Oracle® Retail Demand Forecasting

Configuration Guide

Release 14.1

December 2014

ORACLE®

Oracle Retail Demand Forecasting Configuration Guide, Release 14.1

Primary Author: Melissa Artley

condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, “alteration” refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

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

Contents

B Appendix: Configuring the Clone Procedure

C Appendix: Configuring the Forecast Procedure

D Appendix: AppFunctions

E Appendix: Configuring the Preprocess Special Expression

List of Figures

List of Tables

Send Us Your Comments

Oracle Retail Demand Forecasting Configuration Guide, Release 14.1.

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

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

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

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

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

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

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

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

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

Preface

Oracle Retail Configuration Guides are designed so that you can view and understand the application's behind-the-scenes processing, including information for key system administration configuration settings.

Audience

Anyone who has an interest in better understanding the inner workings of the RDF system can find valuable information in this guide. There are three audiences in general for whom this guide is written:

- System analysts and system operation personnel:
- who are looking for information about RDF processes internally or in relation to the systems across the enterprise.
- who operate RDF on a regular basis.
- Integrators and implementation staff who have the overall responsibility for implementing RDF into their enterprise.
- Business analysts who are looking for information about processes and interfaces to validate the support for business scenarios within RDF and other systems across the enterprise.

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 Demand Forecasting Release 14.1 documentation set:

- *Oracle Retail Demand Forecasting Configuration Guide*

- *Oracle Retail Demand Forecasting Implementation Guide*
- *Oracle Retail Demand Forecasting Installation Guide*
- *Oracle Retail Demand Forecasting Release Notes*
- *Oracle Retail Demand Forecasting User Guide for the RPAS Classic Client*
- *Oracle Retail Demand Forecasting User Guide for the RPAS Fusion Client*
- Oracle Retail Predictive Application Server documentation

The following documentation may also be needed when implementing RDF:

- *Oracle Retail Planning Batch Script Architecture Implementation Guide*

Supplemental Documentation

The following document is available through My Oracle Support at the following URL:

https://support.oracle.com

Oracle Retail Demand Forecasting 14.1 Cumulative Fixed Issues (Note ID 1672842.1)

This document details the fixed issues and defects for all RDF, Curve, and Grade patch releases prior to and including the current release.

Customer Support

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

https://support.oracle.com

When contacting Customer Support, please provide the following:

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

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 14.1) or a later patch release (for example, 14.1.1). If you are installing the base release, additional patch, and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail

document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

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

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

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

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

Oracle Retail Documentation on the Oracle Technology Network

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

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

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

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

Conventions

The following text conventions are used in this document:

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

1

Configuring the Retail Demand Forecasting Solution

Oracle Retail Demand Forecasting (RDF) is a statistical forecasting solution that uses state-of-the-art modeling techniques to produce high quality forecasts with minimal human intervention. Forecasts produced by RDF enhance the retailer's supply-chain planning, allocation, and replenishment processes, which enables a profitable and customer-oriented approach to predicting and meeting product demand.

Forecast information is often required for items at the lowest levels in a hierarchy. Problems can arise when historic sales data for these items is too sparse and too noisy to identify clear selling patterns. In such cases, generating a reliable forecast requires aggregating sales data from a lower level up to a higher level in the hierarchy. After a forecast is generated at the higher level, the resulting data can be allocated (spread) back down to the lower level. This is based on the lower level's relationship to the total. Before you can spread forecast data back down to a lower level, you should have an understanding of the relationship between the lower level and the higher level dimensions. Frequently, an additional forecast will be generated at the lower level to help determine this relationship. This lower level is called the final forecast level. Forecast data at this level might be sufficient to generate reliable percentage-to-whole information, but the actual forecast numbers will be more robust when they are generated at an aggregate level. This aggregate level from which forecast data is spread is referred to as the source forecast level.

Some high-volume items may possess sufficient sales data for robust forecast calculations directly at the final forecast level. In these cases, forecast data that is generated at an aggregate level and then spread down to lower levels can be compared to forecasts that are run directly at the lower level. Comparing the two forecasts, each generated at a different hierarchy level, can be an invaluable forecast performance evaluation tool.

The RDF solution may include multiple final forecast levels. Forecast data must appear at some final level for the data to be approved and exported to other systems.

Using the RDF plug-in, final and source forecast levels are defined for the RDF solution.

Note: The ability to configure the RDF solution may be limited. This is based on your licensing agreement.

Forecasting Calendar Hierarchy Requirement

With any RDF solution, configuration of the calendar hierarchy must always include a day dimension level name. There are no configuration requirements for the dimensions of the merchandise or location hierarchies.

Forecasting Limitations Using the Partition Hierarchy

Any dimension along the partition hierarchy that is used as an intersection to forecast must be unique across all domains. This requirement especially applies to Alternate Hierarchies. For example, if the forecast level is supplier\str\week, my Supplier dimension cannot have a supplier position that exists in multiple domains. However, additional support for clean partitioning of Alternate Hierarchies is provided through the RDF Transformation programs used to integrate RMS foundation data for RDF. See the *Oracle Retail Predictive Application Server Administration Guide for the Classic Client* for more information on data integration programs.

Forecasting Pre-Configuration Data Requirements

There are several parameters within the RDF configuration that may reference other measures that are configured external to the solution, specifically:

- Source Data
- Plan Data
- Spreading Profile
- Seasonal Profile

Prior to configuring an RDF solution, it is required that these measures already exist within the Project.

Source Data

The RDF plug-in populates a pick-list with all non-Boolean and non-string measures that have been created in the Project.

Spreading Profiles and Seasonal Profiles

If Curve will be used to produce Spreading Profiles or Seasonal Profiles to support your Forecasting solution, these profiles should already have been configured in the Curve solution. If these profiles are being defined external to Curve, these measures should already exist within the Project.

Plan Data

If the Plan Data that will be used to support Bayesian forecasting is being defined within another solution, this measure should already exist. The entry of this parameter is not required within the configuration, and it can be entered in the resulting domains.

Registering the RdfFunctions Library

Prior to configuring the RDF Solution, register the RdfFunctions library to support proper validation of the RDF-specific rules:

Open the Function Library Manager and add RdfFunctions.

Note: If Promote is implemented, the following rules will display as invalid; however these should be ignored:

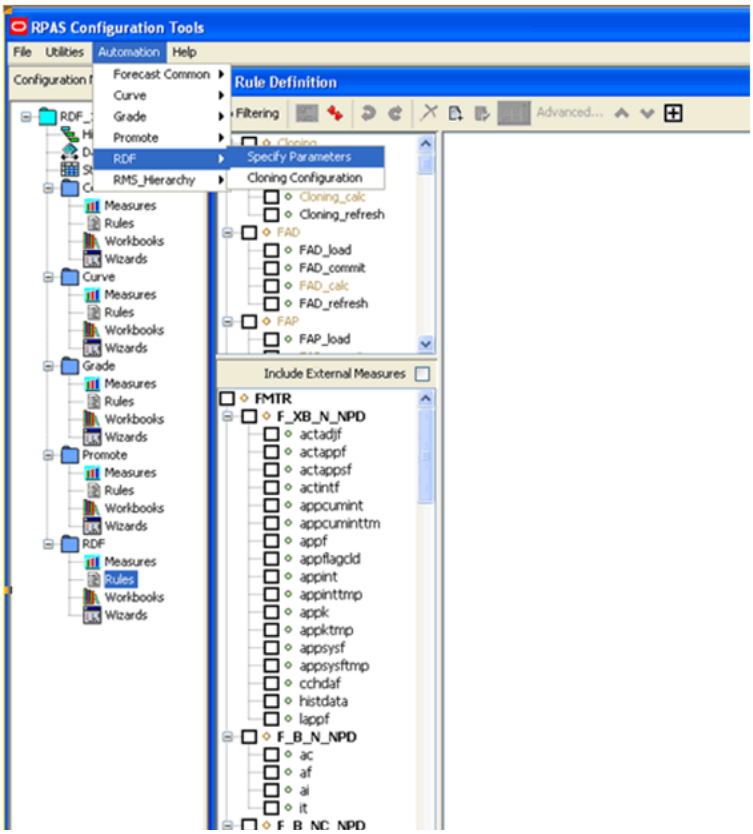
- Rule: PREF_PlHolder
- RuleGroup: PREF_place
- Rule Group: PRMA_place
- Rule Group: PRPL_place

Creating an RDF Solution Extension

To create an RDF solution extension:

1. Open an existing configuration in which the Curve solution has already been defined.
2. From the Configuration Tools toolbar, select the **Automation** menu. From the **RDF** option, select **Specify Parameters**. The following sections outline the process for configuring forecast levels.

Figure 1–1 Configuration Tools Menu Options



Configuring a Final Forecast Level

On the Forecasting Parameters utility, click the **F** icon.

1. A new final level is added, and it is assigned the next available level number.

2. Specify the properties for the final level. See Editing Forecast Level Parameters for details.

Configuring a Source Forecast Level

To create a source level, complete the following steps:

1. On the Forecasting Parameters utility, highlight the final level number in which the new source level will be associated from the Level window.
2. Click the **S** icon.

   A new source level is added, and it is assigned the next available number.
3. Specify the properties for the source level. See Editing Forecast Level Parameters for details.

Note: A new final forecast levels cannot be added or patched into an existing domain, however a new source forecast level can be added to the configuration and patched to an existing domain. Final and source levels already existing in a domain, can not be removed from the domain.

Editing Forecast Level Parameters

Edit forecast parameters:

Parameter	Description
Level Name	The Level Name is the system-assigned level number when a forecast level is created. This is a read-only parameter.
Level Label	The Level label is the level description that will be viewed by the user once the domain is created. ▪ Level labels may not exceed 40 characters. ▪ It is recommended, but not required, that Level labels include the Level Name (the system-assigned level number). Within the Forecast Administration workbook, the Default Source Level may be edited. This pick-list is populated with the Level Name for all levels that are associated with a final level. Since this value can also be specified within this configuration, this recommendation may not be necessary if changes to the Default Source Level are not expected within the application. ▪ RPAS automatically places parentheses **( )** around Forecast Level labels. The configuration specialist should not include these in the level label configuration or the installer will fail. An example of a Forecast Level label that would violate this requirement is (1:itm/str/week - Final). This example is acceptable: 1-item/str/week - Final. ▪ A hyphen '**-**' should not be used before or after the Forecast Level label. An example of a Forecast Level label that would violate this requirement is: -1:itm/str/week - Final-. This example is acceptable as: 1-itm/str/week - Final ▪ A colon '**:**' should not be used at all in the Level label. An example of a Level label that would violate this requirement is 1: itm/str/week-
Intersection	The Intersection is the hierarchy dimensions that define the forecasting level.

Parameter	Description
Default Source Level	Assigned only at the Final level, the Default Source Level is the primary level at which the aggregate, more robust forecast is run. The desired Source Level must first be created within the RDF configuration for it to be a selection in the pick-list. For more information on Source Level Forecasting, refer to the *Oracle Retail Demand Forecasting User Guide*. If no source level is required, the final level should be selected.
Source Data	Assigned only at the Final level, the Source Data is the measure to be used as the input data (for example, POS) for the generation of forecasts. The values in this pick-list are populated with all non-string and non-Boolean type measures that are configured in the Project.
Periodicity	Periodicity is the number of periods within the Calendar dimension, which are defined in the forecast level intersection. For example, if an intersection is defined at Week/item/store, the Periodicity value will be 52 (since there are 52 weeks within a year).
Forecast Method	The Forecast Method window displays all forecast generation methods that may be defined for a forecast level. The Default Forecast Method is also determined here. For additional information, see Selectable Forecast Methods.
Plan Data	Assigned only at the final level, Plan Data (sales plans) provide details of the anticipated shape and scale of an item's selling pattern. This information is required when Bayesian or Load Plan is used as a Forecast Method. The value in this parameter is a measure name.
Seasonal Profile	A seasonal profile provides details of the anticipated seasonality of an item's selling pattern. The seasonal profile is required in conjunction with the Profile-based Forecast Method. The seasonal profile can be generated or loaded, depending on your configuration. The value in this parameter is a measure name. The intersection of the seasonal profile measure must be the same the the intersection of the forecast level.
Spreading Profile	Assigned only at the source forecasting level, the Spreading Profile is used to spread source level forecasts down to the final forecast level. The value in this parameter is a measure name, a profile level name, or any combination of these separated by commas. ■ If Curve is used to dynamically generate the spreading ratios, this parameter should be populated with the final profile level name (profile number) configured. For example: 01 (this is profile level 01). ■ If Curve is used to generate the static (manually approved) spreading ratios, this parameter should be populated with the Approved Profile measure. For example: apvp11 (this is the Approved Profile for Curve level 11).

Note: For more information on Source Level Forecasting, see the *Oracle Retail Demand Forecasting User Guide*.

Selectable Forecast Methods

The following is a list of Forecast Methods that may be selected. See the *Oracle Retail Demand Forecasting User Guide* for more information on each method.

Note: In this chapter, see About Causal for additional information.

- No Forecast
- Average
- Moving Average
- Simple
- Intermittent
- Simple/Intermittent
- Trend
- Additive Seasonal
- Multiplicative Seasonal
- Seasonal
- AutoES
- Causal
- Bayesian
- Profile-based
- Load Plan
- Copy

About Causal

The Causal method should be selected as a valid method only for levels in which causal forecasting will be used.

This method should only be selected as a valid method for levels that will use Causal Forecasting. If Causal is selected and Promote is not licensed or configured, the RDF batch forecast will not generate.

When enabling Causal as a valid forecast method for a source level, note that RDF Promotion variables need to be provided at the same dimension along the product and location hierarchies as the forecast level for which Causal forecasting is run (Final or Source). RDF Causal does not support aggregation of promotion variables along any hierarchies other than Clnd. Aggregation of promotion variables along product and/or location hierarchies needs to be handled externally through configuration. Aggregation along the calendar hierarchy is support by RDF Causal, using specified aggregation and spread profiles. Refer to the *Oracle Retail Demand Forecasting User Guide* for details.

Autogenerating Hierarchies, Measures, Rules and Workbook Templates

The following is the process to autogenerate the hierarchies, measures, rules, and workbook templates that are required by RDF to support the forecasting configuration entered in the RDF plug-in:

On the Forecasting Parameters utility, click **OK**.

The system automatically generates the following:

Autogenerated Item	Description
Hierarchies	The DATA hierarchy will be updated with the flvl, fbrt and fmtr dimensions.

Autogenerated Item	Description
Measures	All measures necessary to support the base RDF solution will be created.
Rules	All Rule Sets, Rule Groups, and Rules to support the base RDF solution will be created.
Workbook Templates	All pre-defined workbook templates to support the base RDF solution will be created.

You may continue to make changes to the RDF plug-in configuration, and the autogeneration process may be repeated as often as needed prior to the installation.

Deleting a Forecast Level

Deleting a forecast level will cause the system-assigned enumerated values in the Level Name to renumber such that levels are in consecutive order starting with forecast level 01. Deleting a forecast level may impact any solution configuration that uses a specific level.

If the domain using the configuration has previously been installed, there is potential to lose data associated to a level that has been deleted or to be renumbered.

To delete a level:

On the Forecasting Parameters utility, highlight the number of the level that you want to delete from the Level window.

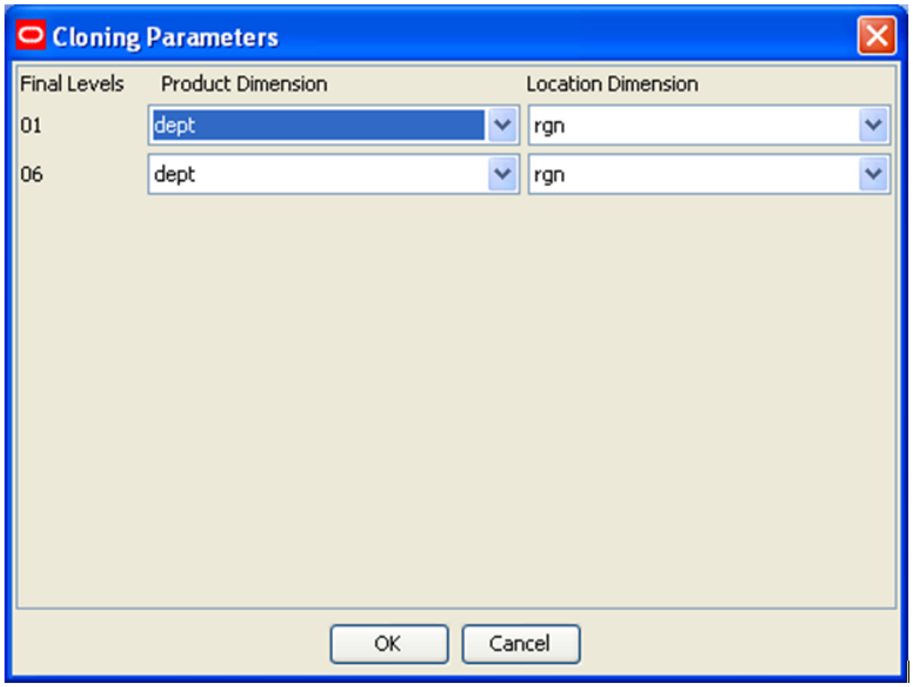
1. Click **X**.
2. The level is deleted. If you delete a final level, any source levels that are associated with it will also be deleted.
3. Select **OK** to regenerate the solution with the changes to the cluster configuration.

Configuring the Cloning Administration Workbook

Product/Location Cloning Administration workbook allows users to specify clone products by a configurable dimension in the location hierarchy and clone stores by a configurable dimension in the product hierarchy. For example, users can specify a different clone item for a different region.

These dimensions can be specified from the **Cloning Configuration** menu option under **RDF Automation**. When the user clicks on this menu, the Cloning Parameters dialog box appears.

Figure 1–2 Cloning Parameters Dialog Box



For each Final forecast level, the user is prompted to select a product dimension and a location dimension. The values selected here drive the dimensionality of the Product Cloning and Location Cloning Worksheets in the Cloning Administration Workbook. Note that the product dimension selected here actually drives the Location Cloning Worksheet and the location dimension drives the Product Cloning Worksheet. For example, the product dimension is the dimension by which clone Users want to specify location clones and vice versa.

For example, if final level 01 is at item/store/week and the user has chosen dept. for product dimension and region for location dimension, then for Final level 01 the Product Cloning Worksheet will be generated at item/region and the Location Cloning Worksheet will be at store/dept.

Note that if the Cloning configuration menu option is not invoked, then the Cloning Administration Workbook and associated measures will not be generated in the configuration.

Editing the RDF GA Configuration

The autogeneration process creates hierarchies, measures, rules, and workbook templates that are required to support the essential RDF functionality. This base configuration is referred to as the GA Configuration. Certain changes to the GA Configuration are allowed. Once edits to the GA Configuration are made and the autogeneration process occurs again, valid changes to the configuration will be preserved. There is nothing in the RPAS Configuration Tools to prevent invalid changes from being made.

Note: When a custom taskflow exists in the configuration, the plug-in automation may cause the workbook template field to be blank. If this occurs, save the configuration and then re-open the configuration and the workbook template field is populated.

The following table outlines acceptable changes and restrictions:

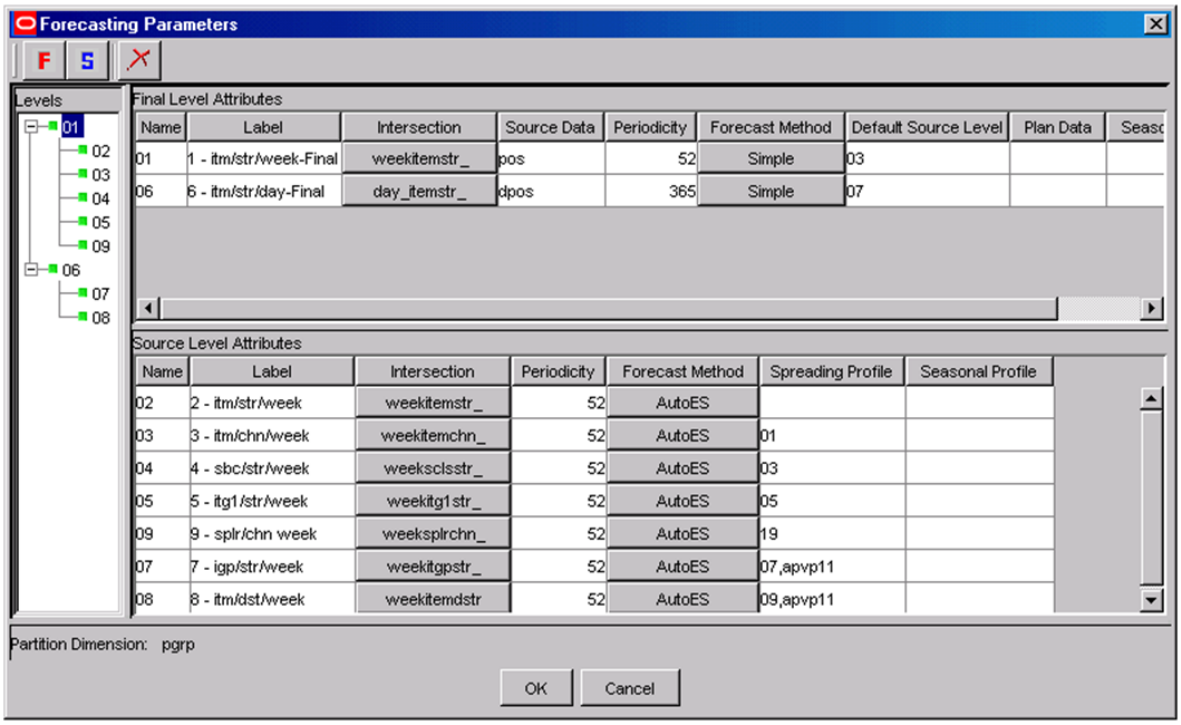
Changes and Restrictions	Description
RDF Solution Extension Name	The name assigned to the resulting RDF solution after autogeneration occurs cannot be edited.
Major and Minor Components	Additional Major and Minor components may be added to the RDF GA Configuration. The Major and Minor components that are part of the GA Configuration may not be edited. This restriction also applies to Measure Names and Measure Labels.
Rules	Additional Rule Sets, Rule Groups, and Rules may be added to the RDF GA Configuration. This includes support for adding new Rules to existing GA Configuration Rule Groups. It is recommended that new Rules added to the GA Configuration Rule Groups include cust (represents Custom) in the Rule Name. This allows for easy identification of Rules that are not part of the GA Configuration. Rule Sets, Rule Groups, and Rules that are part of the GA Configuration may not be renamed. Existing Rules that are part of the GA Configuration may not be modified in any way.
Workbook Templates	Additional Workbook Templates may be added to the RDF GA Configuration. New Measures and Rules may also be added to the GA Configuration Workbook Templates. This is done by adding new Major and Minor components, and adding new Rules to existing Rule Groups in the GA Configuration.

RDF Example

Figure 1–3 shows an example of the Forecasting Parameters utility which is used for:

- Configuring a Final Forecast Level
- Configuring a Source Forecast Level
- Editing Forecast Level Parameters
- Autogenerating Hierarchies, Measures, Rules and Workbook Templates
- Deleting a Forecast Level

Figure 1–3 Forecasting Parameters Window



2

Configuring the Promote Solution

Promote (Promotional Forecasting) is an optional add-on solution to RDF that allows for the effects of promotional and causal events, such as radio advertisements and holiday occurrences, into time series forecasts. The promotional forecasting process uses past sales data and promotional information to forecast future demand.

Using the Promote plug-in, promotions are defined that will be used within the Promote Solution.

Creating a Promote Solution Extension

To create the Promote solution extension:

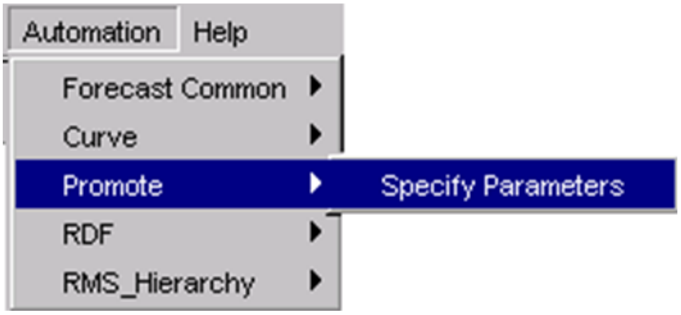
1. Open an existing configuration in which the Curve and RDF solution have already been defined.

   Note: Promote Automation must be run last - after RDF and Curve Automation.

   Note: Promotion/causal forecasting levels are determined within the RDF Solution by selecting Causal as a valid Forecasting Method for source or final forecasting levels.

2. From the Configuration Tools toolbar, select the Automation menu. From the Promote option, select Specify Parameters. The following sections outline the process for configuring forecast levels.

Figure 2–1 Specify Parameters



Creating a Promotion

To create a promotion, complete the following steps.

1. On the Promote Parameters utility, click the P icon.

   A new promotion is added, and it is assigned a default promotion number for the Promotion Name (for example, P001).

2. Specify the properties for the promotion. See Editing Promote Parameters for details.

Editing Promote Parameters

Edit the promotion parameters:

Table 2–1 Promote Parameters

Parameter	Description
Default Intersection	The Default Intersection is the intersection at which any new promotion will be defined. Editing the Default Intersection will not affect any existing promotions.
Promotion Name	The Promotion Name is the internal system identifier of the promotion. The system will initially assign a generic Promotion Name (P001), but this value may be overwritten. The Promotion Name may not be greater than four characters. The following characters may not precede or follow the name that is entered in this field: '( )' Example: (xmas) '-' Example: -xmas- The following must not be used at all in the Promotion Name: ':' Example: xmas:

Table 2–1 (Cont.) Promote Parameters

Parameter	Description
Promotion Label	The Promotion Label is the description of the promotion that will be viewed by the user once the domain is created. Promotion Labels may not exceed 40 characters. The following characters may not precede or follow the label that is entered in this field: '**(**)' Example: (xmas) '**-**' Example: -xmas- The following must not be used at all in the Promotion Name: '**:**' Example: xmas:
Promotion Intersection	Independent of the causal forecasting levels, the Promotion Intersection is the hierarchy dimension that defines the promotion. It is pre-populated with the value set in the Default Intersection at the time when the promotion is created.
Type	The Type is the data type of the promotion variable. Promotion Variables may be defined as Boolean or Real types. The value in this parameter defaults to Boolean.
Model	Model is the model type that the promotion variable is applied into. Model Types maybe defined as Linear or Exp. (Exponential). The value in this parameter defaults to Linear.
Database	The Database displays the database that will be used to store promotion variable information. The value in this parameter defaults to the data/promo database.
PvarDataBase	The PvarDataBase is the database used to store promotion variable information. The value in this parameter defaults to the data/promo database.

Setting Promotion Variable Type

Promotion Variable Type is defined through setting both Type and Model.

Note: Real Exponential and Real Linear promotion variables can not be enabled at the same time for a given forecast level. Boolean with either Real Linear or Real Exponential promotion variables are allowed.

When the Type is	And the Model is	Then the Promotion Variable Type is
Boolean	either Linear or Exponential	Boolean
Real	Linear	Real
Real	Exponential	Exponential

Autogenerating Hierarchies, Measures, Rules and Workbook Templates

The following is the process to autogenerate the hierarchies, measures, rules, and workbook templates required by Promote to support the promotion configuration entered in the Promote plug-in:

- On the Promote Parameters utility, click **OK**.

The system automatically generates:

Autogenerated Item	Description
Hierarchies	The DATA hierarchy will be updated with the ptyp and prom dimensions.
Measures	All measures necessary to support the base Promote solution will be created.
Rules	Only the rules and rule groups necessary to support the installation of the Promote solution are visible in the configuration. Unique to Promote, the additional rules and rule groups needed to support the Promote workbook templates and batch forecast are generated within the domain and not within the plug-in.
Workbook Templates	All pre-defined workbook templates to support the base Promote solution will be created; however, the worksheets are not visible. Unique to Promote, the additional workbook templates needed to support the Promote solution are generated within the domain and not within the plug-in. You may continue to make changes to the Promote plug-in configuration, and the autogeneration process may be repeated as often as needed prior to the installation.

Note: After autogeneration completes, the following rules will display as invalid; however these should be ignored:

Rule: PREF_PlHolder

RuleGroup: PREF_place

Rule Group: PRMA_place

Rule Group: PRPL_place

Deleting a Promotion

Deleting a promotion may impact any solution configuration that references the deleted promotion. To delete a promotion:

1. On the Promote Parameters utility, highlight the promotion to delete from the configuration.
2. Click **X**. The promotion is deleted.
3. Select **OK** to regenerate the solution with the changes to the cluster configuration.

Note: See the *Oracle Retail Demand Forecasting Implementation Guide* for more information on patchable changes to the configuration.

Editing the Promote GA Configuration

The Promote autogeneration process creates all hierarchy dimensions and measures to support the essential Promote functionality; however, only the rules and workbook templates required to support the domain installation are visible in the configuration. Unique to Promote, the additional rules, rule groups and workbook templates needed to support the Promote solution and batch forecast are generated within the domain and not within the plug-in.

Note: When a custom taskflow exists in the configuration, the plug-in automation may cause the workbook template field to be blank. If this occurs, save the configuration and then re-open the configuration and the workbook template field is populated.

Note: This limitation allows for fewer options than in RDF and Curve for edits to the GA Configuration.

The following table outlines acceptable changes and restrictions:

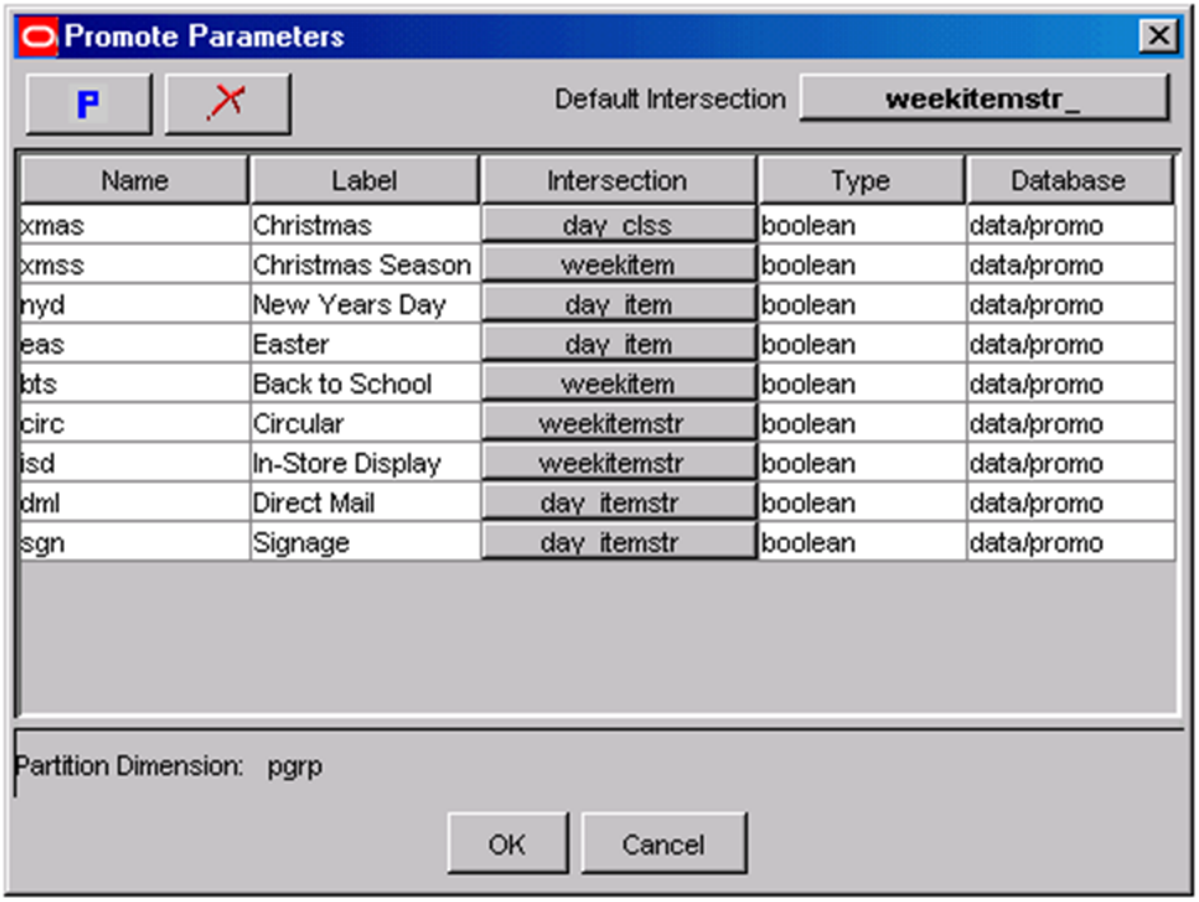
Changes and Restrictions	Description
Promote Solution Extension Name	The name assigned to the resulting Promote solution after autogeneration occurs cannot be edited.
Major and Minor Components	Additional Major and Minor components may be added to the Promote GA Configuration. The Major and Minor components that are part of the GA Configuration may not be edited. This restriction also applies to Measure Names and Measure Labels.
Rules	Additional Rule Sets, Rule Groups, and Rules may be added to the Promote GA Configuration. This includes support for adding new Rules to existing GA Configuration Rule Groups. It is recommended that new Rules added to the GA Configuration Rule Groups include cust (represents Custom) in the Rule Name. This allows for easy identification of Rules that are not part of the GA Configuration. Rule Sets, Rule Groups, and Rules that are part of the GA Configuration may not be renamed. Existing Rules that are part of the GA Configuration may not be modified in any way.
Workbook Templates	Additional Workbook Templates may be added to the Promote GA Configuration; however, new Measures and Rules cannot be added to the GA Configuration Workbook Templates because the Promote worksheets are not visible in the configuration.

Promotion Example

Figure 2–2 shows an example of the Promote Parameters utility that is used for:

- Creating a Promotion
- Editing Promote Parameters
- Autogenerating Hierarchies, Measures, Rules and Workbook Templates
- Deleting a Promotion

Figure 2–2 Promote Parameters Window



3

Configuring the Curve Solution

Curve is an RPAS solution that is used to generate ratios from historical data at user-specified intersections. The profiles generated by Curve can be used for various purposes:

- To convert the organization-level assortment plans into base level weekly sales forecasts
- For generating seasonal forecasts, daily forecasts, or new product forecasting using lifecycle profiles

RDF requires profiles (created by Curve) to determine how a source level forecast (for instance, Item/Chain/Week) is spread down to the execution or final level (for instance, Item/Store/Day). Profiles are generated using historical data and phase definitions that are based on the system configuration. Using the Curve Plug-In, profiles are defined to support the Curve solution.

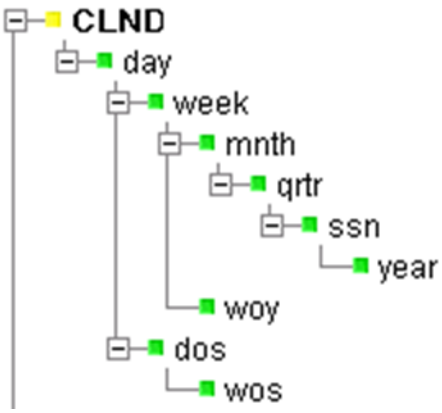
Note: For information on building the Curve domain, refer to the *Oracle Retail Predictive Application Server Installation Guide*.

Curve Hierarchy Configuration Requirements

The following dimensions are required to support different seasonal profiles. If the following defined types of profiles are not required, these hierarchy dimensions may not be necessary:

- **dos** (day of season) - A dimension off day, dos is used to support seasonal profiles that are normalized to day. This profile should use the Daily Seasonal profile type.
- **wos** (week of season) - A dimension off day or dos, wos is used to support seasonal profiles normalized to week. This profile can use the Store Contribution, Product Profile, or User Defined profile types.
- **woy** (week of year) - A dimension off week or year, woy is used to support weekly seasonal profiles normalized to year. This profile can use the Store Contribution, Product Profile, or User Defined profile types.

Figure 3–1 Additional Hierarchy Requirements for Season Profiles

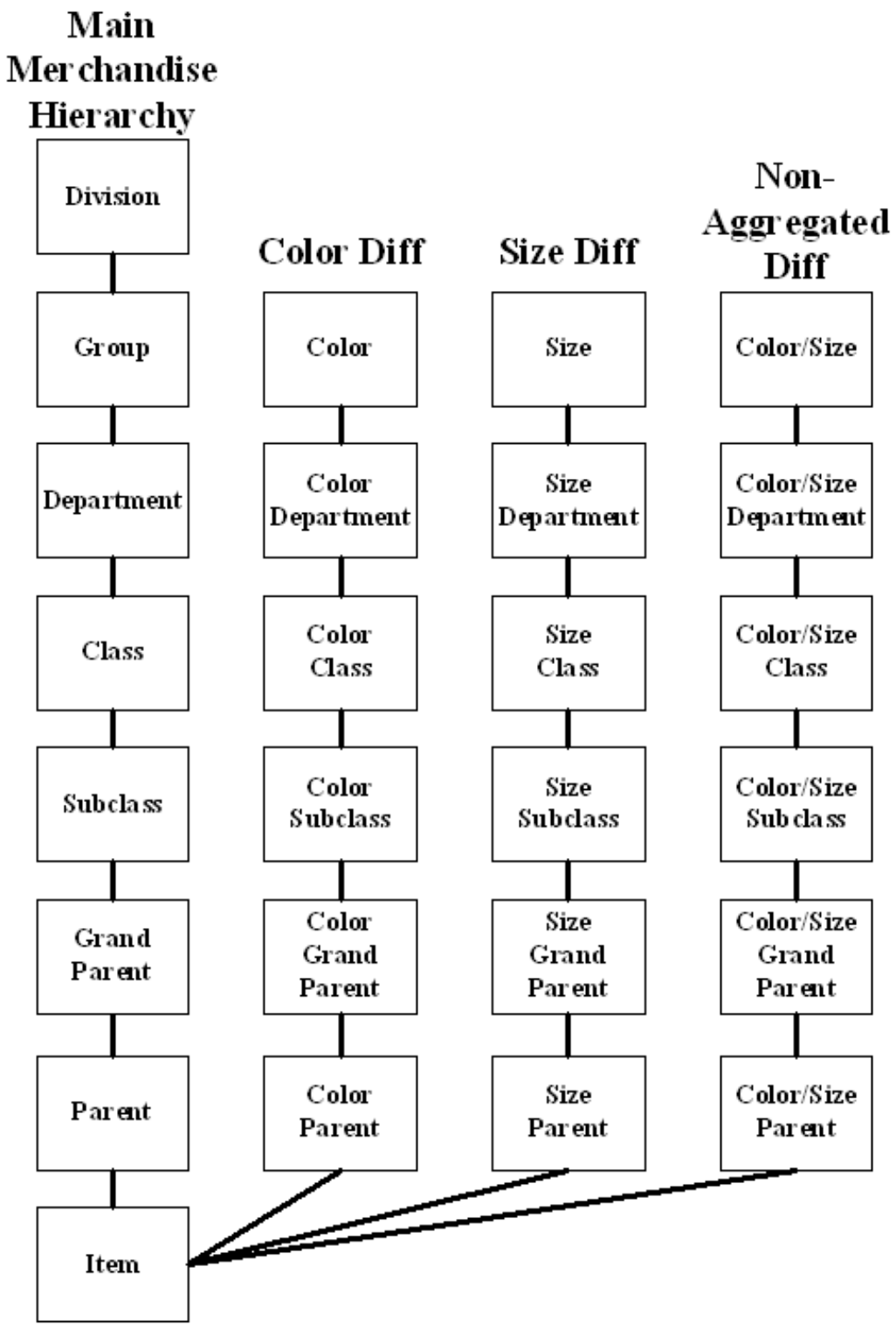


Configuring Curve Differentiator Dimensions (optional) and Merchandise Hierarchy Requirements to Support RMS

Configuring Differentiator dimensions (also referred to as Diff dimensions) within the merchandise hierarchy is optional. Differentiator dimensions allow the merchandise dimensions to be distinguished based on an alternative attribute property such as Color, Size, Flavor, or other attributes properties that are required to support your merchandising needs.

Differentiator dimensions are the combination of each Differentiator and a dimension that is created off of the lowest dimension in the merchandise hierarchy (item). This only goes up as high as Department. The mock install configuration released with Curve is configured with an example of a Differentiator branch along the merchandise hierarchy; however, up to 10 Differentiator branches may be configured. The following diagram provides an example of how a Differentiator branch may be configured.

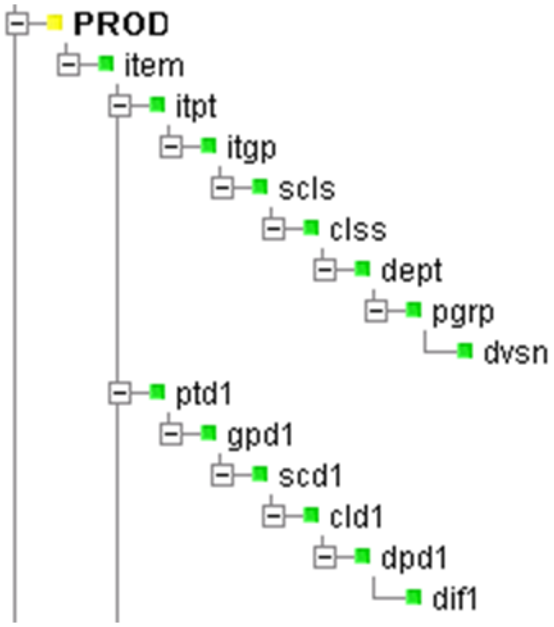
Figure 3–2 Differentiator Branch



If Curve is to be integrated with RMS /Allocation, the Diff dimensions configured in the RPAS Configuration Tools must map to the same Diff dimensions that are or will be configured in the RMS/ Allocation hierarchies. Allocation also requires Non-Aggregated Differentiator dimensions. These dimensions allow for Diff Dimensions to be combined (as shown in the diagram above) and allow for Curve to generate profiles to support Allocation. Within Allocation, these Non-Aggregated Differentiators are represented by Diffs with Aggregation Indicators set to *No*.

Within the merchandise hierarchy, which is also required to support RMS / Allocation, is the itpt (Item Parent) dimension off the item (Item) dimension. And off of the itpt dimension, add itgp (Item Grandparent) dimension. The other aggregate dimensions above item, should be dimensions beginning off of itgp. Figure 3–3 illustrates the GA configuration of the merchandise hierarchies that is configured using Item Parent and Item Grandparent in addition to a Differentiator branch:

Figure 3–3 Merchandise Hierarchy Configuration



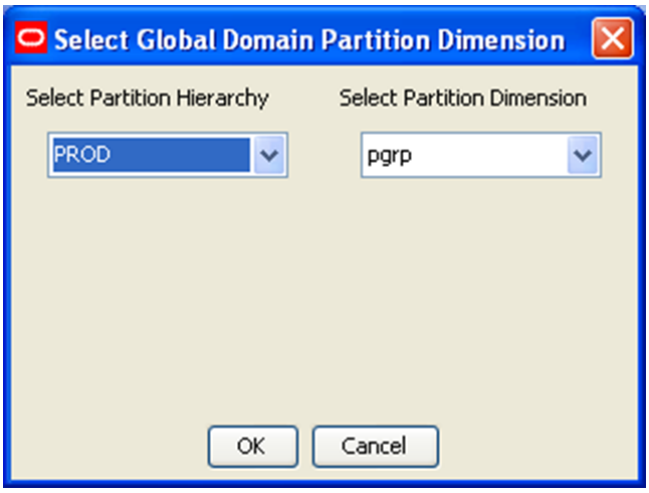
In addition to the above example, profiles 30 through 43 in the mock installations provided in the release packages are diff profile configurations that may be used to support the generation of spreading ratios for RMS / Allocation.

Creating a Curve Solution

To create a Curve solution:

1. Open an existing configuration in which hierarchies (for example, product, location, and calendar) have already been defined.
2. From the Configuration Tools toolbar, select the **Automation** menu. If installing a Global Domain environment, go on to Step 3. If installing a Simple Domain environment, go on to Step 4.
3. Select **Forecast Common**, and then select **Specify Partition Dimension**. The Select Global Domain Partition Dimension dialog box appears.

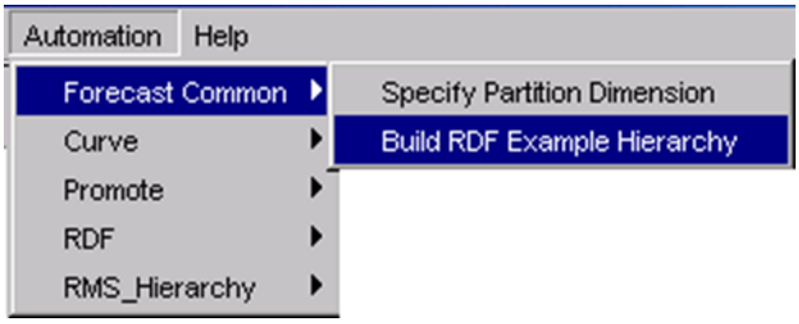
Figure 3–4 Select Global Domain Partition Dimension



Note: To access this dialog, the configuration must already be defined as a Global Domain environment. This is performed by selecting **Workspace Properties** from the File menu and selecting the **GlobalDomain** option.

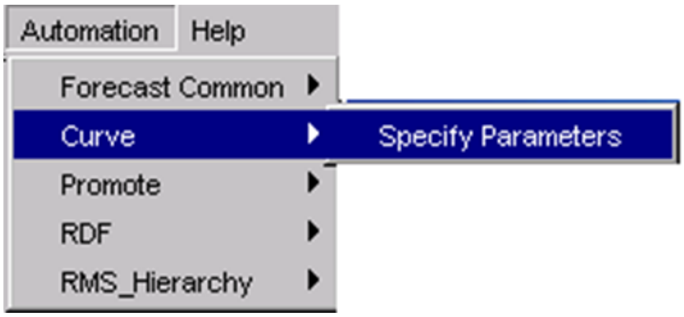
4. Perform the following:
5. From the **Select Partition Hierarchy** list, select the hierarchy in which the domains will be partitioned.
6. From the Select Partition Dimension list, select the appropriate partition dimension.
7. Select **OK**.
8. Optional: The Forecast Common plug-in may also be used to create an example hierarchy configuration. Select **Forecast Common - Build RDF Example Hierarchy**. The resulting hierarchy configuration is the same hierarchy that is used for the mock installation configurations provided in the release packages.

Figure 3–5 Build RDF Example Hierarchy



9. From the Automation menu, select **Curve - Specify Parameters**.

Figure 3–6 Specify Parameters



Configuring Profiles

The following sections provide information on configuring profiles:

Configuring a Final Profile

Configuring Source Level Attributes

Editing Profile Properties

Configuring a Final Profile

To create a final profile:

1. On the Curve Parameters utility, click the **F** icon.
2. A new final profile is added and is assigned the next consecutive number starting with 01.
3. Specify the properties for the final profile. See Editing Profile Properties for details.

Configuring Source Level Attributes

To create a source level:

1. On the Profile and Source Level window, highlight the final profile number in which a source will be created.
2. Click the **S** icon. A new source profile is added and is assigned the next consecutive number.
3. Specify the properties for the Source Level. See Editing Profile Properties for details.

Editing Profile Properties

The following sections describe how to edit profile properties.

Profile Name

The Profile Name is the system-assigned level number when a Final Profile or Source Level is created. This is a read-only field.

Profile Label

The Profile Label is the profile description that will be viewed by the user once the domain is created.

- Level Labels may not exceed forty characters.
- It is recommended (but not required) that Profile Labels include the Profile Name, which is the system-assigned profile number. There are two reasons for this:
- The Profile Name is referenced in the RDF configuration to specify Spreading Profiles. Curve requires that profiles 1 through 9 be referenced as 01, 02,…, 09 when being specified as a Spreading Profile in the RDF configuration.
- The Default Source Profile parameter in the Profile Administration workbook is a pick-list that is populated with the Profile Name of each source level configured for the final profile being viewed in the workbook. If the Default Source Profile set within this configuration, it not expected to change within the domain(s). This recommendation may not be necessary for consideration.
- RPAS automatically puts **( )** around Profile Labels. The configuration specialist should not include these in their level label configuration, or the installer will fail. An example of a Profile Label that would violate this requirement is (01 - chn->str-Final). It is acceptable as 01 - chn->str-Final.
- '**-**' should not be used before or after the Profile Label. An example of a Profile Label that would violate this requirement is -01-chn->str-Final-. It is acceptable as 01 - chn->str-Final.
- '**:**' should not be used at all in the Profile Label. An example of a Profile Label that would violate this requirement is 01: chn->str Final.

Profile Type

Assigned on the final profile, the Profile Type is a pick-list of profile types that are used to determine the profile algorithm and validation required by the profile level. Profile Types are represented with pre-defined configuration information.

Profile Types That Share The Same Profile Algorithm

The following Profile Types share the same profile algorithm. The rationale for providing different types that have the same behavior is strictly to remind the user of the intent of the profile while using the Profile Administration workbook:

Profile Types	Description
Store Contribution Profile	The Store Contribution Profile is used to determine the data relationship between stores to aggregate dimensions in the location hierarchy.
Hourly Profile	The Hourly Profile is used to determine the spreading ratios from aggregate dimensions to the hour, hour of day, or hour of week dimensions. To configure Hourly Profile, you must set the following: ■ The root dimension of the calendar hierarchy must be hour dimension. From configure tools, both RPAS Name and Tool Name for hour dimension have to be *HOUR*. Otherwise, the code will not work. ■ The position format for the configuration must be changed to *HOU%YEAR%MO%DAY%HR* and the calendar hierarchy file must match that format. ■ The training window start and phase start date for hourly profile will be mapped to the first hour of the day. ■ The training window end and phase end date for hourly profile will be mapped to the last hour of the day.
Daily Profile	The Daily Profile is used to determine the data relationship between a given day to the week in which it belongs.

Profile Types	Description
Product Profile	The Product Profile is used to determine the data relationship between any two dimensions along the product hierarchy.
Size Profile	The Size Profile is used to determine the data relationship between any dimension in the size hierarchy and any dimension in the product hierarchy. A size hierarchy must be defined to use this profile type.
User Defined Profile	The User Defined Profile may be used to support any profile configuration.

Profile Types with Unique Behavior

The following Profile Types have unique behavior:

Profile Types	Description
Diff Profile	Diff Profiles are used to determine spreading ratios from aggregate dimensions in the Product hierarchy to diff dimensions. Used to support the spreading of data in RMAS Allocation, Diff Profiles exhibit the same behavior as the previous profile types. However, unique to Diff Profiles is special validation of the relationship between the defined diff dimensions to dimensions along the main branch of the Product hierarchy. See the *Oracle Retail Demand Forecasting Implementation Guide* for more information on validation criteria.
Daily Seasonal Profile	The Daily Seasonal Profile is used to determine the data relationship between a given day of the week to aggregate dimensions in the calendar hierarchy. This profile type uses training window data to compute the profile. The resulting profile is then clipped to fit within the defined phase window.
Life Cycle Profile	The Life Cycle Profile uses data along a user-defined training window, and then stretches or shrinks data to fit a user-defined phase window.
Profile Intersection	The Profile Intersection is the intersection at which an intermediate profile is calculated. This intermediate profile is then replicated down or aggregated up to the Stored Intersection. If the Store Intersection is the same as the Profile Intersection, the values in intermediate profile are copied to the Stored Intersection. The Profile Intersection must be lower than the Aggregation Intersection. If the profile is being used as the Spreading Profile in RDF, this Profile Intersection should be the same as the Final Forecast Level. Once the Profile Intersection is entered at the Final Profile level, the Stored Intersection for both the Final and Source (if created) will populate with the same value. These may be overwritten if necessary. **Note:** If installing a Global Domain environment, all intersections configured to support a profile MUST include a dimension at or below the partition dimension.
Aggregation Intersection	The Aggregation Intersection is the intersection at which the profile will sum to one (or 100%). If the profile is being used as the Spreading Profile in RDF, this Aggregation Intersection should be the same as the Source Forecast Level. Once the Aggregating Intersection is entered, the Approval Intersection will populate with the same value for both the Final and Source (if created). This may be overwritten if necessary. **Note:** If installing a Global Domain environment, all intersections configured to support a profile MUST include a dimension at or below the partition dimension.

Profile Types	Description
Approval Intersection	Assigned only at the Final Profile, the Approval Intersection is the intersection at which the profile is approved. Approval Intersection should be above or equal to the Aggregation Intersection. If the profile is being used as the Spreading Profile in RDF, this Approval Intersection should be the same as the Aggregation Intersection. The Approval Intersection may be pre-populated with the value set for the Aggregation Intersection. This may be overwritten if necessary. **Note:** If installing a Global Domain environment, all intersections configured to support a profile MUST include a dimension at or below the partition dimension.
Stored Intersection	The Stored Intersection is the destination intersection of the profile. The intermediate profile produced at the Profile Intersection is either replicated down to or aggregated up to the Stored Intersection. If the Store Intersection is the same as the Profile Intersection, the values in intermediate profile are copied to the Stored Intersection. The Stored Intersection should not be greater than the Aggregation Intersection. If the profile is being used as the Spreading Profile in RDF, this Stored Intersection should be the same as the Profile Intersection. The Stored Intersection may be pre-populated with the value set for the Profile Intersection. This may be overwritten if necessary. **Note:** If installing a Global Domain environment, all intersections configured to support a profile MUST include a dimension at or below the partition dimension.
Default Source	Assigned only at the Final Profile, the Default Source is the primary Source Level that will be used in the calculation of the Final Profile. The desired Source Level must be created before it is an option in this pick-list.
Source Data	Assigned only at the Final Profile, the Source Data is the measure to be used as the input data (for example, POS) for the generation of profiles. The values in this pick-list are populated with all measures configured external to the RDF, Curve, and Promote solution extensions. If the profile is to be used to support the dynamic generation of spreading ratios (Spreading Profile) in the RDF batch forecast process, no value in Source Data should be specified.

Editing the Curve GA Configuration

The autogeneration process creates hierarchies, measures, rules, and workbook templates that are required to support the essential Curve functionality. This base configuration is referred to as the GA Configuration. Certain changes to the GA Configuration are allowed. Once edits to the GA Configuration are made and the autogeneration process occurs again, valid changes to the configuration will be preserved. There is nothing in the RPAS Configuration Tools to prevent invalid changes from being made.

> **Note:** When a custom taskflow exists in the configuration, the plug-in automation may cause the workbook template field to be blank. If this occurs, save the configuration and then re-open the configuration and the workbook template field is populated.

The following outlines acceptable changes and restrictions:

Item	Description
Curve Solution Extension Name	The name assigned to the resulting Curve solution after autogeneration occurs cannot be edited.
Major and Minor Classes	Additional Major and Minor classes may be added to the Curve GA Configuration. The Major and Minor classes that are part of the GA Configuration may not be edited. This restriction also applies to Measure Names and Measure Labels.
Rules	Additional Rule Sets, Rule Groups, and Rules may be added to the Curve GA Configuration. This includes support for adding new Rules to existing GA Configuration Rule Groups. It is recommended that new Rules that are added to the GA Configuration Rule Groups include cust (represents Custom) in the Rule Name. This allows for easy identification of Rules that are not part of the GA Configuration. Rule Sets, Rule Groups, and Rules that are part of the GA Configuration may not be renamed. Existing Rules that are part of the GA Configuration may not be modified in any way.
Workbook Templates	Additional Workbook Templates may be added to the Curve GA Configuration. As well, new Measures and Rules may be added to the GA Configuration Workbook Templates. This is done by adding new Major and Minor classes, and adding new Rules to existing Rule Groups in the GA Configuration.

Deleting a Profile Level

Deletion of a profile level will cause the system-assigned enumerated values in the Profile Name to renumber such that levels are in consecutive order starting with profile level 01. Deleting a profile level may impact any solution configuration that uses a specific profile level. For example, the following parameters within an RDF Solution configuration may be affected if profile levels are deleted or renumbered:

- Seasonal Profile
- Spreading Profile

If the domain using the configuration has been previously installed, there is potential to lose data associated to a level that has been deleted or renumbered.

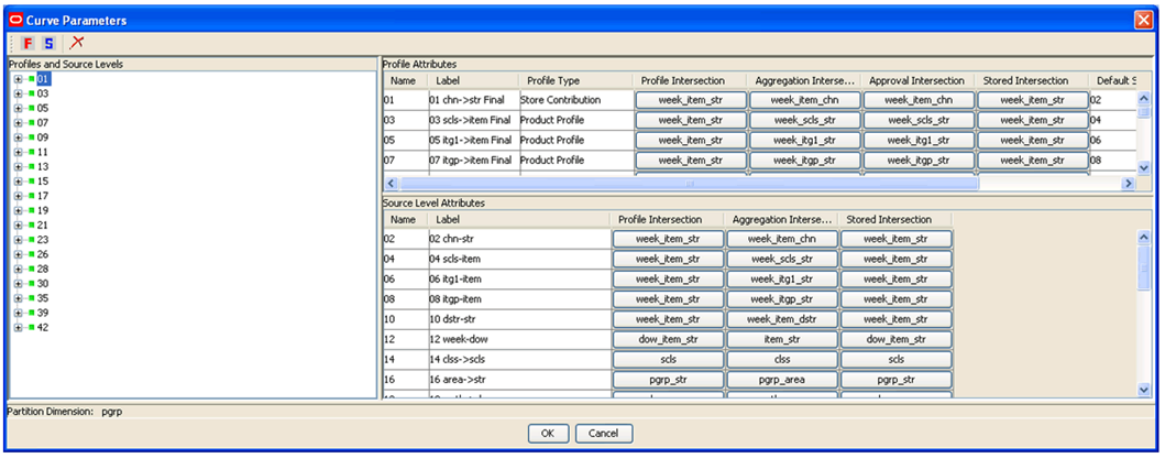
To delete a level:

1. On the Profiles and Source Level window, highlight the number of the profile you want to delete.
2. Click the **X** icon. The profile is deleted. If you delete a final profile, any source profiles that are associated with it will also be deleted.
3. Select **OK** to regenerate the solution with the changes to the cluster configuration.

Curve Plug-In Example

Figure 3–7 shows the Curve Parameters plug-in.

Figure 3–7 Curve Parameters Window



4

Configuring the Grade Solution Extension

Grade is a clustering tool that provides insight into how various parts of a retailer's operations can be grouped together. Typically, a retailer may cluster stores over item sales to create logical groupings of stores based on sales of particular products. This provides increased visibility to where products are selling, and it allows the retailer to make more accurate decisions in merchandising. Beyond this traditional use of clusters, Grade is flexible enough to cluster any business measure based on products, locations, time, promotions, customers, or any hierarchy configured in the solution.

Grades/clusters used within the Grade Solution are defined using the Grade Parameters utility in the RPAS Configuration Tools.

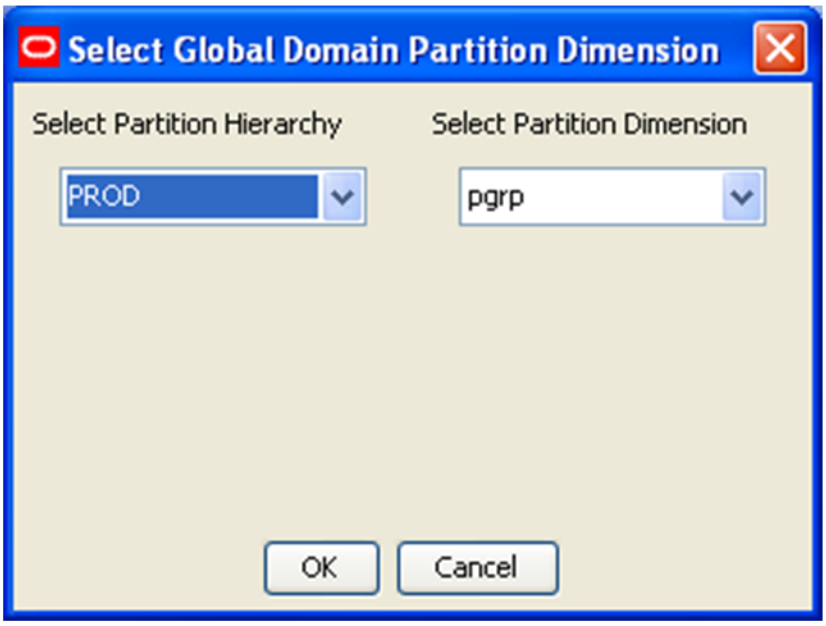
Note: For information on building the Grade domain, refer to the *Oracle Retail Predictive Application Server Installation Guide*.

Creating a Grade Solution Extension

To create a Grade solution extension:

1. Open an existing configuration in which hierarchies (for example, Product, Location, and Calendar) have already been defined.
2. From the Configuration Tools toolbar, select the **Automation** menu. If installing a Global Domain environment, go on to Step 3. If installing a Simple Domain environment, go on to Step 4.
3. Select **Common** and then the **Specify Partition Dimension**. The Select Global Domain Partition Dimension window appears. Select the hierarchy in which the domains will be partitioned, and then the Partition Dimension. Select **OK**, and continue with Step 4.

Figure 4–1 Select Global Domain Partition Dimension Window



Note: To access this dialog, the configuration must already be flagged as a Global Domain environment. To do this, select **File - Configuration Properties**. The Configuration Properties window appears. Select the **GlobalDomain** option and click **OK**.

4. Optional: The Common plug-in may also be used to create an example hierarchy configuration. Select **Common**, and then **Build RMS Example Hierarchy.** The resulting hierarchy configuration is the same hierarchy that is used for the mock installation configurations provided in the release packages.
5. Open the Function Library Manager and add the **ClusterEngine** library.
6. From the Grade option, select **Specify Clusters**. The following sections outline the process for configuring profiles.

Create Clusters

On the Grade Parameters utility, enter the Maximum Number of Clusters that will be required to support any Clustering/Grading process.

1. Click the **Create Clusters** icon.
2. The Cluster and Label parameters will update to reflect the number of clusters specified.
3. Specify the properties for the clusters. See Editing Grade Parameters for details.

Editing Grade Parameters

Edit Grade parameters:

- **Cluster** - Cluster is the system assigned Cluster Name. This value cannot be edited.

- **Label** - The Label is the description of the cluster/grade that will be viewed by the user once the domain is created.

Cluster Labels may not exceed 40 characters.

The following characters may not precede or follow the label that is entered in this field:

- '( )' Example: (cluster01)
- '-' Example: -cluster01-

A colon (:) may not be used in the Cluster Label field.

Example: cluster01:

Example 4–1 Grade Parameters

```
cluster01:
```

Autogenerating Hierarchies, Measures, Rules and Workbook Templates

The following is the process to autogenerate the hierarchies, measures, rules, and workbook templates that are required by Promote to support the promotion configuration entered in the Promote plug-in:

- On the Grade Parameters utility, click **OK**.
- The system automatically generates:
- **Hierarchies** - The CLSH hierarchy will be created with a clst dimension. The GRCH hierarchy will be created with the grcd dimension.
- **Measures** - All measures necessary to support the base Grade solution will be created.
- **Rules** - Only the rules and rule groups necessary to support the installation of the Grade solution are visible in the configuration. A special code is used within the domain to create rules as needed for cluster generation and workbook templates.
- **Workbook Templates** - All pre-defined workbook templates to support the base Grade solution will be created; however, only the worksheets necessary to support the domain installation are visible. Additional processes within the application handles the creation of additional worksheets based on the user's selections in the workbook template wizards.

Note: You may continue to make changes to the Grade plug-in configuration and the autogeneration process may be repeated as often as needed prior to the installation.

Note: After autogeneration completes, the following rules will display as invalid; however, these should be ignored:

Rule: clad_l1

Rule: clad_l2

Rule: clev_l2

Rule: clev_l3

Rule: clev_l4

Rule: clev_l5

Rule: clev_l6

Rule: clev_l7

Rule: clev_l8

Rule: clev_l9

RuleGroup: clad_load

Rule Group: clad_refresh

Rule Group: clrev_load

Rule Group: clrev_refresh

Adding or Deleting Clusters in the Configuration

Follow this process if you need to add clusters to the configuration or remove clusters from the configuration:

1. On the Grade Parameters utility, enter the new Maximum Number of Clusters that will be required to support any Clustering/Grading process.
2. Click the **Create Clusters** icon.

   The Cluster and Label parameters will update to reflect the number of clusters specified.
3. Click **OK** to regenerate the solution with the changes to the cluster configuration.

Editing the Grade GA Configuration

The Grade autogeneration process creates all hierarchy dimensions and measures to support the essential Grade functionality; however, only the minimum rules, workbook templates, and worksheets required to support the domain installation are visible in the configuration. Additional processes within the application handles the creation of rules and workbook template worksheets.

Note: When a custom taskflow exists in the configuration, the plug-in automation may cause the workbook template field to be blank. If this occurs, save the configuration and then re-open the configuration and the workbook template field is populated.

Note: This limitation allows for fewer options than in RDF and Curve for edits to the GA Configuration.

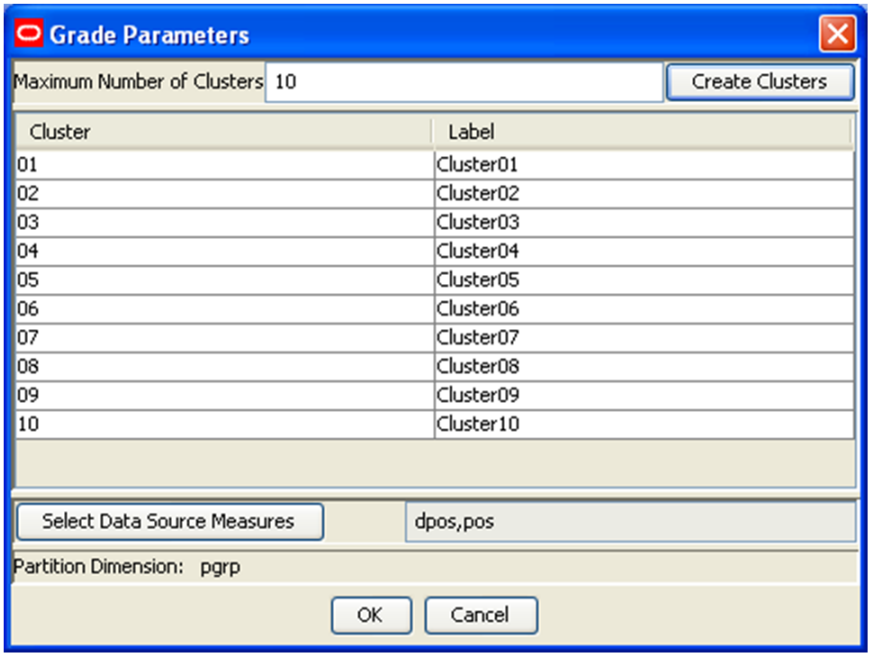
Acceptable changes and restrictions are outlined as follows:

Item	Description
Grade Solution Extension Name	The name assigned to the resulting Grade solution after autogeneration occurs cannot be edited.
Major and Minor Classes	Additional Major and Minor classes may be added to the Grade GA Configuration. The Major and Minor classes that are part of the GA Configuration may not be edited. This restriction also applies to Measure Names and Measure Labels.
Rules	Additional Rule Sets, Rule Groups, and Rules may be added to the Grade GA Configuration. This includes support for adding new Rules to existing GA Configuration Rule Groups. New Rules that are added to the GA Configuration Rule Groups should include cust (represents Custom) in the Rule Name, which makes it easy to identify Rules that are not part of the GA Configuration. Rule Sets, Rule Groups, and Rules that are part of the GA Configuration may not be renamed. Existing Rules that are part of the GA Configuration may not be modified in any way.
Workbook Templates	Additional Workbook Templates may be added to the Grade Configuration; however, new Measures and Rules can only be added to the Configuration Workbook Template worksheets that are visible in the configuration.

Grade Example

Figure 4–2 provides an example of the Grade Parameters utility.

Figure 4–2 Grade Parameters Window



A

Appendix: Configuring the Cluster Procedure

Clustering may be used to provide insight into how various parts of a retailer's operations can be grouped together. Typically a retailer may cluster stores over item sales to create logical groupings of stores based upon sales of particular products. This provides increased visibility to where products are selling, and it allows the retailer to make more accurate decisions in merchandising. Beyond this traditional use of clusters, the Cluster is flexible enough to cluster any business measure based on products, locations, time, promotions, customers, or any hierarchy configured in the solution.

> **Note:** The syntax is slightly different than the standard RPAS functions and procedures that are described in the "Rule Functions Reference Guide" section of the *Oracle Retail Predictive Application Server Configuration Tools User Guide*.

The two approaches available for clustering are BreakPoint and Cluster, or the BaNG approach. See the *Oracle Retail Grade User Guide* for details on these two approaches. The following sections explain the specifics for configuring clustering.

Cluster Requirements

The following libraries must be registered in any domains that will use the Cluster solution extension:

- AppFunctions
- ClusterEngine

Using the Cluster Procedure

The following notes are intended to serve as a guide for configuring the Cluster procedure within the RPAS Configuration Tools:

1. See the section, "Syntax Conventions" on page A-2, for the appropriate syntax for calling this procedure. Parameter labels must always be used.
2. If the ClusterEngine is not registered with the Configuration Tools, this rule will remain red, which indicates that it is invalid because the RPAS JNI cannot validate it at this point in time. Therefore, there is no validation for this rule. Refer to the Grade documentation for the appropriate input parameters and output measures. Make sure to register the ClusterEngine with the Configuration Tools. It is recommended that you register the ClusterEngine when creating the domain to avoid potential issues.

3. Make sure that the resultant measures are at the right intersection levels by using the information based on the input and output parameters.
4. The Cluster procedure is a multi-result procedure, which means that it can return multiple results with one procedure call within a rule. In order to get multiple results, the resultant measures must exist, 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.
5. You must configure/register all required input measures.
6. Be sure to create load and commit rules for the input measures. The RPAS JNI cannot validate the Cluster procedure call, so all input measures must exist within other rules in the rule set in order for them to be available for selection in the Workbook Tool.
7. You must use the latest version of RPAS to build the domain. You will get the following message in the log because the Cluster function is not validated:

   Warning: unable to parse new expression (Unknown special expression: Cluster)

   This message is okay.

   Registering ClusterEngine will eliminate this error from occurring.
8. After the domain build, use the regfunction RPAS utility to register the Grade library. The library, which is located in the $RPAS_HOME/applib directory, is libClusterEngine.so. Do not specify the lib or .so file extension for the function name with the regfunction utility.

Example A–1 ClusterEngine

```
regfunction -d /domains/D01 -l ClusterEngine
```

9. Use the Mace command to run the Cluster rule with the rule group (for instance, grade_batch).

Syntax Conventions

The following table displays the syntax conventions used in this document.

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	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.
BoldItalics	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 name.

Cluster Syntax

The syntax for using the Cluster or BaNG algorithm appears in the following examples. The input and output parameter tables explain the specific usage of the parameters names use in the procedure.

Example A–2 Generic Example

```
POINTMEMBERSHIP: MEMBERMEAS, CENTROID: CENTROIDMEAS [, DISTFROMCENTROID:
DISTCENTDMEAS, COHESION: COHESIONMEAS, CLUSTERPORTION: CLPORTMEAS, CENTROIDTOAVG:
C2AVGMEAS, CLOSESTCLUSTER: CLOSCLUSTMEAS, CLOSESTCLUSTERDIST: CLOSCLUSTDISTMEAS]
<-Cluster(MEASURE: MEASMEAS, METHOD: METHOD, NUMCLUSTERS: NUMCLUST, CLUSTERHIER:
CLUSTHIER, CLUSTEROVERHIER: CLUSTOVERHIER [, BYGROUPDIMS: BYGROUPDIM, AGGMETHOD:
AGGTYPES])
```

Example A–3 Sample - Cluster with Minimum Information:

```
POINTMEMBERSHIP:MEMB, CENTROID:CENT<-Cluster(MEASURE:RSAL, METHOD:"BANG",
NUMCLUSTERS:5, CLUSTERHIER:"PROD", CLUSTEROVERHIER:"LOC")
```

Syntax for Calculate Cluster Statistics (CalculateClusterStatistics)

Example A–4 Generic Example

```
CENTROID: CENTROIDMEAS, [DISTFROMCENTROID: DISTCENTDMEAS, COHESION: COHESIONMEAS,
CLUSTERPORTION: CLPORTMEAS, CENTROIDTOAVG: C2AVGMEAS, CLOSESTCLUSTER:
CLOSCLUSTMEAS, CLOSESTCLUSTERDIST: CLOSCLUSTDISTMEAS]
<-CalculateClusterStatistics(MEASURE: MEASMEAS, POINTMEMBERSHIP: MEMBERMEAS,
CLUSTERHIER: CLUSTHIER, CLUSTEROVERHIER: CLUSTOVERHIER [, BYGROUPDIMS: BYGROUPDIM,
AGGMETHOD: AGGTYPE])
```

Example A–5 Sample - Cluster Statistics with Minimum Information

```
CENTROID:CENT<-CalculateClusterStatistics(MEASURE:RSAL, POINTMEMBERSHIP:MEMB,
CLUSTERHIER:"PROD", CLUSTEROVERHIER:"LOC")
```

Configuration Parameters and Rules

Input Parameters

The following table provides the input parameters for the Cluster procedure and special expressions.

Parameter Name	Description
POINTMEMBERSHIP	This is an input parameter for CalculateClusterStatistics and bpstatistics. Its intersection should be the dimension being clustered and all by group dimensions from other hierarchies. The values state which positions are assigned to which cluster index. Data Type: Integer Required: Yes

Parameter Name	Description
MEASURE	The measure you are trying to cluster. It must have at least two dimensions. Data Type: Real Required: Yes
METHOD	Determines which clustering algorithm to use. Valid values are BANG (preferred) or KMEANS. Data Type: Real Required: Yes - for Cluster.
NUMCLUSTERS	For each by group partition, the maximum number of clusters. Data Type: Integer Required: Yes - for Cluster.
CLUSTERHIER	The hierarchy that contains the dimension to cluster. The results will give you clusters of positions in this dimension. Data Type: String Required: Yes
CLUSTEROVERHIER	The hierarchy that contains the dimension to cluster over. The algorithm uses the positions in this dimension as the co-ordinates when clustering. Data Type: String Required: Yes
BYGROUPDIMS	The algorithm generates clusters one by group combination at a time. Provide the by group intersection. Data Type: String Required: No
AGGMETHOD	The algorithm aggregates the measure data up to the appropriate level. If AGGMETHOD is specified, it will use it; otherwise, it will use whatever is defined on the measure. Data Type: String Required: No

Output Parameters

The following table provides the output parameters for the Cluster procedure.

Parameter Name	Description
POINTMEMBERSHIP	Its intersection should be the dimension being clustered and all by group dimensions from other hierarchies. The values state which positions are assigned to which cluster index. Data Type: Integer Required: Yes - output for bpcluster
CENTROID	Its intersection should be the cluster dimension, the dimension being clustered over and all by group dimensions from other hierarchies. The values are the average of all points in the cluster. Data Type: Real Required: Yes

Parameter Name	Description
DISTFROMCENTROID	Its intersection should be the dimension being clustered and all by group dimensions from other hierarchies. The values are the squared Euclidean distance from that point to its centroid. Data Type: Real Required: No
COHESION	Its intersection should be the cluster dimension and all by group dimensions. The value is the ratio of points in this cluster versus all clusters. Data Type: Real Required: No
CLUSTERPORTION	Its intersection should be the cluster dimension and all by group dimensions. The value is the ratio of points in this cluster versus all clusters. Data Type: Real Required: No
CENTROIDTOAVG	Its intersection should be the cluster dimension, the dimension being clustered over and all by group dimensions from other hierarchies. The values are the ratio of the centroid to the average of all points. Data Type: Real Required: No
CLOSESTCLUSTER	Its intersection should be the cluster dimension and all by group dimensions. The value is the nearest cluster index. Data Type: Integer Required: No
CLOSESTCLUSTERDIST	Its intersection should be the cluster dimension and all by group dimensions. The values are the squared Euclidean distance from the centroid of the cluster to the centroid of the closest cluster. Data Type: Real Required: No

Syntax for Break Point Cluster (bpcluster)

Example A–6 Generic Example

```
POINTMEMBERSHIP <- bpcluster(SOURCEMEASNAME, CONFIGURATIONMEASNAME, CONFIGNAME [,
GROUPBYINT] )
```

Example A–7 Sample - Break Point Cluster with Minimum Information

```
MEMB<-bpcluster(RSAL, GCFG, "GCFG01", "CHN_PGRP")
```

Syntax of Break Point Cluster Statistics (bpstatistics)

Example A–8 Generic Example

```
CENTROID, DISTANCE <- bpstatistics(POINTMEMBERSHIP, SOURCEMEASNAME [, GROUPBYINT]
)
```

Example A–9 Sample - Break Point Cluster Statistics with Minimum Information

```
CENTROID, DISTANCE <- bpstatistics(MEMB, RSAL, "CHN_PGRP" )
```

Configuration Parameters and Rules

Input Parameters

The following table provides the input parameters for the bpcluster and bpstatistics procedures and special expressions.

Parameter Name	Description
SOURCEMEASNAME	The measure you are trying to cluster. Data Type: Real Required: Yes
CONFIGURATIONMEASNAME	Measure defined at Cluster/Configuration intersection. It contains the thresholds for the breakpoint calculation. Data Type: Real Required: Yes - for bpcluster.
CONFIGNAME	Breakpoint Configuration that will be used to produce the grades. (Refer to the *Oracle Retail Grade User Guide* for details on Breakpoint configuration and administration.) Data Type: String Required: Yes
GROUPBYINT	The algorithm generates clusters by group combination one at a time. Provide the by group intersection. Data Type: String Required: No

Output Parameters

The following table provides the output parameters for the bpcluster and bpstatistics procedures and special expressions.

Parameter Name	Description
POINTMEMBERSHIP	Its intersection should be the dimension being clustered and all by group dimensions from other hierarchies. The values state which positions are assigned to which cluster index. Data Type: Integer Required: Yes - output for bpcluster
CENTROID	Measure defined at Cluster/Configuration intersection. It contains the thresholds for the breakpoint calculation. Data Type: Real Required: Yes - for bpcluster.
DISTANCE	This is the point distance of member from centroid. Data Type: Real Required: Yes

B

Appendix: Configuring the Clone Procedure

Cloning allows users to generate forecasts for new items and locations by copying, or cloning history, from other items and stores. Users can map items or stores that have similar business cases, clone the historical data, and begin generating forecasts. Cloning provides the ability to generate forecasts based on historical data and promotional calendar.

The Clone Syntax section contains the specifications and syntax for configuring the Forecast procedure.

The clone procedure can be set up to clone sales history, promotion history. The clone procedure can handle cloning of real, integer, Boolean and string measures.

This appendix details these topics:

- Clone Requirements
- Clone Syntax
- Configuration Parameters and Rules

Clone Requirements

The following libraries must be registered in any domains that will use the clone solution extension:

- RdfFunctions

Using the Clone Procedure

The following notes provide information about clone functionality.

- Refer to the appropriate input parameters and output measures when using the clone procedure.
- The PROD and LOC hierarchies are required by the clone expression. If CLND exists, it must be the innermost hierarchy.
- Cloning supports up to three parent items or three parent stores with contribution percentages for each of these items or stores.
- An adjustment ratio can be defined to modify the level of the cloned history for the new product or location.
- Users can specify different parent items (or stores) for different locations. For example, Item A sells like Item B in Region 1 and like Item C in Region 2. These location or product levels can be configured through the RPAS Configuration Tools.

- The cloning (copying) of historical data is performed as part of the batch process using the clone special expression.
- The clone special expression is generic enough that it can be used to copy not just history, but forecast parameters, Casual histories and more, using the clone special expression.
- A mask measure is used to define when cloning is performed. When the mask measure is *True*, cloning is performed; setting the mask to *False* stops the cloning process. A business rule may be defined (using RPAS rules) to set the mask measure to *False* when it is desired to stop cloning the item/location. Note that if no mask measure is specified, cloning is performed for all item/locations that have a like item and/or like location specified.
- The input parameters include a source array, up to six map measures, two contribution measures, an optional mask measure, AN optional Adjustment Ratio Measure, and a destination array.
- The source and destination array must be at the same intersection, as validated by the special expression. The intersection will be where cloning is performed.
- The map, contribution, and mask measures can be at intersections higher than the source and destination arrays.
- In the event that any of these are at a higher intersection, standard replication will be used for spreading values down to the source and destination Arrays.
- The map arrays and contribution measures are optional; at least one of each is required. The number of contribution measures should be equal to the number of map measures.
- Two additional optional start and end date measures can be passed, which specify the start end and end date indexes of the cloning process.
- Similar restrictions for the intersection of the Start and End Date Index measures apply as mentioned with map measures above.
- The index should be an index along a calendar dimension equal to the Calendar Dimension along Source and Destination array. For example, if the Source and Destination Arrays are at the item/store/week level intersection, then the Start and end date index measures should contain Index values of the Week dimension.
- If these values are not passed, they will default to calendar hierarchy Start and End dates.
- When running a clone at a higher intersection, you need to create a mask measure. If you create a mask, do so at the same intersection as your SRC and DEST measures less the Calendar dimension if the SRC and DEST are at a higher intersection.
- In some instances, it is preferable that the clone expression is run separately for items and locations. Perform the following procedure to run the clone expression separately for items and locations:

  1. Run the clone expression for items.
  2. Update the clone source measure with the results from the item cloning run.

     For example, if the initial source measure for the cloning run was **source_measure**, and the item cloning adjustments are stored in the **item_cloning_adjustments** measure. Then the update could be achieved by an expression of the type:

     source_measure = source_measure + item_cloning_adjustments

3. Run the clone expression for locations with the same **source_measure** as source measure.
4. Aggregate the various demand components to build the forecast source.

About Cloning

Cloning is handled differently, depending on the type of measures being cloned. This topic addresses the various manner in which cloning is handled for the following measure types:

- Real or integer measure
- Boolean measures
- String measures
- Date measure

Cloning Real or Integer Measures

Since up to three parent items and three parent stores can be specified, the number of mapping measures can consist of up to nine combinations of item/stores (# of parent items specified x # pf parent stores specified) and corresponding % contributions.

Example B–1 Calculating Real or Integer Values Using the Clone Expression

The following example illustrates how real or integer values are calculated using the clone expression.

Item1		
Item2	Item3	Item4
20%	20%	60%
STR1		
STR2	STR3	STR4
20%	50%	30%

The Special Expression will calculate mappings and contributions as follows:

- Item2/STR2 at 4%
- Item2/STR3 at 10%
- Item2/STR4 at 6%
- Item3/STR2 at 4%
- Item3/STR3 at 10%
- Item3/Str4 at 6%
- Item4/STR2 at 12%
- Item4/STR3 at 30%
- Item4/Str4 at 18%

Example B–2 Subset of Possible Values Provided

If only a subset of values is populated, then the clone expression performs its calculations as shown in this example.

Item1	
Item2	Item3
20%	80%
STR1	
STR2	STR3
50%	50%

The Special Expression will calculate mappings and contributions as follows:

- Item2/STR2 at 10%
- Item2/STR3 at 10%
- Item3/STR2 at 40%
- Item3/STR3 at 40%

Example B–3 Subset of Possible Values Provided with Adjustment Ratio

If only a subset of values is populated and an Adjustment Ratio is defined, then the clone expression performs its calculations as shown in this example.

Item1	
Item2	Item3
20%	80%
Adjustment Ration = 1	
STR1	
STR2	STR3
50%	50%
Adjustment Ration = 0.5	

The Special Expression will calculate mappings and contributions as follows:

- Item2/STR2 at 5% (=20% x 0.5x50%)
- Item2/STR3 at 5%
- Item3/STR2 at 20% (=80%x0.5x50%)
- Item3/STR3 at 20%

Cloning Boolean Measures

It is possible to clone promotion variables, which could be Boolean measures. This topic provides information about how the cloning of Boolean measures, specifically multiple Boolean measures, is handled.

The special expression supports the use of multiple Like items or Like stores for cloning Boolean measures. Users must specify a method of combining the multiple measures, which could be an *and* or an *or*.

Example B–4 Calculating Boolean Measures Values Using the Clone Expression

The following example illustrates how Boolean measures values are handled using the clone expression.

Item1		
Item2	Item3	Item4
STR1		
STR2	STR3	STR4

The Special Expression will calculate Item1/STR1 as follows:

Item2/STR2 *or* Item2/STR3 *or* Item2/STR4 *or* Item3/STR2 *or* Item3/STR3 *or* Item3/STR4 *or* Item4/STR2 *or* Item4/STR3 *or* Item4/STR4

Example B–5 Cloning Promotion Measures for New Item with or Aggregation Type Defined

When only cloning promotion measures for new items and user has specified an *or* aggregation type:

Item1	
Item2	Item3

For Item1/STR1, the special expression calculates the following: Item2/STR1 *or* Item3/STR1

Cloning and String Measures

It is possible to clone multiple string measures. When more than one clone items or stores are specified, the special expression concatenates individual string measures.

When cloning Boolean or String measures, the following parameters are ignored by the special expression:

- SKUCONTRIBUTION1
- SKUCONTRIBUTION2
- SKUCONTRIBUTION3
- SKUADJUSTMENTRATIOMEAS
- STRCONTRIBUTION1
- STRCONTRIBUTION2
- STRCONTRIBUTION3
- STRADJUSTMENTRATIOMEAS

Valid Parameters during Cloning

When cloning Boolean or String measures, the following parameters are used in the same way as with real or integer measures:

- SOURCE_MEAS
- SKUMAPMEAS1
- SKUMAPMEAS2

- SKUMAPMEAS3
- STRMAPMEAS1
- STRMAPMEAS2
- STRMAPMEAS3
- STARTINDEX
- ENDINDEX
- MASKMEASURE

When cloning Boolean measures, the following additional parameters can be specified:

- Boolean Operator (indicating whether to use an *and* or an *or* operator for combining multiple measures). If this measure is not specified, and when cloning Boolean measures, the special expression defaults to an *or* operation for combining multiple measures.
- Boolean operator parameter can be specified as a scalar constant, a scalar measure, or a non-scalar measure which has a base intersection that is equal to or higher than the dest/src measure intersection.
- The Boolean operator when specified as a measure, it needs to be a real measure, so that it can be displayed as a picklist if desired.

Cloning Date Measures

When cloning a date measure, only one parent item and one parent store can be specified

When cloning Date measures, the following parameters are ignored by the special expression:

- SKUCONTRIBUTION1
- SKUCONTRIBUTION2
- SKUCONTRIBUTION3
- SKUADJUSTMENTRATIOMEAS
- STRCONTRIBUTION1
- STRCONTRIBUTION2
- STRCONTRIBUTION3
- STRADJUSTMENTRATIOMEAS
- SKUMAPMEAS2
- SKUMAPMEAS3
- STRMAPMEAS2
- STRMAPMEAS3

Valid Parameters during Cloning

When cloning Date measures, the following parameters are used in the same way as with real or integer measures:

- SOURCE_MEAS
- SKUMAPMEAS1

- STRMAPMEAS1
- STARTINDEX
- ENDINDEX
- MASKMEASURE

Clone Syntax

The syntax for using the clone procedure is shown in the following examples. The input and output parameter tables explain the specific usage of the parameters names use in the procedure.

Example B–6 Generic Example for Cloning Real or Integer Measures:

DEST_MEASURE <-clone(SRC:SOURCE_MEASURE,

SKUMAP1: SKUMAPMEAS1, SKURATIO: SKUCONTRIBUTION1, SKUMAP2: SKUMAPMEAS2, SKURATIO2:SKUCONTRIBUTION2, SKUMAP3:SKUMAPMEAS3, SKURATIO3: SKUCONTRIBUTION3, SKUADJRATIO: SKUADJUSTMENTRATIOMEAS

STRMAP1:STRMAPMEAS1, STRRATIO1:STRCONTRIBUTION1, STRMAP2:STRMAPMEAS2, STRRATIO2:STRCONTRIBUTION2, STRMAP3:STRMAPMEAS3, STRRATIO3:STRCONTRIBUTION3, STRADJRATIO:STRADJUSTMENTRATIOMEAS [, STARTINDEX:STARTINDEX]

[, ENDINDEX:ENDINDEX] [, MASK:MASKMEASURE] [,BOOLOPT:BOOLEANOPERATOR])

Example B–7 Generic Example for Cloning Boolean Measures:

DEST_MEASURE <-clone(SOURCE_MEASURE,

SKUMAPMEAS1, SKUMAPMEAS2, SKUMAPMEAS3,

STRMAPMEAS1, STRMAPMEAS2, STRMAPMEAS3,

STARTINDEX, ENDINDEX, , MASKMEASURE, BOOLEANOPERATOR)

Example B–8 Generic Example for Cloning String Measures:

DEST_MEASURE <-clone(SOURCE_MEASURE,

SKUMAPMEAS1, SKUMAPMEAS2, SKUMAPMEAS3,

STRMAPMEAS1, STRMAPMEAS2, STRMAPMEAS3,

STARTINDEX, ENDINDEX, MASKMEASURE)

Example B–9 Sample of Clone Function with Real or Integer Measures:

DEST:clnSls <- Clone(SRC: promo, SKUMAP1:prodcln101xb, SKURATIO1:prodcnt101xb, SKUMAP2:prodcln201xb, SKURATIO2:prodcnt201xb, SKUMAP3:prodcln301xb, SKURATIO3:prodcnt301xb, STRMAP1:loccln101xb, STRRATIO1:loccnt101xb, STRMAP2:loccln201xb, STRRATIO2:loccnt201xb, STRMAP3:loccln301xb, STRRATIO3:loccnt301xb, SKUADJRATIO:prodadjpct01xb, STRADJRATIO:locadjpct01xb, MASK:clnmask01xb)

Configuration Parameters and Rules

This section describes the configuration parameters and rules for clones.

Input Parameters

Table B–1 provides the input parameters for the clone procedure and special expressions.

Table B–1 Input Parameters for the Clone Procedure and Special Expressions

Parameter Name	Description
SOURCE_MEASURE	The source measure used for cloning. The source array and destination array need to be at the same intersection, which will be validated by the special expression. This is the intersection in which cloning is performed. Data Type: Date, Integer, Real, Boolean, or String Required: Yes
SKUMAPMEAS1	The first SKU measure that is mapped to the new item/store. Data Type: String Required: Yes At least one map measure is required when using the clone function. SKUMAPMEAS1 needs to be populated with position ID of clone item.
SKUCONTRIBUTION1	The percentage of data used when cloning historical data. Used to weigh or assign importance of item or store map. Data Type: Real Required: Yes—for Real or Integer cloning. Ignored for Boolean and String cloning. At least one contribution measure is required Real or Integer cloning.
SKUMAPMEAS2	The second SKU measure that is mapped to the new item/store. Data Type: String Required: Yes— for Integer, Real, Boolean, or String cloning. Ignored for Date cloning.
SKUCONTRIBUTION2	The percentage of data used when cloning historical data for SKUMAPMEAS2. Used to weigh or assign importance of item or store map. Data Type: Real Required: Yes Ignored for Boolean and String cloning.
SKUMAPMEAS3	The third SKU measure that is mapped to the new item/store. Data Type: String Required: Yes— for Integer, Real, Boolean, or String cloning. Ignored for Date cloning.

Table B–1 (Cont.) Input Parameters for the Clone Procedure and Special Expressions

Parameter Name	Description
SKUCONTRIBUTION3	The percentage of data used when cloning historical data for SKUMAPMEAS3. Used to weigh or assign importance of item or store map. Data Type: Real Required: Yes Ignored for Boolean and String cloning.
SKUADJUSTMENTRATIOMEAS	A value greater than zero (0) used to adjust the level of history calculated at the item level. Data Type: Real Required: Yes
STRMAPMEAS1	The first STR measure that is mapped to the new item/store. Data Type: String Required: Yes STRMAPMEAS1 needs to be populated with position ID of clone store.
STRCONTRIBUTION1	The percentage of data used when cloning historical data for STRMAPMEAS1. Used to weigh or assign importance of item or store map. Data Type: Real Required: Yes — for Real or Integer cloning. Ignored for Boolean and String cloning.
StrMapMeas2	The second STR measure that is mapped to the new item/store. Data Type: String Required: Yes— for Integer, Real, Boolean, or String cloning. Ignored for Date cloning.
StrContribution2	The percentage of data used when cloning historical data for STRMAPMEAS2. Used to weigh or assign importance of item or store map. Data Type: Real Required: Yes Ignored for Boolean and String cloning.
StrMapMeas3	The third STR measure that is mapped to the new item/store. Data Type: String Required: Yes— for Integer, Real, Boolean, or String cloning. Ignored for Date cloning.
StrContribution3	The percentage of data used when cloning historical data for STRMAPMEAS3. Used to weigh or assign importance of item or store map. Data Type: Real Required: Yes Ignored for Boolean and String cloning.

Table B–1 (Cont.) Input Parameters for the Clone Procedure and Special Expressions

Parameter Name	Description
STRADJUSTMENTRATIOMEAS	A value used to adjust the level of history calculated at the store level. Data Type: Real Required: Yes
STARTINDEX	The index value for the Week dimension to be used as the starting point to clone data. Data Type: Integer Required: No
ENDINDEX	The index value for the Week dimension to be used as the ending point to clone data. Data Type: Integer Required: No
MASKMEASURE	A *True* or *False* value that indicates if cloning should be performed. The special expression only updates the DEST_MEASURE for item/stores with MASKMEASURE set to *True*, if this measure is passed. If this measure is not specified, it updates it for all item/stores. Data Type: Boolean Required: No
BOOLEANOPERATOR	An optional parameter that can be specified when cloning Boolean measures. This parameter indicate whether to use an *and* or an *or* operator for combining multiple measures. Data Type: Boolean Required: No
DEST_MEASURE	An array containing the item or location data is being cloned. Data Type: Date, Integer, Real, Boolean, or String Required: Yes Data type and Base Intersection of this measure need to be the same as that of the SOURCE_MEASURE.

Clone Special Expression

This section describes special expressions for clones.

ClonePostProc

The ClonePostProc special expression can be used to adjust the cloned sales history based on actual sales of new items. When a new item start selling, the actual sales and the clone sales can be used to calculate an adjustment ratio. The cloned sales is multiplied with the adjustment ratio so that the scale of cloned sales is close to actual sales.

Calculation

*adjust cloned sales = pow(total actual sales from history start to t period forward /total cloned sales over t period backward from history start,1-alpha) * cloned sales*

Example B–10 Clonepostproc Syntax

DEST:clnadj<-ClonePostProc(SALESHIST:pos, NPERSSLSTHRSHLD:clnslsprdXLXB,

MASKMEAS:clnAdjMask, NPERSCALCRAT:clnclcprdX

LXB, ADJRATIO:clnalphaXLXB)

Table B–2 Parameters for the ClonePostProc Special Expression

Parameter Name	Description
DEST	host cloned sales at prod/loc/clnd before special expression run. After special expression run, it host the adjusted sales: Data Type: Real Required: Yes The Data Type and Base Intersection of this measure needs to be the same as SALESHIST.
SALESHIST	Sales measure contains actual sales. Based on prod/loc/calendar Data Type: Real Required: Yes
NPERSSLSTHRSHLD	Maximum number of periods having actual sales. Based on prod/loc, When the sales history period is more than this threshold. no adjust is made. Data Type: Integer, Real Required: Yes
MASKMEAS	A Boolean measure at prod/loc. It indicates which times eries is eligible for adjustment. Data Type: Boolean Required: Yes
NPERSCALCRAT	An integer measure indicate the number of period to used pre and after actual sales history start to calculate adjustment. Data Type: Integer, Real Required: Yes
ADJRATIO	The alpha number in the calculation formula. A real measure based on prod/loc. Data Type: Real Required: Yes Valid alpha is between 0-1. When alpha=0 or1, no adjustment is made. When alpha is close to 0, the adjusted sales is close to the cloned sales. When alpha is close to 1, the adjusted sales is close to the actual sales. It needs to be on the same intersection as NPERSCALCRAT and NPERSSLSTHRSHLD.

C

Appendix: Configuring the Forecast Procedure

This appendix details these topics:

- About the Forecast Procedure
- Forecast Requirements
- Forecast Parameter/Model Dependencies
- Forecast Procedure Syntax
- Configuration Parameters and Rules

About the Forecast Procedure

Using the RPAS 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:

RDF Allows for...	The Forecast Procedure 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.	A single-level forecast and a single forecasting method to be specified in the calculation of the forecast.
Forecasting methods and forecasting parameters to be modified as needed at all levels in your data.	No standard approval process of the resulting forecast.

The Forecast Procedure Syntax section contains the specifications and syntax for configuring the Forecast procedure.

Note: The syntax is slightly different than the standard RPAS functions and procedures that are described in the Rule Functions Reference Guide section of the *Oracle Retail Predictive Application Server Configuration Tools User Guide.*

Forecast Requirements

The following libraries must be registered in any domains that will use the Forecast solution extension:

- AppFunctions

- RdfFunctions

Forecast Parameter/Model Dependencies

The following models require that the stated measure is to be provided.

- Bayesian model — Plan measure required.
- Profile model — Profile measure required.

Using the Forecast Procedure

The following notes are intended to serve as a guide for configuring the Forecast procedure within the RPAS Configuration Tools.

- Refer to the appropriate input parameters and output measures when using the Forecast procedure.
- The resultant measure (that is, frcstout) 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 multi-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.
- The startdatemeas that specifies the forecast start date needs to be periodically updated (every week or so) by configuring rules.
- The forecast methods are specified via the mask measure. This is an int measure. Refer to the Forecast Model/method list table for the expected values of this measure for each forecast method.

Forecast Procedure Syntax

The syntax for using the Forecast procedure is shown in the following examples. The input and output parameter tables explain the specific usage of the parameters names used in the procedure.

Example C–1 Generic Example:

FORECAST: FORMEAS [, INT: INTMEAS, CUMINT:CUMINTMEAS, PEAKS:PEAKSMEAS, CHMETHOD:METHMEAS, CHLEVEL:LVLMEAS, CHTREND:TRENDMEAS, ALERTS:ALERTSMEAS] <-FORECAST(MASK:MEASKMEAS, {STARTDATE:STARTDATE | STARTDATEMEAS:STARTDATEMEAS}, FORECASTLENGTH:FORECASTLENGTH, HISTORY:HISTORYMEAS, PERIOD:PERIOD [

[,{PROMO_0:PROMO0, PROMOEFF_0:PROMOEFF0, PROMOOVER_ 0:PROMOOVER0, PROMOTYPE_0:PROMOTYPE0} … {,PROMO_N:PROMON, PROMOEFF_N:PROMOEFFN,PROMOOVER_N:PROMOOVERN, PROMOTYPE_ N:PROMOTYPEN}],

HISTSTART: HISTSTARTMEAS, {FRCSTSTARTMEAS:FRCSTSTARTMEAS | FRCSTSTART:FRCSTSTART}, MINWINTERS:MINWINTERSMEAS, MINHOLT: MINHOLTMEAS, MINCROSTON:MINCROSTON, MAXALPHA:MAXALPHA, MAXWINTERSALPHA:MAXWINALPHA, MAXPROFILEALPHA:MAXPROFILEALPHA, BAYESALPHA:BAYESALPHA,

TRENDDAMP:TRENDDAMP, {VALID_DD:VALID_DD, DDPROFILE:DDPROFILE }, PROMO_IN_BASELINE:PROMO_IN_BASELINE, PLAN:PLAN, PROFILE:PROFILE, VERBOSE:VERBOSE, AGGPROF:AGGPROF, SPREADPROF:SPREADPROF, READMODE:READMODE, BAYESIAN_HORIZ,BAYESIAN_HORIZ, MINB:MINB, MAXB:MAXB, KEEPCLAMPEDMAXB:KEEPCLAMPEDMAXB, SMOOTHBASELINE:SMOOTHBASELINE, CAUSALMERGE:CAUSALMERGE, CAPS:CAPSMEAS, CAPRATIOS:CAPRATIOSMEAS, USECAPPING:USECAPPING, MINCAPHIST:MINCAPHIST, PLANINT:PLANINTMEAS, PLANCUMINT:PLANCUMINTMEAS, CAPINTERVALS:CAPINTERVALS]

Example C–2 Sample 1—Startdate as String:

FORECASTOUT <-FORECAST(BAYESALPHA:0.15, FORECASTLENGTH:12, HISTORY:RSAL, MASK:METHMASK1, MAXALPHA:0.99, MAXB:50, MAXWINTERSALPHA:0.99, MINCROSTON:5, MINHOLT:13, MINWINTERS:104, PERIOD:52, PLAN:BAYES_PLAN1, SMOOTHBASELINE:TRUE, STARTDATE:"D19980505", TRENDDAMP:0.5, HISTSTART:H_STARTDATE1, FRCSTSTART:F_STARTDATE1)

Example C–3 Sample 2—Startdate as Parameter Measure:

FORECAST:FORECASTOUT <-FORECAST(BAYESALPHA:0.15, FORECASTLENGTH:12, HISTORY:RSAL, MASK:METHMASK1, MAXALPHA:0.99, MAXB:50, MAXWINTERSALPHA:0.99, MINCROSTON:5, MINHOLT:13, MINWINTERS:104, PERIOD:52, PLAN:BAYES_PLAN1, SMOOTHBASELINE:TRUE, STARTDATEMEAS:TODAY*, TRENDDAMP:0.5, HISTSTART:H_STARTDATE1, FRCSTSTART:F_STARTDATE1)

Example C–4 Sample 3—Getting Multiple Results

FORECAST: FORECASTMEAS, INT:INTMEAS, CUMINT:CUMINTMEAS, PEAKS: PEAKSMEAS <-FORECAST(BAYESALPHA:0.15, FORECASTLENGTH:12, HISTORY:RSAL, MASK:METHMASK1, MAXALPHA:0.99, MAXB:50, MAXWINTERSALPHA:0.99, MINCROSTON:5, MINHOLT:13, MINWINTERS:104, PERIOD:52, PLAN:BAYES_PLAN1, SMOOTHBASELINE:TRUE, STARTDATEMEAS:TODAY*, TRENDDAMP:0.5, HISTSTART:H_STARTDATE1, FRCSTSTART:F_STARTDATE1)

Configuration Parameters and Rules

Input Parameters

Table C–1 provides the input parameters for the Forecast procedure.

Table C–1 Input Parameters for the Forecast Procedure

Parameter Name	Description
BAYESALPHA	The maximum Bayesian alpha value. Data Type: Real Multiple Allowed: No Required: No

Table C–1 (Cont.) Input Parameters for the Forecast Procedure

Parameter Name	Description
BAYESIAN_HORIZ	The horizon to which the Bayesian adjust is applied. Data Type: Integer Multiple Allowed: No Required: No
CAPRATIOS	Cap ratio for each time series. Data Type: Real Multiple Allowed: No Required: No
CAPS	Caps for each time series. Data Type: Real Multiple Allowed: No Required: No
DDPROFILE	De-seasonalized demand measure used only for profile-based forecasting. Data Type: Real Multiple Allowed: No Required: No
FORECASTLENGTH	The length of the forecast. Data Type: Integer Multiple Allowed: No Required: Yes
FRCSTSTART	The forecast start date. Data Type: Datetime Multiple Allowed: No Required: No
FRCSTSTARTMEAS	The measure of the forecast start dates. Data Type: Datetime Multiple Allowed: No Required: No
HISTORY	The input measure the forecast is based on. Data Type: Real Multiple Allowed: No Required: Yes
HISTSTART	The historical start date. Data Type: Real Multiple Allowed: No Required: No

Table C–1 (Cont.) Input Parameters for the Forecast Procedure

Parameter Name	Description
KEEPCLAMOEDMAXB	Determines whether variables exceeding maxb are clamped or values are dropped and regression is re-run. Data Type: Real Multiple Allowed: No Required: No
KEEPCLAMPEDMINB	Determines whether variables exceeding minb are clamped or values are dropped and regression is re-run. Data Type: Real Multiple Allowed: No Required: No
MASK	Array that identifies what forecast method is used for each time series. See the "Forecast Method/Model List" on page C-9. Data Type: Boolean Multiple Allowed: No Required: Yes
MAXALPHA	The maximum alpha value. Data Type: Real Multiple Allowed: No Required: No
MAXB	The maximum ratio between beta and baseline. Data Type: Real Multiple Allowed: No Required: No
MAXPROFILEALPHA	The maximum alpha for the Profile method. Data Type: Real Multiple Allowed: No Required: No
MAXWINTERSALPHA	The maximum alpha in the Winters method. Data Type: Real Multiple Allowed: No Required: No
MINB	The minimum ratio between beta and baseline. Data Type: Real Multiple Allowed: No Required: No
MINCAPHIST	The minimum number of weeks before capping can be used. Data Type: Real Multiple Allowed: No Required: No

Table C–1 (Cont.) Input Parameters for the Forecast Procedure

Parameter Name	Description
MINCROSTON	The minimum Croston history. Data Type: Integer Multiple Allowed: No Required: No
MINHOLT	The minimum Holt history. Data Type: Integer Multiple Allowed: No Required: No
MINWINTERS	The minimum Winters history. Data Type: Integer 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. This measure's intersection may not be higher than the intersection of a corresponding forecast source level. Data Type: Real Multiple Allowed: No Required: No
PLANCUMINT	The cumulative Interval of the plan associated with the plan (PARAMETER forecast); Bayesian only. Data Type: Real Multiple Allowed: No Required: No
PLANINT	The interval of the plan associated with the plan (PARAMETER forecast); Bayesian only. Data Type: Real Multiple Allowed: No Required: No
PROFILE	The Seasonal Profile measure. Data Type: Real Multiple Allowed: No Required: No
PROMO	The Promo variable measure (one for each promotion). Data Type: Integer Multiple Allowed: Yes Required: No

Table C–1 (Cont.) Input Parameters for the Forecast Procedure

Parameter Name	Description
PROMO_IN_BASELINE	An indicator used to identify if the promotion is incorporated in the baseline. Data Type: Boolean Multiple Allowed: No Required: No
PROMOEFF	The calculated promotional effects (one per promotion). Data Type: Real Multiple Allowed: Yes Required: No
PROMOOVER	The promo effect override measure (one for each promotion). Data Type: Boolean Multiple Allowed: Yes Required: No
PROMOTYPE	The promo type measure (one for each promotion). Data Type: Integer Multiple Allowed: Yes Required: No
READMODE	Indicates whether mode is Random or Sequential. Data Type: Integer/enum Multiple Allowed: No Required: No
SMOOTHBASELINE	When value is *True*, historical baseline is smoothed prior to future baseline forecast. Defaults to *True* in casual method. Data Type: Boolean Multiple Allowed: No Required: No
SPREADPROF	The profile to spread to final forecast level. Data Type: Real Multiple Allowed: No Required: No
STARTDATE/ STARTDATEMEAS	The forecast start date. Either STARTDATE or STARTDATEMEAS is required. Data Type: STARTDATE - Date as a string. Data Type: STARTDATEMEAS - Date as measure. Multiple Allowed: No Required: Yes
TRENDDAMP	The trend damping parameter. Data Type: Real Multiple Allowed: No Required: No

Table C–1 (Cont.) Input Parameters for the Forecast Procedure

Parameter Name	Description
USECAPPING	A Boolean measure that indicates whether capping is applied. Data Type: Boolean Multiple Allowed: No Required: No
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
CAPINTERVALS	When set to *True,* interval and cumint are capped. Data Type: Boolean Multiple Allowed: No Required: No
VERBOSE	When set to *True,* detail information is added to log file. Useful for debugging. Data Type: Boolean Multiple Allowed: No Required: No

Output Parameters

Table C–2 provides the output parameters for the Forecast procedure.

Table C–2 Output Parameters for the Forecast Procedure

Parameter Name	Description
ALERT	A high-level forecast alert generated by the forecast engine. Data Type: Boolean Multiple Allowed: No Required: No
CHALPHA	ES alpha. Data Type: Real Multiple Allowed: No Required: No
CHLEVEL	ES level. Data Type: Integer Multiple Allowed: No Required: No
CHMETHOD	Selected method. Refer to Forecast Model/Model List table. Data Type: Integer Multiple Allowed: No Required: No

Table C–2 (Cont.) Output Parameters for the Forecast Procedure

Parameter Name	Description
CHTREND	ES trend. Data Type: Real Multiple Allowed: No Required: No
CUMINT	Cumulative interval forecast. Data Type: Real Multiple Allowed: No Required: No
FORECAST	Forecast output. Data Type: Real Multiple Allowed: No Required: Yes
INT	Interval forecast for Standard Deviation. Data Type: Real Multiple Allowed: No Required: No
PEAKS	Peaks, which are used for calculating baseline of the forecast. Data Type: Real Multiple Allowed: No Required: No

Forecast Method/Model List

Table C–3 provides the numeric value assigned to the forecast model/model list.

Table C–3 Numeric Values Assigned to the Forecast Model/Model List

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

Table C–3 (Cont.) Numeric Values Assigned to the Forecast Model/Model List

Model	Numeric Value
PROFILE	15
MOVING AVERAGE	17

D

Appendix: AppFunctions

The AppFunctions library supports a number of functions and special expressions, most of which are for internal use, and not recommended for customer use, except TransformSpread, which achieves higher performance than its peer in RPAS.

Supported Functions

The following functions are supported by AppFunctions:

AsDouble
clndstart
addperiods
datediff
datefn
dateToString
doubletoint
hiername
union_int_low
union_int_high
ispopulated
merge
parseDate
parseDateFromPosition
range
resizeCal

Supported Special Expression Functions

The following special expression functions are supported by AppFunctions:

activeindex
copyseries
maskedAgg
rangese
RepIndInRangeExpr
ScaleExpr
TransformSpread

TransformSpread

The TransformSpread special expression function converts data across hierarchies using different spreading flavors: transformProp, transformRepl, and transformEven.

Syntax

<target1, [target2, …, targetn]> <- TransformSpread (<transform spreading flavor>, <map>, Source [<hierarchy>].[<dimension>], Desination [<hierarchy>].[<dimension>]

< input1> [, < input2> … ,< inputn>])

Input Parameters

Table D–2 provides the input parameters for the TransformSpread function.

Table D–1 Input Parameters for the TransformSpread Function

Parameter Name	Description
Transform spreading flavor	Spreading flavor map
Source	[<hierarchy>].[<dimension>]
Destination	[<hierarchy>].[<dimension>]
Input1, [input2,...,inputn]	Input measures from which data are spread

Output Parameter

Table D–2 provides the output parameter for the TransformSpread function.

Table D–2 Output Parameter for the TransformSpread Function

Parameter Name	Description
target1, [target2, …, target]	Measures into which the source measures are spread to.

Example D–1 TransformSpread Function

```
mace -d . -run -expression "output1,output2,output3
<-TransformSpread(\"repl\",map,[CLSH].[CLST],[LOC].[STR],input1,input2,input3)"
```

E

Appendix: Configuring the Preprocess Special Expression

This appendix details these topics:

About the Preprocess Module

The purpose of the Oracle Retail Preprocess module, which may also be referred to as Preprocessing, is to correct past data points that represent unusual sales values that are not representative of a general demand pattern. Such corrections may be necessary when an item is out of stock and cannot be sold, which usually results in low sales. Preprocessing will adjust for stock out for both the current week and the following week because it assumes that the out of stock indicators represent end of week stock out. Data Correction may also be necessary in a period when demand is unusually high. The Preprocess module allows you to automatically make adjustments to the raw POS (Point of Sales) data so that subsequent demand forecasts do not replicate undesired patterns that are caused by lost sales or unusually high demand.

The Preprocess Syntax section contains the specifications and syntax for configuring the Preprocess function in the RPAS Configuration Tools. There is an RPAS multi-return function named preprocess and one RPAS special expression named preprocess. The special expression provides better performance; however, it only works in the batch mode. The multiple return function preprocess works in both batch mode and workbook mode. The syntax is exactly the same in both modes, except that procedures use <- instead of = in the expression.

Note: The syntax is slightly different than the standard RPAS functions and procedures that are described in the Rule Functions Reference Guide section of the *Oracle Retail Predictive Application Server Configuration Tools User Guide*.

Preprocess Requirements

The following libraries must be registered in any domains that will use the Preprocess solution extension:

- AppFunctions
- PreprocesssFunctions

Configuration Restrictions

The following restrictions apply to use the Preprocess function/procedure:

- An underscore (_) character may not be used in any measure names and rules unless the measures and rules are to be expanded using the RDF or Curve solution's classification scheme.

The classifications apply the AppFunctions and are as follows:

- _F: Expand measures and rules across final levels
- _S: Expand measures and rules across source levels
- _B: Expand measures and rules across birth dates

Preprocess Parameter/Model Dependencies

The following models require that the stated measure is to be provided.

- Bayesian model—Plan measure required.
- Profile model—Profile measure required.

Using the Preprocess Function

The following notes are intended to serve as a guide for configuring the Preprocess function within the RPAS Configuration Tools:

- The Preprocess function is an RPAS C++ special expression. 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 function call. The resultant measure parameters must be comma-separated in the function call as in the example.
- Because different filtering methods require different input parameters, it is necessary that every input parameter (measure or constant) must be accompanied by the corresponding label. All of the input measure parameters must be configured and registered before the function call. The input parameters must be comma-separated in the function call as in the example.
- The Preprocess function library must be registered after the domain build by using the regfunction RPAS utility.
- The Preprocess function required all the input and output measures using the same intersections. Mixed input/output measure intersections should be aligned to the same calculation intersection with other RPAS function/procedure before calling the Preprocess function. The same procedure can be carried out to the resultant measures to spread or aggregate them to the designated intersections.
- Because of the limitation that the same measure cannot simultaneously appear on both left-hand side and right-hand side, the implementation of the CLEAR filter requires the user to provide a LSOVER_REF measure (a duplication of the

previously calculated LSOVER measure) when you try to retain the results on certain time series but clear the others by providing a mask measure (TSMASK_DENSE). The LSOVER_REF is not required when the results for all the time series need to be cleared.

- The LSTODAY measure is used to specify the end date for the filter processing. It only accepts the index number for the end date along the calendar dimension as valid input. If it is desired that the string position name to be used for the end date specification, the available RPAS time dimension translation function index can be used to do the name-index conversion before calling the Preprocess function.
- The LSTODAY input parameter is designed to be a measure rather than a constant to provide more flexibility. Current implementation only allows one global LSTODAY index value to be used in processing all the time series. To specify the end date, you just need to populate its value for the first time series, and this index will be applied to all the other time series.
- The index value in the LSTODAY measure starts from 0.
- FLP_FIRST and FLP_LAST are the resultant measures to be used for the First-Last-Populated Location calculation. They do not have the calendar dimension, and each of their cell values represent the indices for the first and last populated locations along the calendar dimension from the first time series up to the current time series, respectively.
- TSMASK_DENSE is a Boolean input measure without calendar dimension to specify which time series is going to be processed and which is not. For filtering methods other than the CLEAR method, the true value means that it will be processed if the popcount for the current time series is larger than the hard-coded threshold value. Otherwise, it will not be processed. The false value means that the current time series will not be processed. If the TSMASK_DENSE measure is not specified, all the time series will be processed and the internal hard-coded threshold value will not be considered. For the CLEAR filtering method, the true value means that the previously calculated results for the current time series will be cleared and the false value means the results will be retained. If the TSMASK_DENSE measure is not specified, all the results will be cleared.
- For all the input measures that do not have the calendar dimension, such as UP_ADJ_RATIO and DELTA, you can use a constant as input. In this case, the constant value will be applied to all the time series.

Preprocess Syntax

The syntax for using the Preprocess is shown in the following examples. The input and output parameter tables explain the specific usage of the parameters names use in the function/procedure.

Example E–1 Generic Example 1:

LSOVER: LSOVERMEAS, LS: LSMEAS, [, TSALERT: TSALERTMEAS, SERVICE_LEVEL: SERVICELEVELMEAS, STOCK_LEVEL: STOCKLEVELMEAS, FLP_FIRST: FLPFIRSTMEAS, FLP_LAST: FLPLASTMEAS] <- preprocess(SRC: SRCMEAS, LSTODAY: LSTODAYMEAS, NPTS: NPTSMEAS [, MIN_TSALERT: MINTSALERTMEAS, OUTAGE: OUTAGEMEAS, TSMASK_DENSE: TSMASKMEAS, UP_ADJ_RATIO: UPADJMEAS, DOWN_ADJ_RATIO: DOWNADJMEAS, REFERENCE: REFMEAS, DEVIATION: DEVMEAS {, WINDOW: WINDOWMEAS |, WINDOW1: WINDOW1MEAS, WINDOW2: WINDOW2MEAS, WINDOW3: WINDOW3MEAS, WINDOW4: WINDOW4MEAS, WINDOW5: WINDOW5MEAS} {,

ALPHA: ALPHAMEAS, NPAST: NPASMEAS, NFUT: NFUTMEAS} {, NSIGMA_MIN: NSIGMA_MINMEAS, NSIGMA_MAX: NSIGMA_MAXMEAS |, NSIGMAOUT_MIN: NSIGMAOUT_MINMEAS, NSIGMAOUT_MAX: NSIGMAOUT_MAXMEAS, NSIGMAADJ_MIN: NSIGMAADJ_MINMEAS, NSIGMAADJ_MAX: NSIGMAADJ_MAXMEAS} {, FRCST_MIN: FRCST_MINMEAS, HIST_MIN_FS: HIST_MIN_FSMEAS} {, PRICE: PRICEMEAS, INVENTORY: INVENTORYMEAS, HIST_MIN_MD: HISTMINMDMEAS}, DELTA: DELTAMEAS, LSOVER_REF: LSOVERREFMEAS]

Example E–2 Generic Example 2:

LSOVER: LSOVERMEAS, LS: LSMEAS, [, TSALERT: TSALERTMEAS, SERVICE_LEVEL: SERVICELEVELMEAS, STOCK_LEVEL: STOCKLEVELMEAS, FLP_FIRST: FLPFIRSTMEAS, FLP_LAST: FLPLASTMEAS] <-preprocess(SRC: SRCMEAS, LSTODAY: LSTODAYMEAS, NPTS: NPTSMEAS [, MIN_TSALERT: MINTSALERTMEAS, OUTAGE: OUTAGEMEAS, TSMASK_DENSE: TSMASKMEAS, UP_ADJ_RATIO: UPADJMEAS, DOWN_ADJ_RATIO: DOWNADJMEAS, REFERENCE: REFMEAS, DEVIATION: DEVMEAS {, WINDOW: WINDOWMEAS |, WINDOW1: WINDOW1MEAS, WINDOW2: WINDOW2MEAS, WINDOW3: WINDOW3MEAS, WINDOW4: WINDOW4MEAS, WINDOW5: WINDOW5MEAS} {, ALPHA: ALPHAMEAS, NPAST: NPASMEAS, NFUT: NFUTMEAS} {, NSIGMA_MIN: NSIGMA_MINMEAS, NSIGMA_MAX: NSIGMA_MAXMEAS |, NSIGMAOUT_MIN: NSIGMAOUT_MINMEAS, NSIGMAOUT_MAX: NSIGMAOUT_MAXMEAS, NSIGMAADJ_MIN: NSIGMAADJ_MINMEAS, NSIGMAADJ_MAX: NSIGMAADJ_MAXMEAS} {, FRCST_MIN: FRCST_MINMEAS, HIST_MIN_FS: HIST_MIN_FSMEAS} {, PRICE: PRICEMEAS, INVENTORY: INVENTORYMEAS, HIST_MIN_MD: HISTMINMDMEAS}, DELTA: DELTAMEAS]

Example E–3 Sample 1:

LSOVER:LSOVER1, LS:LS1, TSALERT:TSALERT1 <- preprocess(SRC:POS, METHODID:MTHID, LSTODAY:TODAY1, NPTS:NPTS, WINDOW:WIN)

Example E–4 Sample 2:

LSOVER:LSOVER1, LS:LS1, TSALERT:TSALERT1 <- preprocess(SRC:POS, METHODID:MTHID, LSTODAY:TODAY1, NPTS:NPTS, WINDOW:WIN)

Configuration Parameters and Rules

Input Parameters

Table E–1 provides the input parameters for the Preprocess procedure.

Table E–1 Input Parameters for the Preprocess Procedure

Parameter Name	Description
SRC	The source data. Data Type: Real Required: Yes
METHODID	The filtering method ID. Data Type: Real Required: Yes

Table E–1 (Cont.) Input Parameters for the Preprocess Procedure

Parameter Name	Description
LSTODAY	The end date for filter processing. Data Type: Real Required: Yes
NPTS	The number of points into history that will be filtered. Data Type: Real Intersection: Required: Yes
MIN_TSALERT	The threshold value used to set off TSALERT. Data Type: Real Required: No
OUTAGE	The outage indicator. Data Type: Boolean Required: No
TSMASK_DENSE	A Boolean value to specify which time series will be processed. Data Type: Boolean Required: No
UP_ADJ_RATIO	The upward adjustment ratio that will be applied on LS. Data Type: Real Required: No Default value: 1.0* * If the measure is not specified, the default value will be applied to each of the time series to be processed.
DOWN_ADJ_RATIO	The downward adjustment ratio that will be applied on LS. Data Type: Real Required: No Default value: 1.0* * If the measure is not specified, the default value will be applied to each of the time series to be processed.
REFERENCE	Reference will be used for source data substitution. Data Type: Real Required: No
DEVIATION	The standard deviation for confidence interval calculation by Forecast Sigma filters. Data Type: Real Required: No
WINDOW	Filter window length for Standard Median filter. Data Type: Real Required: No Default value: 13

Table E–1 (Cont.) Input Parameters for the Preprocess Procedure

Parameter Name	Description
WINDOW1	First round filter window length for Oracle Retail Median filter. Data Type: Real Required: No Default value: 13
WINDOW2	Second round filter window length for Oracle Retail Median filter. Data Type: Real Required: No Default value: 19
WINDOW3	Third round filter window length for Oracle Retail Median filter. Data Type: Real Required: No Default value: 7
WINDOW4	Forth round filter window length for Oracle Retail Median filter. Data Type: Real Required: No Default value: 5
WINDOW5	Fifth round filter window length for Oracle Retail Median filter. Data Type: Real Required: No Default value: 11
ALPHA	The exponential coefficient used to evaluate past and future velocities. Data Type: Real Required: No Default value: 0.2
NPAST	The maximum number of historical points to calculate past velocity. Data Type: Real Required: No Default value: 5
NFUT	The maximum number of historical points to calculate future velocity. Data Type: Real Required: No Default value: 5
NSIGMA_MIN	The number of standard deviations for lower bound calculation. Data Type: Real Required: No Default value: 3.0

Table E–1 (Cont.) Input Parameters for the Preprocess Procedure

Parameter Name	Description
NSIGMA_MAX	The number of standard deviations for upper bound calculation. Data Type: Real Required: No Default value: 3.0
FRCST_MIN	The forecast lower bound for Forecast Sigma filters. Data Type: Real Required: No Default value: 0.1
HIST_MIN_FS	The minimum number of historical points required for Forecast Sigma filters. Data Type: Real Required: No Default value: 5
NSIGMAOUT_MIN	The number of standard deviations for lower outlier calculation. Data Type: Real Required: No Default value: 3.0
NSIGMAOUT_MAX	The number of standard deviations for upper outlier calculation. Data Type: Real Required: No Default value: 3.0
NSIGMAADJ_MIN	The number of standard deviations for lower bound calculation. Data Type: Real Required: No Default value: 1.5
NSIGMAADJ_MAX	The number of standard deviations for upper bound calculation. Data Type: Real Required: No Default value: 1.5
DELTA	Ratio of reference will be used to copy or increase for OVERRIDE and INCREMENT filters. Data Type: Real Required: No Default value: 1.0* * If the measure is not specified, the default value will be applied to each of the time series to be processed.
LSOVER_REF	Data will be used to override SRC. Used by CLEAR filter only. Data Type: Real Required: No

Table E–1 (Cont.) Input Parameters for the Preprocess Procedure

Parameter Name	Description
POA	Partial Outage Flag. Used by STD ES LS filter only. If set to *False*, the period immediately following an out-of-stock period will not be adjusted. The default value is *True*. Data Type: Boolean Required: No
EVENT_FLAG	Used by STD ES and STD ES LS filters. If a value is *True*, the given period is not used in the calculation of the past or future velocities. Data Type: Boolean Required: No

Output Parameters

Table E–2 provides the output parameters for the Preprocess function/procedure.

Table E–2 Output Parameters for the Preprocess Procedure

Parameter Name	Description
LSOVER	Adjusted source data. It is the Primary Result. LSOVER = SRC + LS Data Type: Real Required: Yes
LS	The adjustment on the source data. Data Type: Real Required: Yes
TSALERT	Boolean flag set to *True* when more than MIN_TSALERT number of data points have been modified. Data Type: Boolean Required: No
SERVICE_LEVEL	SERVICE_LEVEL = SRC / LSOVER Data Type: Real Required: No
STOCK_LEVEL	Used by Mark Down filter only. Data Type: Real Required: No
FLP_FIRST	First populated position. Used by FLP filter only. Data Type: Real Required: No
FLP_LAST	Last populated position. Used by FLP filter only. Data Type: Real Required: No

Lost Sales Method/Model List

Table E–3 provides the numeric value assigned to the forecast model/model list.

Table E–3 Numeric Values Assigned to the Lost Sales Model/Model List

Model	Numeric Value	Comments
MEDIAN5	0	Oracle Retail Median. Required input parameters: None Optional input parameters: ■ WINDOW1 ■ WINDOW2 ■ WINDOW3 ■ WINDOW4 ■ WINDOW5
MEDIAN1	1	Standard Median. Required input parameters: None Optional input parameters: WINDOW
OVERRIDE	2	Override Required input parameters: REFERENCE Optional input parameters: DELTA
INCREMENT	3	Increment. Required input parameters: REFERENCE Optional input parameters: DELTA
ES_LT	4	Standard ES. Required input parameters: OUTAGE Optional input parameters: ■ ALPHA ■ NPAST ■ NFUT ■ EVENT_FLAG
LS_ES_LT	9	Lost Sales—Standard ES. Required input parameters: OUTAGE Optional input parameters: ■ ALPHA ■ NPAST ■ NFUT ■ EVENT_FLAG ■ POA
FRCST_SIGMA	14	Forecast and standard deviation algorithm. Required input parameters: ■ REFERENCE ■ DEVIATION Optional input parameters: ■ NSIGMA_MAX ■ NSIGMA_MIN ■ FRCST_MIN ■ HIST_MIN_FS

Table E–3 (Cont.) Numeric Values Assigned to the Lost Sales Model/Model List

Model	Numeric Value	Comments
FRCST_SIGMA_EVENT	15	Forecast and standard deviation algorithm with Event. Required input parameters: ■ OUTAGE ■ REFERENCE ■ DEVIATION Optional input parameters: NSIGMAOUT_MAX ■ NSIGMAOUT_MIN ■ NSIGMAADJ_MAX ■ NSIGMAADJ_MIN ■ FRCST_MIN ■ HIST_MIN_FS
CLEAR	17	Clear—clears specified result measures. Required input parameters: None Optional input parameters: ■ TSMASK_DENSE ■ LSOVER_REF
NO_FILT	19	No filtering. Required input parameters: None Optional input parameters: None
DEPRICE	22	Remove pricing effects Required input parameters:price, maximum price Optional input parameters:none
FLP: first populated location	20	Required input: src and method

Preprocess Filtering Methods

This section details the following Preprocess filtering methods:

- Standard Median
- Oracle Retail Median
- Standard Exponential Smoothing
- Lost Sales—Standard Exponential Smoothing
- Forecast Sigma
- Forecast Sigma Event
- Override
- Increment
- Clear
- DePrice

Standard Median

Standard Median is recommended for getting data baselines on long time ranges when promo indicators are not available.

A standard median filter implementation:

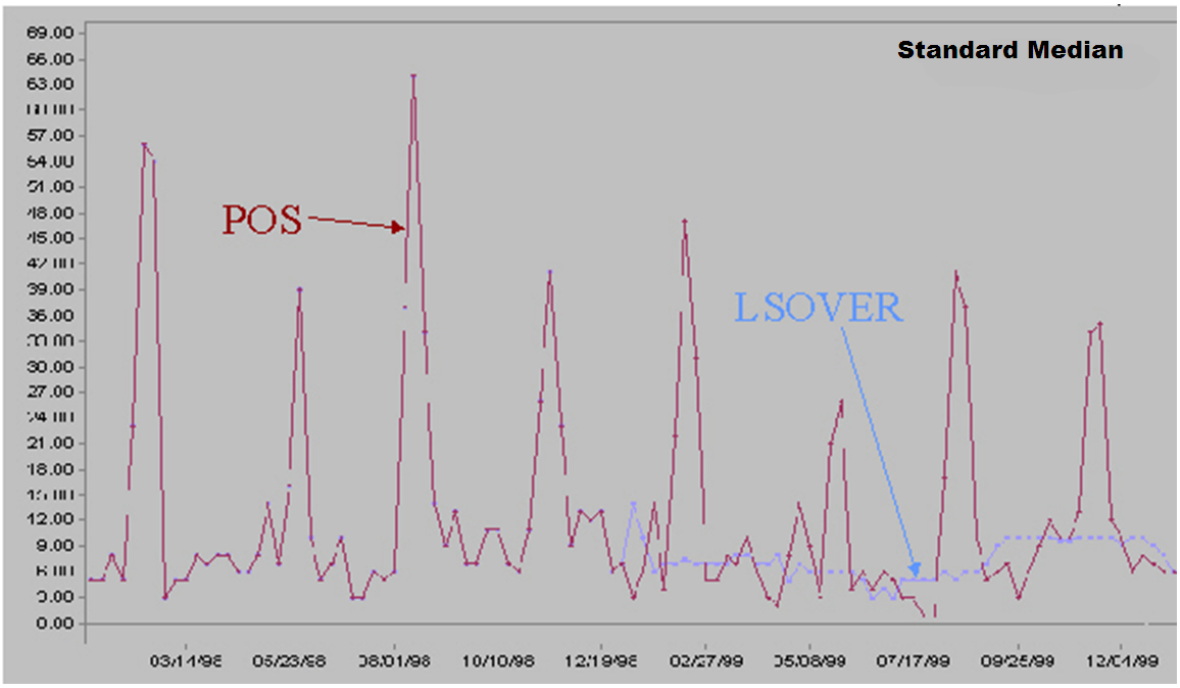
- Does not take outage information as an input.
- Can use one optional parameter: window length.

Mathematical Formulation

LSOVER(t) = median value of SRC over [t-window/2, t+window/2],

Where: window is the parameter window length of the filter.

Figure E–1 Standard Median with Window = 13 points



Example E–5 Standard Median with Window = 13 points

LSOVER:lsover1, LS:ls1, TSALERT:tsalert1 <- preprocess(SRC:pos, METHODID:mthid, LSTODAY:today1, NPTS:npts, WINDOW:win)

Oracle Retail Median

Oracle Retail Median is recommended for getting data baselines on long time ranges when promo indicators are not available.

Oracle Retail Median provides the following features:

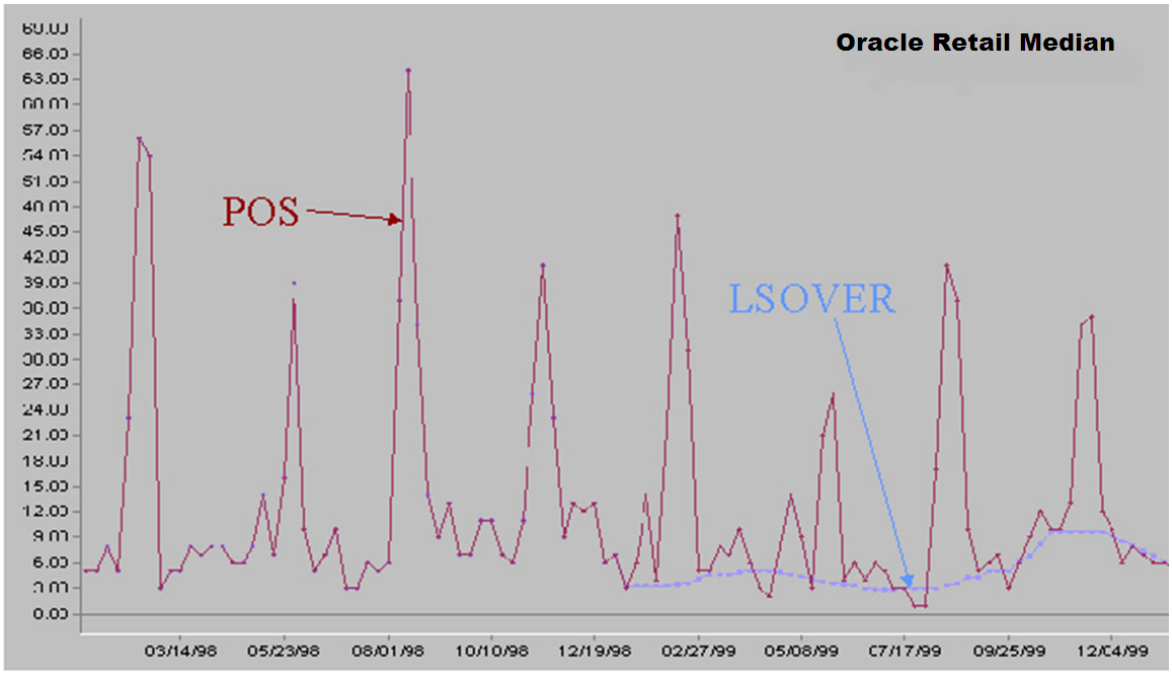
- A sophisticated median filter that takes trends into consideration and improves side effects over the standard median filter. It makes five standard median filter passes.
- Does not take outage information as an input.

- Can accept five optional parameters: window length for each pass.

Mathematical Formulation

1. The first two passes recursively apply the standard median filter. The result is denoted by MEDIAN_2(t). The one-step difference of MEDIAN_2(t) is calculated. That is, DIFF_1(t) = MEDIAN_2(t) - MEDIAN_2(t-1). Then, the standard median filter is applied to DIFF_1(t). The result is denoted by MEDIAN_DIFF_1(t).
2. Using MEDIAN_DIFF_1(t), a first smoothed version (that is, baseline) of the source data is calculated at the third step: SMOOTH_1(t) = SMOOTH_1(t-1) + MEDIAN_DIFF_1(t) on points where the absolute deviation of SRC(t) over its mean is larger than half of the global absolute standard deviation. Otherwise, SMOOTH_1(t) = SRC(t).
3. To prepare for the fourth pass, the one-step difference of SMOOTH_1(t) is calculated. That is, DIFF_2(t) = SMOOTH_1(t) - SMOOTH_1(t-1). An average version of DIFF_2(t) is calculated using the standard median filter. The result is denoted by AVG_DIFF_2(t). The result of the fourth pass is SMOOTH_2(t) = SMOOTH_2(t-1) + AVG_DIFF_2(t).
4. Finally, LSOVER(t) is the result of applying the standard median filter to SMOOTH_2(t).

Figure E–2 Oracle Retail Median with Default Parameters



Example E–6 Oracle Retail Median with Default Parameters

LSOVER:lsover1, LS:ls1, TSALERT:tsalert1 <- preprocess(SRC:pos, METHODID:mthid, LSTODAY:today1, NPTS:npts, WINDOW1:win, WINDOW2:win2, WINDOW3:win3, WINDOW4:win4, WINDOW5:win5)

Standard Exponential Smoothing

Standard Exponential Smoothing (Std ES) is recommended for removing specific spikes of low or non-seasonal data when spike indicators are available.

Standard Exponential Smoothing provides the following features:

- Based on standard Exponential Smoothing calculations of past and future sales velocities.
- Must have unusual event measure info as input (that is, also referred to as outage).
- Does not take into consideration seasonal components.
- Four optional parameters:
  - Alpha (ES parameter used to evaluate past and future velocities)
  - Maximum number of historical points to calculate past velocity
  - Maximum number of future points to calculate future velocity
  - EVENT_FLAG specifies the periods that are used to calculate the future and/or past velocities

Mathematical Formulation

Std ES is the standard Exponential Smoothing filter. It preprocesses a subset of points as predetermined by an input measure. For every contiguous sequence of points to adjust, say between *tf* and *tl*, a past velocity and a future velocity are calculated using an exponentially weighted average. For the points between *tf* and *tl*, the adjustment is calculated as a linear interpolation of the past and future velocities.

Figure E–3 Standard Exponential Smoothing Calculations

$$Past\_Velocity = \frac{\sum_{i=1}^{np}(1-\alpha)^{i-1} * SRC(t_f - i)}{\sum_{i=1}^{np}(1-\alpha)^{i-1}}$$

$$Future\_Velocity = \frac{\sum_{i=1}^{nf}(1-\alpha)^{i-1} * SRC(t_l + i)}{\sum_{i=1}^{nf}(1-\alpha)^{i-1}}$$

$$LSOVER(t) = Past\_Velocity + \frac{Future\_Velocity - Past\_Velocity}{t_l - t_f + 2} * (t - t_f + 1), \forall t \in [t_f, t_l]$$

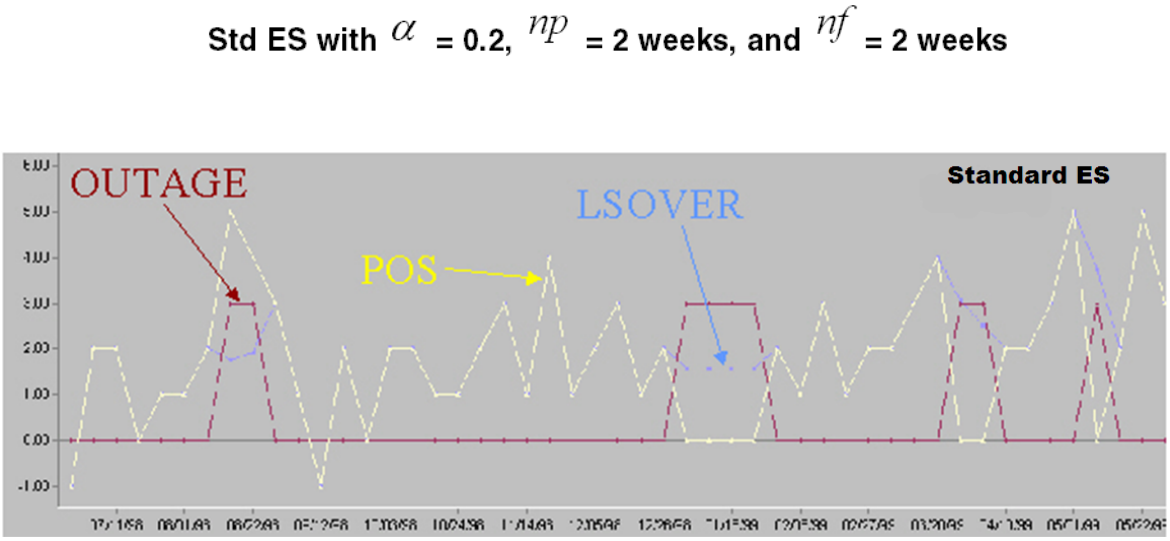
Where:

α is the exponential coefficient used to evaluate past and future velocities.

np is the maximum number of historical points to calculate past velocity.

nf is the maximum number of future points to calculate future velocity.

Figure E–4 Standard Exponential Smoothing



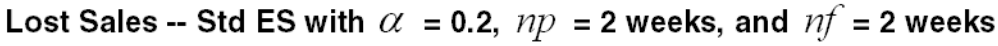
Example E–7 Standard Exponential Smoothing

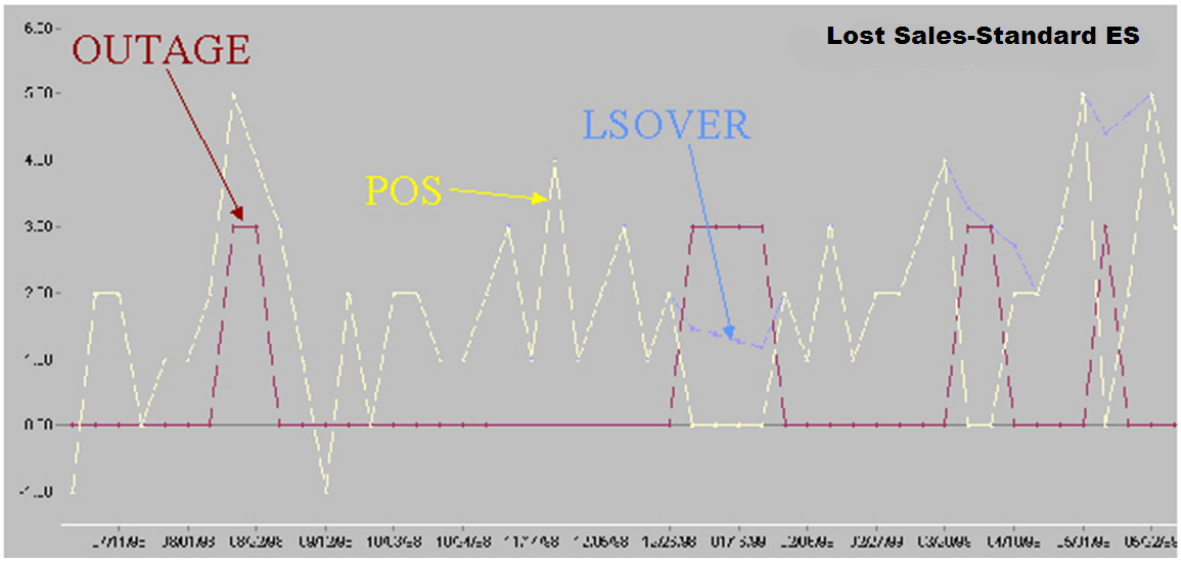
LSOVER:lsover1, LS:ls1, TSALERT:tsalert1 <- preprocess(SRC:pos, METHODID:mthid, LSTODAY:today1, NPTS:npts, OUTAGE:outage1, ALPHA:alpha, NPAST:npast, NFUT:nfut)

Lost Sales—Standard Exponential Smoothing

Lost Sales—Standard Exponential Smoothing functions like Std ES with two exceptions. First, it only adjusts lost sales (that is, negative spikes). Second, it can adjust not only the out-of-stock period but also the period immediately following such a period (partial outage period).

Figure E–5 Lost Sales—Standard Exponential Smoothing





Example E–8 Lost Sales—Standard Exponential Smoothing

LSOVER:lsover1, LS:ls1, TSALERT:tsalert1 <- preprocess(SRC:pos, METHODID:9, LSTODAY:today1, NPTS:30, OUTAGE:outage1, ALPHA:0.2, NPAST:5, NFUT:5,EVENT_FLAG: event_flag_measure, POA:TRUE)

Forecast Sigma

Forecast Sigma is recommended for removing recent spiky data points when approved forecasts and approved confidence intervals are available on the filtering window, but spike indicators are not available. This method is based on the principle that if a data point significantly deviates from an approved forecast, this data point is likely to be an unusual event that should be overridden in the source measure (POSOVER) used by the forecasting engine. It is adjusted by bringing the override value within some bounds of the approved forecast as defined by a proportional coefficient scalar of the forecasts' standard deviation.

Forecast Sigma provides the following features:

- Does not take outage information as an input
- Requires two parameters:
  - Approved forecast array
  - Approved standard deviation array of forecast
- Can accept four optional parameters:
  - Number of standard deviations for upper bound
  - Number of standard deviations for lower bound.
  - Forecast lower bound
  - Minimum item history (# points) required for filtering

Mathematical Formulation

This method relies on approved forecasts with their corresponding confidence intervals. It adjusts the points that are far (as defined by a multiple of the forecast

standard deviation) from their corresponding previously approved forecasts by bringing the override values to their closest confidence interval bounds.

IF # historical points < MinHist **THEN**
LSOVER(t) = SRC(t)
ELSE IF forecast(t) < MinFrcst **THEN**
forecast(t) = MinFrcst **AND** σ = MinFrcst
ELSE IF $\sigma = 0$ **THEN**
IF forecast(t) < 1.0 **THEN**
σ = forecast(t)
ELSE $\sigma = \sqrt{\text{forecast(t)}}$
IF SRC(t) > forecast(t) + nsu*σ **THEN**
LSOVER(t) = forecast(t) + nsu*σ
ELSE IF SRC(t) < forecast(t) – nsl*σ **THEN**
LSOVER(t) = forecast(t) - nsl*σ
ELSE LSOVER(t) = SRC(t)

Where:

- nsu is the number of standard deviations for upper bound.
- nsl is the number of standard deviations for lower bound.
- MinFrcst is the forecast lower bound.
- MinHist is the minimum item history (# points) required for filtering.

Example E–9 Lost Sales—Forecast Sigma with nsu = 3, nsl = 3, minFrcst = 0.1 and minHist = 5 weeks

LSOVER:LSOVER1, LS:LS1, TSALERT:TSALERT1 <- preprocess(SRC:POS, METHODID:mthid, LSTODAY:TODAY1, NPTS:npts, REFERENCE:forecast1, DEVIATION:dev1, NSIGMA_MIN:nsigma_min, NSIGMA_MAX:nsigma_max, FRCST_MIN:0.1, HIST_MIN_FS:hist_min_fs)

Forecast Sigma Event

This is similar to Forecast Sigma. It takes an outage (for instance, event) indicator to further process.

Mathematical Formulation

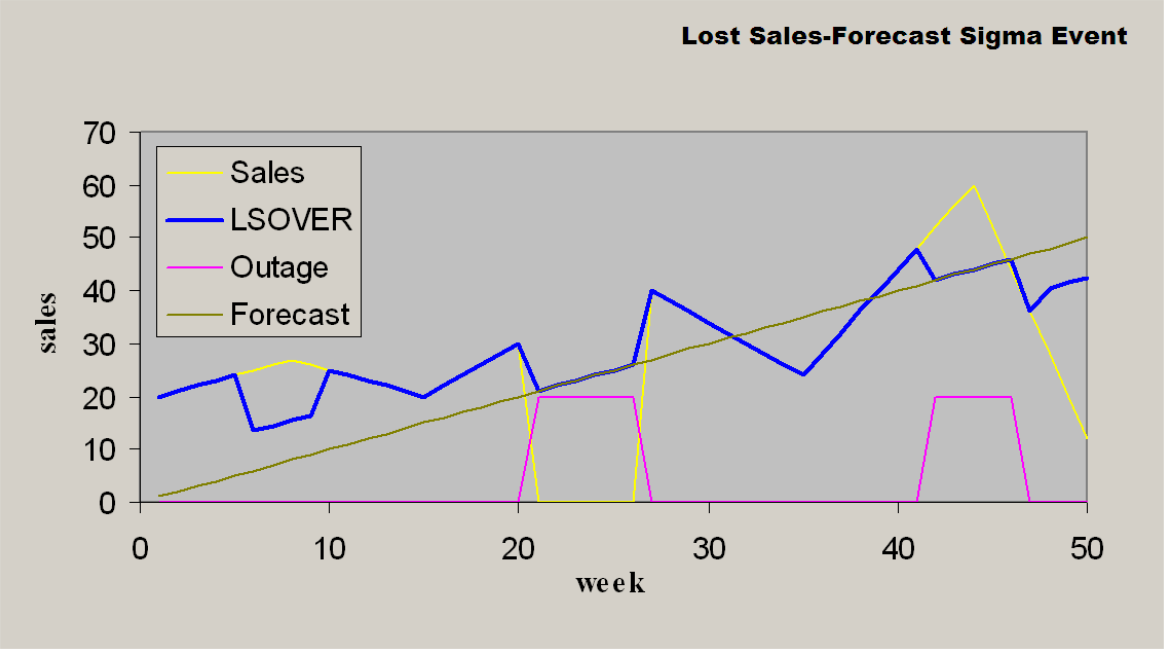
When the outage/event mask is ON:

LSOVER(t) = forecast(t)

When the outage/event mask is OFF:

If the data points that are outside of the outliers calculated through NSIGMAOUT_MIN and NSIGMAOUT_MAX, they will be brought into the confidence interval bounds, which are defined through NSIGMAADJ_MIN and NSIGMAADJ_MAX.

Figure E–6 Lost Sales—Forecast Sigma Event



Lost Sales—Forecast Sigma Event with nsigmaout_min = 3, nsigmaout_max = 3, nsigmaadj_min = 1.5, nsigmaadj_max = 1.5,

minFrcst = 0.1 and minHist = 5 weeks

Example E–10 Lost Sales—Forecast Sigma Event

LSOVER:LSOVER1, LS:LS1, TSALERT:TSALERT1 <- preprocess(SRC:POS, METHODID:mthid, LSTODAY:TODAY1, NPTS:npts, OUTAGE:outage1, REFERENCE:forecast1, DEVIATION:dev1, NSIGMAOUT_MIN:nsigmaout_min, NSIGMAOUT_MAX:nsigmaout_max, NSIGMAADJ_MIN:nsigmaadj_min, NSIGMAADJ_MAX:nsigmaadj_max, FRCST_MIN:frcst_min, HIST_MIN_FS:hist_min_fs)

Override

This method overrides the destination measure with the source measure that is adjusted by the adjustment percentage according to the mask. It is recommended for filling data gaps when an existing reference measure exists as a default value.

Override provides the following features:

- It is a simple data copy of a given percentage of the reference data to copy from.
- This may or may not take outage (for instance, event) info as an input to mask the operation.
- Requires two parameters:
  - Reference measure to copy data from

  - Source measure for the original data
- Can accept one optional parameter, Ratio of reference to actually copy.

Mathematical Formulation

This method uses the following parameters:

- A source measure that can be any measure in the system as long as it has the same intersection as the destination measure
- A reference measure that can be any measure in the system as long as it has the same intersection as the destination measure
- A destination measure that can be any measure in the system as long as it has the same intersection as the source measure
- A mask that is a Boolean measure that has the same intersection as the source and destination measures
- An adjustment percentage

This method overrides the destination measure with the source measure adjusted by the adjustment percentage according to the mask:

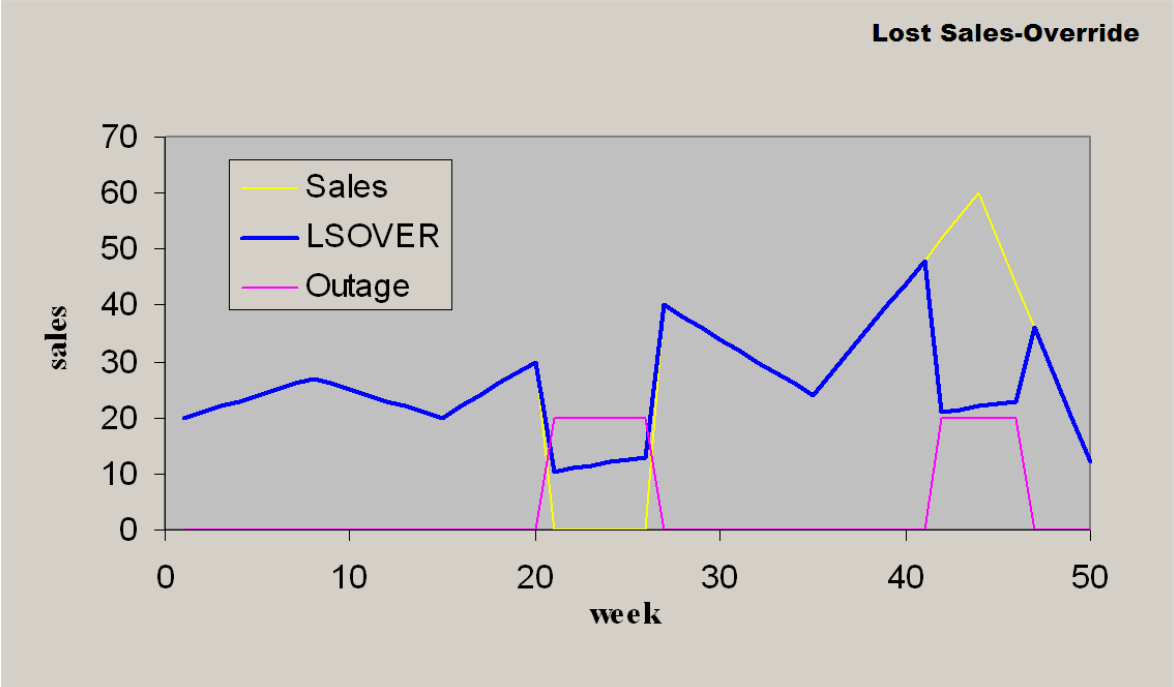
Let:

- S(i) is the value in cell (i) of the source measure
- R(i) is the value in cell (i) of the reference measure
- D(i) is the value in cell (i) of the destination measure
- M(i) is the value of cell (i) of the mask
- a is an adjustment percentage

The result of the override method is:

- D(i) = a * R(i) if M(i) is *True*
- D(i) = S(i) if M(i) is *False*

Figure E–7 Lost Sales—Override with Delta = 0.5



Example E–11 Lost Sales—Override with Delta = 0.5

LSOVER:lsover1, LS:ls1, TSALERT:tsalert1 <- preprocess(SRC:pos, METHODID:mthid, LSTODAY:today1, NPTS:npts, REFERENCE:ref1, OUTAGE:outage1, DELTA:delta1)

Increment

This method increments or decrements the destination measure by the source measure, which is adjusted by the adjustment percentage according to the mask. It is recommended for updating outliers or data gaps when an existing reference measure exists as a default adjustment.

Increment provides the following features:

- It is a simple data increment of a given percentage of the reference data to copy from.
- It may or may not take outage information (for example, event) as an input to mask the operation.
- Has one required parameter, Reference measure to increment by.
- Can accept one optional parameter, Ratio of reference to actually increment by.

Mathematical Formulation

This method uses the following inputs:

- A source measure that can be any measure in the system as long as it has the same intersection as the destination measure.
- A reference measure that can be any measure in the system as long as it has the same intersection as the destination measure.

- A destination measure that can be any measure in the system as long as it has the same intersection as the source measure.
- A mask that is a Boolean measure that has the same intersection as the source and destination measures.
- An adjustment percentage.

This method increments or decrements the destination measure by the source measure, which is adjusted by the adjustment percentage according to the mask.

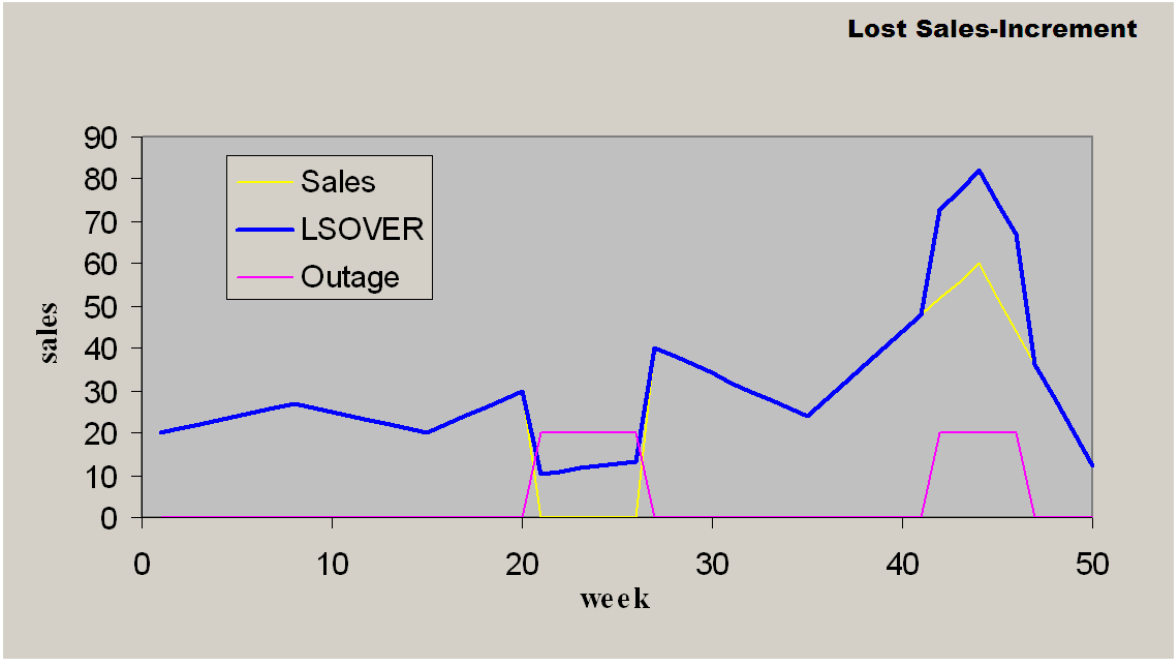
Let:

- S(i) is the value in cell (i) of the source measure
- R(i) is the value in cell(i) of the reference measure
- D(i) is the value in cell (i) of the destination measure
- M(i) is the value of cell (i) of the mask
- a is an adjustment percentage (can be between (-100%) and (+100%))

The result of the reduction method is:

- D(i) = S(i) + a * R(i) if M(i) is *True*
- D(i) = S(i) if M(i) is *False*

Figure E–8 Lost Sales—Increment with Delta = 0.5



Example E–12 Lost Sales—Increment with delta = 0.5

LSOVER:lsover1, LS:ls1, TSALERT:tsalert1 <- preprocess(SRC:pos, METHODID:mthid, LSTODAY:today1, NPTS:npts, REFERENCE:ref1, OUTAGE:outage1, DELTA:delta1)

Clear

This is used for canceling the effect of some former preprocessing adjustments.

Clear provides the following features:

- Does not take outage information as an input.
- May or may not take time series mask (does not have calendar dimension) input to retain results for certain time series.
- If time series mask is specified, one duplicated LSOVER measure must be provided in addition to the original LSOVER measure.

Mathematical Formulation

IF TimeSeriesMask is provided & TimeSeriesMask = false **THEN**

LSOVER(t) = LSOVER_REF(t)

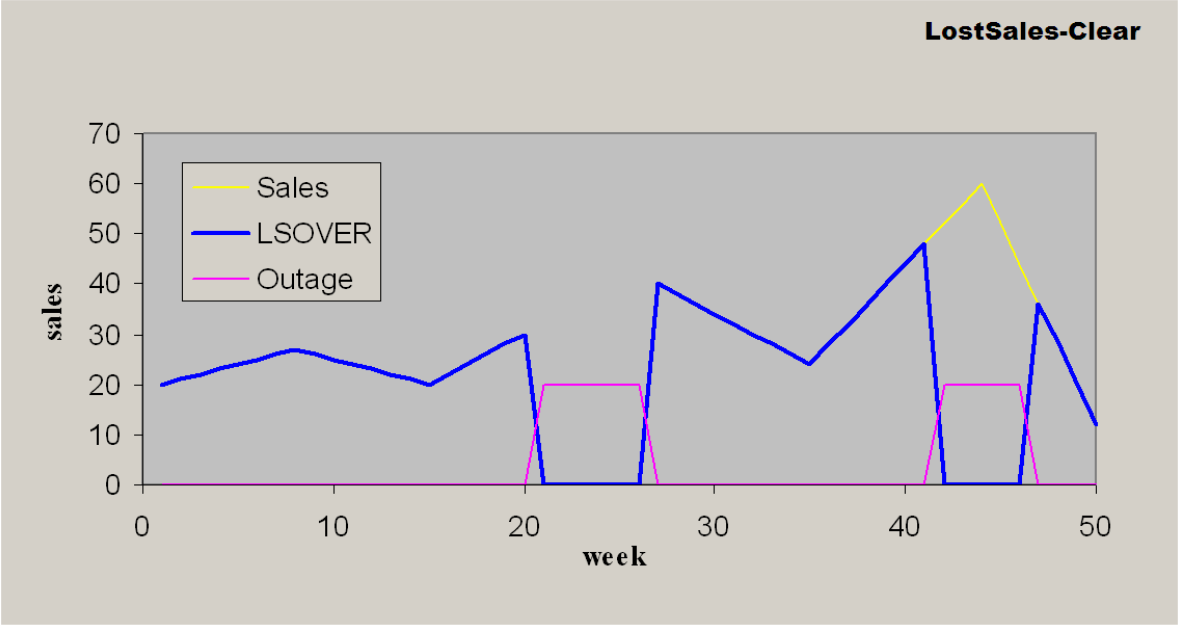
LS(t) = LSOVER_REF(t) - SRC(t)

ELSE

LSOVER(t) = 0

LS(t) = 0

Figure E–9 Preprocess—Clear with TS_Mask



Example E–13 Clear All

LSOVER:LSOVER1, LS:LS1, TSALERT:TSALERT1 <- preprocess(SRC:POS, METHODID:mthid, LSTODAY:TODAY1, NPTS:npts)

Example E–14 Partial Clear with Mask Input

LSOVER:LSOVER1, LS:LS1, TSALERT:TSALERT1 <- preprocess(SRC:POS, METHODID:mthid, LSTODAY:TODAY1, NPTS:npts, TSMASK_DENSE:tsMask1, LSOVER_REF:lsoverref1)

DePrice

A filter removing pricing effects:
smoothed=original*(price/maxprice)*(price/maxprice)

LSOVER:LSOVER1, LS:LS1 <-preprocess(SRC:POS, METHODID:mthid, LSTODAY:TODAY1, NPTS:npts, TSMASK_DENSE:tsMask1,PRICE:price.MAX LSOVER_REF:lsoverref1)

F

Appendix: Customizing Hooks for the RDF Generate Utility and Curvebatch

RDF and Curve provide a number of hooks for running customized computation at certain points of the batch process.

Generate is an RDF utility that runs the RDF batch, such as generating profiles, generating source level forecasts, and approving forecasts. Often, some customized computation is needed during the running of *generate.* For instance, after the generation of forecast and prior to the forecast approval, you may want to run some rules to compute the value of the approval alerts.

Curvebatch is a Curve utility that runs profile generation batch, such as range data source, generate source level profiles, and merge profiles.

This appendix details these topics:

Hooks

There are several hooks provided in the RDF Generate and Curvebatch utilities so that custom expressions can be run at various predefined phases of the batch.

The hooks listed in the following tables are defined in the appcust.xml file.

Phases and Hooks in RDF

Each phase of the batch cycle begins and ends with a hook that links to the next phase. Table F–1 lists and describes the RDF batch phases along with its hooks. These hooks are provided in the RDF Generate utility and defined in the appcust.xml file as shown in Figure F–1.

Table F–1 RDF Phases and Hooks

Phase	Phase Action	This Hook...	Is Run...
Initialization	Prepares initializing environment for forecast generation	preinit	before initialization starts
		postinit	after initialization
Forecast Generation	Generates forecast	pregen	before generate forecast
		postgen	after generate forecast

Table F–1 (Cont.) RDF Phases and Hooks

Phase	Phase Action	This Hook...	Is Run...
Like Functionality Performance	Runs a like item sister store functionality	prelikets	Before likets function
		postlikets	After likets function
Forecast Adjustment	Automatic adjustment for forecast	preadjust	Before adjust forecast
		postadjust	After adjust forecast
Alert Execution	Runs alerts	prealert	Before generate alerts
		postalert	After generate alerts
Forecast Approval	Automatic approval of forecast	preapprove	Before forecast approval
		postapprove	After forecast approval

Phases and Hooks in Curve

Each phase of the batch cycle begins and ends with a hook that links to the next phase. Table F–2 lists and describes the Curve batch phases along with its hooks. These hooks are provided in the Curvebatch utility and defined in the appcust.xml file as shown in Figure F–2.

Table F–2 Curve Phases and Hooks

Phase	Phase Action	This Hook...	Is Run...
Ranging	Ranging the source data based on training window	prerangesource	Before rangeDataSource is run
		postrangesource	After rangeDataSource is run
Profile Generation	Generate profile at all source levels	prerunsource	One time before SourceLevel:run is iteratively performed on all source levels of the profile
		postrunsource	One time after SourceLevel:run is iteratively performed on all source levels of the profile
Profile Merger	Merge profiles	premerge	Before merge is run
		postmerge	After merge is run
Profile Reshape	Reshape profiles	prereshape	Before reshape is run
		postreshape	After reshape is run
Profile Renormalize	Renormalize profiles	prerenormalize	Before renormalize is run
		postrenormalize	After renormalize is run
Profile Clip	Clip profiles based on phase start and end dates	prephaseclip	Before phaseClip is run
		postphaseclip	After phaseClip is run
Profile Approval	Automatic approvals of profiles	preapprove	Before approve is run
		postapprove	After is approve run

About appcust.xml

The hooks listed in "Hooks" on page F-1 are defined in the appcust.xml file. For the setup directory of each domain, appcust.xml must be included.

The format of the RDF appcust.xml file is shown in Figure F–1.

The format of the Curve appcust.xml file is shown in Figure F–2.

Figure F–1 Format of appcust.xml for RDF

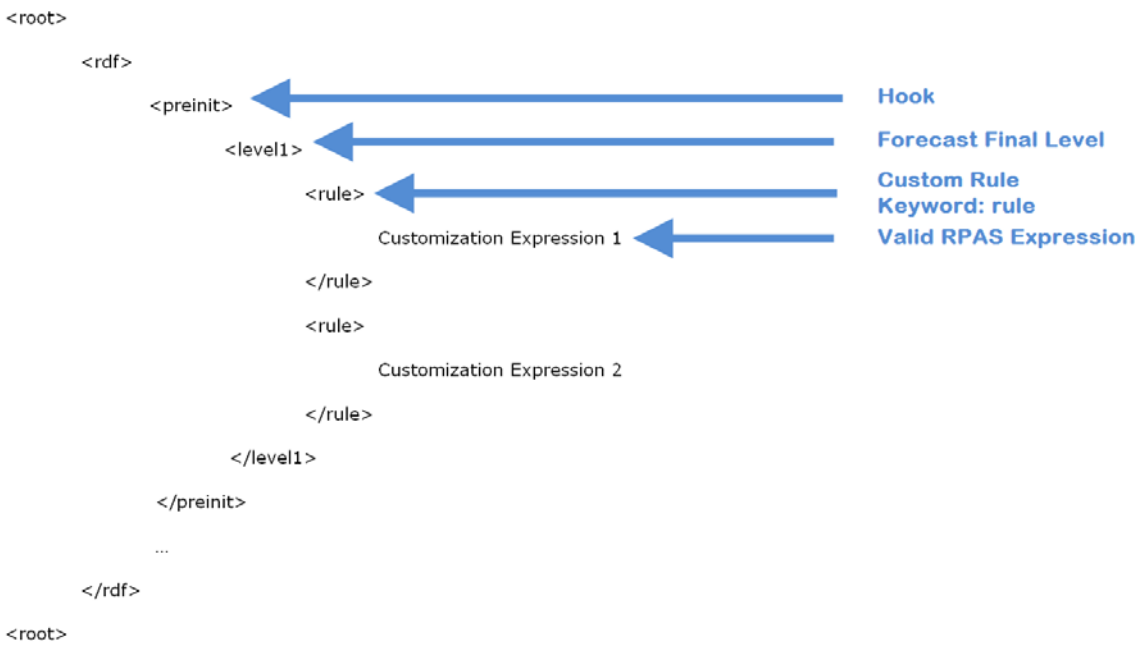
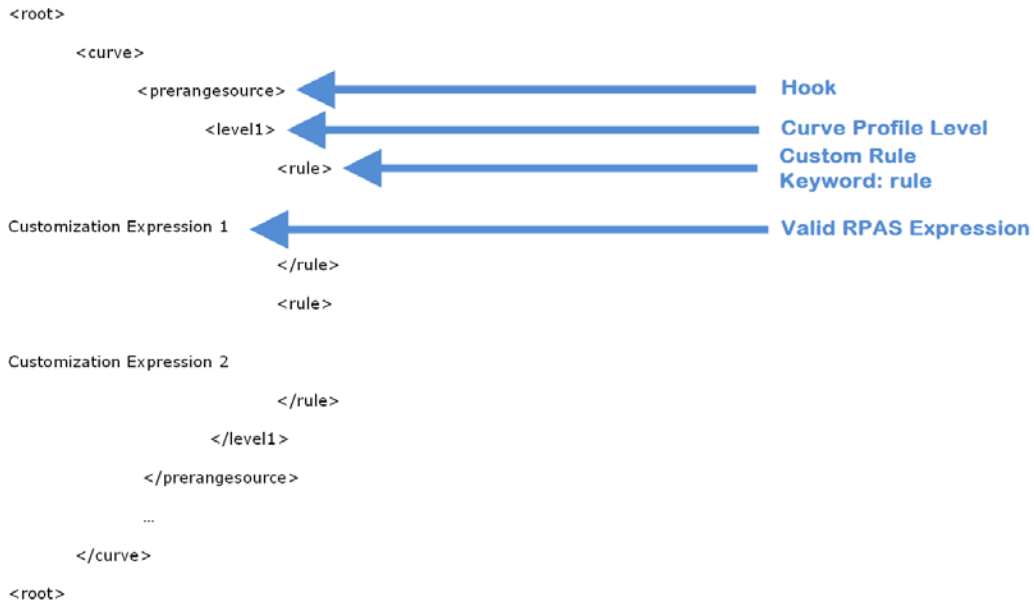


Figure F–2 Format of appcust.xml for Curve



G

Appendix: Configuring Promo Planning

This appendix outlines the special expressions and configuration changes needed for forecast What-if to support the Promotion Planning process.

First, you configure the special expression PromoSelfEffExpr, which computes the self lift of a promoted item, based on pre-computed Promo effects from RDF Causal.

Next, you configure thePromoCrossLift special expression, which uses the self promo lifts, and pre-computed cross elasticity matrix from CPEM, to generate Halo and Cannibalization lifts.

Finally you configure simple rules that apply the self and cross lifts multiplicatively on top of the baseline, to generate a forecast that is influenced by the planned promotions and their cross effects.

This appendix details these topics:

- PromoSelfEffExpr
- PromoCrossLift
- Promotion Modeling Consideration in a Promo Planning Environment

PromoSelfEffExpr

The PromoSelfEffExpr special expression calculates self lift promotional forecast based on baseline, causal model type, and promotional events. The promotion variables can be based on different intersection. The intersection of input promotion variables need to be equal or higher than that of the forecast. The number of promotion variables can be changed based on how the expression is written. The promotion variables need to be converted to real measures before passing into the special expression. Zero (0) means no promotion.

PromoSelfEffExpr Syntax

The syntax for using the PromoSelfEffExpr special expression is shown in the following section and Example G–1. The input and output parameter tables (Table G–1 and Table G–2 explain the specific usage of the parameters names used in the special expression.

Syntax

FORECAST:self_lift_forecast,

SELF_LIFT_UNIT:self_lift_peak

<-PromoSelfEffExpr(

BASELINE:baseline,

MODELTYPE:causal_model,

PROMOMODELTYPE:promo_model,

PROMOENABLE:true,

FCST_START_INDEX:startindex,

FCST_END_INDEX:endindex,

PROMOCAL0:promovar0,PROMOTYPE0:ptypvar0,PROMOEFF0:peffvar0,PROMOEFFOVR0:povrvar0,

PROMOCAL1:promovar1,PROMOTYPE1:ptypvar1,PROMOEFF1:peffvar1,PROMOEFFOVR1:povrvar1,

PROMOCAL2:promovar2,PROMOTYPE2:ptypvar2,PROMOEFF2:peffvar2,PROMOEFFOVR2:povrvar2

…)

Example G–1 Example Syntax for PromoSelfEffExpr

```
FORECAST:psf01xp,SELF_LIFT_
UNIT:prmslfun01xp<-PromoSelfEffExpr(BASELINE:pfb01xp,MODELTYPE:cslmthused01xb,PROMOMODELTYPE:promomodeltype,PROMOENABLE:true,FCST_START_INDEX:startindex,FCST_END_INDEX:endindex,PROMOCAL0:paggXLxmas.level([CLND].[week]).average,PROMOTYPE0:ptyp01xmas,PROMOEFF0:peff01xmas,PROMOEFFOVR0:povr01xmas,PROMOCAL1:paggXLxmss,PROMOTYPE1:ptyp01xmss,PROMOEFF1:peff01xmss,PROMOEFFOVR1:povr01xmss,PROMOCAL2:paggXLnyd.level([CLND].[week]).average,PROMOTYPE2:ptyp01nyd,PROMOEFF2:peff01nyd,PROMOEFFOVR2:povr01nyd,PROMOCAL3:paggXLeas.level([CLND].[week]).average,PROMOTYPE3:ptyp01eas,PROMOEFF3:peff01eas,PROMOEFFOVR3:povr01eas,PROMOCAL4:paggXLbts,PROMOTYPE4:ptyp01bts,PROMOEFF4:peff01bts,PROMOEFFOVR4:povr01bts,PROMOCAL5:paggXLcirc,PROMOTYPE5:ptyp01circ,PROMOEFF5:peff01circ,PROMOEFFOVR5:povr01circ,PROMOCAL6:paggXLisd,PROMOTYPE6:ptyp01isd,PROMOEFF6:peff01isd,PROMOEFFOVR6:povr01isd,PROMOCAL7:paggXLdml.level([CLND].[week]).average,PROMOTYPE7:ptyp01dml,PROMOEFF7:peff01dml,PROMOEFFOVR7:povr01dml,PROMOCAL8:paggXLsgn.level([CLND].[week]).average,PROMOTYPE8:ptyp01sgn,PROMOEFF8:peff01sgn,PROMOEFFOVR8:povr01sgn,PROMOCAL9:paggXLpprc,PROMOTYPE9:ptyp01pprc,PROMOEFF9:peff01pprc,PROMOEFFOVR9:povr01pprc)
```

Table G–1 Input Parameters for the PromoSelfEffExpr Special Expression

Parameter Name	Description
BASELINE	A real measure based on prod/location/calendar. It contains the non-promoted forecast.
MODELTYPE	A real measure based on prod/location. It indicates if the causal model is additive or multiplicative. Additive is 0. Multiplicative is 1.
PROMOMODELTYPE	A real measure based on promo dimension or scalar measure. It indicates if the promotion type is linear or exponential. Linear is 1. Exponential is 2. If the input is based on promo dimension, the promo dimension position name and the promo variable names must be in the same format as that in RDF. In RDF, if the promo position name is *xmas*, the promo variable is named as *pvarxlxmas*.

Table G–1 (Cont.) Input Parameters for the PromoSelfEffExpr Special Expression

Parameter Name	Description
PROMOENABLE	A boolean measure based on promo dimension or scalar measure. It indicates if a promotion should be used in the promotional forecast calculation. If the input is based on promo dimension, the promo dimension position name and the promo variable names must be in the same format as that in RDF. In RDF, if the promo position name is *xmas*, the promo variable is named as *pvarxlxmas*.
FCST_START_INDEX	An integer measure based on prod/location indicating the window start index for the merge operation.
FCST_END_INDEX	An integer measure based on prod/location indicating the window end index for the merge operation.
PROMOCAL[i]	A real measure based on prod/location/calendar indicating promotional events. 0 means no promotion. There can be multiple promotion variable inputs with their label starting from PROMOCAL0 and incrementing.
PROMOTYPE[i]	An integer measure based on prod/location indicating how the promotional events are used in the calculation. There can be one promo type per promotion variable inputs with their label starting from PROMOTYPE0 and incrementing. The integer meanings for promotype are: ■ 0 (Automatic) ■ 2 (Disabled) ■ 3 (Override All)
PROMOEFF[i]	A real measure based on prod/location indicating how much is the promotion effects. There can be one promo effect measure per promotion variable inputs with their label starting from PROMOEFF0 and incrementing.
PROMOOVR[i]	A real measure based on prod/location indicating how much is the user overwritten promotion effects. There can be one promo overwriten effect measure per promotion variable inputs with their label starting from PROMOOVR0 and incrementing.

Table G–2 Output Parameters for the PromoSelfEffExpr Special Expression

Parameter Name	Description
FORECAST	A real measure based on prod/location/calendar. It holds the calculated self-lift promotional forecast.
SELF_LIFT	A real measure based on prod/location/calendar. It holds the calculated self-lift factors. Optional output. Forecast =self_lift*baseline
SELF_LIFT_UNITS	A real measure based on prod/location/calendar. It holds the calculated self-lift promotional peaks. Optional output. Forecast=self_lift_units+baseline

PromoCrossLift

The PromoCrossLift special expression calculates cross promotion Halo or Cannibalization effect ratio.

PromoCrossLift Syntax

The syntax for using the PromoCrossLift special expression is shown in Example G–2. The input and output parameter tables (Table G–3 and Table G–4 explain the specific usage of the parameters names used in the special expression.

Example G–2 Syntax for PromoCrossLift

```
CROSS_LIFT:LiftRatioMeasure <- promocrosslift(EFF_MASK: CrossElasticityMask, EFF_
MATRIX:CrossElasticityMatrix, FB_RATIO:frcstvsbaseMeasure, FCST_START_
INDEX:ForecastStartIndex, FCST_END_INDEX: ForecastEndIndex)
```

Table G–3 Input Parameters for the PromoCrossLift Special Expression

Parameter Name	Description
EFF_MATRIX	promo cross elasticity matrix Data Type: Real Required: Yes Default value: 0
EFF_MASK	promo cross elasticity mask Data Type: Real Required: Yes Default value: 0
FB_RATIO	Forecast divided by baseline Data Type: Real Required: Yes Default value: 0
FCST_START_INDEX	Forecast start date index Data Type: Integer Required: Optional Default value: -1
FCST_END_INDEX	Forecast end date index Data Type: Integer Required: Optional Default value: -1

Table G–4 Output Parameters for the PromoCrossLift Special Expression

Parameter Name	Description
CROSS_LIFT	promo cross ratio matrix Data Type: Real Required: Yes Default value: 1

Promotion Modeling Consideration in a Promo Planning Environment

The expressions documented in this appendix can help build the rules necessary in a promo planning environment. However, there are additional things that can be done to make your activities more interactive.

For instance, if the environment has a percent off promotion, rules and measures can be configured to allow you to enter a desired percent off in a measure. A hidden layer automatically flags the boolean promotion, as well as recalculate the price discount that is modeled as a exponential variable. Based on the updated info, the forecast is calculated and you see the impact.

Example Promotions

For a promotion that is: **Buy X (units) and get Y ($) off**.You can configure two measures, one for X— units to buy, and one for Y— the amount you save. You can enter any desired amount for each of them. The hidden layer translates this information into a boolean flag, as well as calculate the price percent off, according to:

```
price percent off = (1 - Y / X / regular unit price)
```

For a promotion that is: **Buy X (units) and get Y (units) free**. You can have two measures that specify the two quantities. The hidden layer translates the information into a boolean flag, as well as a price percent off, according to:

```
price percent off = X / (X+Y)
```

Such a translation layer allows you greater flexibility with the planning of promotions. However, it may not work for every promotion type.

H

Appendix: CPEM Calculations

Cross Promotional Effects Module (CPEM) is a data mining solution that determines promotional Cannibalization, or Halo relationships, or both between items or groups of items. The special expressions detailed in this appendix calculate cross promotional effects.

This appendix details these topics:

- RdfPromoCrossEffectExpr

RdfPromoCrossEffectExpr

The promo cross effect special expression calculates the cross promotion Halo or Cannibalization effects.

RdfPromoCrossEffectExpr Syntax

The syntax for using the RdfPromoCrossEffectExpr special expression is shown in Example H–1. The input and output parameter tables (Table H–1 and Table H–2 explain the specific usage of the parameters names used in the special expression.

Example H–1 Syntax for RdfPromoCrossEffectExpr

```
CROSS_EFFECTS:RtnCannEff,STD_ERR:CannStdErr <-RdfPromoCrossEffectExpr(DES_
SLS:LogDesnCannSls,NORM_PRICE:LogCannNorPrc,PROMO_VAR:LogCannProVar,CROSS_
LIFT:LogCannLift,CROSS_MASK:CannMask,CROSS_TYPE:EffType,HIST_
START:HistStrIdx,HIST_END:HistEndIdx)
```

Table H–1 Input Parameters for the RdfPromoCrossEffectExpr Special Expression

Parameter Name	Description
DES_SLS	Aggregated deseasonalized sales at the Halo / Cannibalization level Cannibalization) Data Type: Real Required: Yes Default value: 0
NORM_PRICE	Normalized regular price at the Halo / Cannibalization level Data Type: Real Required: Yes Default value: 0

Table H–1 (Cont.) Input Parameters for the RdfPromoCrossEffectExpr Special

Parameter Name	Description
PROMO_VAR	Aggregated promotion variables at the Halo / Cannibalization level Data Type: Real Required: Yes Default value: 0
CROSS_LIFT	Aggregated cross product sales lift at the Halo / Cannibalization level Data Type: Real Required: Yes Default value: 0
CROSS_MASK	The cross promotion calculation mask to specify whether the cross effects will be calculated at Product/RHS Product/Location Data Type: boolean Required: Yes Default value: 0
CROSS_TYPE	Cross Promotion Type : 1(Cannibalization),2(Halo) Data Type: Integer Required: Yes Default value: 0
HIST_START	History Start Date Index Data Type: Integer Required: Yes Default value: 0
HIST_END	History End Date Index Data Type: Integer Required: Yes Default value: 0

Table H–2 Output Parameters for the RdfPromoCrossEffectExpr Special Expression

Parameter Name	Description
CROSS_EFFECTS	Cross effects (Halo or Cannibalization) Data Type: Real Required: Yes Default value: 0
STD_ERR	Standard Error (This Parameter is useless in this release and it is reserved for future enhancement) Data Type: Real Required: Yes Default value: 0

RDF Configuration Options for CPEM

CPEM gives unconstrained estimates of cross promotional elasticities, and verifies that the values are within some preset boundaries.

However, it is only at application time, in RDF, that the actual magnitude of the cross promotional lift can be calculated.

There may be several occasions when additional checks could be introduced. For instance, when an item is on promotion, it usually experiences a spike in sales. However, it may also receive Halo lift from a driver item as well. Appropriately, the cross effect should not be larger than the self lift. Preferably, the cross lift should not be larger than ten percent of the self lift.

RPAS and RDF offer the tools and hooks to implement logic preventing or alerting these types of situations.

Adjusted Forecast Example 1

An example of how the forecast can be adjusted is achieved by implementing the following pseudo code in the RPAS rules, and packaging it in the appcust.xml hook.

The stage where the rules are run can be pre-approved.

Pseudo code:

- if an item is on promotion
- if halo lift > self lift * threshold then
- halo lift = self lift * threshold

These rules prevent the halo lift from being larger than a certain percentage of the self lift for a promoted item.

Adjusted Forecast Example 2

Another example of how the forecast can be adjusted with the following pseudo code in the RPAS rules, and packaging it in the appcust.xml hook.

Pseudo code:

- if an item is not on promotion
- if abs(cannibalization effect) > baseline forecast * threshold then
- cannibalization effect = - baseline forecast * threshold

These rules prevent the cannibalization effects from dragging down the forecast too much.

Similar logic can be implemented to trigger an exception, rather than adjust the forecast, so that you can review it and determine whether or not to act.

In the pre-alert stage of the appcust.xml hook, the following logic alerts you that the cross lift is close to the self lift.

Alert logic:

- if an item is on promotion
- if halo lift > self lift * threshold then
- trigger high_halo_alert

CPEM Configuration Points

There are a few configuration points for CPEM, which define intersections at which the solution mines for Cannibalization and Halo effects, as well as supporting measures. This is handled thru labeled intersections, and the following table gives the label - level assignment for the two main ones.

Table H–3 Level - Labeled Intersection Assignment

Level	Labeled Intersection
Cannibalization	#CannLvl#
Halo	#HaloLvl#

There are additional labeled intersections in CPEM, but they all depend on the two main ones.

Note: Changing the main labeled intersections may require changing the dependent labeled intersections as well, to match the Cannibalization and Halo intersections

If the Halo level is class/region, a dependent labeled intersection may be class/region/week.

If the Halo level is changed to class/area, the dependent intersection must be changed accordingly, that is, to class/area/week.

Configuration Challenges

Finding the correct level for mining Halo and Cannibalization is not an easy task. Mining Halo is difficult because the entire merchandise space needs to be searched.

Cannibalization is difficult because the correct level is hard to determine. For instance, item is too low, but subclass may be too high.

Hierarchical Example

In the following example the item level may be too low, but Small Yogurts may be too high because of no visibility to brand or flavor.

Yogurt (Category) -> Small Yogurts (Sub Category) -> Dannon Light ‘n Fit 8 oz. (Item parent) -> Dannon Light 'n Fit NF 8 oz.:Raspberry (item)

In this example, the right level may be Item parent.