Retek® Predictive Application Server™ 11.0.4

Release Notes

Corporate Headquarters:

Retek Inc.
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403

888.61.RETEK (toll free US)
+1 612 587 5000

Fax: +1 612.587.5100

European Headquarters:

Retek
110 Wigmore Street
London
W1U 3RW
United Kingdom

Switchboard:
+44 (0)20 7563 4600

Sales Enquiries:
+44 (0)20 7563 46 46

Fax: +44 (0)20 7563 46 10

Customer Support

Customer Support hours

Customer Support is available 7x24x365 via e-mail, phone, and Web access.

Depending on the Support option chosen by a particular client (Standard, Plus, or Premium), the times that certain services are delivered may be restricted. Severity 1 (Critical) issues are addressed on a 7x24 basis and receive continuous attention until resolved, for all clients on active maintenance. Retek customers on active maintenance agreements may contact a global Customer Support representative in accordance with contract terms in one of the following ways.

Contact Method	**Contact Information**
E-mail	support@retek.com
Internet (ROCS)	rocs.retek.com Retek's secure client Web site to update and view issues
Phone	1 612 587 5800

Toll free alternatives are also available in various regions of the world:

Australia	1 800 555 923 (AU-Telstra) or 1 800 000 562 (AU-Optus)
France	0800 90 91 66
United Kingdom	0800 917 2863
United States	1 800 61 RETEK or 800 617 3835
Mail	Retek Customer Support Retek on the Mall 950 Nicollet Mall Minneapolis, MN 55403

When contacting Customer Support, please provide:

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

Release Details

Application:	Retek Predictive Application Server (platform)
Version Number:	11.0.4
Code Cut Off Date:	December 19th, 2003
Release Date:	January 16th, 2004
Type of Release:	
Base:	Yes
Patch:	No
RPAS Server Version:	Not applicable
Patch Level:	Original
Supported OS, Server:	Sun Solaris 2.8, AIX 5.1, HP-UX 11i, Windows NT
Supported OS, Client:	Windows NT, XP, 2000
Required 3rd Party Software:	None
Documentation:	RPAS 11.0.4 Installation Guide
	RPAS 11.0.4 Administrators Guide
	RPAS 11.0.4 Users Guide
	RPAS 11.0.4 Rule Functions Reference Guide
	RPAS 11 Calculation Engine
	RPAS 11.0.4 Developers Guide

Overview

Functional Summary

The purpose of this document is to communicate the enhancements and changes that have been implemented in RPAS 11.0.4. The features and changes described herein are available in the base platform and most are supported in the RPAS 11.0.4 Configuration Tools. Certain features might require some configuration or administration outside the RPAS user interface and outside of the Configuration Tools.

Due to some additions to the internal metadata definitions, an upgrade process is required to upgrade 11.0.3 domains to 11.0.4. Note that configuration changes will be required to take advantage of many of these new features (such as position queries in windows, non-materialized measures, new functions, procedures and aggregation types), but that in general the performance enhancements will not require configuration changes.

The features and changes in RPAS 11.0.4 were implemented to support a number of customer requirements and Retek initiatives, including:

- Features to support the port of Retek Demand Forecasting to RPAS 11
- Features and performance enhancements to support Assortment Planning
- Features and performance enhancements to support the development of Retek Advanced Inventory Planning

Performance Summary

Significant performance enhancements were made in RPAS 11.0.4. Improvements made include faster workbook builds and many enhancements to the performance of the RPAS calculation engine.

Technical Summary

The RPAS platform consists of the RPAS server application and client user interface. The RPAS server has been tested and verified on 3 Unix platforms (Sun Solaris, IBM AIX, and HP-UX) and Windows NT. Please see the RPAS Installation Guide for additional technical information.

Summary of Changes made since 11.0.3.1 Release

New Features

- Position queries in workbook windows – position queries provide the ability to define, save, and reuse the definition of a ‘query’ that results in a selection of positions in a hierarchy
  - Implementation of infrastructure to support position queries
  - Support for position queries in workbook windows for automatic ranging in the z-axis; this allows the user to change the position in the z-axis (or the slice) and RPAS automatically executes the “query” such that the set of positions that are relevant for another dimension for that slice are displayed in the client. Thus, for example, the warehouses that are visible in the down orientation will vary according to the current product in the slice area.
  - Positions that are not selected through the query will be treated as hidden in the UI. The behavior of ‘hidden’ positions in the UI has not changed.
  - The definition of the positions that should, or should not, be visible in a given context (i.e. the slice position in another hierarchy) will be held in a normal binary measure. RPAS intends to support more sophisticated queries over time.
  - Position queries are not available in the measures dimension.
- Sanity check – a mechanism that allows dynamic lower and upper limits to be placed on measures such that values are validated on edit, prior to being passed into the calculation engine. The limits are held in measures that may have base intersections that are higher up the hierarchies with the values being ‘replicated down’ if required. Thus, for example, an upper limit at department may be used to perform a sanity check on, say, an IMU at class/week.
- Non-materialized measures, display-only – non-materialized measures are measures that are temporary or transient in nature, and are not persisted to the domain or workbook. This first implementation is for recalc measures that are only displayed in the client, although RPAS intends to support more sophisticated queries over time. These recalc measures (typically variances, etc.) are only calculated on demand (i.e. when required to display on the UI), not during a normal calculation cycle, which for workbooks with many such measures provides a significant performance boost.
- Mechanism for kicking users off the system – the “domainDaemon” utility has been upgraded so that users are kicked off the system when domains are deactivated (existing parameter); this parameter can be set and executed prior to beginning batch processes to remove users that are logged into domains

- Changing descriptions of positions on the UI – a feature to allow the descriptions of positions to be changed through the normal calculation of rule expressions. This affects the descriptions in just the local workbook, not the domain. This is especially valuable in applications, such as 'grading', that require position descriptions to vary according to the context, such as the active class/assortment period.
- Aggregate procedure –This procedure allows a recalc measure to be calculated at aggregated levels by aggregation. The aggregation type can vary by hierarchy. The procedure therefore supports the calculation of measures where the required aggregations may be, for example, total up calendar and product, but average up location.
- flookup function – A function (not a procedure) that may therefore be used in complex expressions with other functions. A limited version of the lookup procedure, that looks up against specified slices (positions). The specification of the positions must be explicit in the function parameters (ie, it cannot be indirect through a measure).
- Several enhancements for functions with multiple results. See a later section in this document for additional details.
- New aggregation method for determining the median value (and median of populated values)
- Tokenized measures – string measures used as variables in an RPAS expression whose values refer to other measures. This functionality is provided to support the generation of configurations of applications such as RDF, and is not supported as general configurable functionality through the tools for planning applications.
- Ability to set an environment variable (TODAY_RETEK) to a specified date that will override the system date when the 'today' keyword is used in expressions. This enables batch jobs to be run with an effective date different to the actual date, for example when updating actuals an hour or two before midnight.
- Ability to delete rules and rule groups using RPAS-Tools interface and calc engine utility (mace) – delete rules that do not belong to any rule group, delete specific rule groups and its rules, and delete all rules and rule groups
- Function that returns the na value of an expression (function navalue)

Updated RPAS Utilities

The following utilities were changed in RPAS 11.0.4. Some of these utilities might be used in external, implementation-specific batch scripts. These scripts might require an update depending on the utility used and if the parameters have changed.

- Updated utility for loading measures (loadmeasure) – utility rewritten in C++ (all MSPL removed), requiring command line arguments to change
- Updated utility for registering measures (regmeasure)– utility updated to allow the update/modification of the position information for a measure along each dimension in the measure hierarchy
- New parameters for calculation engine utility (mace)
  - Validate rule groups – the new parameter (–validate) takes one of 3 possible values (general, calc, refresh) to specify which type of rule group to validate
  - Purge rules and rule groups – it is now possible to delete rules not referenced in any rule group, delete rule groups and its rules (that are not used in other rule groups), and delete all rules and rule groups
- Updated utility for loading hierarchies (loadHier) – functionally equivalent to hierarchy load utility in 11.0.3 but developed in C++ (all MSPL removed); changes to command line options
- Updated utilities for performing certain administrative tasks on hierarchies (refreshHier & checkParents) – rewritten in C++ (all MSPL removed); these utilities are not normally called by an administrator

New RPAS Utilities

The following is a list of new utilities in 11.0.4.

- Adding and removing users in a domain; this utility can be run from a command line or in batch (utility usermgr)
- Listing and removing workbooks; this utility can be run from a command line or in batch (utility wbmgr)
- Creating databases and indicating if they are in normal/spare or hypersparse mode (utility createDb)
- Determine the version of RPAS for a specified domain (utility domainversion)
- Determine the version of RPAS running in a specified location (utility rpasversion)

Other Changes and Additions

- Changes and additions to data in RPAS that can be translated
  - Domains can be designated as single or multi-language domains; this is set during the domain-build process (use –m flag when running createRpasDomain or a flag set in the Configuration Tools)
  - The position name and label of any dimension can now be translated
  - Measure descriptions can be translated – previously only measure names and labels could be translated
  - The format of measure names for loading translated data has been changed; new format is R_{dimname}{property} (for example, R_MEASLABEL)
- Domain daemon has been enhanced to allow domains to be accessed from multiple servers
- Ability to print on legal paper

Definition of New Features

The following section describes some of the larger new RPAS features in more detail.

Non-materialized measures – Display-only

Definition

Non materialized measures are measures that are temporary or transient in nature, and are not persisted to the domain or workbook. In the initial implementations, they are always assumed to be of recalc aggregation type. They are designed to be performance-enhancing for two major reasons:

1 Non materialized measures are calculated only when required.

2 Non materialized measures are not persisted to the domain or workbook

Expressions for normal (materialized) measures are always calculated when the rule is affected and the expression is selected. This is not the case with non-materialized measures. A non-materialized measure that is not used on the rhs of any expressions, does not get calculated during the calculation cycle, even when measures on the rhs of the expression to calculate the non-materialized measure are changed. Such measures only get calculated when they are required. Processes that may require such measures include viewing data through windows in workbooks, extracting data, and so on.

The materialized status of a measure is a measure attribute, which is maintainable through the tools. Validation ensures that all measures flagged as non-materialized have a recalc aggregation type. Any rule group that does not have a rule with a single expression to evaluate every non-materialized measure is invalid. Any rule group that includes any expression with a non-materialized measure on the right hand side is invalid.

aggregate – Procedure that returns the value of a measure aggregated from the base intersection to the current level, using the supplied aggregation type

Syntax

aggregate (<cachemeasure>, <hierspec1> [, <hierspec2> … , <hierspecn>])

Where <hierspec1-n> is [<hierarchy>].<aggtype>

The aggregate procedure provides a mechanism that is the functional equivalent of an aggregation type that varies by hierarchy. Such variable aggregation types cannot be implemented as aggregation types because of the difficulties of spreading changes. The aggregate procedure avoids that difficulty, because the measure being calculated will be of recalc type, and thus is spread indirectly through a mapping rule.

The rule writer specifies a <cachemeasure> which holds the base intersection values to be aggregated, and is also the source of values for ‘recalc’ aggregation.

The rule writer also specifies the aggregation type for each hierarchy and the priority sequence to be used. The priority sequence is required because at levels that are aggregated in more than one hierarchy (say, Department/Region/Month for a measure with a base intersection of Class/Store/Week), different results would usually be obtained by aggregating up each of the hierarchies. Thus, for example, if the requirement is to aggregate up the product hierarchy by using the total aggregation type, up the location hierarchy using the average aggregation type and the calendar hierarchy by using the first aggregation type, there are three potential ways to calculate a value at Department/Region/Month. We could total from Class/Region/Month, average from Department/Store/Month or first from Department/Region/Week. These would almost certainly generate three completely different values. By providing a priority sequence, the rule writer explicitly determines which of these values are required (See worked example, below).

Note that the effect of a series of aggregations of the same type up a single hierarchy may return different results from those of a measure with the same aggregation type. ‘Normal’ aggregation for a measure driven by its aggregation type is performed from all base intersection cells descended from the cell being evaluated. Thus, for example, for a measure with a base intersection of Class/Week and an ‘average’ aggregation type, the value calculated for a cell at Department/Month is the average of all values for all Class/Week cells for the Department/Month. If the measure is a recalc measure, calculated at aggregated levels from a rule with an aggregate function such as aggregate(x, [prod].average, [clnd].average), the value for the Department/Month will be the average of all the Class/Months (not Class/Weeks) that belong to the Department/Month. Other than coincidentally, this would generate a different value.

<cachemeasure> is the measure to be aggregated, and the value of <cachemeasure> is also the value used for cells that are at the base intersection (i.e. bottom levels), and at aggregated levels when the required aggregation type is recalc. <hierarchy> is the name of a valid hierarchy. Each hierarchy may only be specified once in the procedure, but hierarchies may appear in any order. The sequence that the hierarchies are specified in is used to determine which hierarchy to aggregate up if the cell being evaluated is at an aggregated level in more than one hierarchy. In this circumstance, aggregation is performed up the first specified hierarchy that the cell is at an aggregated level in, and the other hierarchies are ignored. <aggtype> specifies the aggregation type to be used. The <aggtype> must be one of the standard aggregation types (see Appendix in back). If any hierarchy that is in the scope of the measure being calculated is not explicitly specified, the aggregation type of that hierarchy is assumed to be total. Such hierarchies are assumed to be sequenced after all hierarchies that are explicitly referenced, and ordered from innermost to outermost.

Note that the value of the <cachemeasure> is used at the base intersection of the measure being calculated. If, for a given cell, the aggregation type to be used is recalc, the value is also obtained directly from the <cachemeasure> at that level, which will normally have an aggregation type of recalc.

Note that aggregate is a procedure and thus cannot be combined with functions, modifiers, or other procedures in any manner. As a procedure it requires a different syntax: "<-" instead of "=" when being assigned.

Inverse: The aggregate procedure does not have an inverse.

Examples

- **aggregate**(x, [clnd].recalc) For cells at the base intersection, the value is calculated from the measure x. For cells at an aggregated level in the calendar hierarchy, the value is also obtained from the measure x (which we can assume has an aggregation type of recalc, and thus the result of the procedure is as if the aggregation type were recalc, using the usual expression to calculate measure x). If the cell is not at an aggregated level in the time hierarchy, and assuming in this example that the other hierarchies are product and location, in that priority, the value for a cell at an aggregated product level is calculated as the total of all cells for products descended from that product, for the same location and time. Otherwise the value for the cell is calculated as the total of all cells for locations descended from the cell's location, for the same product and time.
- **aggregate**(x, [loc].average) In a similar manner to the previous example, cells at aggregated levels in the location hierarchy will be calculated by averaging the values of cells for all descendent locations, otherwise the value will be totaled up the product or time hierarchy as appropriate.
- **aggregate**(x, [clnd].average) Totals up all hierarchies except time, which uses an average aggregation type.
- **aggregate**(x, [prod].average, [clnd].first) Averages up the product hierarchy if possible, otherwise takes the first child value up the calendar hierarchy, otherwise totals up the other hierarchies.

- **aggregate**(x, [prod].average, [clnd].last) Averages up the product hierarchy if possible, otherwise takes the last child value up the calendar hierarchy, otherwise totals up the other hierarchies.

Multi-level calculation example

Consider a measure calculated from the expression **aggregate**(x, [prod].total, [loc].avg, [clnd].first). The measure is assumed to have a base intersection of Class/Store/Week. Examples of the calculations that would be applied at various levels are:

Class/Store/Month: first from Class/Store/Week

Class/Region/Month: avg from Class/Store/Month

Department/Region/Month: total from Class/Region/Month

flookup – 'Fixed lookup' function which returns the value of a measure for an explicitly named fixed intersection.

Syntax

flookup*(*<measure>, <posspec1> *[*, <posspec2> … , <posspecn>*])*

Where <*posspec1-n*> is: [<*hierarchy*>].[<*dimension*>].[<*positionname*>]

<*measure*> is the measure to be looked up. This <*measure*> must conform with the measure being calculated as follows. Some hierarchies may be present in the base intersection of both measures, and these are handled by normal 'non-conforming' logic. For any hierarchies that are only in the base intersection of the measure being calculated, all positions will lookup the same value. For any hierarchies that are only in the base intersection of the <*measure*>, the position to be used must be explicitly named through a <*posspec*>. Note that if the position to be used can only be specified indirectly (for example, if it is held in a measure), the *flookup* function cannot be used, and the more powerful *lookup* procedure should be used instead.

flookup can be used to return a constant or a slice. In case of a constant, the NA value of the flookup function will be the value of the constant. In case of a slice, the NA value of the flookup function will be the NA value of <measure>.

For each <*posspec*> that is specified, the <*hierarchy*> must be the name of a valid hierarchy, the <*dimension*> must be the name of a valid dimension in that hierarchy, and the <*positionname*> must be the name of a valid position in that dimension. If <*hierarchy*> is not a valid hierarchy or <*dimension*> is not a valid dimension in that hierarchy, or <*positionname*> is not a valid position in that dimension an *error* is generated.

Additionally, <*dimension*> must be a dimension in the base intersection of <*measure*>. If a level modifier is used, the aggregated <*measure*> will be used in the *flookup* function, and any <*posspec*> that refers to that hierarchy must conform to the new level.

There is no special ‘cycle breaking’ logic for the *flookup* function. This means that a measure may never be calculated from the *flookup* of the same measure. The *flookup* function returns the value of the expression from the specified fixed intersection.

Inverse:The *flookup* function does not have an inverse.

Examples

- **flookup**(perc, [flvl].[flvl].[flvla]) Returns the value for the measure perc for the position ‘flvla’ in the flvl dimension of the flvl hierarchy.
- **flookup**(leadtime, [prod].[cls].[class1], [loc].[whse].[whseA]) Returns the value for the measure *leadtime* for the class ‘class1’ for the warehouse ‘whseA’.

Enhancements for functions with multiple results

Several enhancements to the supporting infrastructure for writing functions with multiple results were added to RPAS 11.0.4. Below is a summary of the specifications and assumptions about this type of function, full details are in the developer’s guide:

- Functions can contain multiple outputs/results
- Each output can be a different type (string, integer, real, etc.).
- Parameters and results can be named
- Results can be optional either by omitting the rightmost results or by using names to denote which results are desired. The function is aware of which results are desired and can use this information to skip unnecessary computations.
- Arguments can be optional but not named.
- Multiple output functions integrate with the protection processing algorithms
- Arguments passed to multi-result functions can be computed using normal calculation features. But the result of the function cannot have any further calculations applied to it - it must be assigned directly to the LHS.

Position queries

Overview & global design

Position queries provide the ability to define, save, and reuse the definition of a 'query' that results in a selection of positions in a hierarchy, or in the measures dimension. The intention is that, over time, position queries will become all-pervasive in RPAS applications, and may be used at any point where the application requires the selection of positions. 11.0.4 includes the first support for position queries in worksheets only.

Position queries in spreadsheet windows

Position queries in 11.0.4 supports the specific case where a context is supplied in one hierarchy that dictates the positions to be shown in another hierarchy. For example, the selected position in a warehouse dimension may be used to range down the store dimension of the location hierarchy, to only show those stores 'served by' the selected warehouse. The information that defines which positions should be visible for a specific context is held in a Boolean measure (dimensioned on the context and the dimension being queried), which is calculated normally.

The positions to be shown for a dimension on a view are defined as being the result of a position query, using a specific measure, through the configuration tools. Currently, therefore, there is no mechanism to dynamically configure a worksheet window to use a position query. For such worksheets, the current position of the dimension in the slice area is the context. When the user changes this current position, the worksheet is refreshed to show just the positions required for the new context.

Sanity Check

Sanity check provides a mechanism that allows dynamic lower and upper limits to be placed on measures such that values are validated on edit, prior to being passed into the calculation engine. The limits will be held in measures that may have base intersections that are higher up the hierarchies with the values being 'replicated down' if required. Thus, for example, an upper limit at department may be used to perform a sanity check on, say, an IMU at class/week.

Specifically the following functionality is available:

- Upper and lower bound measures can be defined for any registered measure
- This feature will only function with numeric measures (non-numeric measures will be ignored); however, real and integer types can be mixed
- Bounding measures must be materialized (must be stored in the domain)
- The same hierarchies must be present in the measure being evaluated and the bounding measures
- If a bound measure is defined at an intersection that is not an ancestor of the current intersection, and cannot be aggregated to the current intersection, no sanity checking will apply.

navalue – Function that returns the na value of an expression

Syntax: navalue(<expression>)

<expression> can be a constant, a measure, or an expression.

The navalue function does not directly generate errors, but it can propagate errors generated by <expression>.

Inverse:The navalue function does not have an inverse.

Examples:

- **navalue**(<meas>) This returns the NA value of <meas>.
- **navalue**(<meas1> + <meas2>) This returns the NA value of the expression "<meas1> + <meas2>". In this example, if the NA value of <meas1> is 2 and the NA value of <meas2> is 5, then the result of the navalue function will be 7.

RPAS Utilities

The following section contains the syntax of the new RPAS utilities and describes changes made to existing utilities. Only utilities that are intended to be directly run by an administrator or implementer are included in this section.

Loading Measures – loadmeasure utility

Use the loadmeasure utility to load a measure or apply staged loads for a measure. You must specify the measure name and the path to the domain containing the measure. Usage of the utility is:

loadmeasure –d pathToDomain –measure measureName {–logdirectory directoryName} {–applyloads} {–loglevel level}

Argument	Description
–d pathToDomain	Specifies the domain in which to load the measure.
–measure measureName	Specifies the name of the measure to load. The name must be lowercase.
–logdirectory directoryName	Specifies the location of the output error log. The default location is pathToDomain/scripts/err.
–applyloads	Use this argument to apply any staged loads for the named measure.
–loglevel level	Use this argument to set the logger verbosity level. Possible values: all, profile, debug, information, warning, error, or none.

Calculation Engine – mace utility

Purging rules and rule groups

Mace utility now supports the following:

mace –d $TEST_DOMAIN purgeRules

This deletes all rules that are not contained in any rule group.

mace –d $TEST_DOMAIN removeGroup -groupName

This removes the specified rule group. It also removes all rules within the rule group that are not shared by other rule groups.

mace –d $TEST_DOMAIN removeAllRuleData

This removes all rule groups and all rules.

Validating rule groups

New parameter (–validate) that takes one of 3 possible values (general, calc, refresh) to specify which type of rule group to validate.

All of these also require the -ruleGroup <ruleGroupName> parameter be given as well. In addition, if you select the refresh option, you must also specify the -calcRuleGroup <calcRuleGroup> parameter. Each refresh rule group has a corresponding calc rule group.

The usage is:

mace -d $TEST_DOMAIN -validate general -ruleGroup MyRuleGroup

mace -d $TEST_DOMAIN -validate calc -ruleGroup MyCalcRuleGroup

mace -d $TEST_DOMAIN -validate -refresh -ruleGroup MyRefreshRuleGroup -calcRuleGroup MyCalcRuleGroup.

Administering Users – usermgr utility

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

Usage:

usermgr –d domainPath –add userName –label label –password psw –group grp{–admin} {–loglevel level}

usermgr –d domainPath –remove userName {–loglevel level}

usermgr –d domainPath –list {–loglevel level}

usermgr –d domainPath –print –user username {–loglevel level}

usermgr –d domainPath –print –group groupname {–loglevel level}

usermgr –version

Argument	Description
–d domainPath	Specifies the path to a domain that you want to add, remove or get information about a user.
–add userName	Use this argument to add a user with a specified name. Use the other arguments specified in the usage to add those attributes for that user.
–label label	Use this argument to specify the label of the user that you are adding to the domain.
–password psw	Use this argument to specify the password of the user that you are adding to the domain.
–group grp	Use this argument to specify the group of the user that you are adding to the domain.
–admin	Use this argument to specify that the user you are adding to the domain has administrative rights.
–remove userName	Use this argument to remove the user with the specified name from the domain.
–list	Use this argument to list all the users registered to the specified domain.
–print	Use this argument to print the specified user or group information.
–user username	Use this argument to specify the user name in the specified domain that you want to print. This argument is only applicable to –print option.
–group groupname	Use this argument to specify the group in the specified domain name that you want to print. This argument is only applicable to –print option.
–version	Use this argument to get the version information. It does not require –d domainPath.
–loglevel level	Use this argument to set the logger verbosity level. Possible values: all, profile, debug, information, warning, error, or none.

Inspecting and Managing Workbooks – wbmgr utility

Use the wbmgr utility to inspect or remove the existing workbooks. Do not assume that manual removal of the workbook directories will remove the workbook metadata in the domain.

Usage:

wbmgr –version

wbmgr –d pathToDomain –list –all {–loglevel level}

wbmgr –d pathToDomain –list –user userName {–loglevel level}

wbmgr –d pathToDomain –print –wbList wb1,wb2,... {–loglevel level}

wbmgr –d pathToDomain –remove –all {–loglevel level}

wbmgr –d pathToDomain –remove –user userName {–loglevel level}

wbmgr –d pathToDomain –remove –user userName –wbList wb1,wb2,... {–loglevel level}

Argument	Description
–d pathToDomain	Specifies the domain containing the workbooks.
–list –all	Lists all workbooks in the domain.
–list –user userName	Lists all workbooks belonging to the user.
–print –wbList wb1,wb2,...	Prints detailed information about workbooks in the list.
–remove –all	Removes all workbooks from the domain.
–remove –user userName	Removes all workbooks from the domain belonging to the user.
–reomove –user userName –wbList wb1,wb2	Removes all the workbooks in the specified list.
–loglevel level	Use this argument to set the logger verbosity level. Possible values: all, profile, debug, information, warning, error, or none.
–version	Use this argument to get the version information. It does not require –d domainPath.

Loading Hierarchy Structure – loadhier utility

The loadHier utility can be used to load and refresh a hierarchy. Usage of the utility is:

loadHier –version

loadHier –d domainPath –load hiername {–purgeAge purgeage} {–checkParents} {–noClean}{–loglevel level}

Argument	Description
–version	Use this argument to get the version information. It does not require –d domainPath.
–d domainPath	Specifies the domain in which to load the hierarchy.
–load hierName	Specifies the name of the hierarchy to load and refresh.
–purgeAge purgeage	Specifies the purgeage during loadHier. If not specified, loadHier gets purgeage from domain.
–checkParents	Use this argument to check the parents while loading.
–noClean	If specified input files and the meta data used during load process are not cleaned. It is used only for debugging purposes.
–loglevel level	Use this argument to set the logger verbosity level. Possible values: all, profile, debug, information, warning, error, or none.

Domain Version Information – domainversion utility

Use the domainversion utility to get version information for a specified domain.

Usage:

domainversion –d pathToDomain

domainversion –code

Argument	Description
–d pathToDomain	Specifies the path to the domain for which you want to get version information.
–code	Use this argument to get version information that RPAS Libraries expect.

RPAS Version Information – rpasversion utility

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

Usage:

rpasversion –l pathToLibrary

Known issues

Issue	Resolution / Workaround
The following issue relates to the "hiermods" feature and is not relevant if that feature is not in use. It is not currently possible to define only a single 2-dimensional rollup per hierarchy using *hiermods* (functionality used to limit the number of dimensions and the rollups displayed in a workbook). Workbooks with only 2 dimensions defined for a hierarchy will not build.	Define more than one rollup per hierarchy (e.g. CLSS-DEPT will not work by itself, must have another relationship such as SKU-CLSS in addition to CLSS-DEPT). This defect will be fixed in a patch after 11.0.4 has been released.
Database server process continues to run for a domain after adding workbooks to the auto-workbook build queue and exiting the RPAS client.	This defect will be fixed in a patch after 11.0.4 has been released.
RPAS client does not automatically refresh workbook data after transitioning to a rule group via a custom menu.	This defect will be fixed in a patch after 11.0.4 has been released.
Workbooks with time periods that are all elapsed are not properly protected. If a workbook contains a time period that is not elapsed the protection is handled correctly.	This defect will be fixed in a patch after 11.0.4 has been released.
Measure security is defined at the base and aggregated levels in the Configuration Tools; currently all users that are not administrators are given read-only access to the measures in workbook templates. Permission must be explicitly set for each measure for a user to be able to edit measures.	This defect will be fixed in a patch after 11.0.4 has been released.

Upgrading

The following section describes the process for upgrading all the components of an RPAS solution. This upgrade process is only possible with solutions built on the GA version of RPAS and Tools 11.0.3 with patch 1, and for solutions built on pre-releases of RPAS 11.0.4. The process involves upgrading the RPAS platform, the Configuration Tools, configurations, and domains.

Important Notes about Upgrading

Please note the following about the upgrade process.

- The upgrade process is not reversible for configurations or domains
- All existing 11.0.3 workbooks should be committed, if desired, and deleted prior to running the upgrade utility. The upgrade utility does not upgrade workbooks within the domain, so trying to use an 11.03 workbook after the upgrade will result in an error
- All the wizard page user selections are deleted – specifically, users/*/selections directories in the domain are deleted, meaning that all stored wizard page selections will no longer exist and users will have to make selections again
- The domain-upgrade component of the upgrade process performs several upgrade functions. Before performing any of these upgrades, it first validates that the domain has an 11.0.3 format for each component it's trying to upgrade. If any of the components is in an unexpected format, the utility reports an error and leaves the domain unchanged. Otherwise it will proceed with the upgrade.

Upgrade Process

1 Make a backup copy of your configuration, your domain, and the installs directory for the domain.

2 Install the new Tools release on your computer. Starting with this release of the Configuration Tools, we have made some changes with the installation process of the Tools. These changes are particularly important if you are ***not*** using the NT server installation of RPAS to build domains. If you are not using the NT server installation of RPAS on your laptop or desktop, then please refer to the RPAS Installation Guide, *Chapter 6: Installing the configuration tools in Windows*, regarding the changes for this release of the Tools. Even if you have already installed a previous release of the Tools, you ***must*** make these changes for this release.

Specifically, we have separated the RPAS NT-specific files from the Tools release into their own directory called **RPASlibs**. The RPASlibs directory is packaged as part of the Tools release. You need to setup the **RPAS_HOME** environment variable to point to the path where you install the RPASlibs directory. Even if you are not using the RPAS NT server installation to build domains, you need to setup the RPAS_HOME variable to point to the RPASlibs directory to use the Tools Workbench. Once you have setup the RPAS_HOME environment variable, you need to add *%RPAS_HOME%/lib* directory to your **PATH** environment variable.

3 If you want to build domains on NT or Windows 2000, then you need to install the new RPAS NT release on your computer. This installation process has not changed. Please refer to the RPAS Installation Guide for complete details. Change or create your **RIDE_HOME**, **RPAS_HOME**, and **PATH** environment variables as necessary to point to the new Tools and RPAS installations respectively. If you are not building domains using the RPAS NT installation, you can skip this step.

Follow the process below depending on your respective scenario, either Scenario 1 or Scenario 2. Then proceed to the remaining instructions for Both Scenarios.

Scenario 1: Upgrade from RPAS/Tools 11.0.3 to 11.0.4

This scenario is only for configurations that have not been previously upgraded to a pre-release of RPAS 11.0.4. Otherwise, go to Scenario 2.

Step 4: Upgrade the configuration:

- Open your configuration using the new Tools installation.
- You will be prompted to convert the configuration. Select Yes.
- As part of the conversion process, rule set, rule groups, and rule names may be truncated in order to meet new validation requirements. Specifically, rule sets will be truncated to the first 9 characters. Rule groups will be truncated to the first 10 characters. Rule names will be renamed to be the new rule group name plus a rule index autogenerated numeric identifiers. The conversion process will also convert rule group names used in workbook custom menus to reflect the new rule group names. The converted rule set, rule group, rule names, and custom menus should be manually inspected. Note that there should be no impact to rules and expressions as a result of the conversion.
- To assist you in updating any external scripts that reference rule group names, a log file will get created called RuleRenamer.log located in your configuration directory.
- We strongly recommend that you do not make any changes to your configuration during the upgrade process. After upgrading your configuration, save and exit your configuration.

Scenario 2: Pre-existing RPAS/Tools 11.0.4 Configuration and Domain

This scenario is only for configurations that have been previously upgraded to a pre-release of RPAS 11.0.4.

Step 4: Upgrade the configuration:

- Double-click on the RuleRenamer.bat file located in the bin subdirectory of your Tools installation. This will open up a new utility that will convert your configuration. Please note that you should *not* have your configuration open in the Tools Workbench during this scenario.
- Use the Browse button to open up your configuration .xml file. Select the main .xml file of your configuration the same as you would as if you were opening the configuration in the Tools Workbench.

As part of the conversion process, rule set, rule groups, and rule names may be truncated in order to meet new validation requirements. Specifically, rule sets will be truncated to the first 9 characters. Rule groups will be truncated to the first 10 characters. Rule names will be renamed to be the new rule group name plus a rule index autogenerated numeric identifiers. The conversion process will also convert rule group names used in workbook custom menus to reflect the new rule group names. The converted rule set, rule group, rule names, and custom menus should be manually inspected. Note that there should be no impact to rules and expressions as a result of the conversion.

- The renamed rules will be displayed in the RuleRenamer. We recommend that you copy and paste this into a text file in order to save this information for updating any external scripts.
- Exit the RuleRenamer.
- Go to step 6 below. We recommend that you do not make any changes to your configuration during the upgrade process. You do *not* need to open and save your configuration in the Tools Workbench for the rule conversion to take effect. The RuleRenamer modifies your configuration directly.

Both Scenarios (except step 10):

4 Change any external batch scripts to use the new rule group names and to reflect parameter and/or name changes to certain RPAS utilities (including **loadmeasure**, **loadHier**, etc.)

5 On the server, move or rename your old configuration. Zip up your new configuration and transfer it to the server. Do ***not*** unzip the new configuration over the old configuration.

6 If using a UNIX server, change your **RIDE_HOME** path in your .profile to point to the new Tools installation.

7 If using a UNIX server, change your **RPAS_HOME** path in your .profile to point to the new RPAS release. Restart your session for these changes to take effect.

8 For **Scenario 1** only (upgrading from 11.0.3 to 11.0.4), run the upgrade utility on your domain using the command below:

```
domainUpgrade1104 –d domain_path
```

Note: This script will call the RPAS upgrade1104 utility.
This script will remove all rule groups except the RPAS ones.

9 Rebuild all custom libraries using the new RPAS release. Move the custom libraries Into the $RPAS_HOME/**applib** directory.

10 Since RPAS no longer references libraries from the **lib** directory of the domain, remove this directory from the domain.

11 Run a **patchinstall** on your domain using the new configuration. The installs directory must be located in your domain home path at the same level as your domain for the patchinstall to work.

This will complete the upgrade process for the RPAS platform, Configuration Tools, and existing domains.