hierarchies.
- result <- aggregate(x, [prod].average, [clnd].last)
- Averages up the product hierarchy if possible. Otherwise, takes the last child value up the calendar hierarchy. Otherwise, totals up the other hierarchies.

Multi-Level Calculation Example for Aggregate Procedure

Consider a measure calculated from the expression aggregate(x, [prod].total, [loc].avg, [clnd].first). The measure is assumed to have a base intersection of Class/Store/Week.

Examples

Examples of the calculations that would be applied at various levels are as follows:

- Class/Store/Month: first from Class/Store/Week
- Class/Region/Month: avg from Class/Store/Month
- Department/Region/Month: total from Class/Region/Month

Transform Procedures

RPASCE offers the following transformation procedures.

- transformSum
- transformMax
- transformOr
- transformProp

- transformEven
- transformRepl

Transform Procedure Requirements

The following libraries must be registered in any PDS that will use the transform procedures:

- Transform

Example

```
regfunction -d <pathToDomain> -l Transform -add
```

> **Note:**
>
> For all transformation procedures, use of the LABEL parameter in place of the POSNAME parameter when dealing with dimensions with large numbers of positions can adversely affect the performance and memory usage of the procedure. Therefore, use of POSNAME is preferable to use of LABEL when the business case could support either option.

transformSum

transformSum converts data across hierarchies using sum aggregation. The procedure converts data between measures of different dimensionality using a set of map measures to convert positions from the source measure to positions in the target measure. Source measures are aggregated into the target using the sum aggregation method.

Syntax

```
<target> <- transformSum(<source>, <transformspec1> [, <transformspec2> … ,
<transformspecn>])
```

Input Parameters

Table D-8 provides the input parameters for the transformSum procedure.

Table D-8   Input Parameters for transformSum

Parameter Name	Description
source	Measure that is being aggregated into <target> measure using the aggregation type of sum.

Table D-8 (Cont.) Input Parameters for transformSum

Parameter Name	Description
transformspec1-n	This parameters is [<source hierarchy>].[<source dimension>], [<target hierarchy>].[<target dimension>] , [LABEL|POSNAME], <map> The <transformspec> defines which dimension in the source is mapped to which dimension in the target and how the positions are mapped between the dimensions. The <map> measure may either be text or Boolean. If it is text, then the value of the cell contains the position id or label name of a position in the target dimension. The compulsory [LABEL|POSNAME] parameter specifies which method is used. If the <map> measure is Boolean, then its base intersection must include the <source dimension>; any true cells in the map measure will define the positions that are transformed to the target. If a label is not unique within a dimension and the LABEL option is used, then only the first position in the dimension that includes the label will be part of the transformation.

Table D-9 provides the output parameter for the transformSum procedure.

Table D-9 Output Parameters for transformSum

Parameter Name	Description
target	Measure into which the <source> measure is aggregated into using the aggregation type of sum.

Notes

If a hierarchy is in both the source and target measures, then the dimension for that hierarchy in the source and target must be the same, unless the transformation is defined through a mapping transformspec, meaning the source measure cannot have a base intersection of item if the target measure has a base intersection of class and there is no explicit transformation specified from item to class in transformspec.

If a dimension in the target is not in the source and is also not defined by a mapping, then transformation is applied to every position in that dimension.

transformSum only works for numeric measures. Text or Boolean measures will not get transformed.

If a cell in the source cannot be mapped to a position in the target, then it is ignored. The Transform procedure always writes a status message to rpas.log indicating how many cells were successfully transformed, how many cells failed and how many seconds the transformation took to execute.

The source measure and any map measure may be non-conforming. For instance, the source may be defined at month and the map defined at season.

Example

```
WpVRSlsR <-transformSum(WpSlsR, [LOC].[STR], [DVR].[VR], LABEL, WpRankTx)
```

Takes WpSlsR (store/class/month) and transforms it to WpVRSlsR (volume rank/class/month) using label mappings defined in WpRankTx (store/class/season).

transformMax

The transformMax procedure converts data across hierarchies using max aggregation. The procedure operates in the same way as transformSum, except that the aggregation method used is max.

Syntax

```
<target> <- transformMax(<source>, <transformspec1> [, <transformspec2> … ,
<transformspecn>])
```

Input Parameters

Table D-10 provides the input parameters for the transformMax procedure.

Table D-10 Input Parameters for transformMax

Parameter Name	Description
source	Measure that is being aggregated into <target> measure using the aggregation type of max.
transformspec1-n	This parameter is [<hierarchy>].[<dimension>], [<hierarchy>].[<dimension>] , [LABEL|POSNAME], <map>

Table D-11 provides the output parameter for the transformMax procedure.

Table D-11 Output Parameters for transformMax

Parameter Name	Description
target	Measure into which the <source> measure is aggregated into using the aggregation type of max.

Example

```
r_ut_out<-transformMax(r_ut_in, [prod].[sku], [clnd].[week], 0, r_ut_map)
```

Takes r_ut_in (sku/str/day) and transforms it to r_ut_out (sku/str/week) using label mappings defined in r_ut_map (sku/str). Here the maximum across all the days of a week is taken from r_ut_in and stored in the r_ut_out measure using label mappings defined in r_ut_map (sku/str).

transformOr

The transformOr procedure converts data across hierarchies using or aggregation. The procedure operates in the same way as transformSum, except that the aggregation method used is or. Both source and target measures must be Boolean measure types.

Syntax

```
<target> <- transformOr(<source>, <transformspec1> [, <transformspec2> … ,
<transformspecn>])
```

Input Parameters

Table D-12 provides the input parameters for the transformOr procedure.

Table D-12 Input Parameters for transformOr

Parameter Name	Description
Acsource	Measure that is being aggregated into <target> measure using the aggregation type of or. Must be a Boolean measure type.
transformspec1-n	This parameter is [<hierarchy>].[<dimension>], [<hierarchy>].[<dimension>] , [LABEL|POSNAME], <map>

Table D-13 provides the output parameter for the transformOr procedure.

Table D-13 Output Parameters for transformOr

Parameter Name	Description
Target	Measure into which the <source> measure is aggregated into using the aggregation type of or. Must be a Boolean measure type.

Example

```
r_ut_out<-transformOr(r_ut_in, [prod].[sku], [clnd].[week], 0, r_ut_map)
```

Takes r_ut_in (sku/str/day) and transforms it to r_ut_out (sku/str/week) using label mappings defined in r_ut_map (sku/str). Here the Boolean OR across all the days of a week is taken from r_ut_in and stored in the r_ut_out measure using label mappings defined in r_ut_map (sku/str).

transformProp

The transformProp procedure converts data across hierarchies using Proportional spreading. The procedure converts data between measures of different dimensionality using a set of map measures to convert positions from the source measure to positions in the target measure. While the transformSum procedure (and related aggregation procedures) assumes a many->one relationship as it performs the transformation (aggregation), the transformProp assumes a one-to-many relationship between the source and target cells (spreading).

Each source value is spread to a set of target values, leaving the ratio between the target values intact.

If the sum of all target cells is zero, then the source is spread evenly to the targets.

Syntax

```
<target> <- transformProp(<source>, <transformspec1> [, <transformspec2>
… , <transformspecn>])
```

Input Parameters

Table D-14 provides the input parameters for the transformProp procedure.

Table D-14 Input Parameters for transformProp

Parameter Name	Description
source	Measure that is being spread into <target> measure.

Table D-14 (Cont.) Input Parameters for transformProp

Parameter Name	Description
transformspec1-n	This parameter is [<hierarchy>].[<dimension>], [<hierarchy>].[<dimension>] , [LABEL|POSNAME], <map>

Table D-15 provides the output parameter for the transformProp procedure.

Table D-15 Output Parameters for transformProp

Parameter Name	Description
target	Measure into which the <source> measure is spread.

Note

If the <source> measure has a calendar dimension, then the r_elapsed measure has to have a value. (This is true for all TransformSpread flavors: transformProp, transformRepl, transformEven)

Example

```
mace -d . -run -expression "r_ut_out <- transformProp(r_ut_in, [clnd].[day],
[loc] .[str], 0, r_ut_map)
```

transformEven

The transformEven procedure converts data across hierarchies using Even spreading.

Syntax

```
<target> <- transformEven(<source>, <transformspec1> [, <transformspec2> … ,
<transformspecn>])
```

Input Parameters

Table D-16 provides the input parameters for the transformEven procedure.

Table D-16 Input Parameters for transformEven

Parameter Name	Description
source	Measure that is being spread into <target> measure.
transformspec1-n	This parameter is [<hierarchy>].[<dimension>], [<hierarchy>].[<dimension>] , [LABEL|POSNAME], <map>

Table D-17 provides the output parameter for the transformEven procedure.

Table D-17 Output Parameters for transformEven

Parameter Name	Description
target	Measure into which the <source> measure is spread.

Note

If the <source> measure has a calendar dimension, then the r_elapsed measure has to have a value. (This is true for all TransformSpread flavors: transformProp, transformRepl, transformEven)

Example

mace -d . -run -expression "r_ut_out <- transformEven(r_ut_in, [clnd].[day], [loc] .[str], 0, r_ut_map)"

transformRepl

The transformRepl procedure converts data across hierarchies using Replicate spreading.

Syntax

```
<target> <- transformRepl(<source>, <transformspec1> [, <transformspec2>
… , <transformspecn>])
```

Input Parameters

Table D-18 provides the input parameters for the transformRepl procedure.

Table D-18   Input Parameters for transformRepl

Parameter Name	Description
source	Measure that is being spread into <target> measure
transformspec1-n	This parameter is [<hierarchy>].[<dimension>], [<hierarchy>].[<dimension>] , [LABEL|POSNAME], <map>

Table D-19 provides the output parameter for the transformRepl procedure.

Table D-19   Output Parameters for transformRepl

Parameter Name	Description
target	Measure into which the <source> measure is spread.

Note

If the <source> measure has a calendar dimension, then the r_elapsed measure must have a value. (This is true for all TransformSpread flavors: transformProp, transformRepl, transformEven)

Example

```
mace -d . -run -expression "r_ut_out <- transformRepl(r_ut_in, [clnd].
[day], [loc] .[str], 0, r_ut_map)"
```

Normalization and Resizing Functions

This section provides details on the normalization and resizing functions.

resize

This uses the shape of a time series to produce another time series of a different length, but with the same shape.

Syntax

```
resize(<expression>, <start>, <fromlength>, <tolength>, <dst_start>)
```

Where <expression> is a measure or expression whose time series is to be used, and <start>, <fromlength> and <tolength> are expressions that calculate numbers. <start> is assumed to be a calendar index number; if its value is numeric but non-integer, only the integer portion will be used. If its value is a date type, the date value is converted to a calendar index internally. If <fromlength> or <tolength> are less than 0, or either parameter is non-numeric or when added to <start>-1 is outside the scope of the calendar index numbers for the dimension being calculated, an error is generated. If <fromlength> or <tolength> are non-integer, only the integer portion will be used. <dst_start> is an optional input. It can be a date type measure. If so, the date value is converted to a calendar index internally. It can also be a numeric value. If it's a numeric value, it represents the first calendar index that the output time series is written to. If omitted, it is always 0.

The function returns a time series that is resized such that the overall shape of the values is retained, but the number of time periods is stretched or shrunk from <fromlength> or <tolength>. For time periods outside the horizon covered by <start> and <start> -1 + <tolength> (if there are any), the function will return zero – if values other than this are required, or if no update to those periods is required, the function should be wrapped in an if function that can set the appropriate value or use the ignore clause, as appropriate.

The function stretches or shrinks the section of the time series by interpolation or decimation. The algorithm uses upsampling, convolution, and then downsizing. The filter used in convolution is a finite impulse response (FIR) lowpass filter, using a hamming window with cut-off frequency and length determined from greatest common denominator of the source and destination time series lengths.

The values generated for individual cells through this process are not normalized (for a similar function that normalizes the result, see the resizenorm function), and will be of similar magnitude to the cell values for the source cells.

Inverse

The resize function does not have an inverse.

Examples

- resize(profile, first, 10, 17)
- The first 17 periods of the result time series will have values with a shape the same as the first 10 periods of the measure profile. All other periods will be zero.
- resize(lag(profile,startweek), startweek, profilelength, numweeks)
- This example should be compared with the similar example of the normalize function. It uses a profile to generate a sales plan for an item for a specified length of time from a specified period of time. The profile is not necessarily the same length as the period for which sales are to be generated. The measure profile is assumed to have a profile (shape) for the sales of an item, starting in the first period with values for a number of periods given by the measure profilelength. startweek is an index number of the period from which sales should be generated for the item. numweeks has the length of the sales profile to be generated. Periods before the startweek or after the startweek-1+numweeks will have a result of zero. The periods from startweek to startweek-1+numweeks will have the result of the first profilelength weeks of the profile measure, stretched or shrunk to fit the appropriate number of periods.

resizenorm

This uses the shape of a time series to produce another time series of a different length, but with the same shape, normalized to a specific total.

Syntax

```
resizenorm(<expression>, <start>, <fromlength>, <tolength>[, <total>],
<dst_start>)
```

<total> is an expression that returns a numeric value. If <total> is not specified, it is assumed to be the sum of the cells of <expression> from startweek to startweek-1+fromlength. See the resize function for an explanation of the other parameters.

This function is identical to the resize function, except that the calculation engine automatically normalizes the resized values to the specified <total>.

Inverse

The resizenorm function does not have an inverse.

Examples

- resizenorm(profile, first, 10, 17)
- The first 17 periods of the result time series will have values with a shape the same as the first 10 periods of the measure profile. All other periods will be zero. The values of the cells will be such that sum of the 17 generated periods of the result time series will be the same as the first 10 periods of the measure profile.
- resizenorm(lag(profile,startweek), startweek, profilelength, numweeks, targetsales)
- This example should be compared with the similar example of the resize function. The generated sales will be normalized so that their sum is the value of the targetsales measure.
- resizeprofile

  The resizeprofile function is a rewrite of the resizenorm function. The resizeprofile function is intended as a functional replacement for the resize and resizenorm functions.

> **Note:**
>
> Users are encouraged to replace the expressions containing resize and resizenorm functions with resizeprofile. The resize and resizenorm functions are still maintained for backward compatibility.

Syntax

- resizeprofile(expression, start, fromlength, tolength, dststart: <dststart>, total: <total>)
- Where <expression> is a measure or expression whose time series is to be used, and <start>, <fromlength> and <tolength> are expressions that calculate numbers. <start> is assumed to be a calendar index number; if its value is numeric but non-integer, only the integer portion will be used. If its value is a date type, the date

value is converted to a calendar index internally. If <fromlength> or <tolength> are less than 0, or either parameter is non-numeric or when added to <start>-1 is outside the scope of the calendar index numbers for the dimension being calculated, an error is generated. If <fromlength> or <tolength> are non-integer, only the integer portion will be used. <dststart> is an optional input. It can be a date type measure. If so, the date value is converted to a calendar index internally. It can also be a numeric value. If it is a numeric value, it represents the first calendar index that the output time series is written to. If omitted, it is always 0.

- The function returns a time series that is resized such that the overall shape of the values is retained, but the number of time periods is stretched or shrunk from <fromlength> or <tolength>. For time periods outside the horizon covered by <start> and <start> -1 + <tolength> (if there are any), the function will return zero. If values other than this are required, or if no update to those periods is required, the function should be wrapped in an if function that can set the appropriate value or use the ignore clause, as appropriate.
- The function stretches or shrinks the section of the time series by interpolation or decimation. The algorithm uses upsampling, convolution, and then downsizing. The calculation engine automatically normalizes the resized values to the specified <total>.

Usage

The usage text of this function is as follows:

- resizeprofile(expression, start, fromlength, tolength, dst_start: <dst_start>, total: <total>) where:
  - expression: is a measure or expression, whose time series is to be used in the calculations. This is a required input.
  - start: denotes the index number of the calendar dimension of expression. The algorithm processes data points from this index number going forward. This is a required input.
  - fromlength: together with start, this input establishes which portion of the curve should be processed. fromlength determines the length of the original curve. This is a required input.
  - tolength: determines the length of the output curve. This is a required input.
  - dst_start: this is a named value pair denoting an optional input, which determines the index of the starting point of the output curve. If not specified, it defaults to zero.
  - total: this is a named value pair denoting an optional input whose numeric value is used to normalize the resized curve. If total is not specified, skip the Normalize step.

Examples

- "MeasureB=resizeProfile(MeasureA, 0, 10, 30, dststart:1)"
- resizeprofile will take the curve represented by the 10 positions starting at index 0 in MeasureA and resize that curve to fit the 30 positions starting at index 1 in MeasureB.
- "MeasureD=resizeProfile(MeasureA, 0, 10, 25, dststart:1, total:200)"
- resizeprofile will take the curve represented by the 10 positions starting at index 0 in MeasureA and resize that curve to fit the 25 positions starting at index 1 in MeasureD. While performing the resize operation, resizeProfile will normalize the curve to a total of 200 across all destination positions.
- "MeasureD=resizeProfile(MeasureC + MeasureA, 0, 15, 30, dststart:2, total:100)"
- resizeprofile will take the curve represented by the 15 positions starting at index 0 in the expression “MeasureC + MeasureA" and resize that curve to fit the 30 positions starting

at index 2 in MeasureD. While performing the resize operation, resizeProfile will normalize the curve to a total of 100 across all destination positions.

String Functions

This section provides details on the string functions.

uppercase

Converts a string to upper case.

Syntax

```
uppercase(<expression>)
```

Where the value of <expression> is returned as a string with upper case characters. This is useful in making string comparisons.

lowercase

Converts a string to lower case.

Syntax

```
lowercase(<expression>)
```

Where the value of <expression> is returned as a string with lower case characters. This is useful in making string comparisons.

textCompare

Performs a case-sensitive comparison of two strings.

Syntax

```
textCompare(<expression1>, <exprssion2>)
```

Where the two input arguments <expression1> and <expression2> must have a type of string. The result type is Boolean. The result value is a Boolean value of the case-sensitive comparison of the two RHS (right hand side) expressions.

The number of input arguments is 2. The number of output arguments is 1.

textConcat

Concatenates string arguments.

Syntax

```
textConcat(<expression1> ,<expression2>[,… ,<expressionN>])
```

Where the value of all expressions is either a string type measure or a string literal. Concatenates two or more values into a single string.

substr

This returns a portion of the input String measure's value.

Syntax

```
substr(<inputMeasure>[,<startIndex>,<length>])
```

Where <inputMeasure> must be a string type measure and the optional <startIndex> and <length> arguments are either integer measures or literal integer values. The substr function will take the portion of the input string beginning at <startIndex> (which defaults to 0 if not specified) and spanning the amount of characters specified by the <length> argument (which defaults to the length of the input string if not specified). If the end index of the copy (calculated as <startIndex> + <length>) is greater than the length of the input string, substr will pad the input string with spaces to make it long enough to be copied.

ConvertToString

The ConvertToString function will return the string representation of any non-string measure. The number and type of input arguments will change, depending on the type of first argument to the function, but it will always return string values.

The first argument is the input measure which contains the non-string values which needs to be converted to string. This is a mandatory input for ConvertToString.

Syntax - ConvertToString when input measure type is real

```
strMeas = ConvertToString(realMeas,precision:
<precision>,separator:<separator>,decimalmark:<decimalmark>)
```

If we pass in a real measure as first input argument to the ConvertToString function, then the output measure will contain the input measure's real numbers converted to string.

When the input measure is of type real the function will accept an additional optional argument of type integer which is the precision to be used on the input real values before they are converted to string. Precision is an optional integer constant or scalar integer measure. If precision is not specified a default precision value of 14 will be used. This is because, as a rule, the default value is set to be as precise as possible. This is consistent with current default used by RPAS CE, 14 is the internal default precision used by the RPASCE when converting a double to string by utilities like printArray.

Another optional argument is the decimal mark which is also of type string. Decimal mark can be a string constant or a scalar string measure. If the decimal mark string is not provided '.' will be used as the default decimal mark.

Syntax - ConvertToString when input measure type is integer

```
strMeas = ConvertToString(intMeas,separator:<separator>)
```

If we pass in an integer measure as first input argument to ConvertToString function, then the output measure will contain the input measure's integer numbers converted to string.

Syntax - ConvertToString when input measure type is date

```
strMeas = ConvertToString(dateMeas,dateFormat:<dateFormat>)
```

If a date measure is passed as first input argument to the ConvertToString function, then the output measure will contain the input measure's numeric date values converted to string. In addition, if the first input argument is a date type measure, the ConvertToString function will accept a second optional argument of type string that is the date format string. The second argument must be prefixed with the label dateformat followed by a colon, in the same way as the other labels used in RPASCE functions. For example, dateformat:%d%h%Y. The internal numeric dates in the input measure will be converted to string according to the format specified in the format string and stored in the output string measure. If second argument is not specified, the result string will be in the format "%d %h %Y.

The date format string is expected in the form %[variable]. Variable can be any of the following.

- B: month, full name
- h: month, 3-character abbreviation, such as JAN, FEB, MAR
- Y: 4-digit year
- y: 2-digit year
- m: 2-digit month
- d: 2-digit day
- H: 2-digit hour
- M: 2-digit minute
- S: 2-digit second
- s: 3-digit millisecond

Syntax - ConvertToString when input measure type is Boolean

```
strMeas = ConvertToString(boolMeas, trueString:<trueString>,
falseString:<falseString>)
```

If we pass in a Boolean measure as first input argument to the ConvertToString function, then the output string measure will contain the input measure's Boolean values converted to string. This function takes three additional string arguments. We will refer to the first two as trueString and falseString. These two strings are expected to be used in place of Boolean values true and false in the result. Each true cell value in the input measure will be stored as the trueString in the output string measure and each false cell value in the input measure will be stored as falseString in the output string measure. If the second and third arguments are not specified, then by default English strings true and false will be used.

Math Functions

This section provides details on the math functions.

pow

This returns the value of a number raised to the power of another number (x to the power of y).

Syntax

```
pow(<x>, <y>)
```

<x> and <y> are expressions that return real numbers. <y> designates the exponent to which <x> is raised.

exp

This returns the value of the transcendental number e raised to the power of a number (e to the power of x).

e is the base of natural logarithms.

Syntax

```
exp(<x>)
```

<x> is an expression that returns a real number to which the number e (value 2.71828183) is raised.

sqrt

This returns the square root of a number.

Syntax

```
sqrt(<x>)
```

<x> is an expression that returns a real number. This function returns the equivalent of pow(x, 0.5).

log

This returns the logarithm of a number.

This function returns the exponent that indicates the power to which a number is raised to produce a given number.

Syntax

```
log(<x>, [<base>])
```

Where <x> and <base> are expressions that return real numbers. If <base> is not specified, the default value is 10.

Examples

- log(100)
- The logarithm of 100 to the base 10 is 2.
- log(125, 5)
- The logarithm of 125 to the base 5 is 3.

ln

This returns the natural logarithm of a number.

This function returns the logarithm of <x> to base e (2.71828183).

Syntax

```
ln(<x>)
```

<x> is an expression that returns a real number.

mod

This returns the remainder as the result of the division of 2 numbers.

The result of this function is the remainder of <x> divided by <y>.

Syntax

```
mod(<x>, <y>)
```

<x> and <y> are expressions that return real numbers.

Example

- mod(5, 2)
- The remainder of 5 divided by 2 is 1.

abs

This returns the absolute value of a number.

Syntax

```
abs(<x>)
```

<x> is an expression that returns a real number.

rand

This is used to generate random values of type integer, real, and date. The return type of rand and the number and type of input arguments will depend on the type of the LHS measure.

In all cases of the rand function described in the following sections, low and high must be the first two parameters. They do not need to be in a fixed order; the function will select the upper value for high and the lower value for low. Low and high can be constants, scalar measures, or regular measures with a base intersection.

- Base intersection of the low and high measures can be at or higher than the base intersection of the LHS measure. If they are at a higher intersection the values will be spread down using the repl method before being used.
- When the upper limit is lower than the lower limit rand function will internally compare the limit values and generate a random value that falls between the two limit values. Also, low and high are included in the range of possible random values generated by rand. Therefore rand(1, 10) can generate 1 or 10 or anything in between, same as rand(10, 1). If low and high are measures, then rand generates values that fall between the two measures.
- Seed is an optional input into the random generation algorithm. Seed can be either an integer or a real number. The same seed will produce the same set of random values. Note that this is the behavior of random number generator algorithms in general and not specific to RPASCE. The seed argument may be more applicable in the testing/verification process where the same test random data can be generated multiple times using the same seed. When using the seed argument, the value needs to be preceded by the “seed:" label. The use cases have an example of proper usage.

Syntax - For real and integer type

```
resultMeas = rand(lowerLimit,upperLimit[,seed:<seed>])
```

When the LHS measure is an int or real type measure, the rand procedure generates random numbers that falls between the lower limit number (lowerLimit) and the upper limit number (upperLimit). For int or real type LHS measures lowerLimit and upperLimit are mandatory inputs.

If you pass in real numbers as lowerLimit or upperLimit and the LHS measure is an int type measure, then the real limit values will be rounded to the nearest integer before being used.

Syntax - For date type

```
dateMeas = rand(startDate,endDate[,dateformat:<datefomrat>][, seed:<seed>])
```

When the LHS measure is a date type measure, the rand procedure generates random dates that fall between startDate and endDate. startDate is the lower limit date for the range of dates within which random dates need to be generated and endDate is the upper limit of that range. When LHS measure is date type startDate and endDate are mandatory inputs. startDate and endDate can be constants, scalar measures or regular measures with a base intersection.

String type is allowed for startDate and endDate when they are constants or scalar measures. In that case date format string will be used to convert startDate and endDate to internal numeric date values.

If startDate and endDate are measures with base intersection, then they need to be of type date and cannot be of string type. In this case date format argument if supplied will be ignored.

dateformat is an optional input string that specifies the format of the startDate and endDate if they are strings. If dateformat string is not provided the default position format value in the PDS will be used. Internally this format can be found in be found in the diminfo array of the meta.db. Position format is usually specified only for the inner most dimension in the calendar hierarchy (usually day dimension). This argument must be preceded by the label dateformat.

The date format string is expected in the form %[variable]. Variable can be any of the following:

- B: month, full name
- h: month, 3-character abbreviation, such as JAN, FEB, MAR
- Y: 4-digit year
- y: 2-digit year
- m: 2-digit month
- d: 2-digit day
- H: 2-digit hour
- M: 2-digit minute
- S: 2-digit second
- s: 3-digit millisecond

If date format is being used Year, month and day are required. Time (hour, minute, second and millisecond) is optional. If time is not specified all 0's will be used for time which will be 00:00:00:000 using format H:M:S:s.

Other Functions and Procedures

This section provides details on other functions and procedures.

multisource

The multisource procedure calculates a workbook-only recalc measure, based upon the intersection that the measure is currently displayed at. The multisource procedure takes as arguments a list of measures at different base intersections. When evaluating the output measure, the intersection of the current worksheet is used to determine which of the right-hand side (RHS) measure's values must be assigned to the LHS measure.

Syntax

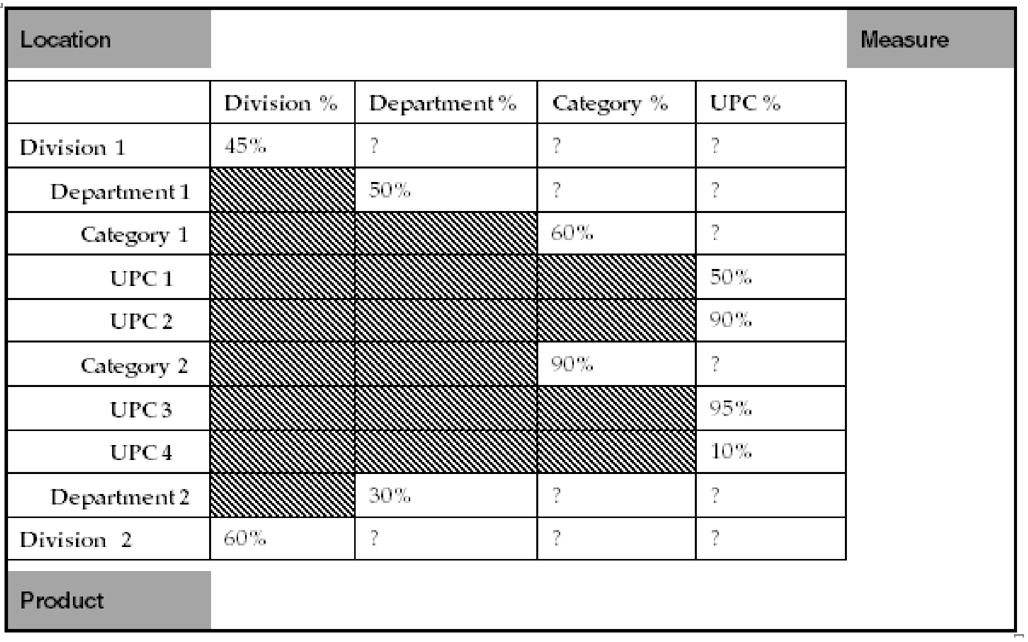
```
multisource(<inputMeas1>,<inputMeas2>…<inputMeasN>[,<performAggregation>])
```

Where <inputMeas1> through <inputMeasN> are measures at differing base intersections. When the multisource expression is evaluated, the input measure with the appropriate base intersection is used.

The <performAggregation> flag is a Boolean value (true or false). When this flag is set to True and there is no input measure at the evaluated intersection, then the closest measure will have its values aggregated based on that measure's aggregation method, and those values are used.

Multisource is useful when you need to load and display non-aggregated data at different intersections within a single measure. To achieve this without multisource measures, you must have multiple measures at each of the required intersections. This decreases usability since the user sees many different measures and many invalid cells. The following example shows the number of measures that would be needed to support just the different levels of the product hierarchy. This would be exploded out for each combination of intersections along the different hierarchies.

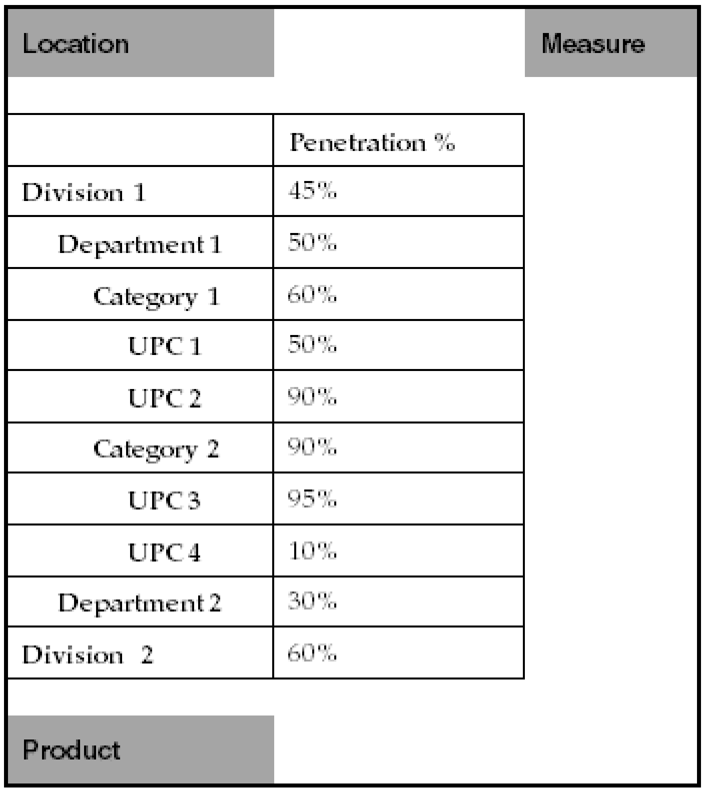
Figure D-7 Without Multisource Measures


Location
Measure
Product

	Division %	Department %	Category %	UPC %
Division 1	45%	?	?	?
Department 1	▨	50%	?	?
Category 1	▨	▨	60%	?
UPC 1	▨	▨	▨	50%
UPC 2	▨	▨	▨	90%
Category 2	▨	▨	90%	?
UPC 3	▨	▨	▨	95%
UPC 4	▨	▨	▨	10%
Department 2	▨	30%	?	?
Division 2	60%	?	?	?

However, with the use of multisource measures, the same requirements met by the previous example are achieved with a single measure, the result of which is shown in Figure D-8.

Figure D-8   With Multisource Measures


Location
Measure
Product

	Penetration %
Division 1	45%
Department 1	50%
Category 1	60%
UPC 1	50%
UPC 2	90%
Category 2	90%
UPC 3	95%
UPC 4	10%
Department 2	30%
Division 2	60%

Left Hand Side (LHS) Measure Properties

LHS measures must have the following attributes and properties:

- Base intersection: At or higher than the worksheet intersection like any other workbook measure.
- Aggregation Type: RECLC. (This is recalculated whenever the display intersection changes.)
- Spread Type: None (gets recalculated whenever the measure gets spread down)
- Materialized Type: Persistent. (This is persisted only in the workbook database. It cannot have a application database). LHS measures database property must not be specified.
- Base State: Read only
- Agg State: Read only
- Measure Type: Numeric, Date, String and Boolean
- The special expression must be part of a calc rule only. This validation check must be exclusively performed in the RPASCE Configuration Tools.

LHS Measure Restrictions and Validations

If the expression is triggered as part of a calc rule through workbook and any of the LHS measure properties are not met, then a marshallable exception is thrown by the server. An error message is displayed that describes the cause of exception. After closing the error message window, you can either close the workbook normally or continue working on the same workbook without performing any calculation. On the Configuration Tools side, a validation failure results in displaying the context of validation in red text and prevents the rpaInstall from building the PDS. The properties previous listed must be validated for the LHS measure on the RPAS CE Server and Configuration Tools.

Right Hand Side (RHS) Measure Properties

RHS measures must have the following attributes and properties:

- MeasureType: RHS measures inputMeas1, inputMeas2,inputMeas3 are of the same type as the type of the LHS measure.
- Base Intersection: RHS measures inputMeas1, inputMeas2,inputMeas3 must have different base intersections.
- Min and Max Number of RHS measures: The minimum number is 1 and maximum number is 200.
- Scalar measure: If LHS measure is a scalar, there can be only one RHS measure, and it must be a scalar measure. Since the number of RHS arguments can vary from 1 to 200, the condition when LHS measure is scalar can be met by providing a single RHS measure which is also a scalar.
- Since aggregation of RHS measures is allowed, the RHS measures base intersection can be at or higher or lower than the LHS measure base intersection. Whenever an exact match for the LHS measure display intersection is found on the RHS side, that measure gets used otherwise whichever RHS measure that has the nearest base intersection to the LHS measure's display intersection gets used after aggregation.
- RHS measures database property can have a value as they can be persisted in both PDS and workbook.
- The right most argument of RHS must be checked for a constant after the previous validation check. If it is a constant, then the remaining RHS arguments must be all measures. There can be only one optional constant in the expression and that must be the right most argument.

RHS Measure Restrictions and Validations

The conditions previously described must be validated for the RHS measures on the RPASCE Server and Configuration Tools. If the validation fails, a marshallable exception is thrown on the RPASCE Server side.

Rule Group Restrictions

The following validation checks must be exclusively performed in the Configuration Tools:

- The multisource expression can be used only in calc rule groups.
- LHS measure cannot appear on the RHS of any expression in any rule group.

cover

The cover function returns the number of future periods for which stock covers sales.

Alternately phrased, that is a forward weeks of supply, or the number of future periods of sales that could be satisfied from the stock with no further receipts.

The cover function allows for two sales expressions, where the second is a wrap around expression to provide a well-defined cover for periods at or near the end of the calendar horizon that would otherwise run out of forward sales. An offset is also specified to allow the cover function to behave appropriately for both opening and closing stock.

Unlike other functions, use of the level modifier is not supported in the stockexpression, salesexpression, offsetexpression, or wraparoundsalesexpression. Calculation at aggregate levels is possible, but the aggregates need to be computed first, and then the aggregated expressions can be used in cover/uncover.

Syntax

```
cover(<stockexpression>, <salesexpression>[, <offsetexpression>,
[<wraparoundsalesexpression>]])
```

Where <stockexpression> is an expression or measure that represents the ‘stock' <salesexpression> is an expression or measure that represents the sales. <offsetexpression> is an expression that calculates a number that represents the offset to apply. If the value is non-integer, only the integer portion is used. If the value is non-numeric, an error is generated. If <offsetexpression> is not provided, the default value will be 1. <wraparoundsalesexpression> is an expression or measure that represents the wrap around sales. If <wraparoundsalesexpression> is not provided, there will be no wraparound, and the function will generate an error if there is insufficient forward sales to calculate the cover.

The <salesexpression> can be considered to define a time series of sales data values, starting at the current period offset by the <offsetexpression>, and stretching until the end of the calendar horizon. If this time series is too short to evaluate the cover value, it can be considered extended by one or more copies of the time series implied by the <wraparoundsalesexpression>, if specified, from the start until the end of the calendar horizon. The cover value is calculated by summing down the time series until a sum is reached that is equal to or greater than the value of the <stockexpression>. If the sum is equal to the <stockexpression>, the number of periods used is returned. If the sum is greater than the <stockexpression>, the value returned is the number of periods used minus 1, plus the proportion of the last period reached that is required to exactly reach the value of the <stockexpression>. If the <offsetexpression> causes the start of the time series to be before the start of the calendar horizon, or no <wraparoundsalesexpression> is specified, and there is insufficient ‘forward sales' to determine the cover, an error is generated.

Inverse

The cover function has an inverse function, uncover. uncover returns the amount of stock that is required to give a specified number of forward periods cover. There is no inverse function that solves this relationship for sales (which is used as a time series, rather than a single value).

Note:

The inverse can only apply if the <stockexpression> is a single measure, rather than an expression.

Examples

- cover(EOP, Sales)

  This provides an EOP based forward cover. There is no wraparound sales expression, so this function will generate errors towards the end of the plan horizon.

- cover(BOP, Sales + MD, 0)

  This provides a BOP based forward cover, using Sales plus markdowns as the expression to be covered. There is no wraparound sales expression, so this function will probably generate errors towards the end of the plan horizon.

- cover(EOP, Sales, 1, Sales)

  This provides an EOP based forward cover. Sales itself is used as the wraparound sales expression (this is typical where the plan horizon is a year, since the Sales measure has the appropriate seasonality; where this is not the case, another measure, such as next season sales would be used) so this function will return reasonable values towards the end of the plan horizon when the cover is greater than the number of weeks remaining.

Note:

The cover function is always calculated at the current time dimension. For example, in a plan where the bottom time dimension is week, a measure with an aggregation type of recalc that is calculated from a cover function at the month level will calculate forward months of supply. If forward weeks of supply are required to be calculated for the month dimension, it would be more appropriate to specify the measure with an aggregation type of first or last, so that aggregation, rather than calculation through the rule, is used to generate the values at the month dimension.

Make sure that the wrap around expression, if used, is seeded with appropriate values.

Both the stock and the sales used in the cover function are expressions. This supports various business needs, such as using covers based on sales plus markdowns. If the stock is provided as an expression, rather than just a single measure, the function will not have an inverse.

The offset expression is used to define the offset: from which period to start using the sales expression. It is assumed to be an offset from the current period, so that a value of zero means that the sales for the current period must be used in evaluating the cover (which is appropriate for an opening stock based cover), and an offset of 1 means start in the period following the current period (which is appropriate for a closing stock based cover). Values other than 0 and 1 may be used.

uncover

The uncover function returns the amount of stock required to cover sales for the specified number of forward periods.

The uncover function allows for two sales expressions where the second is a wrap around expression that provides a well-defined cover for periods at or near the end of the calendar horizon that would otherwise run out of forward sales. An offset is also specified to allow the uncover function to behave appropriately for both opening and closing stock.

Unlike other functions, use of the level modifier is not supported in the stockexpression, salesexpression, offsetexpression, or wraparoundsalesexpression.

Calculation at aggregate levels is possible, but the aggregates need to be computed first, and then the aggregated expressions can be used in cover/uncover.

Syntax

```
uncover(<coverexpression>, <salesexpression>[, <offsetexpression>,
[<wraparoundsalesexpression>]])
```

Where <coverexpression> is an expression or measure that represents the cover value <salesexpression> is an expression or measure that represents the sales. <offsetexpression> is an expression that calculates a number that represents the offset to apply. If the value is non-integer, only the integer portion is used. If the value is non-numeric, an error is generated. If <offsetexpression> is not provided, the default value will be 1. <wraparoundsalesexpression> is an expression or measure that represents the wrap around sales. If <wraparoundsalesexpression> is not provided, there will be no wraparound, and the function will generate errors if there is insufficient forward sales to calculate the stock.

The <salesexpression> can be considered to define a time series of sales data values, starting at the current period offset by the <offsetexpression>, and stretching until the end of the calendar horizon. If this time series is too short to evaluate the stock value, it can be considered extended by one or more copies of the time series implied by the <wraparoundsalesexpression>, if specified, from the start until the end of the calendar horizon. The stock value is calculated by summing down the time series for some periods equal to the integer portion of the <coverexpression> and adding the value of the next period, multiplied by the fractional portion of the <coverexpression>. If the <offsetexpression> causes the start of the time series to be before the start of the calendar horizon, or no <wraparoundsalesexpression>, is specified, and there is insufficient forward sales to determine the stock, an error is generated.

Inverse

The uncover function has an inverse function, the cover function. This function returns the number of forward periods of cover implicit in the specified stock. There is no inverse function that solves this relationship for sales (which is used as a time series, rather than a single value).

Note:

The inverse can only apply if the <coverexpression> is a single measure, rather than an expression.

Examples

- uncover(WOS, Sales)

  This provides an EOP stock value that gives the specified weeks of supply. There is no wraparound sales expression, so this function will generate errors towards the end of the plan horizon.

- uncover(WOS, Sales + MD, 0)

  This provides a BOP stock value that gives the specified weeks of supply, using Sales plus markdowns as the expression to be covered. There is no wraparound sales expression, so unless the value of WOS is less than 1, this function will generate errors towards the end of the plan horizon.

- uncover(WOS, Sales, 1, Sales)

This provides an EOP stock value that gives the specified weeks of supply. Sales itself is used as the wraparound sales expression (this is typical where the plan horizon is a year, since the Sales measure has the appropriate seasonality; where this is not the case, another measure, such as next season sales would be used) so this function will return reasonable values towards the end of the plan horizon when the cover is greater than the number of weeks remaining.

Note:

The uncover function is always calculated at the current time dimension. For example, in a plan where the bottom time dimension is week, a rule or mapping rule that uses an uncover function at the month level will calculate the stock on the assumption that the <coverexpression> provides a forward months of supply. Make sure that the wrap around expression, if used, is seeded with appropriate values. Both the cover and the sales used in the uncover function are expressions. This supports various business needs, such as using covers based on sales plus markdowns. If the cover is provided as an expression, rather than just a measure, the function will not have an inverse. The offset expression is used to define the offset: from which period to start using the sales expression. It is assumed to be an offset from the current period, so that a value of zero means that the sales for the current period should be used in evaluating the cover (which is appropriate for an opening stock based cover), and an offset of 1 means start in the period following the current period (which is appropriate for a closing stock based cover). Values other than 0 and 1 may be used.

min

The min function returns the minimum value from a series of expressions or set of measures.

Syntax

```
min(<expression1>, <expression2> [, <expression3> … <expressionn>])
```

Where <expression1-n> are expressions or a set of measures (denoted by {<measureset>}), which return numeric values. The function returns the minimum value of the expressions.

Inverse

The min function does not have an inverse.

Example

- min (A, B, C)
- Returns the minimum of the measures A, B, and C.

max

The max function returns the maximum value from a series of expressions or set of measures.

Syntax

```
max(<expression1>, <expression2> [, <expression3> … <expressionn>])
```

Where <expression1-n> are expressions or a set of measures (denoted by {<measureset>}), which return numeric values. The function returns the maximum value of the expressions.

Inverse

The max function does not have an inverse.

Example

- max (A, B, C)
- Returns the maximum of the measures A, B, and C.

sum

The sum function returns the sum of a series of expressions or measure set.

Syntax

```
sum(<expression1>, <expression2> [, <expression3> … <expressionn>])
```

Where <expression1-n> are expressions or a set of measures (denoted by {<measureset>}), which return numeric values. The function returns the summed value of the expressions or measure set.

Inverse

The sum function does not have an inverse.

Example

- sum (A, B, C)
- Returns the sum of the measures A, B, and C.

lag

The lag function returns the value of an expression from the previous time period in the dimension being evaluated.

The lag function cannot be used in rule groups containing recalc measures.

Syntax

```
lag(<expression>)
```

Where <expression> is any valid expression. The function returns the value of the expression in the previous period. If the current period being evaluated is the first period in the calendar horizon (so that there is no previous period), an error is generated. For that reason, lag functions are usually embedded in if functions or prefer functions to check for that case.

Inverse

The lag function does not have an inverse.

Example

- lag(EOP)
- Returns the value of the measure EOP from the previous period.

Note:

The lag function is deliberately intended as a simple version of the timeshift procedure for one of the most frequently used cases, which is that the offset is one period in the past. Use the timeshift procedure for lagging with a variable offset. The lag function has special cycle breaking logic that enables a series of expressions to be calculated in a manner that allows them to be evaluated period wise. This allows an apparent deadly embrace to be broken. Thus the following two expressions are allowed, and can be calculated in the same rule group, even though EOP seems to depend on BOP, which seems to depend on EOP: EOP = BOP + Rec – SLs – MD BOP = lag(EOP) Note, however, that the cycle breaking logic does not support the measure being calculated being lagged on the RHS of the expression. Thus the following expression is not allowed: AccumSls = Sls + lag(AccumSls)

lead

The lead function returns the value of an expression from the next (following) time period in the dimension being evaluated.

The lead function cannot be used in rule groups containing recalc measures.

Syntax

```
lead(<expression>)
```

Where <expression> is any valid expression. The function returns the value of the expression in the following period. If the current period being evaluated is the last period in the calendar horizon (so that there is no following period), an error is generated. For that reason, lead functions are usually embedded in if functions or prefer functions to check for that case.

Inverse

The lead function does not have an inverse.

Example

- lead(BOP)
- Returns the value of the measure BOP from the next period.

Note:

The lead function is deliberately intended as a simple version of the timeshift procedure for one of the most frequently used cases, which is that the offset is one period in the future. Use the timeshift procedure for leading with a variable offset. In a similar manner to the lag function, the lead function has special cycle breaking logic that enables a series of expressions to be calculated in a manner that allows them to be evaluated period wise. This allows an apparent ‘deadly embrace' to be broken. Even when an error is generated because the current period is the last period in the calendar horizon, the lead function itself, if not guarded by if or prefer functions, returns the re-evaluated NA value of the measure. For example, for the following expression group: A = lead(B) B = A + 1 ...assume that the NA value for both A and B is 0. The system first re-evaluates B's NA value to be A's NA value + 1 = 1 based on the second expression. The system will attempt to retrieve the time period after B's last time period when A = lag(B) is evaluated. Because that time period does not exist, the lead function will return B's re-evaluated NA value instead, which is 1.

timeshift

The timeshift procedure that returns the value of a measure from a time period in the dimension being evaluated that is lagged by a designated number of periods.

This procedure has the following special uses and restrictions:

- Measures used in this procedure can be modified with the master modifier.
- Currently timeshift cannot be used in calculation or commit rule groups.
- Used for lagging the values of a measure by more than one period.
- Used for lagging/shifting PDS measure data in the PDS batch run.
- Used for retrieving values from time periods outside the scope of the workbook.
- Used for addressing 52-53-week year differences.

Syntax

```
<output> <- timeshift(<input>, {<lagvalue> | <lagmeas> | <lagmap>})
```

<input> is the measure that is being lagged and must have the same base intersection as <output> or must be forced to evaluate at the base intersection of <output> by using the level modifier. <input> must include a dimension in the calendar hierarchy and must be the same data type as <output>.

<lagvalue> is a scalar value that designates the number of periods each position in <input> is shifted. A negative value refers to shift forward in calendar dimension (lead), and a positive value refers to shift backward in calendar dimension (lag).

<lagmeas> is a numeric measure that contains values that determine how each position is shifted. <lagmeas> cannot have a calendar dimension and all non-calendar dimensions must be identical to <input>.

Note:

This implies that if either <input> or <lagmeas> measure is modified with the master modifier, the other measure must also be modified with the master modifier.

<lagmap> is a string measure used for sophisticated mappings. It must have the same calendar dimension as <input>. If <lagmap> has no calendar dimension, then the non-calendar dimension must be at same or aggregated level of <input>. The measure contains position names that indicate how each time period is mapped, and it must only contain positions from the dimension from the calendar hierarchy. The value of each position (called the source position here) in the lagmap measure is the name of position in the destination measure to which the data for that source position in the input is mapped. Multiple positions can be specified by separating them using a space. In other words, <lagmap> defines a mapping of positions from the input measure to the destination measure along time. Entries in <lagmap> that are not the names of valid positions in the dimension from the calendar hierarchy are ignored.

Note:

This implies that if either <input> or <lagmap> measure is modified with the master modifier, the other measure must also be modified with the master modifier.

This mapping technique is primarily used when lagging measures between 52 and 53-week years. When mapping multiple positions to a single position (such as mapping the last 2 weeks in a 53-week year to the last week in a 52-week year), the resulting value is the sum of the source values (that is, the sum of the last 2 weeks of the 53-week year). When mapping a single position to multiple positions (such as mapping the last week in a 52-week year to the last 2 weeks in a 53-week year), the source value is replicated to the resulting values (that is, weeks 52 and 53 in the 53-week year are updated to week 52 in the 52-week year).

Note:

`timeshift` is a procedure so it cannot be combined with functions, modifiers, or other procedures in any manner. As a procedure, it requires a different syntax: “<-“ instead of “=" when being assigned.

Inverse

The timeshift procedure does not have an inverse.

Examples

- `salesly <- timeshift(sales.master, -4)`

Updates the positions in the workbook measure for last year's sales with the values from the PDS measure sales where each position is advanced by four periods.

Table D-20 Input for salesly <- timeshift(sales.master, -4)

sales	chnl	scls	week	Value
-	Catalog	Loafer	w01_2007	10
-	Catalog	Loafer	w02_2007	20
-	Catalog	Loafer	w03_2007	30
-	Catalog	Loafer	w04_2007	40
-	Catalog	Loafer	w05_2007	0
-	Catalog	Loafer	w06_2007	0
-	Catalog	Loafer	w07_2007	0
-	Catalog	Loafer	w08_2007	0

Table D-21 Output for salesly <- timeshift(sales.master, -4)

salesly	chnl	scls	week	Value
-	Catalog	Loafer	w01_2007	0
-	Catalog	Loafer	w02_2007	0
-	Catalog	Loafer	w03_2007	0
-	Catalog	Loafer	w04_2007	0
-	Catalog	Loafer	w05_2007	10
-	Catalog	Loafer	w06_2007	20
-	Catalog	Loafer	w07_2007	30
-	Catalog	Loafer	w08_2007	40

- `salesly <- timeshift(sales.master, saleslag)`

  Where sales and salesly have a base intersection of SKU-week, the numeric measure saleslag contains a value for each SKU that indicates the number of periods to lag by SKU.

Table D-22 Input for salesly <- timeshift(sales.master, saleslag)

sales	chnl	scls	week	Value
-	Catalog	Loafer	w01_2007	10
-	Catalog	Loafer	w02_2007	20
-	Catalog	Loafer	w03_2007	30
-	Catalog	Loafer	w04_2007	40
-	Catalog	Loafer	w05_2007	0
-	Catalog	Loafer	w06_2007	0
-	Catalog	Loafer	w07_2007	0
-	Catalog	Loafer	w08_2007	0

Table D-22 (Cont.) Input for salesly <- timeshift(sales.master, saleslag)

sales	chnl	scls	week	Value
-	Catalog	Boots	w01_2007	10
-	Catalog	Boots	w02_2007	20
-	Catalog	Boots	w03_2007	30
-	Catalog	Boots	w04_2007	40
-	Catalog	Boots	w05_2007	0
-	Catalog	Boots	w06_2007	0
-	Catalog	Boots	w07_2007	0
-	Catalog	Boots	w08_2007	0
saleslag	chnl	scls	-	Value
-	Catalog	Loafer	-	2
-	Catalog	Boots	-	-2

Table D-23 Output for salesly <- timeshift(sales.master, saleslag)

salesly	chnl	scls	week	Value
-	Catalog	Loafer	w01_2007	30
-	Catalog	Loafer	w02_2007	40
-	Catalog	Loafer	w03_2007	0
-	Catalog	Loafer	w04_2007	0
-	Catalog	Loafer	w05_2007	0
-	Catalog	Loafer	w06_2007	0
-	Catalog	Loafer	w07_2007	0
-	Catalog	Loafer	w08_2007	0
-	Catalog	Boots	w01_2007	0
-	Catalog	Boots	w02_2007	0
-	Catalog	Boots	w03_2007	10
-	Catalog	Boots	w04_2007	20
-	Catalog	Boots	w05_2007	30
-	Catalog	Boots	w06_2007	40
-	Catalog	Boots	w07_2007	0
-	Catalog	Boots	w08_2007	0

- `salesly <- timeshift(sales.master, salesmap)`

  Where salesmap is a string measure that contains position names indicating which position in sales to use for each position in salesly; the position in salesly whose name is cell contents of salesmap receives the corresponding value from sales. In cases where the current year in the workbook contains 52 weeks, the previous year that is not in the workbook contains 53 weeks; the 52nd position in salesmap could contain the position names w52_2007 and w53_2007, causing the value of salesly to contain the sum of the two positions.

Table D-24 Input for salesly <- timeshift(sales.master, salesmap)

sales	chnl	scls	week	Value
-	Catalog	Loafer	w02_2007	20
-	Catalog	Loafer	w03_2007	30
-	Catalog	Loafer	w04_2007	40
-	Catalog	Loafer	w05_2007	0
-	Catalog	Loafer	w06_2007	0
-	Catalog	Loafer	w07_2007	0
-	Catalog	Loafer	w08_2007	0
salesmap	-	-	week	Value
-	-	v	w01_2007	w05_2007
-	-		w02_2007	w06_2007
-	-		w03_2007	w07_2007
-	-		w04_2007	w08_2007
-	-		w05_2007	w01_2007
-	-		w06_2007	w02_2007
-	-		w07_2007	w03_2007
-	-	-	w08_2007	w04_2007

Table D-25 Output for salesly <- timeshift(sales.master, salesmap)

salesly	chnl	scls	week	Value
-	Catalog	Loafer	w01_2007	0
-	Catalog	Loafer	w02_2007	0
-	Catalog	Loafer	w03_2007	0
-	Catalog	Loafer	w04_2007	0
-	Catalog	Loafer	w05_2007	10
-	Catalog	Loafer	w06_2007	20
-	Catalog	Loafer	w07_2007	30
-	Catalog	Loafer	w08_2007	40

round

This returns the value of an expression rounded up or down to the nearest multiple.

Syntax

```
round(<expression>[, <multipleexpression>])
```

Where <expression> is any valid expression, which specifies the value to be rounded, <multipleexpression> is an expression that calculates a number that represents the multiplier to use. If the value is not specified, it is assumed to be 1. The value may be non-integer. If the value of either expression is non-numeric, an error is generated. The rounding is up or down to the nearest multiple of the multiplier. If there are 2 multiples equally near to the value (for instance, rounding 1.5 to the nearest integer), then rounding is up (away from zero).

Note:

When the round function is called with the <multipleexpression> argument, round first determines the nearest integer to the input and then determines the nearest multiple of the <multipleexpression>. For instance, round(2.8, 2) first rounds 2.8 to 3 and then returns 4 as the final value. However, round(3.8, 3) first rounds 3.8 to 4 and then returns 3 as the final value.

Inverse

The round function does not have an inverse.

Examples

- round(qty)

  Returns the value of the measure qty, rounded up or down to the nearest integer. If the qty is 14.324, this returns the result of 14, if the qty is 14.824, this returns the result of 15.

- round(qty, packsize)

  Returns the value of the measure qty, rounded up or down to the nearest multiple of the pack size. If the qty is 14.324 and the packsize is 6, this returns the result of 12. If the qty is 16.824 and the packsize is 6, this returns the result of 18.

roundup

The roundup function returns the value of an expression rounded up to the nearest multiple.

Syntax

```
roundup(<expression>[, <multipleexpression>])
```

Where <expression> is any valid expression, which specifies the value to be rounded. <multipleexpression> is an expression that calculates a number that represents the multiplier to use. If the value is not specified, it is assumed to be 1. The value may be non-integer. If the value of either expression is non-numeric, an error is generated. Rounding is always up (to the nearest multiple of the multiplier further away from zero).

Inverse

The roundup function does not have an inverse

Examples

- roundup(qty)

  Returns the value of the measure qty, rounded up to the nearest integer. If the qty is 14.324 or 14.824, this returns the result of 15.

- roundup(qty, packsize)

  Returns the value of the measure qty, rounded up to the nearest multiple of the pack size. If the packsize is 6 and the qty is 14.324 or 16.824, this returns the result of 18.

rounddown

The rounddown function returns the value of an expression rounded down to the nearest multiple.

Syntax

```
rounddown(<expression>[, <multipleexpression>])
```

Where <expression> is any valid expression, which specifies the value to be rounded, <multipleexpression> is an expression that calculates a number that represents the multiplier to use. If the value is not specified, it is assumed to be 1. The value may be non-integer. If the value of either expression is non-numeric, an error is generated. Rounding is always down (to the nearest multiple of the multiplier closer to zero).

Inverse

The rounddown function does not have an inverse.

Examples

- rounddown(qty)

  Returns the value of the measure qty, rounded down to the nearest integer. If the qty is 14.324 or 14.824, this returns the result of 14.

- rounddown(qty, packsize)

  Returns the value of the measure qty, rounded down to the nearest multiple of the pack size. If the packsize is 6 and the qty is 14.324 or 16.824, this returns the result of 12.

Note:

The round functions have no inverses. Great care should be used in designing rule groups that use these functions, and the preferred technique for rounding is often to not round during calculation, but to round values on display only. The round functions cause problems because they can compromise the integrity of rule and expression relationships. Consider a typical relationship between value, units and price. If the units are calculated through a round function (on the apparently reasonable assumption that units should be integers) after a change to, say, the value, then the integrity of the rule relationships is immediately compromised because the price is no longer the value divided by the units.

navalue

The navalue function returns the NA value of the specified expression.

Syntax

```
navalue(<expression>)
```

<expression> can be a constant, a measure, or an expression.

The navalue function does not directly generate errors, but it can propagate errors generated by <expression>.

Inverse

The navalue function does not have an inverse.

Examples

- navalue(<meas>)

  This returns the NA value of <meas>.
- navalue(<meas1> + <meas2>)

  This returns the NA value of the expression <meas1> + <meas2>. In this example, if the NA value of <meas1> is 2 and the NA value of <meas2> is 5, the result of the navalue function will be 7.

propspread

propspread is a multiple result function that spreads a value across a collection of measures while retaining their relative proportions. The multiple results are not named and are therefore positional only. The typical usage of this function is to allow spreading of hierarchical measures.

Syntax

```
propspread(<totalexpression>, <childexp1>, … <childexpn>)
```

Where <totalexpression> is an expression that returns a numeric value to which to balance the results of the function, <childexp1> - <childexpn> are expressions that provide the shape of the results. They will typically be the same measures as those assigned to the result of the function but using the old modifier. A measure defined as a result cannot be used on the right-hand side without old.

The function generates n positional results, where n is the number of child expressions. The results will sum to the <totalexpression>, using the shapes of the child expressions in the order of the child expressions.

The number of results must be equal to the number of child expressions, which means that there should be one more argument on the right-hand side than output measures on the left-hand side. Additional child expressions are ignored. If too few child expressions are defined, the function will fail. Currently, there is no validation to warn when this condition occurs.

If the sum of the child expressions is zero, the spread will be even.

Inverse

The propspread function does not have an inverse.

Example

The old modifier can be used in conjunction with the propspread function to implement a hierarchical relationship among measures. In the following example, Total sales (TotalSls) is the “parent" measure and regular sales (RegSls), promotional sales (PromSls), and markdown sales (MkdSales) are the “child" measures. Using old and propspread to configure this relationship allows the manipulation of any combination of these measures before calculating, except for all of them.

In the following example and in other such hierarchical measure relationships, the order of the expressions within a rule is critical for the measures to be correctly calculated.

```
TotalSls = RegSls + PromoSls + MkdSls
RegSls, PromoSls, MkdSls = propspread(TotalSls, RegSls.old, PromoSls.old, MkdSls.old)
PromoSls, MkdSls = propspread(TotalSls - RegSls, PromoSls.old, MkdSls.old)
RegSls, MkdSls = propspread(TotalSls - PromoSls, RegSls.old, MkdSls.old)
RegSls, PromoSls = propspread(TotalSls - MkdSls, RegSls.old, PromoSls.old)
RegSls = TotalSls - PromoSls - MkdSls
PromoSls = TotalSls - RegSls - MkdSls
MkdSls = TotalSls - RegSls - PromoSls
```

passthrough

passthrough is a multiple result function that is used to encapsulate any number of normal computations into a single expression.

Note:

The passthrough function is not allowed for measures with a recalc agg type.

Syntax

```
passthrough(<exp1>, <exp2>, …, <exp-n>)
```

Where <exp1> - <exp-n> are normal expressions used to calculate the resulting measures.

All measures on the left-hand side must be computed at the same base intersection. The number of results must be less than or equal to the number of calculation expressions (additional calculation expressions are ignored). If too few calculation expressions are defined, then function will fail. Currently, there is no validation to warn an individual when this condition is met.

There are two main reasons for using this function:

- Use passthrough in an expression for a rule when computing values for multiple measures without having to write (develop) a multiple-result function or procedure.
- To improve performance. If many measures are computed using the same or similar set of RHS measures, combining those calculations using passthrough may be faster because there is less physical input/output with the data.

Inverse

The passthrough function does not have an inverse.

Examples

- A, B = passthrough(C + D, C - D)
- Computes the sum and difference of two measures simultaneously.
- SalesA, SalesB =passthrough(SalesA.old * TempMeas / TotalSales.old, SalesB.old * TempMeas / TotalSales.old)
- Proportionately spread TotalSales down to its components, SalesA and SalesB.

rankagg

This procedure is to use a numeric ranking to assign a value. When used in conjunction with the recalc aggregation type, rankagg will return the value associated with the highest/lowest rank cell as the aggregate value.

Syntax

The special expression is of the form `D <- rankagg(R, B [, S])`, where

On the left-hand side:

- D is a string type measure, with aggregation type recalc.

On the right-hand side:

- R is a numeric measure that contains rank information for each cell in the base intersection of S (if S is available), or each cell in the base intersection of D (if S is not available). This ranking can be generated as a result of calculation or can be loaded at workbook build time.
- B is a Boolean value (or a scalar Boolean measure). If B is TRUE, the procedure will select the highest rank value, otherwise, the lowest rank value.
- S is a source measure, which is optional. The rankagg procedure will assign the values of S to the LHS measure D, when evaluated at the base intersection of D. When evaluated at the aggregated intersection, S will provide the value returned by the procedure. When S is not provided, D will be used as both source and destination. At this time, the rankagg procedure will have no effect when evaluated at the base intersection of D. When evaluated at the aggregate level, the procedure will populate the aggregate array from base intersection of D based on the value of R, and then pass the value.

Example

Here is an example for the usage of rankagg procedure, D<-rankagg(R, B, S).

SKU-A, SKU-B, and SKU-C, are positions in a hierarchy that aggregate to SCLS-1.

In the procedure, the measure R is a numeric-valued measure. This is the measure from which the rankagg procedure will identify the maximum or minimum value, based on the Boolean measure B. A value of TRUE instructs the procedure to select the maximum value; FALSE corresponds to the minimum value. In the worksheet, R corresponds to the Sales value measure. For the purposes of this example, Sales value has an agg type of max, just to show what's going on at the SCLS-1 level.

The measure S represents a measure from which the procedure will draw the values that will eventually feed into the “Cell value measure on the worksheet. The “Cell value measure would be the expression's D (for display) measure. S (for source) in this example is a hidden measure. S contains values that could be specified by some calculation. The values in S could be, for example, strings that represent images, though they need not be so.

By evaluating the procedure, at the lowest (SKU/week) level, the rankagg procedure will just copy over the values of S to D. At higher levels, say SCLS-1/week, the rankagg procedure will select a representative value for the “Cell value measure s that is equal to the SKU-level value where the corresponding “Sales value measure (R) has its max (from B) value over the SKU positions.

Note:

In cases where there is more than one cell in the rank measure with the highest (or lowest, if lowest rank is specified) value, RPAS CE will select one of the tied cells to be the highest (or lowest) ranked position

In cases where the source data need not be calculated by an expression, then the rankagg procedure will also support a variant syntax. That is, the rule writer could specify simply D <- rankagg(R, B). In this case, D will be the source (at the base level) and the target (at the aggregate level).

ranksort

The ranksort procedure returns the rank of intersections given the rank order (ascending or descending), the measure to rank upon and the dimensions to rank over.

Syntax

```
<output> <- ranksort(<input>,<rank order> [,<dimspec1>, …, <dimspecn>])
```

<output> is the measure that will contain the ranking results. The result of ranking will always be an integer value, so the data type of <output> must be integer or numeric.

<input> is the measure to be ranked.

<rank order> is {ascending | descending}. ascending is a keyword meaning that the intersection will be ranked in ascending value of the <input> measure, and descending is a keyword meaning that the intersection will be ranked in descending value of the <input> measure. All keywords which need to be passed to a function must be wrapped in double quotes ( ). Any other syntax will throw an error.

<dimspec1-n> is [<hierarchy>].{[<dimension>] | top} <dimspec1-n> specifies the dimensions to rank over. For each <dimspec> that is specified, the <hierarchy> must be the name of a valid hierarchy and the <dimension> must be the name of a valid dimension in that hierarchy. top is a keyword that refers to the highest dimension (all) in the hierarchy. <dimspec1-n> is optional, and if omitted the value for each hierarchy in the base intersection of <output> will be [hierarchy].top. A <dimspec> for a hierarchy that is not in the base intersection of <output>, or that references a dimension that is not higher than (a parent/grandparent and so on, of) the dimension in that hierarchy in the base intersection of <output>, or which references a hierarchy that already has a <dimspec>, is an error.

The base intersection of <output> determines the intersections that will be ranked. If the base intersection of <input> is different to that of <output>, as will usually be the case, then the values of <input> used for ranking will be the values at the intersections implied by the base intersection of <output> obtained by normal non-conforming measure handling, with replication from higher dimensions and/or aggregation from lower dimensions.

The scope of the ranking is dictated by <dimspec1-n>. These will usually be implied rather than explicitly specified and be at the top of the hierarchy. However, when a dimension is specified, there will be a separate ranking for each position (or combination of positions where a dimension is specified in two or more hierarchies) in that dimension. Thus, for example, when evaluating a measure calculated from the ranksort procedure that has a base intersection of sku/week, where the <dimspecs> reference the dimensions class and season, in a workbook with 4 classes and 2 seasons, there will be eight sets of ranks, one per class/

season, and the value for each sku/week intersection will be the order of that sku/week within its class/season.

The ranking process sorts the intersections in ascending or descending value of <input>, as required, and the ranking number is the order that each position is after sorting. The intersection with the highest value of <input> (lowest when ranking ascending) will have a rank of 1, with subsequent intersections having a rank higher by one. Where two or more intersections have the same value of <input>, they will be given the same rank, but the next rank value will account for the number of intersections with identical rankings. For example, the first few rankings might be 1, 2, 3, 3, 5, 6, …

Note:

ranksort is a procedure and thus cannot be combined with functions, modifiers, or other procedures in any manner. As a procedure, it requires a different syntax: “<-“ instead of “=" when being assigned. The level modifier cannot be used on the LHS of an expression that uses the ranksort procedure. That is, the level at which the ranking is executed will always be determined by the base intersection of <output>. The ranksort procedure must, by its nature, calculate a rank value for every intersection within a scope, not just those that have changed values for measures on the right-hand side of the expression. In incremental calculation mode (for example, when planning online) this may cause longer than expected calculation times, especially when the measure calculated through the ranksort procedure is used on the right-hand side of other expressions, as those expressions, plus any knock-on effects will also have to be calculated for every intersection within the scope.

Examples

- **Rank <- ranksort(WpSlsR, descending)**

  If Rank has a base intersection of sku, the result of this procedure is the integer value representing where each SKU's Sales value ranks amongst all SKUs. The SKU with the highest WpSlsR value will have a rank of 1.

- **Rank <- ranksort(WpSlsU, descending, [prod].[clss])**

  If Rank has a base intersection of sku, the result of this procedure is the integer value representing where each SKU's Sales units ranks amongst all SKUs within its class. The SKU with the highest WpSlsU value in each class will have a rank of 1, and there will be several SKUs with a rank of 1, one per class.

- **Rank <- ranksort(WpSlsR, descending, [prod].[clss])**

  If Rank has a base intersection of sku/week, the result of this procedure is the integer value representing where each SKU/Week's Sales value ranks amongst all SKU/weeks within its class for the entire time horizon. The SKU/week with the highest WpSlsR value in each class will have a rank of 1, and there will be several SKU/weeks with a rank of 1, one per class.

- **Rank <- ranksort(WpSlsR, descending, [prod].[clss], [clnd].[seas])**

  If Rank has a base intersection of sku/week, the result of this procedure is the integer value representing where each SKU/Week's Sales value ranks amongst all SKU/weeks within its class/season. The SKU/week with the highest WpSlsR value in each class/season will have a rank of 1, and there will be several SKU/weeks with a rank of 1, one per class/season.

positionLocked

Position locking allows a user to lock one or more positions at any level of a hierarchy in a workbook. Once locked, the values of cells corresponding to locked positions cannot change as a result of incremental evaluation, that is, calculations resulting from user edits in the workbook. However, those values can change as a result of a full evaluation resulting from either a full transition to the Calc rule after a refresh, a custom menu execution, or from the execution of an expression that cannot be evaluated incrementally; for example, lhsmeasure = 25. None of the RPASCE procedures, such as lookup or flookup, honor position locking.

The positionLocked function has been provided to help solution designers honor position locking during full evaluation. However, since usage of functions cannot be combined with the evaluation of special expressions (procedures), this provision does not help designers in honoring position locking when special expressions (procedures) are used. Position locking is not supported for special expressions. Position locking is also not supported or honored in scripts, whether they appear in batch or as part of a custom menu.

The positionLocked function is evaluated at the level at which calculation is being performed. It returns a FALSE if none of the positions for the current cell's dimensions are locked and returns TRUE otherwise. Since position locking is only available in a workbook and can only be used after a workbook has been built, the function always returns a FALSE when evaluated in a PDS, used in the load rule group, or used in a workbook. When used in a PDS, the function will log a warning in the RPASCE logs but will not fail evaluation. The function will work as expected when used in refresh, calc and commit rule groups. It will also work as expected in custom menu rule groups that operate on the workbook.

The positionLocked function does not provide for hierarchical protection processing. Refer to the following examples for a better understanding of this behavior.

Syntax

```
positionLocked()
```

The function does not take any arguments and returns a Boolean value. The function can be used by itself on the RHS.

Inverse

The function does not have an inverse.

Examples

- **Output = positionLocked()**

  Populates the Output measure with values that tell whether a cell is locked as a result of position locking along any of the dimensions in the base intersection of the measure.

- **Output = !positionLocked()**

  Populates the Output measure with values that tell whether a cell is not locked as a result of position locking along any of the dimensions in the base intersection of the measure.

- **Output.level([PROD].[clss]) = positionLocked()**

  Populates the Output measure (default spread type: repl) with values that tell whether a cell is locked as a result of position locking along any of the dimensions, except those belonging to the PROD hierarchy, in the base intersection of the measure. For the PROD hierarchy, the check is done at the clss dimension and the results would be spread down to the base intersection of the Output measure. Therefore, at the base intersection level, the cell value will be True if all child positions of the cell's clss are position locked, and False if any of them are unlocked irrespective of whether the position itself is locked or not; that is, the positionLocked function does not provide for hierarchical protection processing.

- **Price = if (positionLocked(), ignore, Price.master)**

  When used in a load rule group, this expression behaves the same as “Price = Price.master" because positionLocked() always returns FALSE. When used in a refresh rule group, the LHS is not updated for positions that are locked.
- **Price.level([PROD].[clss]) = if (positionLocked(), ignore, Price.master)**

  When used in a load rule group, this expression behaves the same as Price.level([PROD].[clss]) = Price.master because positionLocked() always returns FALSE. When used in a refresh rule group, the LHS is not updated for clss level positions that are locked, that is, all their children are locked. However, if any of the children is unlocked, and hence the clss level position is unlocked, the result of the spread after the expression evaluation will alter the child position's value, even if the child position was locked, that is, positionLocked function does not provide for hierarchical protection processing.

randMask

Used to create a randomly populated mask measure. In contrast to rand, the return type is always Boolean. In contrast to rand, it is a procedure rather than a function.

Syntax

```
boolMeas <- randMask(density:<density>, seed:<seed>)
```

Labels are required with all parameters since randMask can be called without any input parameters. A description of each of the optional parameters is provided later:

- Density is the percentage of cells in the LHS measure that need to be filled with randomly determined true or false values. The cells that are set with random values depend on the iterator order but randMask will try to space out the values evenly, so they are not concentrated in any particular area in the dimspace. Density is an optional parameter and can be either a real or integer constant or scalar measure. Default value of density is 5, corresponding to 5% of the total cells being set.
- Seed is an optional input into the random generation algorithm. Seed can be either an int or a real number. The same seed will produce the same set of random values. Note that this is the behavior of random number generator algorithms in general and not specific to RPAS CE. The seed argument may be more applicable in the testing/verification process where the same test random data can be generated multiple times using the same seed. When using the seed argument, the value needs to be preceded by the seed: label. The use cases have an example of proper usage.
- Because both arguments are either literals or scalar measures, incremental mode evaluation of randMask is the same as full mode

Examples

- **boolMeas <- randMask()**

Set a LHS measure of type Boolean with random Boolean values. Since density is not provided, a 5% (default density) of the LHS measures cells will be set with random values.

- **boolMeas <- randMask() (density:12.2, seed:1729)**

  Set approximately 12.2% of the cells in the LHS measure to random Boolean values and use 1729 as the seeded to the random generator algorithm.

loadagg

The loadagg procedure allows the loading of PDS measure data to a workbook using aggregation according to the roll-up information present within the workbook in which the loadagg procedure is run. It may be used within both load and refresh rule groups. Measure data aggregated to the result dimension of a dynamic hierarchy using a load or refresh rule that does not use the loadadd procedure is aggregated according to the parent-child relationships defined within the PDS.

Use of the loadagg procedure has the following constraints:

- The LHS measure must be a workbook measure.

  The LHS measure must be based at an intersection containing a dimension modified by a dynamic hierarchy in the workbook.
- The RHS measure must be a PDS measure.

  Loadagg may only be used in load and refresh rule groups.

Syntax

```
<measure1> <- loadagg(<measure2.master>)
```

Example

```
ClusterValues <- loadagg(StoreValues.master)
```

In theexample, ClusterValues is a measure defined on the cluster dimension, StoreValues is a measure defined on the store dimension, and the aggregation of stores to clusters is modified by a dynamic hierarchy in the workbook.

The values present in the measure StoreValues for those stores that roll-up to any given position in a cluster according to the dynamic hierarchy are aggregated according to the aggregation type of ClusterValues to determine the value assigned to ClusterValue for that cluster.

Were the loadagg procedure not used and the expression:

```
ClusterValues = StoreValues.master
```

were used instead, then the values in the measure StoreValues for those stores that roll-up to any given cluster in the PDS (as opposed to the workbook) would be aggregated by the ClusterValues aggregation method to be the value of ClusterValues for that cluster.

spreadcommit

The spreadcommit procedure allows the committing of workbook measure data to the PDS using spreading according to the roll-up information present within the workbook in which the spreadcommit procedure is run. It may be used only in commit rule groups and custom menu rule groups that perform commit operations. Measure data spread from the result dimension of a dynamic hierarchy using a commit rule that does not use the spreadcommit procedure is spread according to the parent-child relationships defined within the PDS.

Use of the spreadcommit procedure has the following constraints:

- The RHS measure must be an PDS measure.
- The LHS measure must be a workbook measure.
- The LHS measure must be based at an intersection containing a dimension modified by a dynamic hierarchy in the workbook.
- spreadcommit can only be used in the commit rule group and in the custom menu rule groups that perform a commit operation.
- By default, spread operations that rely on data already present in the destination (for example, proportional spread) use the data present in the PDS. An optional argument <seed_measure> may be used to specify an alternate measure to use for this purpose.

Syntax

Labelled notation is also supported. If labels are not provided, then positional notation is supported.

```
<measure1.master>  <-
spreadcommit(<measure2>{,<seed_measure>,<mode_measure>,<maskMeasure>})

<measure1.master>  <-
spreadcommit(SOURCE:<measure2>{,TARGET:<seed_measure>,MODE:<mode_measure>,COMMITM
ASK:<maskMeasure>})
```

If COMMITMASK is passed in, then commit is performed only if mask is true. This must be Calendar dimension and must not be part of the alternate hierarchy.

Example

```
StoreValues.master <- spreadcommit(ClusterValues)
```

In the previous example, ClusterValues is a measure defined on the cluster dimension, StoreValues is a measure defined on the store dimension, and the aggregation of stores to clusters is modified by a dynamic hierarchy in the workbook.

The values present in the measure ClusterValues for any given cluster are spread to the cells of StoreValues whose store positions roll-up to that cluster according to the dynamic hierarchy defined in the workbook.

Were the spreadcommit procedure not used and the expression:

```
StoreValues.master = ClusterValues
```

were used instead, then the values in the measure StoreValues for those stores that roll-up to any given cluster in the PDS (as opposed to the workbook) would receive the spread values of ClusterValues for that cluster.

```
StoreValues.master <- spreadcommit(ClusterValues, Clst2StrSeed)
```

In the second example, values of CusterValues are being spread to StoreValues in a manner similar to the first example. However, in this example, a seed measure Clst2StrSeed is providing the initial data as opposed to the PDS data in StoreValues in the first example. In cases where ClusterValues uses a spread method dependent on data present in the destination (such as with the proportional spread method), the starting data is provided by Clst2StrSeed and not by StoreValues.

dynHierRefresh

The dynHierRefresh function can be used to trigger a refresh of the parent-child relationships in the dynamic hierarchies of a workbook. When called, dynHierRefresh reassigns the roll-ups of a workbook based upon the data present in the mapping measures driving those dynamic hierarchies. In cases where that mapping information has changed since the workbook was build, use of dynHierRefresh results in the new information being applied to the workbook's hierarchies.

Notes on the use of dynHierRefresh:

- May only be used in custom menu rule groups.
- Returns a string value of Success if the refresh could be performed; otherwise, it returns a value of Failure.
- Cannot be used to refresh dimensions upon which dimension attributes are defined. Attempts to do so results in a value of Failure.

Syntax

<result_measure> = dynHierRefresh([<DynDimName1>,...<DynDimNameN>])

Example

```
RefreshFlag = dynHierRefresh()
```

```
RefreshFlag = dynHierRefresh("dm1", "dm2")
```

where RefreshFlag is a scalar String measure that contains either the string Success or the string Failure to indicate the success or failure of the refresh operation.

The dynHierRefresh function takes a list of dynamic dimension names as its input. The input list is optional. If no input is provided, it will refresh all dynamic dimensions defined in the workbook. If input is provided, only the dynamic dimensions specified will be refreshed.

preModified

The preModified function returns a value of true if the current cell has been modified at the beginning of the current expression's evaluation step.

The preModified function only works for workbook's incremental evaluation mode. The function takes a single input that must be a valid measure name that must exist in the current workbook. It returns a value of true for all cells of the rhs measure that have been modified when current expression is first evaluated.

If used in Full Mode evaluation, preModified will return a value of false for all cells.

Syntax

preModified(<measure>)

where <measure> is a valid measure name; no modifier, nor any other calculations are allowed.

Inverse

The preModified function does not have an inverse.

Example

preModified(A) returns a value of true at cells A has been calculated in the current workbook's incremental evaluation step.E = if (preModified(A), A+B, C+D)preModified is used to calculate lhs E using different logic, based on whether the measure A is modified or not. Assume all A, B, C, D are at sku/str/week level and filled with value 1, E initially has Value 2. If the user modified A for a sku/str/week combination to 100, and this rule is triggered. Because preModified(A) returns true, E will be evaluated to A+B, which is 101. If the user modified measure C at the same sku/str/week to 10 and this rule is triggered. Because preModified(A) is false, E will be evaluated to C+D, which is 11.

AggPEClc

The AggPEClc procedure computes the inner expression using the specified preserve Calendar level, then performs the final Calendar aggregation in the PET (Period Ending Total) method. The procedure is designed to wrap a RECLC expression within the CALC rulegroup within a workbook template. The LHS measure must have aggregation of RECLC, and the inner expression is expected to be a RECLC expression.

Syntax

<result_measure> = AggPEClc(<A RECLC expression>, <preserve calendar level name>)

Example

Lhs_dstrsclsmnth = aggpeclc(A_storskupweek + B_storskupday, "week")

The second parameter of AggPEClc indicate the computation of the inner expression must be maintained at calendar level week. When Lhs_dstrsclsmnth measure has aggregation method set as RECLC, then for each intersection that is viewable and is higher or equals to dstr_scls_mnth on UI Client grid, the AggPEClc function will first aggregate the RHS measures to LHS NonClndLevels+WEEK using their respective aggregation methods, will compute the expression, and finally, will set the all periods of LHS calendar level with values from last week values within the period. For example, if the UI view is displaying at rgn_clss_qrtr level, it computes the inner expression at rgn_clss _week level, aggregating measures to rgn_clss_week if needed, then PET aggregate to rgn_clss_qrtr.

Expr A = AggPEClc(B + C, "week")

A base intx = dstr+scls+mnth with RECLC aggregation

B base intx = stor+skup+week with total aggregation

C base intx = stor+skup_week with average aggregation.

And

B C

STR_X1, SKU_A1, WEEK_30 10 20

STR_X2, SKU_A1, WEEK_30 20 30

STR_X1, SKU_A1, WEEK_31 20 30

STR_X2, SKU_A1, WEEK_31 30 40

Agg to

DSTR_X, SCLS_A, WEEK_30 30 25 (B+C = 55)

DSTR_X, SCLS_A, WEEK_31 50 35 (B+C = 85)

FINAL PET To bring WEEK TO MNTH

A @ DSTR_X, SCLS_A, MNTH_5 will be set to last week (WEEK_31) of the sum, which is WEEK_31 and the value is 85. To compute the initial view at A base intx dstr+scls+mnth, the RHS will aggregate B and C to dstr+scls+week using their respective agg method, total for B and average for C. Then the inner expression will be evaluated. Resulting the sum of total of B and average of C @ dstr+sclr+week level. Finally, the respective mnth of the LHS is set with the last week of that mnth from the RHS. When the UI GRID is change to different level, for example, rgn+dept+year. the same process is used to computer the LHS values: Bring the RHS measures to rgn+dept+week using their respective agg method, evaluate the inner expression, then finally set the calendar period using the last week of the year.

intradayexport

The intradayexport procedure submits one or multiple EEBATCH export measure task to the task queue to perform measure export without going through the Configured Batch Tasks in Online Admin Tools. This procedure is expected to be only used by custom menu action. It offers a way for non-admin users to trigger measure export. The export operation does not lock the PDS normal batch tasks. Export operations created by this procedure are submitted to the task queue only and the JTD will determine the time of the actual export should occur.

Syntax

<result_measure> = intradayexport(<ExportTaskName1>[,...< ExportTaskName2>])

Example

ExportFlag = intradayexport("export_mg1", "export_mg2")

where ExportFlag is a scalar string measure that contains either the string Success or the string Failure to indicate the reason of failure. The export task group should use the |T| directive and without the |O| directive to prevent name collision. For detail explanation of these directives, please refer to *Oracle Retail Predictive Application Service Cloud Service Implementation Guide*, Release 19.0.

rmsNewItemExpr

Special Expression rmsNewItemExpr directly calls RMS Micro Service to enable new items. When called, the special expression will iterate all the new items that have the item mask set to true, collect all the associated attributes for each new item, and put all the new item and attributes into a micro service request. The service request is then sent to RMS to be processed.

Syntax

strscalar1←rmsNewItemExpr(mask:itemMask,[item:itemIdMeas, AttrName1:AttrVal1, AttrName2:AttrVal2, …])

To produce the RMS new item request, the special expression will produce the json request based on the list of arguments, so if an argument is not supplied as input, the request json will not have the entry for it. If a new json entry is needed for the request, a new labeled argument can be added to the input and the argument will be added to the request json object automatically. The special expression will use the label of the argument as the key to the json request and the value as the data for that key.

Example

msg:scalarString, itemmsg:itemStringMeas, status:scalarBoolMeas, retcode:scalarIntMeas←rmsNewItemExpr(mask:itemMask,item:itemIdMeas,itemParent:itemParentIdMeas,itemLevel:itemLevelMeas,diff1:diffMeas,dept:deptMeas,class:1,classMeas,itemDescription:itemDescMeas,itemSuppliers_supplier:splrMeas,itemAggregateInd:itemAggMeas,diff1AggregateInd:diff1IndMeas,brand:brandMeas)

The RHS arguments of the expression must use labeled input. Except for the 'mask' argument, all other arguments are optional.

For the RHS measure, the mask measure must be a one-dimensional Boolean measure and the base intx of the measure must be the dimension for the RMS item. For each position on this dimension, if the mask is true, the special expression will create the json object for this item and ask RMS to create new item for it.

The top level of the request will be a json object named as "items", within the "items", there will be a list of json objects, each populated with key/value pairs extracted from the list of input arguments, based on the requirement in the confluence page.

Note that each position can be created only once in RMS, so if one position is created a second time, the RMS micro service will report an error.

The "item" argument provides the item ID to be passed to RMS; if omitted, the position ID for the current item will be used.

For other arguments on the RHS, the input can be either a measure or a constant value for default. If it is a measure, it can be either a one-dimensional measure at the same intersection of the mask or a two-dimensional measure that includes the item dimension plus an extra dimension.

Each labelled input argument, for example arg1, can have a mask input associated with it labeled as arg1mask. The mask must be at the same intx of the arg1's input. It can also have specific labelled measure input, for example, arg1_attr1, arg1_attr2, and so on. The attribute name and the 'arg1' must be separated by '_'.

If the input is a one-dimensional measure, and if it has an associated mask measure, the mask will determine whether the attribute for the current item will be set in the request. If the mask is not present, the input attribute will be set.

If the argument arg1 has no associated attribute such asarg1_attr1 and so on, the following entry for "arg1" will be added to the request json:

"items": [{

"item": "1000",

"arg1": "arg1value",

...

If the argument arg1 does have associated attributes such as arg1_attr1 and arg1_attr2, then the json entry for arg1 will be:

"arg1": [

{

"attr1": "attr1value" ,

"attr2": "attr2value"

}

]

The arg1 entry will have a collection with just one object per requirement, and each attribute will be an entry in the object.

If the input is a two-dimensional measure, the argument label must be in the form of argName_attrName: the attribute name is required and separated by the "_". It can also have the companion mask, such as argNamemask, and the mask measure must be at the same intersection as the other input measures of the same argName.

Since the input is two-dimensional, the arg will be a collection; each positions of the extra dimension, if the mask is present and is true, will be an element in the collection. Within the element, each attribute will be a data member. In the following example, the expressionis configured with:

... v2dmask:skustrb, v2d_attr1:skustrval1, v2d_attr2:skustrval2

Assuming the extra dimension is str and the v2dmask is true for all stores, and there are two stores in the workbook, the following entry will be created:

"v2d" : [

{ "attr1":"val11",

"attr2"::val12

},

{ "attr1":"val21",

"attr2":"val22"

}

]

Note that the "attr1" and "attr2" are the keys inside the element for each store. That is the reason the attribute name must be specified for two—dimensional measures.

Also, assume the first entry in the collection is for store 1 and the second entry is for store 2: If the v2dmask measure has 'false' for store 2, then the second entry will be removed from the request.

The LHS has four labeled output measures; all are optional but at least one must be specified. The msg specifies the scalar string output that stores the returned status/message from the microservice call.

The item msg specifies the item level string measure that stores the validation error for each item, if there is validation error.

The retcode stores the microservice return code in a scalar integer measure. The status stores the information indicating whether the microservice call is success or not in a scalar boolean measure.

E

Appendix – Aggregation and Spread Types

This appendix provides details on the aggregation and spread types.

Aggregation Types

Table E-1 describes the supported aggregation types.

Note:

If aggregation (any) is trying to aggregate from a partitioned level to a non-partition level (HBI), then the aggregation occurs on the whole fact table. RPASCE does not conduct step up aggregation in batch. Also, RPASCE does not conduct any step up aggregation during the workbook load as well.

Table E-1 Aggregation Types

Aggregation Type	Description	Valid Data Types	Recommended Spread Types	Aggregate over Partition Dim
recalc	The measure is not aggregated but is recalculated at all aggregated levels through a recalc expression. The passthrough function is not supported with this agg type. Note that RPASCE recommends not using recalc measures on both the left hand (lhs) and the right hand (rhs) of an expression to avoid possible run time failure. Currently, the RPASCE calculation engine cannot do a recalc aggregation while it is inside another measure's recalc aggregation. In the situation where LHS_Recalc = func(RHS_recalc), if the rhs recalc measure has already been aggregated as part of other expressions, this expression has no issue; if the rhs recalc has not been aggregated and must be aggregated now, this evaluation would fail.	numeric string date Boolean	none	Yes
ambig	The measure is aggregated by considering the values of all child cells. If all child cells have the same value, the aggregated value is the same as the child cells. Otherwise, it is ambig.	numeric string date Boolean	none	No

Table E-1 (Cont.) Aggregation Types

Aggregation Type	Description	Valid Data Types	Recommended Spread Types	Aggregate over Partition Dim
ambig_pop	The measure is aggregated by considering the values of all populated child cells. If all populated child cells have the same value, the aggregated value is the same as the child cells. Otherwise, it is ambig.	numeric string date Boolean	none	No
popcount	The measure is aggregated by counting the number of child cells that are populated (meaning that they have a value different from the NA value for the measure).	numeric string	none	No
mode	Picks the most frequently occurring cell value from the base intersection to represent the cell value of the aggregate dimension.	string	repl	No
mode_pop	Very similar to the mode agg type, except that it will skip all NA values on the base intersection	string	repl	No
hybrid	The measure is aggregated using a specific aggregation type for each hierarchy. This is selected from the valid aggregation types for the measure type.	numeric string date Boolean	none	No
total	The measure is aggregated by taking the total (numeric sum) of the values of all child cells at the base intersection.	numeric	prop	Yes
total_pop	The measure is aggregated by taking the total (numeric sum) of the values of all populated child cells at the base intersection.	numeric	prop_pop	Yes
average	The measure is aggregated by taking the numeric average of the values of all child cells at the base intersection.	numeric	prop	No
average_pop	The measure is aggregated by taking the numeric average of the values of all populated child cells at the base intersection.	numeric	prop_pop	No
max	The measure is aggregated by taking the maximum of the values of all child cells at the base intersection.	numeric date	repl_pop	Yes
max_pop	The measure is aggregated by taking the maximum of the values of all populated child cells at the base intersection.	numeric date	repl_pop	Yes

Table E-1 (Cont.) Aggregation Types

Aggregation Type	Description	Valid Data Types	Recommended Spread Types	Aggregate over Partition Dim
min	The measure is aggregated by taking the minimum of the values of all child cells at the base intersection. Note: For most purposes, the min_pop aggregation type is appropriate because the minimum value of all child values is typically the NA value, which is usually zero.	numeric date	repl	Yes
min_pop	The measure is aggregated by taking the minimum of the values of all populated child cells at the base intersection.	numeric date	repl_pop	Yes
pst	The measure is aggregated by selecting the first value along the innermost hierarchy and by taking the total of all child values along all others. Note: First only has a meaning in the calendar hierarchy. Therefore, this agg type must only be used for measures whose innermost hierarchy is the calendar hierarchy.	numeric	ps	No
pet	The measure is aggregated by selecting the last value along the innermost hierarchy and by taking the total of all child values along all others. Note: Last only has a meaning in the calendar hierarchy. Therefore, this agg type must only be used for measures whose innermost hierarchy is the calendar hierarchy.	numeric	pe	No
median	The measure is aggregated as the median value (the middle value when sorted from lowest to highest) of the values of all child cells.	numeric	repl	No
median_pop	The measure is aggregated as the median value (the middle value when sorted from lowest to highest) of the values of all populated child cells.	numeric	repl_pop	No
and	The measure is aggregated by performing a Boolean And operation on the values of all child cells.	Boolean	repl	Yes
or	The measure is aggregated by performing a Boolean Or operation on the values of all child cells.	Boolean	repl	Yes

Spread Types

Table E-2 describes the supported spread types.

Table E-2 Spread Types

Spread Type	Description	Valid Data Types
none	Values are not spread	numeric string date Boolean
repl	Replicate the value to each cell	numeric string date Boolean
repl_pop	Replicate the value to each populated cell	numeric string date Boolean
prop	Spread value proportionally (previous total non-zero) or evenly (previous total zero)	numeric
prop_pop	Spread value proportionally (previous total non-zero) or evenly (previous total zero) to all populated cells	numeric
even	Spread value evenly	numeric
delta	Increment/decrement each cell evenly. Effectively the even spreading of the change (delta).	numeric
ps	Apply delta to starting period	numeric
pe	Apply delta to ending period	numeric

Arithmetic Operators

This section provides information about the arithmetic operators supported in Configuration Tools.

Unary Operators

Table E-3 describes the supported unary arithmetic operators.

Table E-3 Unary Arithmetic Operators

Symbol	Type	Function
-	real	Negation
!	Boolean	Compliment

Binary Operators

Table E-4 describes the supported binary arithmetic operators.

Table E-4 Binary Arithmetic Operators

Symbol	Type	Function
=	real Boolean string date	Assignment
+	real	Addition
-	real	Subtraction
*	real	Multiplication
/	real	Division
&&	Boolean	Boolean and
||	Boolean	Boolean or
==	real Boolean string date	Equality Note: When used to compare two strings, this operator performs a case-insensitive compare.
!=	real Boolean string date	Inequality
<	real Boolean string date	Less than
<=	real Boolean string date	Less than or equal to
>	real Boolean string date	Greater than
>=	real Boolean string date	Greater than or equal to

F

Appendix – RPASCE Configuration Manager and rpasConfigMgr

Note:

Users who want to run the RPASCE Configuration Manager on a UNIX or Linux system will need to do so over Xserver or another remote window system. Contact the administrator of the UNIX or Linux system for information about availability and setup.

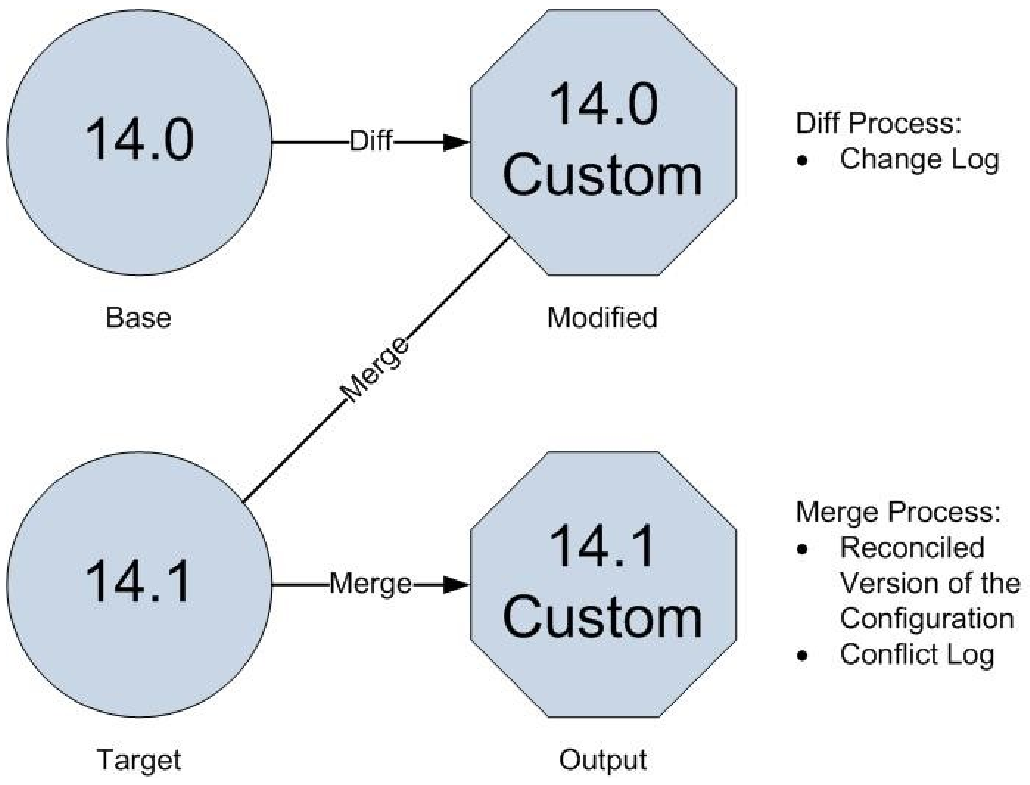
Using the rpasConfigMgr

Many RPASCE users customize their configurations of RPASCE and RPASCE applications in order to be more aligned with their business needs. When those users want to upgrade to a new RPASCE or RPASCE application release, they can use the RPASCE Configuration Manager or the rpasConfigMgr utility to replicate their customizations on the new base release with as little manual configuration as possible. By using these tools to upgrade to a new release, users can take advantage of the new features of that release while maintaining their customized configuration.

rpasConfigMgr Process

The rpasConfigMgr utility consists of two processes: diff and merge. Diffing is the identification of modifications between two versions of a configuration. Merging is the reconciliation between a base version and two modified versions of a configuration. These processes are shown at a high level in Figure F-1.

Figure F-1 The Diff and Merge Processes



When a user has modified the base release (base) to create a customized configuration (modified), that user can use the rpasConfigMgr process to upgrade to the next base release (target) and then apply their configuration to create a customized configuration of the new release (output).

To do this, the user first runs the diff process which finds the customized aspects of the modified version by identifying the differences between the base and modified version.

The output of the diff process is a change log file which records the differences between the base and the modified versions.

Then, the user runs the merge process which automatically applies the customized configuration to the target version if possible. If an aspect of the customized configuration cannot be applied to the target version, it is recorded in the conflict log file. The output of the merge process is the reconciled version of the configuration, a change log, and a conflict log.

Alternatively, the user can run a diffAndMerge process that combines the two processes and performs them together. The result of that process is a reconciled version of the configuration, a change log, and a conflict log.

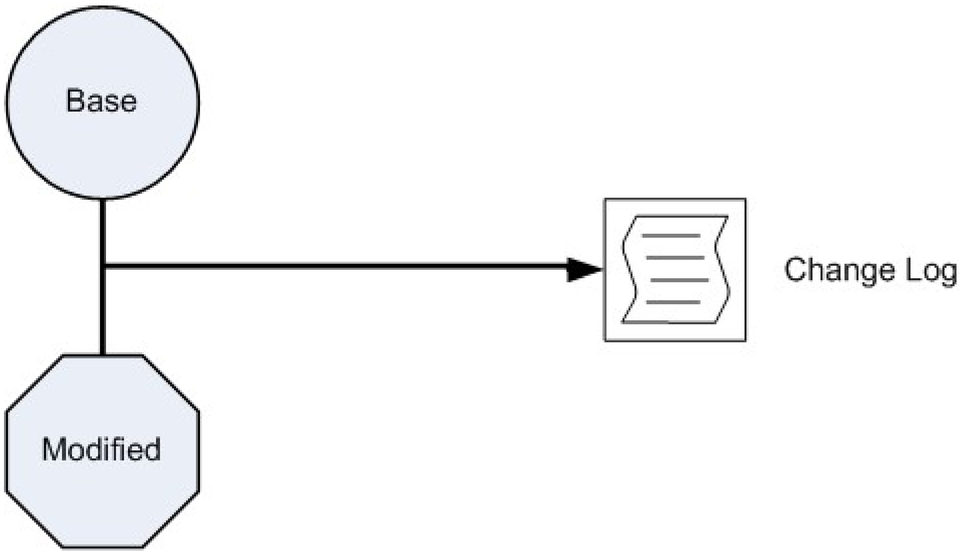
These three processes are explained in greater detail in these sections:

- diff
- merge
- diffAndMerge

diff

The diff process analyzes two different versions of a configuration in order to determine the differences in the configuration between them. Examples of such differences include property changes, altered parent-child relationships, or reordered information of elements that obey strict ordering. These differences are recorded in a change log file.

Figure F-2 Diff Process



The change log is an XML file that contains two distinct classes of elements: scope elements and operations elements.

Scope elements contain the structural information of the RPASCE application configured content. Scope elements by nature do not encode any information about modifications; they only provide information about the area of the configuration in which any given modification occurs.

Since the structural information can be very large and complex, the change log can contain numerous nested scope elements. These elements have an overall structure that mirrors the directory structure used within a configuration. The top level of the scope represents the entire application. This level contains sub-scopes that represent the changes that directly affect hierarchies, styles, and data interface entries. Other sub-scopes represent the solutions in the application. Solution scopes contain within them sub-scopes that represent changes that affect measures, rules, workbooks, and wizards.

To reduce the overall size and complexity of the structural information, unnecessary scope elements are removed from the change log. Unnecessary scope elements are those that contain only other scope nodes within its sub-tree. These elements usually occur when an aspect of a configuration has not changed between versions. For example, a change log that is generated for a configuration that has modifications made only to workbook measure labels would not contain any meaningful change information about hierarchies since no changes to the hierarchy information were made. Therefore, there is no need for scope elements relating to hierarchy and dimension structure to be included in the change log. For this reason, the hierarchy scope elements are removed in order to reduce the size of the change log.

Operation elements, also called change elements, are the other class of element in the change log. Within the structure created by the scope elements are the modifications made to the content of the application. The operation elements describe the nature of the change so that the corresponding modification can be made in the merging process.

There are several operations that can be performed during the configuration process. Each of these requires its own operation tag. The operations are:

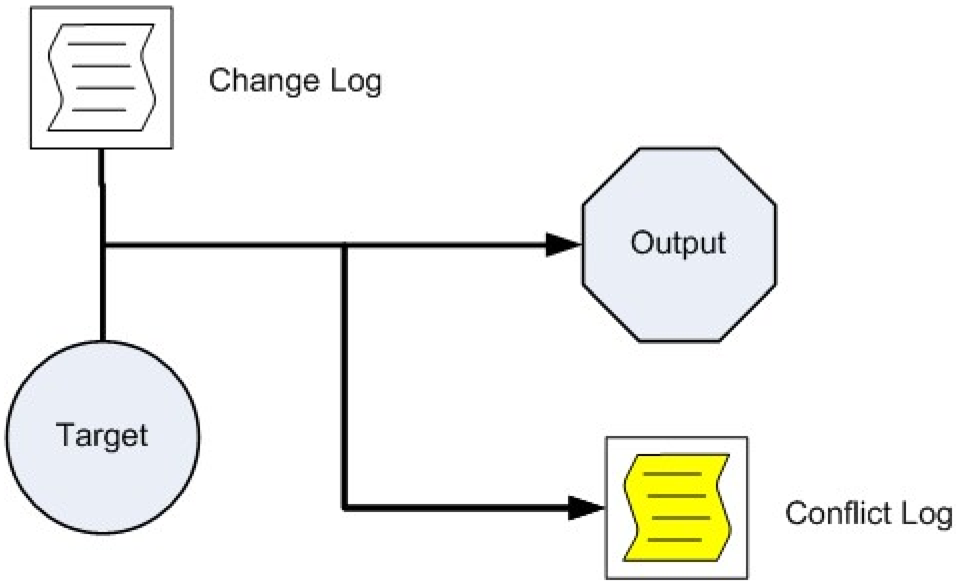
- **Element removal**: an operation that removes a piece of configured content from its parent container. This operation requires no additional information.
- **Element addition**: An operation that inserts an additional piece of configured content to a parent container. This operation requires specification of any attributes of the new content. If the new element obeys a strict ordering (such as a position of a new rule within a rule group), then information about the position of the new element is required as well.
- **Element attribute modification**: an operation that modifies one or more attributes of a piece of configured content. This operation requires information that identifies which attribute was modified and its new value.
- **Element reordering**: an operation that specifies a change in the order of a piece of configured content that has a meaningful order (such as the order of a rule within a rule group). This operation requires information about the new order of the element.

Each of the operation elements previously described must provide enough information in order for the configuration merging process to recreate the modification within the target configuration. The total collection of the operation elements of a change log should provide enough information to completely capture all changes made to the input configuration.

merge

The merge process automatically applies the modifications that were identified in the change log to a target configuration. The merge process uses the change log and the target configuration to create the output configuration and a conflict log, as shown in Figure F-3.

Figure F-3 Merge Process



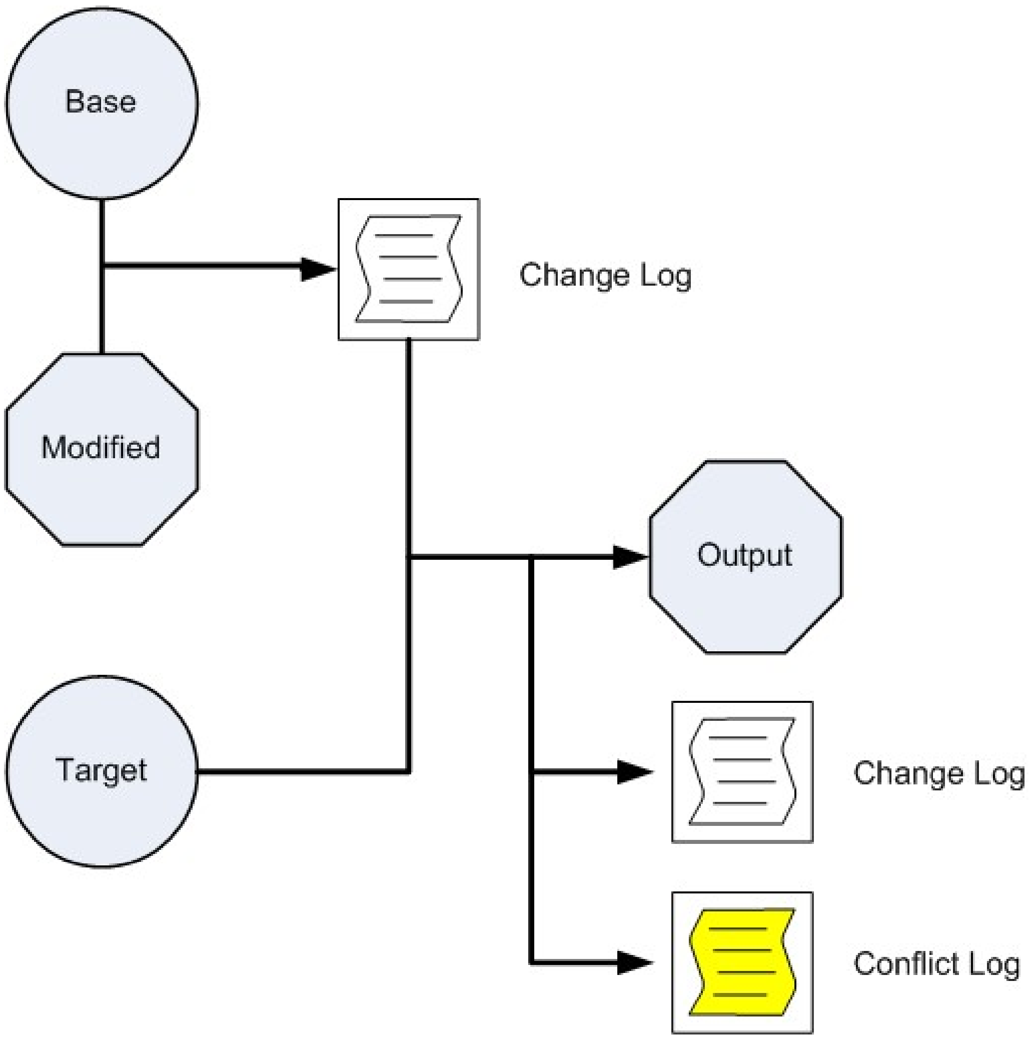
The merge process attempts to recreate each operation represented in the change log. In cases where the modification can be correctly applied, the merge process performs the modification on the target configuration. However, if the differences between the base and target configurations make the operation impossible to perform, then the modification is not applied to the target configuration. In those cases, the operation is documented in the conflict log.

The conflict log is an XML file that contains the same scope structure as the change log. The conflict log also contains conflict nodes that describe the operations that could not be performed automatically. This information is provided so that you can manually configure them after the merge process.

diffAndMerge

The diffAndMerge process is the combination of the diff and merge processes. Use the diffAndMerge process if you do not want to review the change log before continuing to the merge process. The diffAndMerge process loads the three versions of a configuration (the base, the modified, and the target versions), performs the diff command on the base and modified versions, and then immediately applies the detected changes to the target. The output of the diffAndMerge process is the reconciled version of the configuration, a change log, and a conflict log, as shown in Figure F-4.

Figure F-4 diffAndMerge Process



For more information about the diff and merge steps of the diffAndMerge process, see the Diff Process and Merge Process pages.

rpasConfigMgr Usage

The rpasConfigMgr utility supports three commands:

- –diff

  The –diff command loads two versions of a configuration and produces a change log that describes all of the changes in the modified version of the configuration.

- –merge

  The –merge command loads and parses a change log and attempts to apply the changes described within it to a configuration that you specify. It then outputs a modified version of the configuration and a conflict log to the specified location.

- –diffAndMerge

The –diffAndMerge command loads three versions of a configuration (the base, the modified, and the target versions), performs the diff command on the base and modified versions, and then immediately applies the detected changes to the target. The output of the diffAndMerge process is the reconciled version of the configuration, a change log, and a conflict log. These are output to the location you specify.

Examples

Here are examples of rpasConfigMgr commands. In these commands, [baseConfigPath], [modifiedConfigPath], and [targetConfigPath] are paths to the root document of the base, modified, and target versions of a configuration, respectively. [outputDirectory] is the location you specify for the outputs of the commands, and [changeReportPath] is the path to the location of the change log used in the –merge command.

```
rpasConfigMgr -diff -base [baseConfigPath]-mod [modifiedConfigPath]-output
[outputDirectory]
rpasConfigMgr -merge -target [targetConfigPath] -change [changeReportPath] -output
[outputDirectory]
rpasConfigMgr -diffAndMerge -base [baseConfigPath]-mod [modifiedConfigPath]-target
[targetConfigPath] -output [outputDirectory]
```

RPASCE Configuration Manager

The RPASCE Configuration Manager extends the rpasConfigMgr utility to provide a more powerful and flexible tool to manage differences between different versions of a configuration. This is accomplished through two operations:

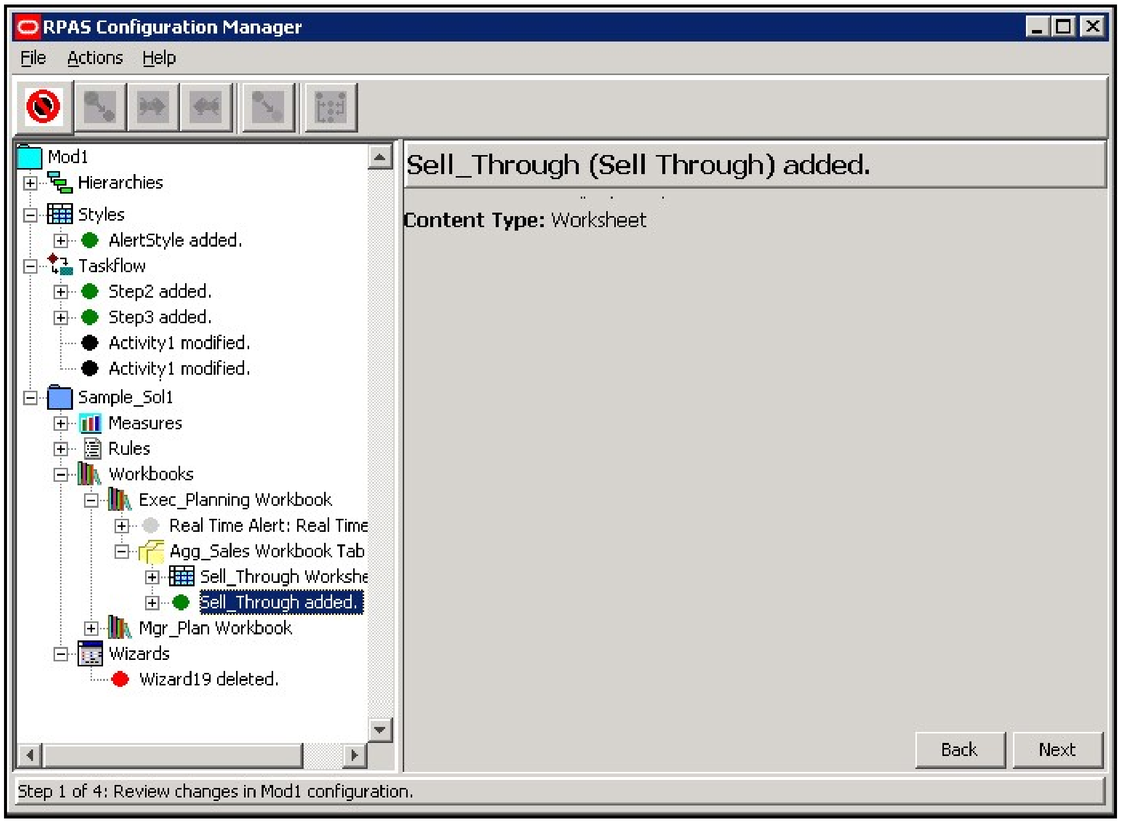
- Creating a detailed Change Report that describes the differences between two versions of a configuration.
- Creating different modified versions of a configuration in reference to a base version of the configuration.

RPASCE Configuration Manager uses the same diff and merge functionality that rpasConfigMgr uses but expands on that functionality in order to provide the user with more information about and control over the merge process.

- RPASCE Configuration Manager gives users the ability to interact with the process, enabling the user to select from a subset of all of the potential changes which diffs to apply to the upgrade.
- The rpasConfigMgr is sensitive to the changes in the base and the mod only. RPASCE Configuration Manager has two diffs:
- Between base and modified
- Between base and the update

A set of algorithms compares these two diffs and detects changes present in the two modified configurations that could potentially create conflicts. RPASCE Configuration Manager then allows users to resolve these conflicts before attempting the merge operation.

Figure F-5 Change Report



Sell_Through (Sell Through) added.

Content Type: Worksheet

Step 1 of 4: Review changes in Mod1 configuration.

When the user wants to determine the set of changes present between two versions of a configuration, they select the Change Report activity from the main UI. After being prompted to supply a base configuration and a modified configuration, the user is presented with a description of the changes present between the two versions.

Field/UI Item Description

- Green dot: Content that was added
- Red dot: Content that was deleted from the configuration
- Blue dot: Content that has been moved in the context of something where ordering matters (ordering of rules inside of a rule group matters).
- Black dot: Piece of content that has had some property of that content modified
- Attribute: Displays the name of the property that was changed

Note:

Selecting a node/dot will make more information about that change appear in the content area.

In addition to allowing the user to inspect the changes between two versions of a configuration, the UI provides the ability to output information about the changes in the form of a report. This report differs from the Change Log created as a part of the original rpasConfigMgr functionality in that it is intended to describe the detected changes in a readable format

The change report provides summary information about changes to the configuration as well as separate summaries for the various functional areas of the configuration (for example, Hierarchies, Measures). It also provides detailed information about individual changes.

Merge Functionality

RPASCE Configuration Manager also provides support for and expands the functionality of the merge operation of the rpasConfigMgr. As with the diff operation, the user can specify a new merge activity. The user is then prompted for a base version of the configuration, as well as two modified versions of the configuration.

RPASCE Configuration Manager performs the diff operation to compare each of the modified configurations to the base configuration. This creates two distinct change lists. These change lists are used as the inputs to the conflict reconciliation process. The reconciliation process identifies conflicts between the changes in the two change lists and the user's directives for those changes to resolve conflicts that would prevent a successful merge of changes to the configurations.

After the reconciliation process is complete, the utility merges the two distinct change lists into a single, comprehensive list of changes. This merged change list is then applied to the base configuration to generate a merged output configuration containing all changes present in both modified configurations that have not been discarded by the user as part of the reconciliation process. This merged configuration is then saved to a location specified by the user along with a Change Report which contains changes from the merged change list.

Conflict Resolution Functionality

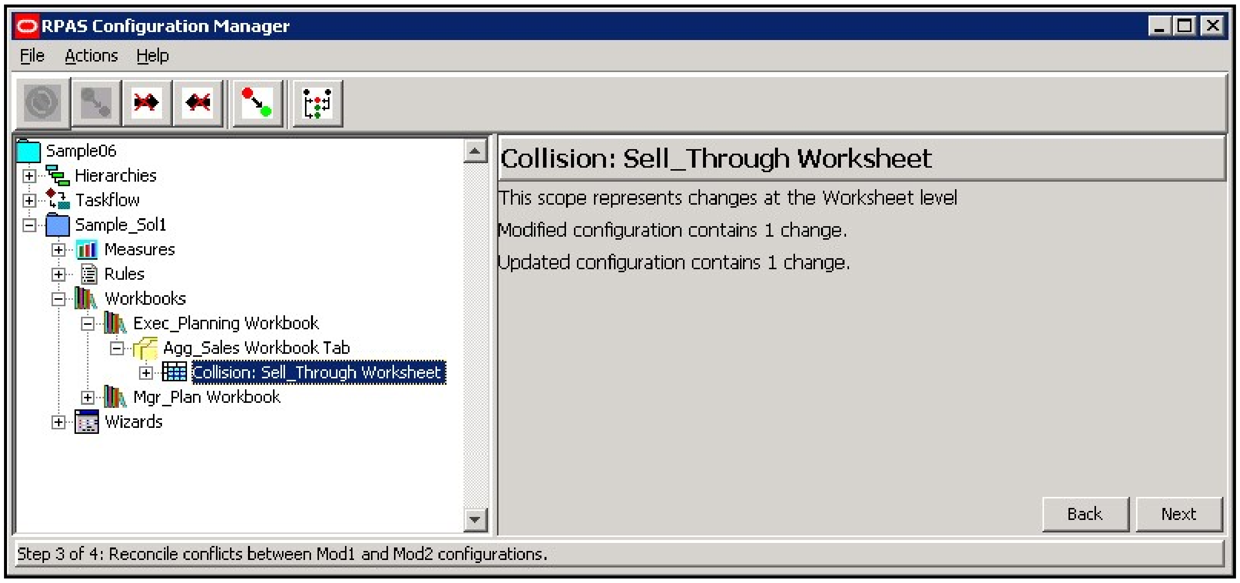
The user specifies the base and two modified versions of the configuration. Oracle Retail Predictive Application Server Configuration Manager performs a diff between the base and each of the modified versions.

The user can examine each of the Change Log summaries. The user can discard any changes they do not wish to merge before moving forward.

The user initiates an action to detect conflicts between the two sets of changes. In its most basic form, this entails performing a series of basic merge operations from the rpasConfigMgr and recording the resulting merges.

After conflicts (whether potential or actual) have been discovered, the utility presents the conflicts in the UI. The user can examine the conflicts present between the two modified versions of the configuration and begin resolving the conflicts.

Figure F-6 Conflict Reconciliation



Collision: Sell_Through Worksheet

This scope represents changes at the Worksheet level

Modified configuration contains 1 change.

Updated configuration contains 1 change.

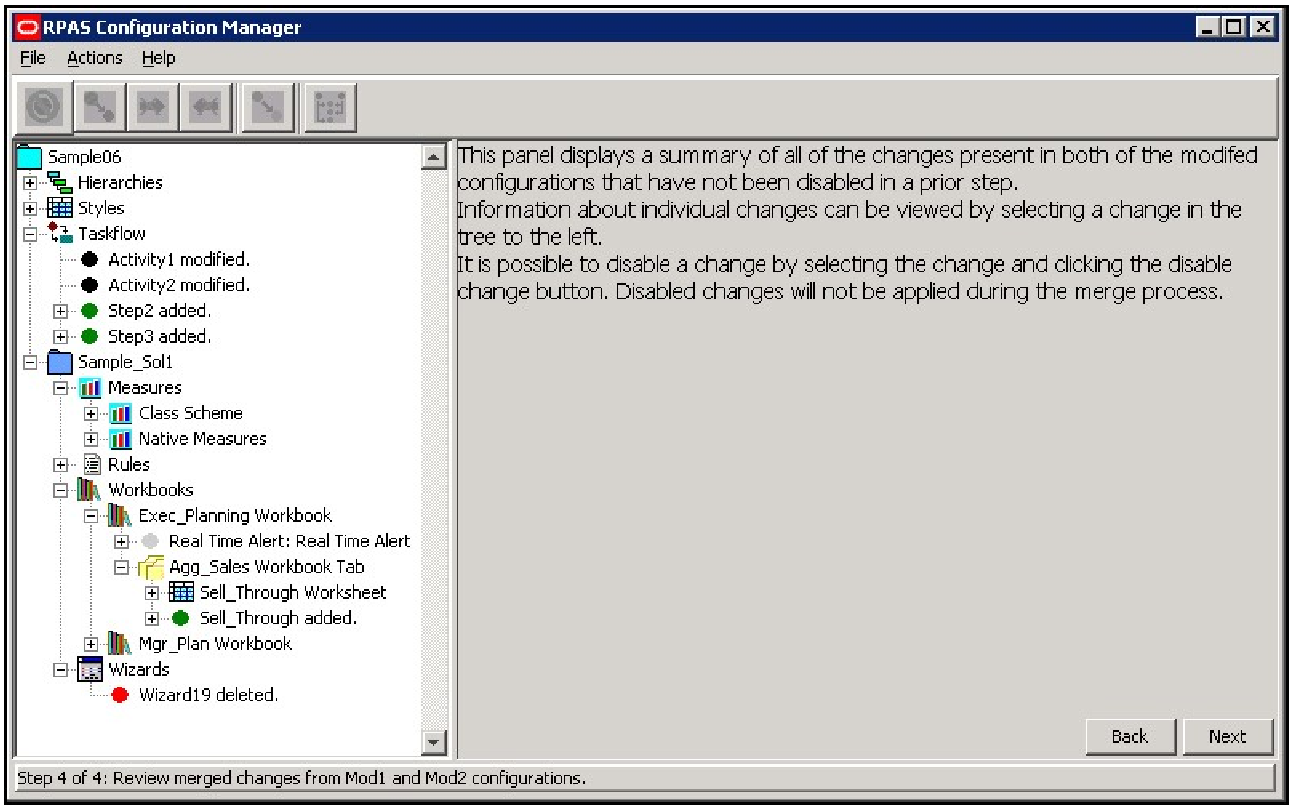
Step 3 of 4: Reconcile conflicts between Mod1 and Mod2 configurations.

This screen shows information about the new set of algorithms, the ones that find the differences between the two diffs. Oracle Retail Predictive Application Server Configuration Manager takes the two sets of changes and runs a collision detection algorithm. This algorithm produces an enhanced node structure, which identifies every place that could potentially have a conflict during the merge step.

After all collisions are resolved, a change list opens. This change list describes all changes from the two configurations except for the ones discarded. This list is a union of both sets of changes, all the changes between the original base configuration and the output at the end. This list can be used to review the changes before the merger is committed.

Oracle Retail Predictive Application Server Configuration Manager attempts to do the merge. If there are still conflicts interfering with the merge process, the Conflict Resolution screen summarizes the conflicts. This enables the user to disable the change or modify it is that stops that change from working. The user can then attempt the merge again. This is an iterative process.

Figure F-7   Merged Changes Review



This panel displays a summary of all of the changes present in both of the modifed configurations that have not been disabled in a prior step.
Information about individual changes can be viewed by selecting a change in the tree to the left.
It is possible to disable a change by selecting the change and clicking the disable change button. Disabled changes will not be applied during the merge process.

Step 4 of 4: Review merged changes from Mod1 and Mod2 configurations.

To reduce the complexity of the resolution process and to provide a greater level of assurance that the output configuration is self-consistent, conflicts are grouped into scopes by default. These scopes correspond to the entities inside an RPASCE application (for example, a measure or a workbook). Users select which of the two modified versions of the configuration they want to accept changes from. The selected version has its changes retained, while any conflicting changes in the other configuration are discarded.

For example, if both modified versions of a configuration modify properties of a measure in the base configuration, it is possible that some of the changes cannot be merged without causing a non-consistent configuration. One modified configuration might change the na value or range of the measure, whereas the other modified configuration might change the type. Although no two changes affect the same property, the combination of the changes results in an invalid measure configuration. For this reason, the user takes either one or the other of the two sets of changes to the measure to be the final change set.

This all-or-nothing approach be too limiting for the user. For example, consider a case where one modified configuration modifies data-centric properties of a measure, such as type or base intersection, and the other modified configuration modified purely cosmetic properties such as the measure label. In this case, both sets of changes can be retained without creating a non-consistent output configuration.

The conflict is presented to the user on a conflict-by-conflict basis (as opposed to an aggregate level of the entity) and users can select individual conflicts. Users can then manually select which changes result in conflicts and disable those changes. After satisfied that conflicts have been eliminated, the user can mark that collision as resolved and continue to the next collision.

For example, if both modified versions of a configuration contain instructions to modify the label of a measure, the user experiences a conflict. The merge process must accept one of the new label values or the other (or at the scope level, one set of changes to measure properties or the other). The simplest way of resolving this conflict is for the user to decide which of the two new label values is desired. After this has been determined, the instruction for the undesired label change is unavailable in the Change Logs (an unavailable change is one that is not applied as part of the merge process). At this point, the merge is successful; only the desired modification is processed.

A more complicated scenario would involve some change present in one modified configuration that can no longer be performed due to changes made in the other modified configuration that do not result in direct collisions, as previously shown, but instead result in changes to some dependent piece of content.

For example, consider the situation in which one configuration deletes a measure from the solution. The second configuration, having not deleted the measure, might attempt to add it to a measure profile. This operation cannot be performed because the operation has a dependency on some non-local content (the base measure) that can no longer exist within the merged configuration.

The utility also attempts to detect these non-local collisions. When a collision is detected, the utility reports that collision the same way that it reports local collisions. As with local collisions, users can reconcile the collision by selectively disabling changes.

RPASCE Configuration Manager Application

The RPASCE Configuration Manager application can be launched by calling the `ConfigManager.sh` script located in the bin directory of your Tools distribution. The application provides two main functions:

- Merge Operation
- Change Report Operation

Merge Operation

Table F-1 shows the buttons used for this function.

Table F-1 Merge Operations Buttons

Button	Function
🚫	Disable Change

Table F-1 (Cont.) Merge Operations Buttons

Button	Function
🔴↘●	Enable Change
✖	Discard Mod1 Changes
✖	Discard Mod2 Changes
⋮	Recompute Collisions
🔴↘🟢	Mark as Resolved

The following steps detail the merge operation function.

1. Click **New Merge Operation** to begin a new operation. Click **Open** to resume a suspended operation.
2. You are prompted to enter locations for three configurations:
   - base
   - mod1
   - mod2

   These represent the original unmodified configuration and two updated versions of that configuration. The field expects the path to the `root.xml` document of the configuration (the file selected during the open operation in the tools).

Note:

Mod1 and Mod2 are labels that can be changed using buttons, if this is desired. The new labels are used wherever Mod1 or Mod2 are referenced in the UI.

3. Click Next to proceed to the first step. This step is a review of the changes between the base configuration and the Mod1 configuration. In this step, there is only a single operation available. If a change node is selected, then **Disable Change** is available. Click **Disable Change** to mark the selected change so that it is ignored for the remainder of the operation and is not applied when the merged configuration is generated.

   If a previously unavailable change is selected, then **Enable Change** is available. Click **Enable Change** to clear the status of the change so that it is processed in future steps.

4. When satisfied with the set of changes between base and Mod1, click Next. You are provided with the changes between the base and Mod2 configurations. The same options exist in this step as in the previous review of the changes between the base configuration and the Mod1 configuration.

5. Click **Next**. In this step, you are presented with a list of locations where changes from the previous two steps might conflict have been identified. A conflict of this type is called a collision, and the navigation tree can be used to navigate between and view details about the conflicts.

6. You cannot progress until all collisions have been resolved. There are multiple ways to resolve collisions.

   a. Select the collision node. Resolve the collision by disabling all changes related to the conflict in Mod1or Mod2.

      If either of these options is selected, all changes listed in the Mod1 Changes or Mod2 Changes section of the collision are marked as unavailable, exactly as if they had been individually selected and you clicked **Disable Change**. Selecting either Discard Mod1 Changes or Discard Mod2 Changes marks the collision as resolved so that the merge can proceed.

   b. You can manually resolve collisions. Individual changes in either the Mod1 list or the Mod2 list can be selected and unavailable using **Disable Change**. Unavailable changes cannot be re-enabled using **Enable Change** in this step; that operation can only be performed in the first two steps.

      Note also that changes unavailable by using either **Disable Change** or **Discard Mod1/Mod2 Changes** are unavailable throughout the UI, even if they appear in more than one collision. It can often be the case that resolving one collision results in one or more other collisions automatically being marked as resolved because the unavailable changes were the cause of those other collisions, as well as the resolved collision.

   c. Sometimes, you are satisfied that a collision does not create a problem during the merge step. This might be because the changes in the collision do not interfere with each other in any meaningful way (this is especially common if one of the sets of changes are to cosmetic attributes such as labels), or because any troublesome changes have been manually unavailable. In such cases, you can manually mark a collision as resolved using **Mark as Resolved**.

Click **Mark as Resolved** to cause the collision to be treated as resolved without disabling any additional changes in the collision (changes that were previously unavailable remain unavailable).

   d. Click **Recompute Collisions** to cause the collision detection to be re-applied to the sets of changes that are an input to the process. However, any changes that have been unavailable are ignored for this new pass. As a result, the collisions those now-unavailable changes would cause do not appear after the re-application of the algorithm. This is useful when presented with many collisions. As collisions are resolved, the collision set can be periodically recomputed to reduce the amount of information and make remaining collisions easier to deal with.

7. After all collisions are resolved, click **Next** to proceed to the final step of the process. You are presented with a set of changes, much like in the first and second steps. However, this step of changes is the combination of the changes found in both the first step (between base and Mod1) and the second step (between base and Mod2). This list contains the union of all changes from either change set that have not been unavailable at some point in the process. In this step, you can still choose to disable any changes that you do not want applied.
8. After satisfied with the list of changes, click **Next**. At this point, the combined set of changes is applied to the base configuration to produce a new configuration that represents the merging of the changes in Mod1 and Mod2. You are prompted for a location in which to save this new configuration, as well as a summary report. In some cases, it might be possible that changes in the separate configurations cannot be applied even though the collision detection algorithms did not detect the conflicts. In this case, the user is presented with a description of the conflicts. If this should occur, the user must go back to the merge step and manually disable changes to resolve the undetected conflict. You can then move forward to the output step.

A Note on Saving and Loading Merge Operations

Because the merge process can take a considerable amount of time for complex configurations where many changes differentiate the Mod1 and Mod2 versions, RPASCE Configuration Manager provides a mechanism to capture the current state of the merge operation and save it disk. This saved record can then be reloaded later to enable the user to pick up on the process where the user left off without starting again from scratch. This functionality is available by making use of the **Save** and **Open** menu items within the File menu.

Note:

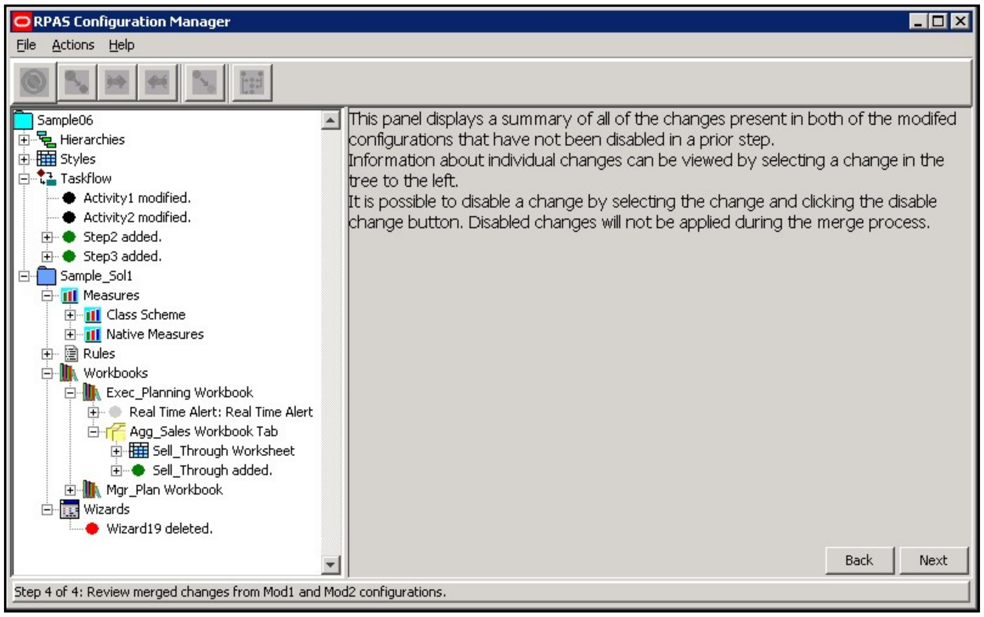
The save operation prompts you to select a directory in which resources describing the state of the merge operation will be saved. This directory should not contain any existing files or directories to prevent conflicts with the resources created by RPASCE Configuration Manager. When loading a saved operation, you should provide the directory specified during the save operation.

Change Report Operation

The following steps detail the change report operation function.

1. Select **New Change Report** from the file menu. You are prompted to enter the paths to a base version of a configuration and a modified version of the configuration.
2. Click **Next** to move to a visual description of the changes present in the two versions of the configuration. This visual description is identical to the view presented during the merge operation when reviewing differences between a base and mod configuration. However, this view is only intended to allow a visual inspection of the detected changes; there are no meaningful operations that can be performed at this point.
3. Click **Next** to move to the final step. In this step you are prompted to provide a directory for output. In this directory a summary of the changes detected is generated.

Figure F-8 Merged Changes Review



This panel displays a summary of all of the changes present in both of the modifed configurations that have not been disabled in a prior step.
Information about individual changes can be viewed by selecting a change in the tree to the left.
It is possible to disable a change by selecting the change and clicking the disable change button. Disabled changes will not be applied during the merge process.

Step 4 of 4: Review merged changes from Mod1 and Mod2 configurations.

G

Appendix – Dynamic Hierarchies

A dynamic hierarchy is a dimension within a workbook whose relationship is dynamic based on the context of the workbook and thus can vary from one workbook to another. The positions within a dynamic hierarchy are built using measure data during the workbook build process. They may vary each time a workbook is built, but the hierarchical relationships within the workbook remain constant.

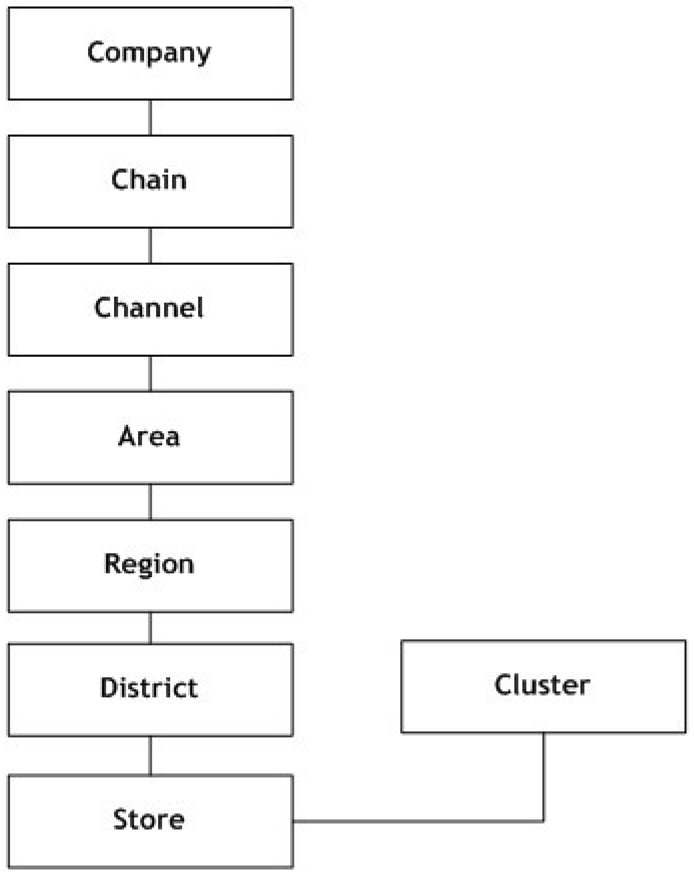
There are two types of dynamic hierarchies available in RPASCE. The first is referred to as application modified dimensions. These dimensions exist in the application hierarchy but are modified in the workbook. The second is referred to as workbook-only dimensions. These dimensions only exist in the workbook and are available for viewing purposes. This appendix will provide additional details on these two types of dynamic hierarchies and how they are configured and used within RPASCE.

Application-Modified Dimensions

As explained previously, application-modified dimensions are dynamic hierarchies that have the dimension defined in the application hierarchy structure but the positions in the workbook are dynamically assigned on workbook build. The positions are driven based on the content of measures defined in the workbook configuration of the Configuration Tools. A measure is defined to drive the dimension name and a different one for the label.

The following is an example of an application-modified dimension. Here is an example of a location hierarchy that is configured in an application.

Figure G-1 Hierarchy with Levels That Correspond to CDTs



Within a specific workbook, the relationship between store and cluster will vary based on the class in the workbook. The dimension information in the application is basically a placeholder and is replaced with the contents of the mapping measures. If the workbook contains more than a single class, then RPASCE will only use the mapping for the first class.

There are multiple uses of application modified dimensions. Some users may need to use application modified dimensions as follows:

- To bring in more than one dynamic hierarchy dimension into workbooks.
- Dynamic hierarchies need to depend on more than one other dimension to determine the value. For example, you can have clusters based on the department that you are working within, but in the future, you may need the cluster to be defined based on the department as well as the time period contained in the workbook.
- To display dynamic branches of a hierarchy only when applicable. For example, a given branch may only apply when within a specific category of department.

Multiple Application Modified Dimensions in Single Workbook

In the hierarchy example, there are dynamic positions in the Location hierarchy. The roll-up of Store Cluster is dynamic based on the Class selected. In addition to this relationship, the workbook could also have a dynamic position within the Product hierarchy. The roll-up of an item to an alternate level can also be dynamic based on at least one dimension within another hierarchy like location or calendar. Both of these hierarchies can be brought into the same workbook. This is an example of two dynamic hierarchies configured for the same workbook template.

Application Modified Dimensions Dependent on Multiple Dimensions

An application modified dimension can vary by more than a single other dimension. Based on the previous hierarchy example, the cluster dimension can vary in a workbook based both on the class and the year brought into the workbook. This is done by RPASCE, allowing you to define measures that are more than two dimensional and contain the dynamic position information.

To help you understand the multiple dimension concept, the single dimension concept is explained here first. In the example, there is a Cluster dimension that is a roll-up of Store. The Cluster that a Store is assigned to varies by Class within the Merchandise hierarchy. Through the Configuration Tools you can create a normal dimension and then create the dynamic hierarchy for the workbook. When defining the dynamic hierarchy, you must set several values. This list shows the values that relate to multiple dimensions:

- **Measure**: This is the measure name that holds the name of the parent position. The measure must have a base intersection of the dimensions that the parent-child relationship is dependent on. In the example, this is Class. The dimension that is the child in the parent-child relationship is Store in the example. The content of the measure is the name of the parent position in the relationship: in the example, this is the name of the Cluster that the Store belongs to for the Class.
- **Label Measure**: This measure holds the label of the parent position. In the example, this is the label for the Cluster. This label measure should have the same intersection as the measure that contains the name.
- **Measure Hier**: This is the name of the hierarchy that the parent-child relationship is dependent on. In the example, this is prod (Product).
- **Measure Dim**: This is the name of the dimension that the parent-child relationship is dependent on. In the example, this is clss (Class).
- **Hier**: This is the name of the hierarchy that the parent-child relationship belongs to. In the example, this is loc (Location).
- **Dim:** This is the name of the dimension that is the child in the parent-child relationship. In the example, this is str (Store).
- **Modified Dim**: This is the name of the dimension that is the parent in the parent-child relationship. In the example, this is clstr (Cluster).

With values in the previous list, the multiple dimension functionality can be described more clearly. The Measure that is defined that contains the name of the parent position needs the ability to be more than just two-dimensional. Continuing from the example, assume that the Cluster dimension varies not only by Class within the Merchandise hierarchy but also by Year within the Calendar hierarchy. The Measure intersection could be set to Store/Class/Year. Based on the Class and Year included in the workbook, the positions of the Cluster dimension are determined from the value in the Measure. If more than one Class or Year are in the workbook, the value of the Cluster positions are determined based on the first value in the measure, similar to how this is handled without multiple dimensions.

Multiple Dimension Notes

- The multiple dimensions functionality is not limited by the dimensionality of the measures. This does, however, increase the configuration load. Therefore, an updated configuration process must be analyzed. This configuration process must allow you to select a measure, and then the hierarchy and dimension information is automatically determined based on the intersection of the measure.
- When defining the measures that contain the position names and label that will define the dynamic positions, the measure must contain the dimension that is the child in the parent-child relationship. Based on the example, this means that the measures must include Store. This allows the measures to be based at a higher level of the dynamic hierarchy that is predefined. For example, assume that the Location Hierarchy in the example has a main branch that has the following relationships:

  ```
  Store -> District -> Region -> Area
  ```
- RPAS CE allows the measures to be based at Region. In this case, the value at the Region can be spread down to Store to determine the correct parent Cluster value.

Refreshing Dynamic Hierarchy Rollups

When a workbook is built, the RPASCE DB Server makes use of the information contained with the mapping measures of a dynamic hierarchy to determine the roll-up behavior of the positions for that workbook. However, changes to the contents of a mapping measure are not automatically reflected in the roll-ups defined for a workbook after it has been built. The function dynHierRefresh can be used to refresh the positions of the dynamic dimension without rebuilding or closing and re-opening the workbook. This process updates the position and roll-up information of both traditional dynamic dimensions and the workbook-only dimension.

Configuring Dynamic Hierarchy Refresh

In order to enable the refresh of dynamic hierarchies, the following must be configured:

- A custom menu item must be created to trigger the refresh.
- This custom menu item must contain a rule making use of the dynHierRefresh() function.
- The rule containing dynHierRefresh must be the only rule in that custom menu rule group.
- Prior to the call to dynHierRefresh, the mapping measures driving any dynamic dimensions that must be updated must have their cell contents updated to contain the roll-up information desired for the refresh.

Note:

It is not possible to refresh a dynamic hierarchy whose modified dimension has dimension attributes defined upon it.

Applying Changes to Data in Dynamic Hierarchy Refresh

Although the dynHierRefresh function updates the parent-child relationships in the dynamic hierarchy, those changes may affect data held within measures based at intersections containing those dynamic hierarchies. It is therefore recommended to follow the custom menu rule group that contains the call to dynHierRefresh with an additional rule group that recalculates measures whose base intersection contains the dynamic hierarchy. In this way, their values reflect the change in roll-ups performed by dynHierRefresh.

Any rule whose LHS is based at a dimension modified by a dynamic hierarchy and whose RHS measures are based at a lower intersection must be recalculated, as the set of RHS positions belonging to any LHS position may have changes as a result of the execution of dynHierRefresh.

Dynamic Hierarchies in the Wizard Process

Although the position and roll-up information contained within a dynamic hierarchy is calculated at workbook build time, it is possible to view the information in the wizard process used to build the workbook under certain circumstances. If the constraints for showing dynamic hierarchy roll-ups in the wizard are not met, the wizard will show the roll-ups defined within the application. This wizard display of dynamic hierarchies occurs when:

- All driving dimensions for the dynamic hierarchy have already had selections made in the wizard process.
- The dimension modified by the dynamic hierarchy is the selectable dimension of the wizard page.
- There are no roll-up conflicts in the positions selected for the driving dimensions in the wizard process.

To illustrate, consider a scenario in which the roll-up behavior of stores to clusters within the location hierarchy is driven by selections made in the product hierarchy. In this scenario, the location wizard shows the dynamic roll-up when:

- Selections have already been made for the product hierarchy.
- The location wizard has a cluster as the selectable dimension.
- For the selections made in the product hierarchy, only one store-to-cluster roll-up is contained within the mapping measures driving the dynamic hierarchy.

Should any of the constraints not hold true, the location wizard shows the store-to-cluster roll-ups defined within the application.

Loading and Committing Aggregated Data with Dynamic Hierarchies

When executing the load and commit operations in workbooks that contain dynamic hierarchies, a difficulty can occur when the PDS data and workbook data are not at the same base intersection. In cases in which the workbook data is at a higher intersection than the

PDS data, an aggregation operation must be performed as part of the load and a spread must be performed as a part of the commit.

The need to aggregate and spread during load and commit is true of any workbook. However, in workbooks not modified by a dynamic hierarchy, the parent-child relationships of the positions being aggregated or spread over is identical in the application and the workbook. When dealing with dynamic hierarchies, it is possible that the workbook contains a different set of roll-ups than the PDS.

To handle these situations, the loadagg and spreadcommit procedures allow control of how data is aggregated or spread between a PDS and a workbook modified by a dynamic hierarchy.

In cases where the application’s set of parent-child relationships is desired the standard a = b.master notation (to use load as an example) may be used. Data is aggregated according to the application’s roll-ups and then copied to the workbook. In cases where the dynamic hierarchy's set of parent-child relationships is desired, the a <-loadagg(b.master) procedure may be used. The data in the application is aggregated according to the dynamic hierarchy's roll-ups.

See the Appendix – Rules Function Reference Guide, for more information on the use of loadagg and spreadcommit procedures.

H

Appendix – RPASCE Rule Writing Tips

This appendix includes tips and information to help you write RPASCE rules that are as efficient as possible. In many cases, a good understanding of the internal workings of the calculation engine and RPASCE I/O fundamentals is required to create good RPASCE rules. This appendix also provides functional solutions to some general functional problems that may be encountered including:

- Basic RPASCE Rules Information
- Principles for Writing Efficient Rules
- Tips
- Expression Iteration Examples

Basic RPASCE Rules Information

The following sections provide basic RPASCE rules information. RPASCE standard practice does not encourage using RPASCE security measures in custom rules. Instead, the security settings for measures at position, template, and user level are set through the Security Administration Workbook.

- Full and Incremental Evaluation Modes
- Rule Group Transitions
- NA Values and Iterators

Full and Incremental Evaluation Modes

As a rule writer, you must know that there are two evaluation modes used in the RPASCE calculation engine: full and incremental.

Full evaluation mode is used in instances when the whole workbook is being calculated. These instances include the following:

- Committing, loading, or refreshing workbooks
- Using custom menus
- Running mace in batch mode
- During rule group transitions, such as between the load and calculate operations that are part of a workbook build

Incremental mode is used when calculating workbooks with the Calculate function. Incremental mode evaluates only the cells that the user has changed as well as any cells that are affected by the user's changes. For instance, if a user changes one cell in a sales measure, the RPASCE calculation engine evaluates that cell and all cells associated with calculations that use that sales measure, such a variance measures, inventory measures, and so on. However, the RPASCE calculation engine does not evaluate measures that are not affected by that sales change, such a receipt variance of last year's sales. In addition, the RPAS CE calculation engine does not calculate any unchanged cells in that sales measure, meaning that if the user changed the sales in the Week1 cell, the calculation engine evaluates only the Week1 cell and not any other week's cell. In short, unaffected cells are not calculated.

Note:

As a rule writer, you can use techniques to optimize rules, but these techniques rarely apply when rules are run in incremental mode.

Rule Group Transitions

Rule group transitions occur when the active rule group is changed. There are automatic rule group transitions when transitioning from the load rule group to the calc rule group during a workbook build as well as when transitioning between rule groups for refreshing and committing. Rule group transitions may also occur through user-defined menu options.

Rule group transitions ensure that the integrity of the rule group being transitioned to is enforced. This is done by evaluating an expression from every rule (and knock-on effects) that is not already known to be correct. The only rules known to be correct are those that were active in the rule group being transitioned from. Rule group transitions are inherently very expensive because they must operate in full evaluation mode. Therefore, logically at least, every intersection is visited, although the iteration efficiencies outlined previously are employed. Therefore, rule group transitions must be avoided wherever possible.

NA Values and Iterators

NA values (also known as navals) are designed to store values that occur often so that the measures can be iterated efficiently. Cell values in the internal RPASCE array are stored only when the cell value differs from the naval. Maximum efficiency is achieved if the naval is the value that is most often (logically) present in the data. This is typically zero for numeric data. The sparsity levels are likely to vary considerably from application to application and customer to customer. When data is sparse, efficient iteration patterns that visit only the populated cells are needed to support fast response times. RPASCE will change the naval of the array as necessary to allow it to use this technique. Thus, the naval of the array may differ from the naval of the measure (which do not dynamically change). Since the array naval may change dynamically and expressions use the array naval instead of the measure naval, rule writers must not write expressions that assume a specific naval.

In full evaluation mode, the measure is logically recalculated at every cell location, including those where the value is the naval. RPASCE employs a collection of iteration optimizations that reduces the number of physical cell evaluations. The optimization decisions are dependent on the following: expression syntax, navals, logical cell counts, and populated (non naval) cell counts. Therefore, it is vital that data only be added or changed through RPASCE processes that maintain this information. Otherwise, the wrong optimization decisions may be made that result in poor performance results.

In full evaluation mode, there are two fundamentally different iteration approaches. The basic iterator is to pass the cells sequentially (in the order of the positions in the hierarchies). Although, this can be very expensive if the data, and thus the cells that need to be evaluated, is sparse. There is also an iterator that iterates over just the cells that have a value different from the naval. These cells are referred to as populated. Processing is likely to be much faster when this iterator is used, especially where the data is sparse. For example, an expression such as `a = b + c` is evaluated by iterating over just those cells where b or c are populated.

With an expression such as `a = if(condition, b,0)` where the naval of a and b is zero, the fastest way to evaluate the expression is to remove all the data for a and then perform one of the following two options:

- Iterate over the intersections where the condition is true, calculating `a=b`
- Iterate over the positions where b is populated, setting `a=b` if the condition is true

The intersections that are not visited already have the correct value for a because they were effectively set to zero (the naval) when the data for a was removed. If the naval of a is not zero, RPASCE sets it to zero so that the efficient iteration pattern can be deployed. RPASCE automatically selects the methods to use based on the population density of the expressions along with other factors. RPASCE used a runtime heuristic to guess which of these two evaluation modes is most efficient and then uses that method.

Principles for Writing Efficient Rules

The following sections describe the principles for writing efficient rules in RPASCE.

- Expensive Functions, Modifiers and Procedures
- Caching Intermediate Results
- Automatic Caching of Expression Phrases

Expensive Functions, Modifiers and Procedures

As a rule writer, you should understand the relative cost of rule functions, modifiers, and procedures. If you know these costs, you can understand that you should use expensive functions only where necessary and that you must avoid any unnecessary evaluation of those functions.

Functions

You can assume that most basic functions are inexpensive. Functions that can be more expensive are those that require large amounts of data to be processed.

- All of the time series functions (such as tssum) can potentially require large amounts of data (depending on how long the time series being used is) and are therefore potentially expensive.
- The same is true of the normalization functions (such as norm, resize, resizenorm).
- Cover and uncover functions can also use large time series, depending upon the data contents.

Modifiers

The use of aggregation type modifiers is not in itself expensive; although it does mean that extra aggregation effort is required to support them.

Procedures

In general, you should assume that all procedures are expensive.

Caching Intermediate Results

If an expression must be evaluated, then the whole expression is evaluated, and the results of intermediate phrases in the calculation of the expression are not cached. Many expressions are relatively short and simple, and therefore no issues arise. There are, however, two particular cases where you must be careful when writing expressions so that you avoid expensive, redundant calculation of phrases.

Case 1

The first case is where a phrase changes infrequently and is relatively expensive to evaluate. Consider an expression in the following format:

```
a = b + c + functionof(d, e, f)
```

Here, if `functionof(d, e, f)` is relatively expensive to evaluate, you should keep its evaluation to a minimum. If d, e, and f change infrequently, then functionof(d, e, f) will also change infrequently. However, if b and/or c change frequently, then every time b and/or c changes, the whole expression is evaluated, including the `functionof(d, e, f)` phrase. This portion of the calculation is usually redundant since d, e and f change infrequently. The result from evaluating the phrase is likely to be the same as from the previous evaluation. You can avoid this inefficiency by forcing the intermediate result of the phrase to be cached by writing:

```
x = functionof(d, e, f)
```

```
a = b + c + x
```

Note that the rule syntax forces the instantiation of the result of a procedure, since procedures must be the only phrase in an expression. Since procedures often are particularly expensive to evaluate, the technique of caching intermediate results is automatically applied.

It is important to note that this method does not always increase performance. In some cases, it may actually decrease performance. For instance, if the normal usage of the rule group is in full evaluation mode, the caching intermediate results approach may be less efficient. This is because the phrase may only be evaluated once, and there could be an unnecessary write (and subsequent read) of the measure. See the Expression Iteration Examples section to learn how the expression is iterated. This can help you determine if using this method is beneficial.

Case 2

The second case when you should avoid expensive calculations is when the same phrase is used repeatedly in the same or multiple expressions. Since the phrase is not automatically cached, it can be evaluated multiple times. There are cases with nested If statements where, down some paths, the same condition can be evaluated three or four times. Using the cache intermediate results technique in these situations could lead to a significant performance increase.

Automatic Caching of Expression Phrases

The conditions that RPASCE automatically instantiates behind the scenes are single time-based conditions where the current keyword (which translates to the index number of the current position in the time dimension) is compared (using one of the comparison operators `==, !=, >, <, >=, <=)` directly with one of the following keywords: first, last, elapsed, or today. For example, the expression

```
if(current > elapsed, x, y)
```

will have the condition automatically instantiated. However, the expression

```
if(current > elapsed + leadtime, x, y)
```

where lead time is a measure, will not have the condition automatically instantiated, and the condition current > elapsed + leadtime may need to be instantiated into a measure for improved performance.

Similarly, an expression such as

```
if(current > elapsed && current < last, x, y)
```

is not automatically evaluated by instantiating the condition. Either of the sub-conditions would be evaluated by instantiation if they stood alone, but when they are combined, RPASCE cannot automatically evaluate them. The efficient way to write such conditions is to instantiate the result of each sub-condition as described in Caching Intermediate Results.

This specific expression could be written as

```
z = current > elapsed && current < last
```

```
if(z, x, y)
```

Tips

The following sections describe tips for writing efficient rules in RPASCE.

- Rule Groups
- Non-Materialized Measures
- Display-Only Non-Materialized Measures
- The If Statement

Rule Groups

The RPASCE engine recognizes rules that are the same in the rule group being transitioned from and to by rule name. Therefore, it is important to ensure that the same rule (and not a copy of the rule with a different name) is used in both rule groups.

Non-Materialized Measures

Non-materialized measures must be used whenever possible. These measures are not necessarily calculated as part of the calculation cycle, but rather they are calculated only if needed to evaluate another expression. Therefore, these non-materialized measures have the potential to significantly decrease the regular calculation time. Note that several different types of non-materialized measures are available, and their usage and performance profile can vary.

Display-Only Non-Materialized Measures

Display-only non-materialized measures are intended for display-only use. They cannot be manipulated and cannot be used in calculations. They must have an aggregation type of recalc, and they must be at the very end of a calculation. Since they cannot be used in calculations (other than the one used to calculate them), they do not need to be calculated by the calculation engine during a normal calculation. They are therefore ignored during the calculate operation. They are calculated only when required and only for the positions required at the time. Typically, the calculation is initiated during the fetch cycle when the measure is to be displayed.

Normally, all measures that are candidates to be display-only non-materialized measures should be defined that way. The exceptions to this rule are when the measure is likely to be viewed much more frequently than it would be calculated, such that calculating it normally through the calc cycle would provide an overall saving.

The If Statement

The following sections describe the If statement.

- Caching the If Condition Phrase
- The Ignore Keyword

See Expression Iteration Examples for examples that show how the If statement is iterated.

Caching the If Condition Phrase

The simple way for RPASCE to iterate over the cells where a condition is true, is for that condition to be instantiated into a Boolean measure with a naval of false. This allows RPASCE to iterate over just the populated cells. Otherwise, the engine must pass all intersections to determine whether the condition is true, which may be a very expensive operation. There are some specific time-based conditions where RPASCE instantiates the condition behind the scenes (see the “Automatic Caching of Expression Phrases section for these examples). Otherwise, you must instantiate the result of evaluating the condition to have efficient calculations. This can be done by setting a measure to the result of the condition. This is especially important if the same condition is used repeatedly in expressions or if the input measures used to calculate the condition change infrequently.

This is not necessary if the condition phrase meets the requirements to be automatically cached (see Automatic Caching of Expression Phrases for more information).

The Ignore Keyword

The If function supports an Ignore argument. This argument means that the calculation must not change the current value of the cell. Under some circumstances, expressions that use this construct in full evaluation mode can be particularly expensive to evaluate, so it should be used only when necessary. For instance, with an expression such as:

```
a = if(b, ignore, 1)
```

where the naval of a is 0, and the naval of b is false, the effective naval of the expression is 1. RPASCE cannot simply change the naval of a from 0 to 1 because of the ignore. Therefore, RPASCE must iterate over all logical cells of b. This problem happens when using ignore whenever the naval of the expression is different from the naval of the measure being calculated.

Note:

Using multiple ignore keywords in the if expressions should be avoided while configuring expressions as such expressions can cause performance issues.

Expression Iteration Examples

This section demonstrates how RPASCE iterates over various expressions. Use Table H-1 as a reference for evaluating the relative performance of an expression due to iteration. None of these expressions are necessarily bad. The relative populated/logical size and naval of the measures must be taken into consideration to determine if the performance can be improved.

The naval of all measures are 0 or false unless otherwise noted. Measures are 10% populated unless otherwise noted. The calendar dimension has 100 positions.

Table H-1 Evaluating the Relative Performance of an Expression Due to Iteration

Sample Expression	Iteration Based On	Notes
N1 = if(B1, 0, 1)	B1	This is the simplest case, iteration-wise.
N1 = if(B1, 0, N2)	N2	Navalue of N2 equals navalue of 0. N2 and 0 has a fill factor of 10%, while N2 combined with B1 has a combined fill factor of 19%. Therefore, iterate just N2.
N1 = if(B1, 0, N2) Same as previous but navalue of N2 is 1.	B1, N2	Navalues do not match.
N1 = if(B1, N2, N3) (N2 fill factor = 30%)	B1, N3	N2 x N3 fill factor: 1 – (1 - .3) * (1 – 0.1) or 37%. B1 x N3 fill factor: 1 – (1 – 0.1) * (1 – 0.1) or 19%. Thus, iterate B1 and N3.
N1 = if(B1, N2, 0)	B1 (N2 could be chosen if it had a lower fill factor).	Navalue of if is 0.
N1 = prefer(10 / N2, N3)	N2, N3	Navalue of first prefer subexpression causes error, so no optimization applies here.
N1 = prefer(10 / (N2 + 1), N3)	N2	Navalue of first prefer subexpression does not cause error, so optimization applies here.
N1 = if(B1, ignore, 0)	B1, N1	Expression navalue is 0.
N1 = if(B1, 0, ignore)	B1	Expression navalue is ignore.

Table H-1 (Cont.) Evaluating the Relative Performance of an Expression Due to Iteration

Sample Expression	Iteration Based On	Notes
N1 = if(B1, ignore, N2)	B1, N2	Because B1's navalue is false, the RHS navalue comes from N2. No optimization applies here. However, if the navalue of the RHS (in this case, N2) does not match the pre-existing navalue of N1, then “all cells are dirty" mode is invoked to avoid changing the navalue of N1 (and thus changing the value of unpopulated cells that should have been ignored).
N1 = if(B1, N2, ignore)	B1	Expression navalue is ignore.
N1 = if(B1, N2, ignore) + if(B2, ignore, N2)	B1, B2, N2	Expression navalue is ignore. **Note:** RPASCE does not support the syntax if() + if(). Although, it still illustrates a useful point from an iteration point of view.
N1 = current	All logical cells	current is not used in a comparison so no optimization applies here.
B1 = (current == first)	first calendar position	current is used in a comparison so simple comparisons to current apply here.

N1 = if (current > elapsed, N2, N3)
elapsed = 25; N3 is 50% populated
	First 25 calendar positions and N2.	Fill factor of N2 x N3 is 55%. Fill factor of current > elapsed x N2 is 33%. Therefore, iterate current > elapsed and N2.

Tips to Design Efficient RPASCE Expressions

Sometimes fairly simple expressions take a very long time to run. Most of the time, the problem exists in the way the expressions are designed and configured. The following is an attempt to explain how to design efficient expressions.

1. Consider the following nested if expression:

```
L1 = if (R1 > 0, V, if (R2 > 0, V, if (R3 > 0, V, if( R4 > 0, V,
if( R5 > 0, V, 0)))))
```

Table H-2 Nested if Expression

Measure	Type	BaseInt
L1	Real	SKU/STR/DAY
R1, R2, R3, R4, R5	Real	DEPT/DAY
V	Real	SKU/STR

The application is partitioned on DEPT where SKU rolls up to DEPT.

This expression is not designed correctly and may have a performance hit because the left-hand side (LHS) measure L1 is at SKU/STR/DAY, the R1, R2, R3, R4, R5 measures are at DEPT/DAY, and the V measure is at SKU/STR. Here, you need to first spread the R1, R2, and so on, measures from DEPT/DAY to SKU/STR/DAY and the V measure from SKU/STR to SKU/STR/DAY. Spreading is a time-consuming process, but any effort to reduce it within an expression should give a performance benefit.

If you redesign the expression as follows, performance can be improved:

a. Register a temporary measure, such as tmpmask, which is at SKU/DAY.

b. Add an expression to generate the tmpmask measure as follows:

   ```
   tmpmask = R1 > 0 || R2 > 0 || R3 > 0 || R4 > 0 || R5 > 0
   ```

c. Modify the previous expression as follows:

   ```
   L1 = if (tmpmask, V, 0)
   ```

d. Those two expressions will only take a fraction of the time taken to run the original expression.

2. Consider an expression which uses the RPASCE prefer function:

   ```
   L2 = prefer(A/B, 0)
   ```

Table H-3 Prefer Expression

Measure	Type	BaseInt
L2	Real	SKU/STR/DAY
A	Real	SKU/STR/DAY
B	Real	SKU

Here also, there is a lot of spreading to be done for the B measure from SKU to SKU/STR/DAY.

There are two approaches to optimize this expression:

a. Make use of the RPASCE CalcEngine's division by zero support where it will now return a 0 when it encounters a division by 0. It effectively is mimicking the prefer behavior and it evaluates faster than prefer.

   The function can be revised as:

   L2 = A / B

b. When the preferred value is not equal to 0, then use the following approach as the RPASCE CalcEngine only returns zero in division by 0 situations.

   If baseint of B is much higher than A, use a temporary intermediate measure.

   Since B is at SKU and A is at SKU/STR/DAY, use an intermediate measure C at either SKU/STR or SKU/STR/DAY and spread B to C using the expression:

   C = B

   Then, modify the prefer expression as follows:

   L2 = prefer(A/C, 5)

   The spreading is much less when C is used inside prefer compared to B and it should evaluate faster than:

   L2 = prefer(A/B, 5)