Retek® Predictive Application Server™ 11.1

Release Notes

Corporate Headquarters:

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

888.61.RETEK (toll free US)
Switchboard:
+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 email, 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
Hong Kong	800 96 4262
Korea	00 308 13 1342
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.

Contents

RPAS 11.1 Overview

Release Information

Application:	Retek Predictive Application Server (platform, client and server)
Version Number:	11.1
Code Cut Off Date:	August 17, 2004
Release Date:	August 31, 2004
Type of Release:	Full release
Base:	11.1
Patch:	N/A
RPAS Client Build Number:	1.1789
Supported OS, Server:	Sun Solaris 8, AIX 5.1/5.2, HP-UX 11i (11.11), Windows NT
Supported OS, Client:	Windows NT, XP, 2000
Required 3rd Party Software:	None
Documentation:	RPAS 11.1 Installation Guide
	RPAS 11.1 Administrators Guide
	RPAS 11.1 Users Guide
	RPAS 11.1 Rule Functions Reference Guide
	RPAS 11.1 Calculation Engine Users Guide
	RPAS 11.1 Extension Development Standards

Summary of New Functionality

The following is a summary of the new functionality that is available in version 11.1 of the Retek Predictive Application Server (RPAS). Additional information is provided in other sections of this document.

- Global Domain effectively provides the ability for users to create workbooks from multiple domains. The initial implementation includes the infrastructure for building and managing a Global Domain environment (a “master” domain and a collection of child/local/sub domains) and will provide users the ability to build, save, and commit workbooks in a master domain with data from the local domains.
    - Global Domain measures – ability to handle measures in the Global Domain that do not contain the dimension on which the Global Domain is partitioned, or where a measure’s base intersection is higher than the partition level
    - Centralized administration with Global Domains – the ability to administer and manage RPAS in the master domain of a Global Domain environment and have all changes propagated to the local domains (a few configuration/ administration activities will not be performed centrally in 11.1).
- Manipulatable percent to parent measures (also referred to as participation measures) – ability to have editable measures that are a percentage/ proportion of another measure up one or more hierarchies; this supports multi-level edits to percent to parent measures; additionally it is possible to add these measures to windows without having them pre-defined.
- Hierarchical measures – ability to define a hierarchical relationship between a number of measures. The typical usage is for total sales made up from regular, promo and clearance sales. This provides a mechanism to treat the relationship between those measures in a hierarchical manner. It is delivered through two independent enhancements:
    - Mechanism that references the pre-calculated version of a measure (“old” modifier).
    - Function with multiple results that can spread a total to several child values while still retaining the shape of those values (“propspread” function).
- Function for performing several computations in a single expression (“passthrough” function).
- Commit as soon as possible – RPAS has supported commit now (which can time out if resources are not available) and commit later (which has batch window implications) in previous releases of RPAS. RPAS now provides a mechanism to queue commits to be done when the system is able to do so, but without the batch window and process implications of using commit later.
- Configurable parallel workbook build and refresh – provides faster throughput when other machines have spare capacity.
- Partial/selective refresh – Efficiency enhancement for partial refresh through supporting multiple refresh rule groups, and allowing the user to select which refresh rule group to use. These refresh rule groups are defined in the RPAS Configuration Tools.

- New/updated aggregation and spreading
  - Measures that have an aggregation type of “average” can use the spread method “proportional”
  - New aggregation type for calculating the total of populated cells (total_pop), and subsequently spreading those values proportionally (prop_pop)
- Ability to limit the number of saved workbooks that a user can have – limits are defined by workbook template and are set for a given user, user group, or all users
- Saved selections – RPAS has expanded the implementation of position queries to allow a user to save and reuse a selection of positions; the selections can be made at the lowest level/dimension of a hierarchy or higher, and the “position query definition” is executed at the level at which it was saved. This is functionally equivalent to the 9.x feature named “saved selections”.
- Ability to update an existing saved profile – users can now update an existing profile in the “Show/Hide” dialog window.
- Web tunneling – ability to install/launch the RPAS client and have an on-line session with the server over the internet; RPAS 11.0.4 could be deployed via a web server but could not communicate over the web. RPAS 11.1 allows the same deployment option, but allows the deployment and communication of a client-server connection to be run outside of a company’s network and firewall.
- Position name indirection – the indirection of position names provides an infrastructure that allows the renaming of positions and ultimately (currently not committed for a release) provide a mechanism to reduce the frequency with which arrays need to be reshaped by adding dummy reusable index entries to RPAS arrays. This initial implementation has removed the current limitation of the RPAS position name (24 characters), provides a utility for renaming positions (which can be used to rename ‘placeholder’ positions), and allows some special characters in external position names; the internal position name is used and managed only by RPAS.
- New RPAS utilities
  - Retrieving miscellaneous information about a domain – domaininfo
  - Copying domains (simple/traditional and global domains) – copyDomain
  - Mapping data between domains – mapData
  - Change the sparsity settings of all the databases in a domain to/from hypersparse – changeDomainSparseness
- Updated RPAS utilizes
  - Ability to reshape arrays in parallel – reshapeArrays
  - Change the width of position names (as documented with position name indirection) – inithier
  - Enhancements to hierarchy loading utility – loadhier
  - Ability to generate multiple user accounts with a single invocation of user manager utility – usermgr

- Miscellaneous improvements to server-side logging (domainDaemon and rpasDbServer)
- Support for displaying NA values and checking for them in calculations – a configurable display of values, which allows for cells with the NA value to be shown as empty
- Password encryption – passwords stored in “Foundation.fcf” and on a web server can be encrypted by using the new RPAS client configuration utility
- Reduction of contention by re-factoring of meta-data – RPAS has changed the schema for some meta-data (replicating meta data, etc), to reduce the number of database opens with write access, which will reduce user contention (users trying to access the same databases in the domain).
- Platform performance and user contention – various improvement in performance and contention management based on restructuring workbook meta data and low level optimizations in the Acumate Ali library.
- New API’s for efficient batch registration/unregistration of measures in global and normal domains.
- Certification on IBM AIX 5.2 UNIX platform

Impact and Implications of Upgrading to 11.1

The following should be noted about any upgrades to RPAS 11.1. Some of these are simply important notes while others are unavoidable implications of upgrading due to an architectural change or the implementation of a new feature.

- User selections in wizards will be removed during the upgrade to RPAS 11.1. RPAS has implemented a new approach for storing the selected positions that a user makes in the wizard process; selections are now stored as “Position Query Definitions”, which can be saved/named and reused in other wizard processes.
- Existing workbooks must be deleted when upgrading to 11.1. This is required for all RPAS patches as the structure of data must be the same between the domain and the workbooks. Workbooks can be deleted by using the –purgeWorkbooks parameter of the **upgradeDomain** utility.
- The workbook batch queue must be deleted when upgrading to 11.1. This implies that all workbooks currently scheduled to be built will need to be rescheduled and added back to the queue; additionally all workbooks scheduled to be refreshed and committed will also be removed from the queue sine all existing workbooks are deleted during the upgrade. The removal of this queue is required due to the implementation of Position Query Definitions that allows users to save selections. The workbook batch queue can be deleted by using the –purgeWorkbooks parameter of the **upgradeDomain** utility.

- New Limitation for Hierarchy and Dimension Names
  - A new limitation is being imposed in RPAS 11.1 that does not allow a hierarchy to contain a dimension with the same name (e.g. the PROD hierarchy cannot contain a dimension named PROD). It is being imposed only as a "soft" limitation in 11.1 in that RPAS will allow this condition but will issue a warning to the administrator.
  - This limitation is being imposed to address a specific Internationalization issue in the RPAS Translation Administration workbook template (used for modifying labels in multi-lingual environments). The names of hierarchies and dimensions are stored as positions and the template cannot handle these identical position names.
  - Existing configurations will be checked by the Configuration Tools to detect this condition, and users will be warned in the RPAS client if a workbook is built with this condition (including the Translation Administration template).
  - Although this is only a soft limitation in 11.1 and does not prevent a domain from having this condition, RPAS currently plans to enforce this as a hard limitation in 11.2.
- The model for position level security has changed in 11.1 and will impact existing security settings where position level security is enabled. The impact of the change is that previously access granted at ANY level (user, group, or world) gave a user access to a position; in 11.1 a user must be given access to a position at ALL levels. More information is included in the section below (Changes in Position-Level Security Behavior).

Upgrading to RPAS 11.1

Customers currently running RPAS 11.0.4.6 or 11.0.4.8 have the ability to upgrade to RPAS 11.1. As with any major release, there have been numerous changes to the RPAS platform and Configuration Tools; as such it is important that the following upgrade process is carefully followed.

Upgrade Process Overview

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

- Acquire the new version of the platform which includes the following key components:
  - RPAS Server
  - RPAS Client
  - RPAS Configuration Tools
- Create a backup copy of any domains that will be upgraded.
- Commit any existing workbooks that have not yet been committed (if desired); the upgrade process will purge all existing workbooks as they are not compatible with the new version of the platform.
- Run the upgrade utility on existing domains; note the utility must be run separately on each domain.
- Upgrade existing configurations.

Note the following important implications of upgrading to the new version of the platform:

- User selections in wizards will be removed during the upgrade process – the positions that a user selects during the wizard process were saved in RPAS 11.0.x. These selections are removed in 11.1 because of a new use of “position queries” that allows a user to save and load these selected positions.
- All existing workbooks must be deleted before or during the upgrade process. Additionally, all workbooks in the workbook batch queue must be deleted as well; workbooks scheduled to be built must be removed due to the change in the mechanism for storing user selections; workbooks scheduled to be refreshed and committed and no longer valid since old workbooks are incompatible with the new release. All of these workbooks are removed using the “**-purgeWorkbooks**” option of the **upgradeDomain** utility.

Upgrade Domains

1. After the new version of the 11.1 platform has been acquired and installed the administrator should create a backup copy of any domains that will be upgraded. This can be done in UNIX by simply copying the domains:

   ```
   cp –Rf olddomainname newdomainname
   ```

2. Ensure that all existing workbooks have been committed if necessary. They will not be compatible with RPAS 11.1 after the upgrade has been completed.
3. The RPAS utility upgradeDomain will be used to upgrade each domain. This utility is located in the directory: $RPAS_HOME/bin/

   If necessary, an administrator can verify the version of the upgradeDomain utility by typing the following command; the version should be 11.1.0.

   ```
   upgradeDomain –version
   ```

4. To upgrade a domain execute the upgradeDomain utility with the following usage.

   upgradeDomain –d *pathtodomain* {–n} {–verbose} –purgeWorkbooks

Argument	**Description**
–d *pathtodomain*	The path to the domain that is being upgraded.
–n	Optional parameter to report which changes would be applied without applying the changes.
–verbose	Optional parameter to show the detail about each change that is applied to the domain.
–purgeWorkbooks	Required parameter that purges all existing workbooks and clears the workbook batch queue.

Notes:

- If an administrator attempts the run the upgrade domain utility on a domain that has already been upgraded the following message will return:

  "Domain is already at version 11.1.0. Nothing to do."

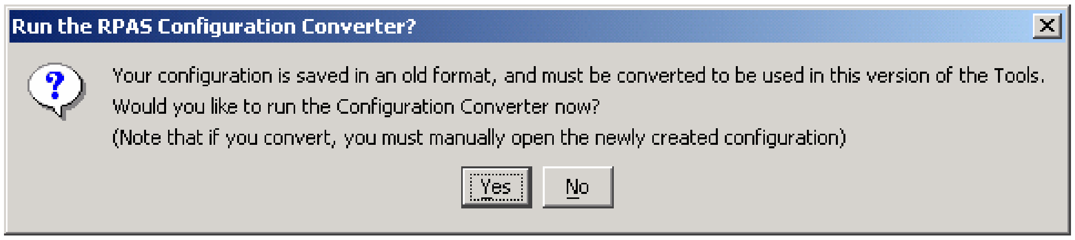
- An administrator can also run the Domain Information utility to verify the upgrade process.

  domaininfo –d *pathtodomain* –domainversion

Upgrade Configurations

Once the RPAS 11.1 Configuration Tools have been installed and the domains have been upgraded an administrator should upgrade existing configurations to 11.1.

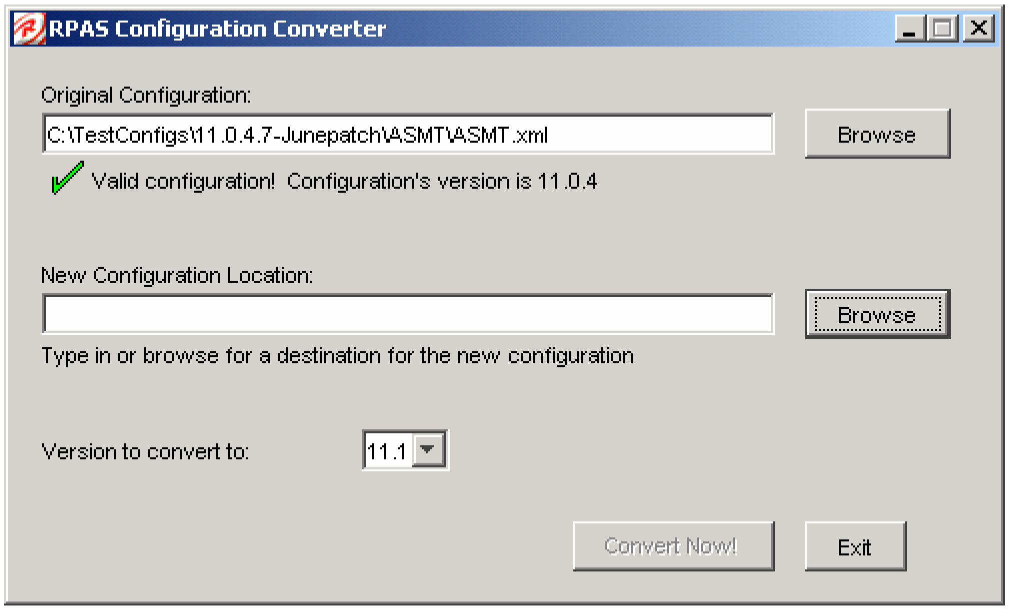
1. Launch the new version of the Configuration Tools; open the existing configuration using File/Open. Select the configuration, click **OK**, and the following message should appear:



Run the RPAS Configuration Converter?

Your configuration is saved in an old format, and must be converted to be used in this version of the Tools.
Would you like to run the Configuration Converter now?
(Note that if you convert, you must manually open the newly created configuration)

2. After selecting **Yes** the following window will appear. The administrator must specify a new location for the upgraded configuration.



RPAS Configuration Converter

Original Configuration:
C:\TestConfigs\11.0.4.7-Junepatch\ASMT\ASMT.xml
Valid configuration! Configuration's version is 11.0.4

New Configuration Location:
Type in or browse for a destination for the new configuration

Version to convert to: 11.1

3. Once you have chosen a new location and name, click **Convert Now**!. The system then allows an Administrator to convert another configuration.

4 If desired open the upgraded configuration using File/Open. Changes can now be made to the configuration. Note that if changes are made the domain must be updated through the normal patch install process (see the Configuration Tools User Guide for additional information).

> **Note**: For the 11.1 upgrade process, it is not required to run a **patchinstall** on the domain once the configuration has been upgraded. A patchinstall is only necessary if changes are made to the configuration after the upgrade process. Please refer to the Configuration Tools Release Notes for additional information.

Functional Enhancements

Global Domain

Overview

“Global Domain” is a new infrastructure feature in 11.1 that effectively provides the ability for users to create workbooks from multiple domains and to administer multiple domains from within a “master” domain. This is intended to be approximately functionally equivalent to the so-called “Cross Domain” functionality implemented in RPAS 9.4 for select customers. That functionality supported an ability to build and commit workbooks from positions spread across a number of independent, though identical (other than the positions) domains.

While “Global Domains” does provide some of the intended functionality of “partitioning”, it should NOT be taken to be a full solution to the problems that partitioning is intended to solve, especially the removal of contention at the Domain level, the ability to run domain level calculations that span local domains, and the automatic parallelization of a single stream of input data or metadata. The longer-term intention is to implement solutions to those issues within a “Global Domain” environment, whilst maximizing the reuse of “Global Domain” features.

In summary, RPAS has provided the following set of features as a part of the initial implementation of Global Domain:

- **Infrastructure** – infrastructure and tools to build and manage a global domain and a collection of local domains
- **Global Workbooks** – users can build, save, and commit workbooks in the global domain with data that is gathered from and committed to multiple local domains. Note that as per ‘cross domain’ in 9.4 this is intended to be an occasionally used feature and not the ‘normal’ way for most users to build most workbooks; for performance reasons users must continue to be build workbooks from the ‘local domains’ wherever possible.
- **Centralized Configuration and Administration** – a majority of the standard administrative tasks of RPAS are centralized in the global domain; the remaining administrative functions will be considered for centralization in future releases

Functionality and details not explicitly stated in this document should be assumed to be out of scope.

Definition

With the release of Global Domain, RPAS now has *three* (3) types of domains that can be used:

- *Simple domain* – traditional, stand-alone domain that has been used prior to 11.1; these domains have no visibility to other domains.
- *Master domain* – this is the primary/parent domain in a global domain environment. There will only be a single master domain per global domain environment. Global workbooks can be built from the master domain by collecting data from the local domains.
- *Local or sub domain* – the child domain in a global domain environment. There will be one or more local domains per global domain environment; the large majority of all transactions will be conducted in local domains within a global domain environment.

A *Global Domain environment* refers to a collection of domains built with a single master domain and multiple local domains.

Infrastructure & Features

RPAS has provided the ability to build a single global domain that has knowledge of and visibility to a collection of "local" domains. These local domains must be partitioned on a single dimension (e.g. department), but can contain data from one or more positions in each local domain. Thus if, for example, department is chosen as the partitioning level, some domains will contain a single (very big) department, whereas other domains may contain several small departments.

The configuration of a Global Domain environment is completed through the RPAS Configuration Tools or via a back-end procedure (documented separately). Domains built on 11.0.4.x can be upgraded to 11.1, but only as simple domains; customers wishing to take advantage of the Global Domain infrastructure must update their configurations and rebuild their environments.

Note that it is possible to invoke multiple RPAS clients and log into different domains with the same user in a Global Domain environment. This means that a single user can be simultaneously logged into the master domain and one or more sub-domains.

Global Workbooks

The primary feature of Global Domain is the ability to build a workbook with data from the collection of local domains. Users can log into the global domain and can run through the regular wizard processes to build workbooks. These workbooks can contain data from any local domain that is configured to be a member of the global domain.

Users are able to do the following with workbooks:

- Build global workbooks with data from local domains
- Save global workbooks
- Commit the data from a global workbook to the individual local domains
- Refresh data in these global workbooks from the local domains

Centralized Configuration and Administration

Many components of domain/environment configuration and administration are now centralized in the Global Domain. Components that are “centrally administered” are used in the global domain and those changes are propagated to all of the local domains.

To avoid data inconsistencies between the master and local domains most of the standard RPAS wizards/templates cannot be used in local domains and can only be performed in the master domain of a Global Domain environment.

These components are broken down into: 1) standard RPAS administrative templates used by an administrator in the RPAS client and 2) standard RPAS utilities (also referred to as stand-alone executables or binaries) that are run on the back end against a domain.

Administrative Templates

- Alert Manager dialog window – results of the alert finder run on the global domain are collated and displayed
  - Applies for all alerts registered in the global domain
  - Results will be based on data from all the individual local domains
  - Results are consolidated (added together) to display a single result per measure
- Alert manager workbook template – used to build alert workbooks from the Alert Manager dialog window; data will be retrieved from the local domains
- Measure Analysis – for analyzing measure data from local domains
- Security administration – ability to set security by template, measure, and positions; this workbook template can only be used in the master domain and has been disabled for use in local domains
- User administration – user information will be set up and maintained in the global domain, but will be replicated to the local domains; updates will be effective immediately after the changes are committed; this workbook template can only be used in the master domain and has been disabled for use in local domains
- Translation administration – template used for modifying the labels of translatable data in RPAS; this workbook template can only be used in the master domain and has been disabled for use in local domains
- Hierarchy maintenance – for setting up and maintaining positions of user-defined dimensions (user-defined dimensions must be registered in the Global Domain using utility reguserdim)

Utilities

- Alerts (alertmgr)
  - Alerts registered in the global domain will be propagated to local domains; note, however, that alerts registered in local domains will not (and do not have to) be included in the global domain
  - Utility for finding alerts (a.k.a. alert finder) runs against the local domains and collates results in the global domain for a centralized view of the alert results
- Loading hierarchies (loadhier) – it is required that hierarchy information be centrally administered in the Global Domain and replicated to the local domains
- Reshaping arrays (reshapeArrays) – execution of this administrative utility in the global domain will perform this function on all local domains
- Users (usermgr) – the back-end utility for managing users will propagate changes from the global domain to local domains; ultimately this will also be updated for the RPAS Administrative Template “User Administration”
- Domain properties (domainprop) – manipulating properties such as specifying password properties, locking user accounts, etc.
- Miscellaneous registration utilities – some of the registration utilities have been updated to operate in the global domain where all changes are propagated to the local domains
  - Measures (regmeasure)
  - Templates (regtemplate)
  - Functions (regfunction)
  - Token measures (regTokenMeasure)
  - User-defined dimensions (reguserdim)

Global Domain Measures (Base Intersections Higher than the Partition-Level)

In a global domain environment, measure data can be physically stored in two different ways: across the local domains or in the master domain.

Measure data stored in local domains is split across domains based on the partition-level that is defined for the global domain environment. However, measures that do not contain the hierarchy on which the environment is partitioned, or measures that have a higher base intersection in that hierarchy than the partition-level must be handled differently. These measures will be referred to as ***HBI (higher base intersection)*** or ***global domain measures***.

HBI Example

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

However, measures that have a higher base intersection in the Product hierarchy than Department (e.g., Division) or measures that do not contain the Product hierarchy (such as a measure based at Store-Week, no Product Hierarchy) cannot be split across the local domains. These measures will reside in the master domain and will be accessed from there when these measures are required in workbooks.

HBI Configuration

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

The physical location of the measure data will be invisible to the user after the measure has been registered. Implementers will be able to reference measures. The only implication will be on the implementer to load the measures in the proper domain (all in the master or split across the local domains).

Additionally, these different measure storage options will be invisible to implementers and configurers when configuring workbook templates in that a measure will be treated the same in the Tools regardless of where the data will ultimately be stored.

Assumptions & Limitations

- Performance and contention warnings – Usage of the functionality to build workbooks from the global domain is assumed to be low frequency. Most workbooks will continue to use just a single local domain, so most workbook builds will not use (or even be aware of) the global domain.
- Templates with MSPL code cannot be built from the global domain.
- There is no provision for global domain level calculations using the RPAS calculation engine utility (mace, multi-dimensional calculation engine). However, "mace" can be run on the local domains; additionally mace can use measures whose data is stored in the master domain (referred to as "global domain measures" or "higher base intersection measures").
- RPAS allows for positions to be added or removed at the partition level in a Global Domain environment requiring the use of a utility; however, sub/local domains cannot be added to an existing environment in 11.1. For example, if a Global Domain environment is partitioned at the department dimension of the product hierarchy, new departments can be added to a sub/local domain or moved from one sub-domain to another; however, new departments must be added to an existing sub-domain.

Reconfiguring Positions at the Partition Level - reconfigGlobalDomainPartitions

RPAS provides the ability to add, remove, or reassign positions at the partition level in a local domain. The following is an outline of the process that must be followed:

- Administrator will be notified in advance that a new position is being added to or removed from the system.
- Run the utility **reconfigGlobalDomainPartitions** to add/remove the positions by specifying the sub-domain to which the positions do or will belong.
- Load the updated hierarchy file (that contains the new positions or doesn’t contain the removed positions) using the **loadhier** utility; the next run of **loadhier** following the addition or removal of positions using the **reconfigGlobalDomainPartitions** utility must contain the updated hierarchy file or **loadhier** will fail.
- Run **reshapearrays** utility to update the domain to reflect the changed positions.

Note this only applies to positions at the partition level; positions can be added or removed **below** or **above** the partition level simply by being added or removed from the hierarchy file.

reconfigGlobalDomainPartitions –d *pathToMasterDomain* –add *posName1,posName2*, ... –sub *pathToSubDomain*

reconfigGlobalDomainPartitions –d *pathToMasterDomain* –remove *posName1, posName2*, ...

Argument	Description
–d pathToMasterDomain	Specifies the path to the master domain in a Global Domain environment
–add posName1, posName2, ...	Adds one or more positions to a specified local domain; the path to the sub-domain must follow the list of positions to add
–sub pathToSubDomain	Specifies the path to the sub-domain to which new positions are being added
–remove posName1, posName2, ...	Removes the designated positions from the sub-domain to which the positions belong

Notes:

- Position names are separated by commas and must be valid **external** position names without the prefix of a dimension
- Note that the path to the sub-domain does not need to be specified if removing positions

Assumptions & Limitations

- Positions can only be added or removed in EXISTING sub-domains.
- User has to specify the sub-domain to which the positions are being added.
- User must call this utility before loading the hierarchy.
- User must run **reshapeArrays** utility after loading the hierarchy.
- The remove option will NOT remove the sub-domain if sub-domain does not have any positions.
- Multiple positions can be added to a sub-domain in a single call to the utility.
- If new positions are included in the hierarchy load file prior to running the **loadhier** utility, RPAS will reject the entire load for that hierarchy file, generate an error message and use the previous load.

Manipulable Percent-to-Parent Measures

Overview

A 'percent to parent measure', or 'participation measure', is a measure that, for a particular measure, contains the value of the current positions as a proportion of the value at a 'parent' level, for example, sales as a percent of the class sales. These measures can be viewed and edited, and may be preconfigured through the RPAS Configuration Tools or dynamically defined in the RPAS client in a worksheet.

Typical uses of this functionality will be to define measures that are percentage participations of sales measures. Typically these are either to a fixed level, such as class, so the participation of each item to the class can be viewed and manipulated, or are to the ‘next level up’ in the product hierarchy.

The following sections describe the percent to parent feature and how to create these measures in the RPAS client. Examples will use the following sample product hierarchy structure: “SKU-style-subclass-class-company-all”.

Definition & Usage

To define a percent-to-parent measure right-click on the measure axis and select “Create Percent to Parent Measure” in the quick-menu.

Select the measure for which a percent to parent measure is desired. The drop-down list contains all measures defined for the active worksheet.

Select the desired percent to parent measure type. The dialog box is context sensitive and the available parameters will change based on which type is selected. The following sections contain detailed information about the different types and how they are defined.

Once the type and the required parameters have been selected press **OK** and the new measure will appear in the worksheet.

Note the following important points when using this feature:

- Note that changing the percentage of the percent to parent measure will cause the values of the underlying measure to change to reflect the newly set percentage.
- Multiple percent to parent measures can be defined for the same underlying measure; however, only one percent to parent measure or the underlying measure can be edited before calculation; all other “versions” will be protected.
- The value of a percent-to-parent measure is a fraction that will usually be between zero and one. Users must format the measure to be displayed as a percentage if desired.

Absolute

The “absolute” type of percent to parent measures allows a user to explicitly define the parent level(s) that are used to calculate the percentage at all child levels.

Using the previously defined sample hierarchy structure if a user sets the absolute parent level to the "class" dimension in the product hierarchy, the percent to parent measure will show the "SKU" as a percent of the "class" at the "SKU" level, the "style" as a percent of the "class" at the "style" level, and the "subclass" as a percent of the "class" at the "subclass" level.

After selecting the "Absolute" type and the measure in the “Create Percent to Parent Measure” dialog box, the user will see an "Intersection" area. This area will contain drop-down lists for all hierarchies on which the measure is dimensioned with each drop-down list containing the dimensions available for that hierarchy.

Select the dimension/level for each hierarchy that is to be explicitly set. After clicking **OK** the user will see the new percent-to-parent measure in the worksheet.

For the "absolute" relationships, cells at or above the explicitly set dimension/level will be hashed out.

Relative

The "relative" percent to parent measure type calculates the value for a given level that is the percentage of that level and its immediate parent (meaning one level higher). This type can only be set for a single hierarchy.

Using the previously defined sample hierarchy structure the percentage displayed at the "SKU" level is the "SKU" as a percent of the "style", at the "style" level the "style" as a percent of the "subclass", and at the "subclass" level the "subclass" as a percent of the "class".

After selecting the "Relative" type and the measure in the “Create Percent to Parent Measure” dialog box, the user will see a drop-down list named “Hierarchy”. This drop-down list contains the hierarchies on which the measure is dimensioned. Select the hierarchy up which the percentage will be calculated.

Note that only a single hierarchy is possible with the “relative” percent to parent measure type.

After clicking **OK** the user will see the new percent-to-parent measure in the worksheet.

In the case where there are multiple branching hierarchies certain dimensions will have multiple parents; for relative percent to parent measures the percentage will always be calculated based on the active roll-up in the current window.

Note that the calculated value is based on the **actual** next level up in the hierarchy (based on the hierarchy structure), not necessarily the one that is being displayed. In the previous examples imagine that SKU-subclass-class is displayed in the client, but the underlying structure is SKU-style-subclass-class. When viewing the value at SKU, the percentage will be based on style, not subclass even though style is not displayed.

Cells at the top of the hierarchy will be hashed out as those values cannot be calculated.

Hierarchical Measures

Overview

The following are specifications for new features that provide the functional behavior of “hierarchical measures” when used together to write expressions. In merchandise planning, users are familiar with the use of hierarchical relationships in planning. Certain measures have a relationship that is fundamentally hierarchical, but without these new features, the calculation engine could not support fully hierarchical manipulation of such measures. There are a number of such relationships that regularly occur in merchandise planning type applications (for example, regular, promotional and clearance sales rolling up to total sales, promotional and permanent markdowns rolling up into a total markdown, or DC, In transit and Store inventory rolling up into total inventory).

These enhancements allow the manipulation of any combination of the measures, except all of them. The changes to multiple such measures are resolved in the same manner as such changes would be resolved in, say, a product hierarchy.

Mechanism for Referencing the Pre-Calculated Value of a Measure – old

The new modifier (“old”) allows the reference/use of the pre-calculated value of a measure. Any measure modified with “old” (e.g. sales.old) will use the value that was available at the start of the calculation process, which means that these modified measures can be ignored for such things as protection processing. Most importantly, this means that a measure can effectively be calculated from itself, as the **.old** modifier breaks the cycle.

old – References the previous value of a measure as of the previous calculate.

Assumptions/restrictions:

- Can only be used in a calc group.
- Can only be used on the right-hand side of an expression.
- Cannot be used in combination with .master, level, or .aggtype modifiers.
- Cannot be used with non-materialized measures.

Syntax: <measure>.**old**

Use of the **.old** modifier has no effect on calculation sequence or protection processing, as the values of **.old** measures are known before the calculation starts.

Example uses of the **old** modifier have been provided with the functions listed below.

Function for Proportionally Spreading a Value Across Multiple Measures – propspread

Definition & Syntax

propspread – ‘proportionally spreads’ a value across a collection of measures whilst retaining their relative proportions.

This function has multiple results. The multiple results are not named, and are therefore positional only. The typical usage of this function is to allow spreading of ‘hierarchical measures’.

Syntax: *propspread***(<totalexpression>, <childexp1>, … <childexpn>***)*

Where <*totalexpression*> is an expression that returns a numeric value to which to balance the results of the function. <*childexp1*> **-** <*childexpn*> are expressions that provide the ‘shape’ of the results. They will typically be the same measures as those assigned to the result of the function, but using the **old** modifier (a measure defined as a result cannot be used on the right hand side without **old**).

The function generates *n* positional results, where *n* is the number of ‘child expressions’. The results will sum to the <*totalexpression*>, using the ‘shapes’ of the child expressions, in the order of the child expressions.

The number of results should be equal to the number of child expressions, meaning there should be one more argument on the right hand side than output measures on the left hand side; additional child expressions are ignored. If too few child expressions are defined then the function will fail.   Currently there is no validation to warn an individual when this condition is met.

If the sum of the child expressions is zero, the spread will be even.

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

Examples

- Hierarchical measures – the **.old** modifier can be used in conjunction with the **propspread** function to implement a hierarchical relationship among measures. In the following example, TotalSls is the "parent" measure and RegSls, PromSls, and MkdSales are the "child" measures. This example shows 8 expressions which should all be in the same rule. It allows any combination of the 4 measures to be manipulated, except all 4 of them. In the absence of 'forcing' from other rules, changes to one or more 'child' measures, without changing the 'parent' measure will result in 'aggregation' to the parent. If the parent is changed, then all the children that are unchanged are 'spread' in such a manner as to keep their relative proportions.

  TotalSls = RegSls + PromoSls + MkdSls
  RegSls, PromoSls, MkdSls = **propspread**(TotalSls, RegSls**.old**, PromoSls**.old**, MkdSls**.old**)
  PromoSls, MkdSls = **propspread**(TotalSls - RegSls, PromoSls**.old**, MkdSls**.old**)
  RegSls, MkdSls = **propspread**(TotalSls - PromoSls, RegSls**.old**, MkdSls**.old**)
  RegSls, PromoSls = **propspread**(TotalSls - MkdSls, RegSls**.old**, PromoSls**.old**)
  RegSls = TotalSls - PromoSls – MkdSls
  PromoSls = TotalSls - RegSls – MkdSls
  MkdSls = TotalSls - RegSls - PromoSls

- Shrinkreg, shrinkclear = propspread(shrinktot, EOPreg, EOPclear) [A total shrinkage value is allocated to regular or clearance shrinkage, based on the proportions of regular and clearance inventory]

Function for Performing Several Computations in a Single Expression – passthrough

Definition & Syntax

passthrough – function with multiple results that is used to encapsulate any number of normal computations into a single expression.

Syntax: **passthrough**(*<exp1>*, *<exp2>*, …, *<exp-n>*)

Where *<exp1>* **-** *<exp-n>* are normal expressions used to calculate the resulting measures.

All measures on the left hand side must be computed at the same intersection. The number of results should be less than or equal to the number of calculation expressions (additional calculation expressions are ignored); if too few calculation expressions are defined then function will fail. Currently there is no validation to warn a user when this condition is met.

There are two main reasons for using this function:

- Use **passthrough** in an expression for a rule when you need to compute values for multiple measures without having to write (develop) a multiple-result function or procedure.
- For performance reasons: If many measures are computed using the same or similar set of RHS measures, combining those calculations using 'passthrough' may be faster because there is less physical input/output with the data.

Examples

- Compute the sum and difference of two measures simultaneously:

  A, B = passthrough(C + D, C - D)

- Proportionately spread TotalSales down to its components, SalesA and SalesB:

  SalesA, SalesB = passthrough(SalesA.old * TotalSales / TotalSales.old, SalesB.old * TotalSales / TotalSales.old)

Commit As Soon As Possible (Commit ASAP)

Overview

Commit As Soon As Possible (Commit ASAP) allows users to schedule the commit process of workbook data so that it executes as soon as all the system resources are available. Commit ASAP is an option in the File menu of the RPAS client; procedures for using Commit ASAP are provided in the “RPAS 11.1 Users Guide”.

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

Using Commit ASAP

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

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

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

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

Users can filter the entries in a variety of ways. If the checkbox ***All Users*** is not checked the user will see only their entries; if it is checked they will see the entries for all users. The checkboxes in the ***Status To Display*** group allow you to filter the output so that you see only the processes with the specified statuses. The window can be updated using the ***Refresh*** button. Also note that the dialog remembers the settings based on the last use.

Important notes:

- If a user attempts to commit a workbook ASAP that already has a process in the queue, the original processes will be removed from the queue. That means that there can only ever be 1 pending commit ASAP in the queue for a given workbook/user/template name combination.
- Workbooks must have been saved at least once before attempting a Commit ASAP (a workbook hasn't been saved if the label says "untitled")

Managing the Workbook Queue – showWorkbookQueues

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

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

Usage:

showWorkbookQueues -version

showWorkbookQueues -d *domainPath* –show [all|pending|waiting|working|success|failed]*

showWorkbookQueues -d *domainPath* -purge *date* [success | failed]*

Argument	Description
–version	Prints the RPAS version, revision, and build information of the utility.
-d *domainPath*	Specifies the path to the domain.
-show	Lists the contents of the queue in the order in which the parameter is specified. Possible values: all, pending, waiting, working, success, failed
All	Used with the –show parameter this lists all of the workbooks in all statuses
pending	Used with the –show parameter this lists all workbooks that are waiting to be committed
waiting	For Retek development use only
success	Used with the –show parameter this lists all workbooks that have been successfully committed
failed	Used with the –show parameter this lists all workbooks that did not successfully commit

Argument	Description
-purge *date*	Purges entries in the Commit ASAP status window; entries before the date provided will be removed. The date should be a string of the following DateTime format: YYYYMMDDHHmm For example "200406071529" equals June 7, 2004 3:29 PM Administrator must select to purge commit processes that either succeeded or failed

Commit ASAP Settings – configCommitAsap

There are 2 settings for Commit ASAP that are managed by an administrator. Both are set using the utility **configCommitAsap**.

- Maximum number of simultaneous commit processes (property **MaxProcesses**, default value is 4)
- Deadline for which all pending processes must be completed, after which they will be cancelled and marked as failed
  - This deadline will likely be used by administrators before beginning nightly batch processes (property **deadline**, default value is 00:01 [meaning 12:01 AM], in 24-hour time)
  - A commit process that starts before the deadline is reached will be processed; commit requests that were in the queue before the deadline that did not get processed will be cancelled and marked as failed; commit requests added to the queue after the deadline will use the deadline the following day

Usage:

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

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

Logging and Technical Information

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

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

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

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

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

Miscellaneous New Functionality

Parallel Workbook Build & Refresh

RPAS allows for workbooks to be built and refreshed in parallel when using the RPAS utility ***wbbatch***. A new parameter (**-parallel**) has been added to both the **-build** and **-refresh** parameters that requires the specification for the maximum number of parallel processes that can be simultaneously run.

The RPAS utility will execute the builds and refreshes in the order in which they are entered; each parallel process will execute the next entry in the queue as it completes the previous operation.

Note that this parallel option will not operate on the Microsoft Windows operating system for the RPAS server.

Partial/Selective Refresh

RPAS 11.1 provides the ability to have multiple refresh rule groups that are configured in the Tools. Each named rule group is configured to contain one or more measures whose data will be retrieved from the domain and updated in current workbook when selected.

Note for RPAS 11.0.4 customers – many 11.0.4 customers were able to use equivalent functionality with multiple refresh rule groups by using custom menus. A menu option could be used to create a rule group that contained the measures that a user would need to refresh. While these were not technically “refresh” rule groups (i.e. they were just “regular” rule groups that RPAS executed when calling the custom menu option), this approach provided similar functionality to what is available in RPAS 11.1. This approach in 11.0.4 presented issues with performance under certain circumstances and customers will need to update their configurations to use this new approach if they wish to take advantage of this functionality. Customers must re-test their configuration after using this new approach due to the way RPAS treats these types of rule groups; it is possible that certain complicated refresh rule groups will behave slightly differently.

Populated total aggregation (total_pop) and proportional spread (prop_pop)

New aggregation type **total_pop** that calculates the total of populated cells. In conjunction with this aggregation method is a new spread type, **prop_pop**, to proportionally spread values to populated cells.

Limiting Number of Saved Workbooks

A feature has been implemented that can be used to limit the number of workbooks a user can create. This limit can be set at the following levels:

- User – workbook template
- User group – workbook template
- Workbook template (all users)

These limits are maintained in the RPAS Security Administration workbook template. They are evaluated in the above order, and the lowest level of detail takes precedence and becomes the final limit. For example, if a limit has been set at the level user-template, values for user-group and template are ignored even if they are populated. If no limit is defined the default value is one billion.

The limit is checked when a user begins the workbook build process and selects a workbook template. If the user's limit has been reached an error message appears that informs the user that the workbook build process cannot complete because the user has reached their limit and what that limit is. The wizard process then terminates.

Workbooks that are on scheduled to be built automatically are handled consistently with manually built workbooks. The workbook batch utility **wbbatch** will generate an error that is printed to the terminal when the utility attempts to build a scheduled workbook that will break a designated limit for a user.

Saved Selections in Wizard Processes – Position Queries

Background & Overview

There has been a long stated intention that RPAS would implement 'position queries', which would be used to select position required for various processes, and that 'position queries' would become very powerful and sophisticated, and all pervasive. This enhancement provides some of the necessary infrastructure to support that long term intention, whilst providing functionality that is functionally equivalent to the 'saved selection sets' that were available in RPAS 9.4.

A form of position queries can now be used in the wizard process when selecting position for a workbook. Users are able to load and save the set of positions that have been made. These selections are referred to as "**Position Query Definitions**", or **PQD's**.

At this stage, just a single form of position queries is supported in these wizards (though others will be added in the future). In this form the selections are saved at the level at which they are made. This means that as the children of a given parent position change over time the selection is also updated, when made at that level. For example, if, for a process that requires a collection of SKUs, a user selects a sub-class, the list of positions that are generated when the PQD is run will always reflect the current SKUs (children) of that sub-class. The PQDs can also be defined at multiple dimensions/levels of a hierarchy. For example, users can select individual SKUs and individual sub-classes in a given PQD; when the PQD is executed in the future, the individual SKUs will be included as well as all current children of the selected sub-classes.

Features

In the wizard process the right side of the 2-tree will contain 2 tabs; one for selected products (as in pre-11.1) and another for the position query definitions. The PQD tab will list all the available PQDs for the current hierarchy to which a user has access. When saving a PQD the user specifies the name and the access level (user or world, group is not supported in 11.1). Users also have the ability to overwrite and delete existing PQDs.

Note that template wizards still remember the current selections when the user selects NEXT or FINISH (as in pre-11.1); these are stored as unnamed PQDs. Those selections are the default selections the next time the user builds a workbook. Users must explicitly save these selections if they wish to store them for use at another time, or by another user.

Ability to Update and Save an Existing Profile

When in the Show/Hide dialog in the RPAS client, users can now edit an existing profile and save any changes. Previously it was only possible to delete and re-add the profile after making changes.

Web Tunneling

An enhancement to web support is available in RPAS 11.1 referred to as “web tunneling”. RPAS 11.0.4 included a feature referred to as “web launch” that provides the ability to install and launch the RPAS client on a user’s computer from a URL in a web browser.

“Web tunneling” provides the ability to deploy the RPAS client and to run a client-server connection, both over the internet. This means that it is possible for a user to install/launch the RPAS client and communicate with the RPAS server outside of a company’s network/intranet/firewall.

Changes to Existing Functionality

Changes in Position-Level Security Behavior

There are some important behavioral changes for position-level security in RPAS 11.1. Position-level security is controlled in the RPAS administrative workbook template Security Administration and controls to which positions a user, a user group, or all users (world) have access to positions in hierarchies for which position-level security is enabled. When enabled access is controlled with a true/false (granted/denied) setting (Boolean measure) for all positions in the hierarchy.

In earlier versions of RPAS 11.0.4 all positions were/are given a value of "true" (meaning the user is **granted** access) by default; this required that users be explicitly denied access to a given position. More specifically, however, this required that ALL levels (user, group, and world) were explicitly set to "false" to deny access to a position for a user. This reportedly caused confusion with many customers and administrators as it seemed to require excess maintenance.

In RPAS 11.1, all positions still default to a value of true at all levels, but RPAS now requires that all 3 levels (user, group, world) be **granted** access for a user to have access to a given position. This means that an administrator only has to deny access to a position at any level to deny a user access.

The following table briefly describes how the behavior works and how it compares to previous releases.

Security set by Position (Denied = False, Granted = True)			**Based on settings on left, user is Granted or Denied access**	
User	**User Group**	**World**	**11.0.4**	**11.1**
Denied	Denied	Denied	Denied	Denied
Denied	Denied	Granted	Granted	Denied
Denied	Granted	Denied	Granted	Denied
Granted	Denied	Denied	Granted	Denied
Denied	Granted	Granted	Granted	Denied
Granted	Denied	Granted	Granted	Denied
Granted	Granted	Denied	Granted	Denied
Granted	Granted	Granted	Granted	Granted

New RPAS Client Configuration Utility

A new version of the Configure has been provided in RPAS 11.1. The “Configure.exe” utility is used for setting up and configuring the RPAS client. The new utility, “**EConfigure.exe**” is being provided **in addition** to the previous/existing version of Configure. The new utility is functionally equivalent to the previous Configure utility, except that the new version stores passwords encrypted. This will provide a higher level of security for user passwords, especially for customers choosing to deploy or run RPAS over an internet connection.

The Configure utility is used for setting up the connection between the RPAS client and server by specifying the port to which the client should connect and the path to a domain, amongst other things.

Technical Enhancements & Changes

Position Name Indirection

The indirection of position names provides an infrastructure that enables the renaming of positions and will ultimately provide a mechanism to reduce the frequency with which arrays need to be reshaped (by adding dummy reusable index entries to RPAS arrays). This initial implementation will remove the current limitation of the RPAS position name (24 characters); the actual internal position name will be used and managed only by RPAS.

Overview

The principle of position name indirection (PNI) is to separate the 'internal key' used for physical storage of data and meta data within RPAS, and an 'external key' – the identifier used when communicating with external systems. PNI is a key enabling technology to provide support for a number of features:

- The ability to have identifiers longer than Acumate-imposed limits (24 characters). With PNI, there is no conceptual limit on an identifier length (included 11.1)
- The ability to quickly and easily rename a position, without having to rebuild Acumate arrays, or copy any data (included 11.1)
- The ability to provide dummy positions in Acumate arrays, so the addition of new positions does not always necessitate the expensive process of reshaping arrays (out of scope for 11.1)

The implementation of position name indirection in 11.1 establishes a 'mapping table' that maps 'external names' to 'internal names'. Where required, internal names will be system generated, and have no intrinsic meaning, but will be used for all internal processes; external names will be the equivalent of the current names, and will be used in all data and meta-data feeds and exports.

In the second phase of PNI development (not in scope for 11.1), when arrays are built or reshaped, there will be the capability to generate 'dummy' entries (that is internal names which have entries in the array and the 'mapping table' that do not map to an external name). When new positions are added to the dimension, they will be assigned to one of the 'dummy' entries (if any are available), and when positions are deleted they will be flagged as deleted in the mapping table; in both cases reshaping of the arrays will not be required.

Features of PNI

The following features are included with the initial implementation of Position Name Indirection.

- Mapping table that stores the relationship between internal and external position names; RPAS will manage the internal names and these internal names will not be visible to an end-user; domains that are upgraded to 11.1 will use the same internal and external names until new positions are added or existing positions are renamed
- Measure load and export utilities use external position names
- Configurable maximum length of external position names for a dimension. Once set, this width may not be reduced without a complete rebuild of the environment (since keys that are currently unique may no longer be unique). The width may be subsequently increased through the **inithier** utility.
- Application Programming Interfaces to map internal and external names; standard RPAS functions and procedures that can use position names in the syntax have been updated to use external rather than internal names (includes timeshift, lookup, flookup, index, and position)
- Rename the external name of a position
- Export sheet (in the RPAS client) exports the external position names

Setup & Configuration

Position name indirection is automatically enabled in all 11.1 domains. The default width of position names will remain at 24 characters unless the use of longer position names is explicitly set. Note that the **position name width is set by dimension**.

Changing Position Name Length before Building a Domain

By default all domains in RPAS 11.1 will have a position name length of 24 characters. To use position names longer than 24 characters for a particular dimension the **hierarchy.xml configuration file** must be updated with the correct setting.

In the following example the administrator is setting the size of position names to 30 for the SKU dimension of the product hierarchy. This process must be completed prior to building a domain.

```
<hierarchy name="prod">
        <label>Product</label>
        <purgeage>30</purgeage>
        <order>20</order>
        <dimension name="sku">
            <label>SKU</label>
            <db>hmaint</db>
            <prefix>4</prefix>
            <start>1</start>
            <width>30</width>
            <labelstart>31</labelstart>
```

```
                <labelwidth>100</labelwidth>
```

Note that the associated .dat file that contains the actual position names must be configured to reflect the new length.

Changing Position Name Length in an Existing Domain

To change the length of position names to more than 24 characters in an existing domain the RPAS utility **inithier** must be run. The syntax is as follows:

inithier –d pathToDomain –dim dimensionName **–width width**

Note that the designated length of the position names must correctly match the associated data of those positions names in the input data files. Note that the length can only be increased and cannot be decreased after a change has been made.

Renaming Positions (renamePositions utility)

Positions can be renamed by using an RPAS named **renamePositions**. Positions that are to be renamed should be included in a hierarchy data file that is located in a designated input directory (specified when running utility) and that follows the naming convention **hierName.rn.dat** (e.g. "prod.rn.dat").

After the hierarchy data file(s) has been updated an administrator must run the **renamePositions** utility with the following usage:

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

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

Note the following about the input file:

- Input File should be named as hierName.rn.dat (e.g "prod.rn.dat")
- You should specify the directory in which this input file locates by using –input option.
- There will be three columns of data in each input line corresponding to dimension name, old position name, and new position name. The three fields will be tab-delimited. Any line not formatted this way will be ignored. Empty lines are also ignored.
- Old position names should be either an existing external name or (after phase 2 of PNI) an existing internal name, which hasn't yet been associated with an external name.

- New position names cannot be an already used external name or existing internal name. Lines having this kind of new position names will be ignored and added to the log file.
- Old Position Name and New Position Name should not be prefixed.
- The 'width' attribute in the domain must be greater than or equal to the max length of the new external names in the input file; otherwise, width reconfiguration must be done before the rename process. If the width of the new name is greater than the width attribute of the corresponding dimension, we log an error in the log file and ignore that line.
- Dimensions specified in input file should belong to the hierarchy specified in arguments. Otherwise that line will be ignored and we log an error into log file.

Note the following about the utilty:

- In a Global Domain environment, this utility is centralized and can only be run in the master domain. Calling it in a sub-domain will result in an error message.
- The –n is a dry run meaning it does everything as a true run (e.g writing to a log file) except that it doesn’t actually commit the changes to the domain.
- –log is an optional argument that you can use to name the log file other than the default, which will be created as hierRename.log in the current directory.

Assumptions & Limitations

- Existing workbooks cannot be updated with new or changed external position names (as with any updates from loadhier); users must rebuild any workbooks to use any new or changed positions.

New RPAS Utilities

Retrieving Information about a Domain – domaininfo

The **domaininfo** utility is replaces the former **domainversion** utility (with additional functionality). Its purpose is to provide basic information about a given domain, specifically to determine whether a domain is a simple domain (not created in global domain environment), a master domain (parent of a global domain), or a local domain (child of global domain).

domaininfo –d pathToDomain [COMMAND]

domaininfo –expectedversion

domaininfo –version

Argument	Description
-domainversion	display version of referenced domain
-expectedversion	display domain version expected by this code
-type	display domain type (see types below)
-sparsity	display domain sparsity (see types below)
-listsubdomains	list all subdomains of a global domain

Argument	Description
-subdomain dim,pos	find subdomain containing given position
-all	display combined details of domain
-history	display version history of domain
-version	display version of this utility

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

The **–domainversion** command gives the version of the domain pointed to by the **-d** parameter. The **–expectedversion** command prints the domain version that the current code would expect to find.

The **-type** command returns which of three types of domain is being inspected:

- Simple: domain is a standard, non-partitioned domain
- Master: domain is the 'master' of a global domain set
- Sub: domain is one partition of a global domain set

The **-sparsity** command reports whether a domain has been designated as containing sparse or hypersparse measure databases.

If a Global domain is being inspected, then the **-listsubdomains** and **-subdomain** commands may be useful. The **-listsubdomains** command will display a list of all subdomains of this domain, and will tell which positions of the partitioned dimension are in each subdomain. The **-subdomain** command allows searching for a particular position of the partitioned dimension or any dimension that rolls up to the partitioned dimension. For example, if the global domain environment is partitioned on DEPT, one may query which local domain contains a particular DEPT position or a particular SKU position.

The **-history** command lists the different versions and upgrade actions taken on the domain.

The **-all** command reports all relevant information on the domain.

Copying Domains – copyDomain

The **copyDomain** utility is used to copy a simple domain or all domains included in a Global Domain environment.

For a standard, simple domain (i.e., non-global), **copyDomain** merely copies the domain directory recursively from one location to another.

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

The arrays within a global domain environment were previously required to contain absolute filename paths. However, relative paths (based on the full pathname of a domain's root directory) are now supported, and **copyDomain** places relative paths in the arrays in the new copies. A benefit of this feature is that the new copy can be archived or copied just like any other directory, and all of the global domain paths in the arrays will still be valid.

copyDomain -d *pathToSrcDomain* -dest *pathToDest* { -f }

copyDomain -version

Argument	Description
-d pathToSrcDomain	Specifies the path of the domain to be copied.
-d pathToDest	Specifies the path to where the domain is to be copied.
-f	Forces the deletion of the existing destination path before copying.

Mapping Data between Domains – mapData

The **mapData** utility is used to move data from one domain to another. Specifically, it copies data from an existing domain, database, or array to a new domain, database or array. This utility is the functional equivalent to three utilities previously available in RPAS 9.x (**mapdomain**, **mapdb**, and **maparray**).

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

The usage of this utility follows:

mapData -srcDomain *srcDomainPath* -destDomain *destDomainPath*
{-db *dbName* {-array *arrayName*}} {-loglevel}

Argument	Description
-srcDomain srcDomainPath	Specifies the full path of the source domain.
-destDomain destDomainPath	Specifies the full path of the destination domain.
-db dbName	Instructs **mapData** to map data only on the given database. If this parameter is not specified the utility will map the entire domain.
-array arrayName	Instructs **mapData** to map data only on the given array within the given database specified by **–db**.
-loglevel	Sets the logger verbosity level. Possible values include: all, profile, debug, information, warning, error, or none.

Changing the Sparseness of Databases – changeDomainSpareseness

The **changeDomainSparseness** utility can be used to change the database sparsity from sparse to hypersparse (or *vice-versa*) for all databases within a specified domain.

The general usage of the utility is:

changeDomainSparseness -d *domainPath* [-tohypersparse | -tosparse] {-nobackup}

If the conversion fails (*e.g.*, due to sparsity issues), the original databases (**.gem** files) will be restored from the backup directory, and the user will be notified of the operation.

If the **-nobackup** switch is specified, the databases in the previous sparsity format will be deleted after the databases with the supplied sparsity are created. Otherwise, a directory named **data/backup** in the domain will be created, and the older files will be placed there.

Modified RPAS Utilities

Reshaping Arrays in Parallel – reshapeArrays

A new parameter –processes has been added to the **reshapeArrays** utility that can be used in a Global Domain environment. **reshapeArrays** is used to update the data structures of the domain after hierarchy changes have been loaded, such as the addition or removal of positions.

reshapeArrays **–d domainPath –registered {–purge} {–processes max}**

Changing the Width of Position Names – inithier

The RPAS utility **inithier** has been updated to allow an administrator to extend the length of position names. A more complete definition of this is available in the section describing "Position Name Indirection".

Note that Position Name Indirection is included in all 11.1 domains; the default width of position names is 24. This width can be increased but cannot be decreased without a rebuild of the environment. To increase the width of position names for a designated dimension use the following syntax:

inithier **–d** ***pathToDomain*** **–dim** ***dimensionName*** **–width** ***width***

Enhancements to the Hierarchy Loading Process – loadhier

The **loadHier** utility now supports the following functionality:

- Recognition of multiple data files associated with the same hierarchy
- Automatically-generated backups of load results and other clean-up

loadHier can now simultaneously load multiple input files for a specific hierarchy. The extra input files need to be named with a secondary extension for **loadHier** to properly recognize them. For example, if the hierarchy to be loaded is **prod**, files such as **prod.dat.1** and **prod.dat.english** would be included with **prod.dat** (which is required to be present) when the **prod** hierarchy is loaded. However, the extra input files can *only* be loaded with the main input file. In other words, the **prod.dat.1** file cannot be loaded by itself in a separate **loadHier** call.

loadHier now also automatically generates a backup copy of a hierarchy's database, containing dimension arrays for that hierarchy, prior to performing the load for that hierarchy. If any type of error occurs during the hierarchy loading process, the hierarchy is restored from the backup copy.

One other change to the hierarchy loading process must be noted, however: the domain creation utilities *must* either use the default input files for all hierarchies or provide input files for all hierarchies. The domain configuration process can no longer use custom hierarchy input files for only some of the hierarchies and rely on RPAS to provide information for the other hierarchies.

Generation of Multiple Domain User Accounts – usermgr

In RPAS 11.1 the domain account management utility **usermgr** has the ability to generate multiple user accounts with a single invocation.

The usage for this particular scenario is as follows:

usermgr -d *domainPath* -addUsers *userFormat* -label *label* -group *grp*
 -begin *startValue* -end *endValue* {-admin}

Argument	Description
-d domainPath	Specifies the full path to the domain.
-addUsers *userFormat*	Directs the utility to generate multiple users. The *userFormat* string should contain exactly one variant of the C language style printing specification **%d**, which acts as a placeholder for the integral counter. This counter serves as the differentiator between the new accounts.
-label *label*	Specifies the label that the newly generated users will have.
-group *grp*	Specifies the group that the newly generated users will belong to.
-begin startValue	Specifies the starting value of the account generator.
-end endValue	Specifies the terminal value of the account generator.
-admin	Gives each newly generated account administrative privileges.

Consider the following example:

```
usermgr -d /home/rpas/thisDomain -addUsers “guest%03d” -label
“Guest” -group “Guests” -begin 10 -end 255
```

This command line will generate 246 unique accounts, with the first account being named **guest010** and the last account being named **guest255**.

Miscellaneous New Features & Notes

Improvements to Server-Side Logging

There are two changes to the **DomainDaemon** logging and one change to the **RpasDbServer** logging. Both daemons now look for the presence of an environmental variable named **RPAS_LOG_BACKUPS** to define the number of backup logs to keep when a new log file needs to be created. If this environmental variable is not set or is set to **1**, the default behavior is to rename the old log file to ***.log.old**. If this environmental variable is set to another value, however, it will determine the number of backup log files kept. For example, if this is set to **4**, then the four newest backup files will be kept. The names of the backup files will be in the form of **rpas*YYYYMMDDhhmm*b*NN*.log** (for **RpasDbServer**) or **Daemon*YYYYMMDDhhmm*b*NN*.log** (for **DomainDaemon**), where ***YYYY*** is the year, ***MM*** is the two-digit month, ***DD*** is the two-digit day of the month, ***hh*** is the two-digit hour, ***mm*** is the two-digit minute, **b** is just **b**, and ***NN*** is a number used to handle two log files created in the same minute. ***NN*** is **00** for the first file created in that minute, **01** for the second, and **02** for the third.

Also, **DomainDaemon** now checks the current time once every tenth of a second to determine if a new day has started, and starts a new log file if it has.

Special Values

'Special values' refers to values of cells that have a special meaning, typically not of any formal data type. Ultimately each special value type (such as **naval**, **ambig**, **novalue**, or **error**) will have a specific behavior defined for various actions (such as **display**, **calculate**, **aggregate**, or **spread**) that can use that special value. These special value behaviors will be defined separately for each measure.

RPAS 11.1 only supports the **naval** special value type and the **display** action. The behavior for **display**ing an **naval** can be **null**, **cellvalue**, or a valid data value for the target measure type.

If this special value is specified, it will define how the client displays all cells for that measure whose values are equal to the navalue at the aggregation level being displayed. If **null** is specified as the behavior for displaying navalue, the client will display an empty cell for date, numeric, and string data types, and it will display a grayed-out checkbox for Boolean data types. If **cellvalue** is the specified behavior, the client will display the array's actual NA value. If a valid data value is specified, such as **-99** for a numeric measure, the client will display that value for all NA-valued cells.

Password Encryption

A new RPAS client configuration utility is available that stores passwords in an encrypted format. Encryption is also used when storing and authenticating passwords on a web server (if using web deployment functionality).

Known Issues

The following are known issues in RPAS 11.1 and will be addressed in a future patch or release.

- The **propspread** and **passthrough** rule functions do not validate that an expression has the correct number of arguments; specifically, **propspread** requires that there is one more argument (on the right hand side) than output measures, and **passthrough** requires that the number of arguments be the same as the number of output measures.
- There is currently an issue when attempting to paste data into cells with read-only aggregates
- There is an error when using the **Back** button in the wizard process when no positions have been selected
- Cannot use "Export Sheet" in the RPAS client (File menu) if "All" calendar is selected as a rollup