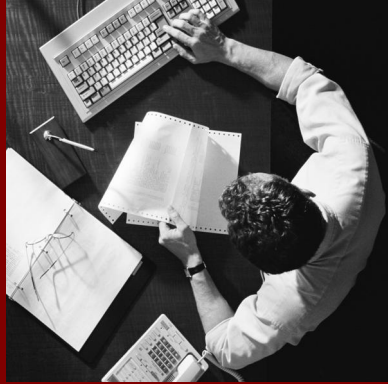


PUBLIC



SAP Library  
BPC Administration Guide

# SAP Business Planning and Consolidation 7.0

## SP03

version for the Microsoft platform

### Target Audience

- System Administrators
- Technology Consultants

Document version: 2.0 – January 30, 2009



© Copyright 2009 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.  
Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP AG  
Dietmar-Hopp-Allee 16  
69190 Walldorf  
Germany  
T +49/18 05/34 34 24  
F +49/18 05/34 34 20  
[www.sap.com](http://www.sap.com)

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP Library document classification: PUBLIC

#### Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressly prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP's Support Services and may not be modified or altered in any way.

Documentation in the SAP Service Marketplace

Documentation is available on SAP Service Marketplace at

<http://service.sap.com/instguidescpm-bpc>

# Table of Contents

Welcome to BPC Administration .....	7
Starting BPC Administration.....	7
Understanding action panes.....	7
Minimizing/maximizing action panes.....	9
Admin Console vs. Web Admin Tasks.....	9
Managing Application Sets.....	11
Setting application set status.....	11
Adding new application sets.....	11
Setting template version.....	12
Refreshing client-side dimension files.....	13
Deleting application sets .....	13
Changing application set descriptions.....	13
Deleting dimensions .....	13
Viewing who is online .....	13
Using Apshell, the sample application set .....	13
Managing Dimensions.....	17
About Managing dimensions .....	17
Adding dimensions .....	19
Copying dimensions.....	20
Processing dimensions .....	20
Exporting Dimensions .....	20
Scheduling dimension processing .....	21
Maintaining dimension members .....	21
Assigning dimensions to applications .....	23
Using dimension properties .....	24
Managing Applications .....	34
About working with applications .....	34
Adding new applications.....	34
About the Rate application.....	35
Copying applications.....	36
Changing work status settings for applications.....	36
Setting concurrent locks.....	37
Optimizing applications .....	37
Required dimensions in applications .....	38
Managing data audit.....	39
Creating a YTD storage application .....	39
Deleting applications .....	40
Managing Security .....	41

About security .....	41
Using SSL (HTTPS) Web Site Security .....	41
Setting up users.....	42
Setting up teams.....	43
Setting up profiles.....	44
Viewing security reports.....	52
Managing Business Process Flows .....	54
About business process flows.....	54
Copying a Business Process Flows .....	54
Resetting Business Process Flows .....	54
Reporting on Business Process Flows .....	54
Adding new business process flows.....	55
Managing Work States .....	59
Managing Business Rules .....	63
About business rules.....	63
Setting up a legal consolidation application set .....	63
About currency conversions .....	63
About intercompany eliminations.....	65
Adding business rules tables to applications.....	67
Defining rules .....	67
Using the Business Rule Library .....	79
Using Script Logic .....	82
Logic Assistant.....	82
Adding UDF functions to the Logic Assistant.....	82
The BPC MDX library.....	83
The BPC SQL library .....	95
Rules keyword reference .....	102
Managing custom menus .....	154
Adding custom menus.....	154
Managing Journals .....	156
Creating journal templates .....	157
Setting up journal security .....	158
Clearing journal tables .....	158
Limiting journal dimension member lists .....	159
Locking data for journal input only .....	159
Setting up Journals Application parameters .....	159
Defining journal validation rules .....	160
Managing Insight .....	162
Setting up Insight .....	162
Setting up the RootCauseEvent table .....	162
About the root cause/KPI association .....	163
Managing users.....	164

Importing/Exporting configurations .....	167
Setting system configurations .....	167
Viewing services and logs .....	170
Maintaining a BPC server .....	171
BPC Server Manager .....	171
Importing Non- BPC reports .....	171
Editing default messages .....	173
Database specifications .....	174
Web Admin tasks .....	176
Setting application set parameters .....	176
Setting application parameters .....	179
Deleting books .....	183
Enabling activity auditing .....	183
Reporting on activity audit .....	183
Setting up drill-through .....	185
Managing document types and subtypes .....	187
Tips and Troubleshooting .....	189
Microsoft Office tips and tricks .....	189
Setting client options .....	191
Updating application set information .....	192
Troubleshooting .....	192
Appendix A: Security Management in BPC .....	194
About task profiles .....	194
About member access profiles .....	194
Appendix B: Best Practices for Writing Logic .....	199
The Golden Rules .....	199
Rule 1: Stay away from MDX logic .....	199
Rule 2: Load in memory only the required data .....	202
Rule 3: Carefully select the "triggers" of your calculations .....	203
Rule 4: Keep the logic structure as compact as possible .....	204
Rule 5: Minimize the number of COMMITs .....	206
Rule 6: Minimize the number of GO statements .....	209
Rule 7: Keep in default logic only the calculations that are absolutely required to be performed in real time .....	209
Rule 8: Always review the LOG files .....	209
Rule 9: Avoid refresh-after-send in Excel .....	210
Rule 10: Run a stress test before going live .....	210
Appendix C: Script logic documentation updates .....	211
Ability to control SQL time-out .....	211
Calling the same stored procedure multiple times .....	211
Running the last stored procedure within a given COMMIT section .....	211
Added support of %WHEN% keyword in *REC instruction .....	212



Ability to disable CALC_EACH_PERIOD in one GO section .....	212
Ability to disable CALC_EACH_PERIOD in one CALC_ORG or CALC_DUMMY_ORG section .....	212
Index .....	213

## Welcome to BPC Administration

Business Planning and Consolidation (BPC) Administration is a tool that allows administrators to perform setup and maintenance tasks for BPC client applications.

### Starting BPC Administration

BPC Administration has two interfaces: A client application and a web interface. The Administration action pane lists the available tasks for both interfaces.

To start BPC Administration

1. Do any of the following:
  - Open a browser and type `http://<server name>/osoft`, where `<server name>` is the name of your BPC server.
  - From the Windows Start menu, select **SAP > BPC**.
  - From your Windows Desktop, click the BPC icon.
2. From the Launch page, select BPC Administration.
3. From the Administration action pane, select the desired task.

### Understanding action panes

"Action panes" provide quick access to tasks that are relevant to your current context. In addition, you can control your current view, view login information, jump to other programs, such as BPC for Excel and BPC for Word, and launch context help from the action pane.

A BPC action pane consists of the following sections: Navigation, Session Information, Task Selection or Input Fields, Available Interfaces, and See Also. These sections are described below.

#### Navigation

The Navigation section provides a Back, Forward, and Home button. The Back button brings you to the previous action pane; The Forward button brings you to the next action pane (only if you previously pressed Back); and the Home button brings you to the first action pane for the current process.

#### Session Information

The Session Information section contains login and current view information.

##### Login

This area shows the active user ID and application set. To change the application set, select the link, make your selection, then click OK.

##### Current View

This area controls which members, parent members, or top-level members are represented in the active report or input schedule. It allows you to dynamically change the current view by entering various members, or selecting them from the Member Lookup. You can expand the section to view all available dimensions and members to which you have access.

#### Sample Session Information

##### Sample Session Information

Login:

ID: HHalter

Appset: ApShell

CV:

Application: FINANCE

```
Account: ExtSales
RptCurrency: USD
Time: 2005.Total
Category: ACTUAL
IntCo: All_InterCo
DataSrc: Input
Entity: Worldwide 1
```

### Task Selection or Input Fields

The Task Selection or Input Fields section displays either a list of tasks, or input fields that require action. If a task list displays, clicking a link will perform a task, open a dialog box, or open another action pane.

If input fields display, you must take action by completing the required fields. If you make a mistake, you can always click the Back button at the top of the action pane.

### Sample Task Selection section

```
Teams
Manage Teams - All Users
Security Tasks
Process security
Team Tasks
Add Teams
All Users Tasks
Modification Team
Delete Team
```

### Available Interfaces

The Available Interfaces section contains links to other components: BPC Web, BPC for Excel, BPC for Word, and BPC for PowerPoint. You can expand or collapse this section to see more or less, respectively, of the Task Selection/Input Field section.

Click a link to start the desired program.

### Available Interfaces Sample

```
Available interfaces:
BPC Web
BPC for Excel
BPC for Word
BPC for PowerPoint
```

### See Also

The See Also section contains a link to the context help topic associated with items in the Selection or Input area in the action pane. Click the link to view field-level help, an overview, and a how-to.

You can expand and collapse this section to see more of the Task Selection/Input Field section.



Sample See Also Selection

See Also  
Admin Help

## Minimizing/maximizing action panes

You can minimize action panes to display more of the BPC Administration interface, and then maximize it when you need to navigate or perform a task.

To minimize an action pane

- From a displayed action pane, click the minimize button in the upper right corner.

To maximize the action pane

- Click View action pane from the BPC for Office toolbar.

## Admin Console vs. Web Admin Tasks

BPC Administration has two interfaces: the Client Admin Console, which requires a small footprint, and the Web Admin tasks, which is a zero-footprint client.

The console client is a Microsoft Explorer-like window where you manage such items as application sets, applications, dimensions, business rules, and business process flows.

The browser client allows you to control application set and application properties, and maintain BPC Web.

The following tables describe the tasks contained in each.

### Admin Console

The Admin Console contains links to tasks you can perform in the Admin client. You must have access to the Admin client in order to perform these tasks.

Task	Description
Manage Security	Opens the Security node where you can define and manage BPC security.  See Managing security
Manage Application Sets	Opens the Application Set node where you can add, delete and manage application sets.  See Managing application sets
Manager Applications	Opens the Application node where you can add, delete, and manage and delete applications.  See Managing applications
Manage Dimensions	Opens the Dimension Library node where you can add, delete, and manage dimensions in the application set.  See Managing dimensions
Manage Business Rules	Opens the Business Rules node where you can define and manage business logic.  See Managing business rules
Manage Business Process Flows	Opens the Business Process Flows node where you can manage business process flows.  See Managing business process flows

## Web Admin Tasks

Task	Description
Set AppSet Parameters	Opens the Application Set Parameters page, where you can control application set-level settings.  See Setting application set parameters
Set Application Parameters	Opens the Application Parameters page, where you can control application-level settings.  See Setting application parameters
Manage Books	You can manage the library of books stored on BPC Web by removing them.  See Deleting books
Manage Document Types	See Managing document types
Manage Document SubTypes	See Managing document subtypes
Manage Activity Audit	Opens the Manage Activity Audit page where you can set up activity auditing.  See Enabling activity auditing
Manage Data Audit	Opens the Manage Data Audit page where you can set up data auditing.  See Managing data auditing
Manage Insight	See Managing Insight
Publish reports	
Edit Drill Through Table	Opens the Edit Drill Through Table page where you can set up the drill through table.  See Editing the drill through table

## Managing Application Sets

An application set is a SQL database that stores all the data from each individual application in the set.

Each application is comprised of an individual Analysis Services database (or application). Each application contains the metadata that controls the data in the application set. Applications can share one or more dimensions with other applications within the set.

### Setting application set status

You can check or change the status of an application set. The available statuses are "Available" and "Unavailable." Some administration tasks automatically set the application set to unavailable. You can also make it unavailable manually. See [Setting the status to Unavailable](#).

The system will automatically be set back to Available when the process is complete, or you can manually make it available. See [Making an application set available](#).

#### Setting the status to Unavailable

You can manually set the application set to Unavailable.

Note that when an administrator logs on to an application set that is "Not Available" the following message is displayed:

Application set: <ApShell> is not available now.  
Do you want to continue?

You can select Yes to proceed with logon, or No to cancel. This message is just a reminder. You can still work in BPC for Excel so that you can do testing while doing application maintenance. (An end user does not have this option.) However, remember that if an application has been modified and not fully processed, you can experience errors or be looking at old application information.

#### To set the status to Unavailable

1. From the Admin Console, select the application set you want to make unavailable.
2. Click Set application set status.
3. Select the Not Available radio button.
4. In the Message box, enter informative text that will be displayed when users try to logon. We recommend including the current date and time so that users can see that the message is current, and to give an estimated time when they can log back on again.

#### Setting the status to Available

When you have finished updating the application, and are ready to set it back to "Available," you can select the Available radio button. System administrators can set the status of an application set, making it available or not available to users. This is useful as a way of preventing access to an application when you take it offline for maintenance, for example.

#### To set the status to Available

1. On the BPC Administration main screen, go to the hierarchical list in the left pane and click the plus sign next to the application name at the top level. The Manage Application Sets action pane is displayed.
2. Under the action pane heading Application Set Tasks, click Set application set status. The Set Application Set Status action pane is displayed.
3. Using the radio buttons, specify if you want the application set to be Available (online) or Not Available (offline). If you specified Not Available you may enter a text message that will appear when users try to access the offline application set.
4. Click Update Application Set Status to set the application set status.

### Adding new application sets

Administrators add a new application set by copying selected information from an existing one, such as the ApShell application set provided with BPC. You can copy security settings, database data, collaboration data, and/or journals.

---

**Note:** We recommend that you add a new application set only when initially setting up BPC for a company, or when expanding the use of BPC as a result of a license upgrade. See Editing system administrators.

---

Adding a new application set does the following:

- Copies the application set Webfolders to the new application set.
- Creates a copy of the relational database (i.e., a SQL database).
- Creates a SQL database login role for the new relational database.
- Creates a copy of the Analysis Services database.
- Copies the published books, by copying the appropriate FileDB subdirectories.
- Sets subdirectory security on the Webfolders subdirectories.
- Sets configuration parameters.

After the copying is complete, you can begin to modify the default applications and add new ones, and assign users to the application set.

To add a new application set

1. From the Admin Console, log on to the desired application set. The Manage Application Sets action pane is displayed.
2. Under the action pane heading Application Set Tasks, click Add a new application set. The Add a New Application Set Step 1 of 2 action pane is displayed.
3. Enter the new application set name and description. Use the drop down menu to select the existing application set upon which you want to base the new one. Click Next. The Add a New Application Set Step 2 of 2 action pane is displayed.
4. This action pane displays the records used in the existing application set you specified in the previous step. Using the check boxes, select all the records that you want to copy to the new application set. These include:
  - Database records
  - Data States (as per the business process flow)
  - Database Security
  - Business process flows
  - Content library
  - Live Reports
  - Journals
5. When all the desired records have been selected, click Add New Application Set. The new application set has been created.

## Setting template version

If you have made changes to any of the dynamic templates for reports or schedules, you can force an update of template files by changing the template version. When you force an update, clients that log on to the application set will be updated with the new template(s).

This topic describes how to update the template version from the Admin Console. You can also update the version in the TEMPLATEVERSION field on the Setting Application Set Parameters page. See Setting application set parameters.

To set the template version

1. From the Admin Console, select the application set for which you want to reset the template versions.
2. From the Manage Application Sets action pane, select Set template version.
3. Increment the number by 1. For example, if the current version is 10 (ten), enter 11 (eleven).
4. Click Update Template Version.

## Refreshing client-side dimension files

The only time you need to use the Refresh client-side dimension files option is when you have processed your application outside of BPC, in Analysis Services. Otherwise the client-side dimension files are automatically updated when you process your application in Validate and Process Members or in Optimize application.

## Deleting application sets

An administrator can delete application sets.

To delete an application set

1. From the Admin Console, select the application set name at the top of the tree.
2. From the Delete application sets action pane, select one or more application sets to delete.
3. Select Delete Selected Application Sets.
4. From the confirmation message, click Yes.

## Changing application set descriptions

You can change the descriptive text associated with an application set. You might want to change the description if you have made a significant change to the application set, such as adding a new application that performs new functions for your business.

To change application set descriptions

1. On the BPC Administration main screen, go to the hierarchical list in the left pane and click the plus sign next to the application name at the top level. The Manage Application Sets action pane is displayed.
2. Under the action pane heading Application Set Tasks, click Add a new application set. The Add a New Application Set Step 1 action pane is displayed.
3. Enter the new application set description and click the green arrow. The Add a New Application Set Step 2 action pane is displayed. Click the green arrow. The new application set description has been saved.

## Deleting dimensions

Administrators can remove dimensions from an application set.

To delete a dimension

1. From the Admin Console, select Dimension Library.
2. From the Manage Dimensions action pane, select Delete dimension.
3. Select one or more dimensions to delete, then click Delete Selected Dimension(s).
4. Select Yes in the confirmation message box.

## Viewing who is online

Administrators have the ability to monitor administrator activities.

To see who is online

1. From the Admin Console, log on to the application set and application that you want to monitor.
2. Select Who is online from the Manage applications action pane.
3. When done viewing the Who is online window, select OK.

## Using Apshell, the sample application set

This section contains an overview of the ApShell application set. Apshell is not only used as a sample application set, but also as the repository for BPC system information and defaults. Because ApShell is the repository for system information and defaults, it gets overwritten when you upgrade to a new version of

BPC, so you should always start your application set by creating a copy of ApShell. See Adding new application sets.

ApShell does not contain any data and has only limited metadata (dimension members). The dimension metadata is limited to default members in most dimensions (the Time dimension has real data) so that the application set will work when an administrative task is run.

ApShell contains all of the required components needed to build a functioning application set:

- Sample best practices for a BPC Web implementation, which includes report wizards, system logic, and support for Journals
- Required dimensions for each application
- Standard report and input schedule templates
- Data Manager packages and sample transformation files

ApShell is only an example. Because it is a shell, it needs to be "filled in" with publications and reports in order to become a fully-functioning AppSet.

### ApShell applications

The ApShell application set contains several sample applications. Two of those applications are Finance and Rate, which contain most of the functionality you will need to get started building your own application set. This topic describes the application sets.

If you need to build more complex application sets, for example, with legal reporting capabilities, you should start with LegalApp and LRate.

Finance is an application whose type is financial. Rate is a rate-type application that supports the Finance application. ApShell does not contain any generic (non-financial) applications. See Managing applications.

### The Finance application

Finance is a multi-currency financial application and has the Intercompany Eliminations option enabled. Because it is multi-currency, it is attached to the Rate application (at design time) for currency rates.

Finance has the following dimensions:

Dimension name	Dimension type
Account	Account
Entity	Entity
Time	Time
Category	Category
IntCo	Intercompany Eliminations
DataSrc	User defined Used for tracking the source of input data.
RptCurrency	Currency Finance is a financial-type application, so a currency-type dimension is required. See Managing dimensions.

### Advanced rules

The ApShell Finance application has standard default advanced rules that run currency translation. The default formula includes the FXTRANS.xls advanced rule, which sets up currency translation by running the system TRANSLATE\_LDI formula.

The Finance application default advanced rules also have a line commented out that runs the ROLLTOBS function. This function rolls differences from prior periods into your balance sheet accounts. See The BPC SQL\_Library.

#### The Rate application

The Rate application contains currency translation rates and Intercompany elimination logic. The rate application was created first, then the Finance application was created and tied to the Rate application when the multi-currency option was chosen.

Rate has the following dimensions:

Dimension name	Dimension type
Rate	Account Contains currency rates.
RateSrc	Entity
Category	Category
Time	Time
InputCurrency	Currency Stores currency rates for currency translation.

#### Advanced formulas

The ApShell Rate application has a standard default advanced logic that calculate divide or multiply rates for currency translation.

The logic requires the following properties and members:

- The currency-type property must have a property named MD whole value is either "M" or "D", which stand for Multiply currency and Divide currency. In the Apshell Rate application, the currency-type dimension is named InputCurrency.
- The Entity-type dimension must have two members named RateInput and RateCalc. In the ApShell Rate application, the Entity-type dimension is named RateSRC.

#### ApShell reports and input schedules

ApShell contains a set of sample reports and input schedules. They are located in the directories specified below. You can put your own custom reports and schedules in these folders and add your own folders to further organize the directories.

When a user selects to open an existing report in the Reporting & Analysis Options action pane in BPC for Excel, it opens the Reports directory. For the ApShell application set and Finance application, this is: `x:/BPC/data/Webfolders/ApShell/Finance/eExcel/Reports/`.

There are several folders in the Reports folder. Some are highlighted below:

Book Reports	This folder contains sample reports to be used in book publication. All reports to be used in book publication must be stored in this folder. Please see BPC for Excel help for information on book publication. This is an optional folder and is provided as a best practices example for organizing your BPC folders.
HQ, Manager	These folders contain sample reports. These are optional folders and are provided as a best practices example for organizing your BPC folders.
Wizard	This folder contains report templates, Hot Analysis reports, and custom menu templates. The Wizards folder is a required folder. Folders inside this folder, such as HotAnalysis and ProcessMenu, are provided for organizational and best practices purposes.

---

	For example, the ProcessMenu folder's contents could be placed directly in the Wizard folder, but it would be hard to differentiate custom menu templates files from regular dynamic template files.
--	--

---

When a user selects to open an existing schedule in the Data Input Options action pane in BPC for Excel, it opens the Input Schedule directory. For the ApShell application set and Finance application, this is: `x:/BPC/data/Webfolders/ApShell/Finance/eExcel/Input Schedules/`.

There are two folders in the Input Schedules folder. These are:

HQ, Manager	These folders contain sample reports. These are optional folders and are provided as a best practices example for organizing your BPC folders.
Wizard	This folder contains the dynamic input schedule templates.



## Managing Dimensions

Working with dimensions involves creating new dimensions, defining members, assigning properties, and other tasks. BPC gives you a graphical view of your dimension members.

### About Managing dimensions

You manage dimensions in an application set using the Dimension Library. The dimensions added to an application set's Dimension Library can be added to one or more applications in the application set.

#### Required dimension types

Every application is required to have one dimension of each of the following types: Account, Category, Entity, Time, Currency, and Intercompany. While a Currency-type dimension is required to be present in all application sets for validation purposes, it does not have to be part of any applications within the application set.

In addition, you can create user-defined dimension types as well, which are referred to in the system as U1, U2, U3, and so on.

Note that an application set can have multiple dimensions of one type, but each application within the AppSet can have only one of each of the required types. For example, the application set might have entity type dimensions named *EntityB* and *EntityF*, with *EntityB* used in the Budgeting application and *EntityF* in the Forecasting application.

The following table describes the required dimension types. Additional dimension types are described in Adding dimensions.

Dimension type	Type ID	Definition
Account	A	Contains your chart of accounts. See Account dimension required properties
Category	C	Contains the types of data you are going to track, such as Actual, Budget, Forecast, etc. You can set up categories to store versions, such as BudgetV1, BudgetV2. See Category dimension required properties and Setting up a legal consolidation application set
Entity	E	Contains the business units that are used to drive the business process. Depending on your application design, the Entity type can be an operating unit, a cost center, a geographic entity, etc. This dimension will be used to supply the members that are used in the Status Board approval process. See Entity dimension required properties
Time	T	Contains the time periods for which you want to store data. For a given Time dimension, you can create an unlimited number of Time hierarchies with different LEVELS. <hr/> Note: We recommend that you create no more than three time hierarchies for a dimension. <hr/> The Time hierarchies can have different starting points, configured using the ISBEGINNING property (for instance, you may want one hierarchy that follows a calendar year and another one that represents the fiscal year). See Time Dimension required properties.

Dimension type	Type ID	Definition
Currency	R	Contains the currency rates for all currencies in which your company does business. Required in the application set, but not in each application.  See Currency dimension required properties and Setting up a legal consolidation application set
Intercompany	I	The Intercompany dimension type contains the Intercompany codes for the entities. It is required for ICmatching and Legal applications.  See Intercompany dimension required properties

#### About the Measures dimension

In addition to the dimensions listed above, BPC also requires a dimension called Measures. This dimension is automatically included in all BPC application sets. It is not listed as a dimension-type when you are creating new dimensions, but is displayed in the current view. Measures allow you to change the view of your data. You can view Periodic, Half-year-to-date (HYTD), Quarter-to-date (QTD), Year-to-date (YTD), Month-to-date (MTD), and Week-to-date (WTD) views of your data using the Measures dimension.

#### About securing dimensions

You can have a mixture of secured and unsecured dimensions in an application. Securing dimensions allows you to control which users (or teams) have read only or read/write access to dimensions and their members. You only need to define a dimension as secured if you want to assign read/write access to a dimension. Unsecured dimensions are accessible to all users.

When defining access to secured dimensions, be sure to define access for all secured dimension in the application. Failure to do so will result in an incomplete security profile for the user or team. If this happens, users of a team will not be able to access the application at logon. For example, the application *Budget2006* has three secured dimensions, Entity, Account and Category. When defining member access for the *Analyst* team you must define secured access for all three dimensions. If you define access for one or two, the profile will be incomplete.

#### Setting application-level dimension security

You can define a dimension as secure for individual applications. If you want to restrict users from accessing an entire application you must have at least one dimension secured in the application. For information on setting security on dimensions, see [Assigning dimensions](#).

#### Setting dimension member-level security

In addition to defining dimensions as secure, you can assign read/write access to members within the dimension using member access profiles. Since by default, users do not have access to any members of a secured dimension, member access profiles must be set up for the users you want to give read and/or write access. For more information on member access profiles, see [Adding member access profiles](#).

#### Maximizing levels and properties

When you are creating dimensions for your application set you should be aware of the following maximums for any one dimension.

- The maximum number of fields in a table (a dimension = 1 table) is 1024.
- The maximum record size is 8064 bytes (a record = 1 row in a table)

The two maximums above relate to the underlying SQL database in which BPC information and data is stored. They need further explanation as to how they relate to a BPC dimension. In BPC a field equals a property. So you can have up to 1024 properties in a dimension. One other factor that has an impact on the actual number of properties you can have in a dimension is the number of levels you have defined. SQL Server creates a set of properties for each level within the dimension. For example, you have 10 properties and three levels in your dimension, your total number of fields is 30. The second limiting factor is the size of the record. To determine record size you have to figure out the number of bytes (a byte equals a character) in each level. Since levels are repeated you only need to figure out the number of bytes in the first level and then multiply that number by the number of levels. To come up with the total number of bytes for a level you simply add up the field size for each field and multiply it by 2 (1 character = 2 bytes). The BPC field sizes are as follows:

- SEQ (hidden) = 6

- ID = 20
- EVDESCRIPTION = 50
- CALC = 1
- SCALING = 2
- ALL User defined properties = 10

You have the ability to control the length of the user-defined properties to meet your needs. For more information on changing the length of a property, see [Adding properties to dimensions](#).

The real variable is the number of additional properties a dimension contains, as it has a direct impact on both the record size and the number of fields in a table. For more information on the maximum and recommended number of dimensions, see [Database specifications](#).

## Adding dimensions

You add dimensions to the Dimension Library of an application set to make them available for your applications. You can use any name you want for a dimension with the following exceptions:

- You cannot name a dimension after an existing BPC database table.
- You cannot use these reserved names.

To add a dimension

1. From the Admin Console, select Dimension Library under the appropriate application set.
2. From the Manage Dimensions action pane, click Add a new dimension.
3. Enter a name for the dimension. The name can be up to 20 characters and cannot contain single quotation marks ('), double quotation marks ("), backslashes (\), or ampersands (&).
4. Enter a description for the dimension, then click Add a new dimension Step 2 of 3. The description can be up to 50 characters and cannot contain a double quotation mark (").
5. Select the dimension type from the Dimension Type drop down list. The dimension type controls the behavior of the dimension and the default properties included in the dimension. See [About managing dimensions](#). The type can be:
  - A - Account
  - C - Category
  - E - Entity
  - D - Data Source
  - I - Intercompany
  - R - Currency-type dimension
  - S – Subtable (Used to break down the account activity or flow. For example, some accounts, like Fixed Assets, have a Subtable dimension containing Opening, Additions, Deletions, Transfers and Ending Balances. The Subtable type dimension is important for writing business rules that require currency translation amounts to be calculated by account. Since the Subtable information can be used for multiple accounts, it requires its own dimension.)
  - T - Time
  - U - User defined
6. Based on your selection in step 5, select a reference dimension, then click Add a new dimension step 3 of 3.
7. Review the dimension properties, and add new ones if desired. See [Assigning properties to a dimension](#).
8. Select Execute.

## Copying dimensions

You can create a new dimension by copying an existing one.

You cannot use any of the following terms as dimension names:

App, AppAccess, AvlObject, CategoryAccess, CollabDoc, CollabIcons, CollabRecipient, CollabSupport, CollabType, DBVERSION, Defaults, DesktopStyleDef, Dimension, DrillDef, DTIParam, Function, Formula, Group, InvestParam, MemberAccess, MessageLog, Packages, PageDef, Permission, PublishedBooks, Rate, ReportParam, SectionDef, Status, StatusCode, TaskAccess, User, UserGroup, UserPackages, UserPovDef, WebContents, SOURCE, SIGNEDDATA

In addition, you cannot name a dimension after an existing BPC database table.

To copy a dimension

1. From the Admin Console, expand the application set from which you want to copy a dimension.
2. Expand Dimension Library, then select the dimension you want to copy.
3. Enter a name for the dimension in the Dimension text box. The name can be up to 20 characters and cannot contain a space ( ), single quotation mark ('), a double quotation mark ("), or a backslash (\).
4. Enter a description in the Description text box, then click OK. The description can be up to 50 characters and cannot contain a double quotation mark (").
5. Select the dimension in the tree on the left to modify its members or properties. See Adding members to dimensions and Maintaining dimension properties.

## Processing dimensions

After you add a new dimension or make changes to a dimension, you must process it. Processing occurs automatically after you create or modify a dimension, so you only need to perform this procedure when you need to manually process dimensions. Modifying the member table requires that you manually process dimensions.

When you process a new dimension, the system constructs a map of the dimension in Analysis Services. This sets up the entire hierarchy, so that the dimension is easy to find upon querying the application. In addition, processing a new dimension sets up a member table with some default properties. For information about adding members to a dimension, see Adding members to dimensions. When you process an existing dimension, the system stores the changes in the database.

Certain changes to a dimension forces the system to process the entire application. For information on processing applications, see Processing applications.

You can schedule the processing of dimensions. See Scheduling dimension processing.

To process dimensions

1. From the Admin Console, select Dimension Library.
2. From the Manage Dimensions action pane, select Process dimensions. The Process Dimensions dialog box is displayed. (If you selected one or more dimensions under Dimension Library, those dimensions will be preselected.)
3. Select the dimensions you want to process. To select all the items, press Select all dimensions.
4. To perform a full process, select the Full Process check box. A full process processes all the members in the dimension. See Full vs. incremental processing.
5. When all the desired items are selected, click OK to complete the dimension processing.

## Exporting Dimensions

You can export dimensions from BPC to an MS EXCEL workbook on a central repository. You can use the exported workbook to make changes to the dimension's members. These changes take effect in BPC once the dimensions are processed.

See Processing Dimensions.

To export dimensions

1. From the Admin Console, select Dimension Library.
2. From the Manage Dimensions action pane, select Export dimensions.

---

**Note:** If the number of exported members is greater than MS EXCEL's worksheet row limitation, the workbook will create additional worksheets as needed.

---

## Scheduling dimension processing

You can validate your changes to dimensions, and schedule the processing for another time. The schedule can be set up to run once or on a recurring basis. The validation is done in the Admin Console, but the scheduling is done in Data Manager.

See Scheduling dimension member processing in the Data Manager Help.

To schedule dimension processing

1. From the Admin Console, select Dimension Library under the appropriate application set.
2. From the Manage Dimensions action pane, select Process dimensions. The Process Dimensions dialog box is displayed.
3. Select the check box for scheduling dimension processing, then click OK.
4. Follow the procedure for Scheduling dimension member processing in the Data Manager Help.

## Maintaining dimension members

You maintain dimension elements by adding and modifying members associated with a particular dimension.

About maintaining dimension members

You maintain dimension members by adding and modifying members associated with a particular dimension.

You add members to dimensions based on your business needs. For example, your company may open a new office, and that office's financial information needs to be reflected in the Entity, Category, and Currency dimensions.

When you create a new dimension, the system creates a template for that dimension. Based on the dimension type, the template contains a set of predefined properties, such as ID, NEWID, DESCRIPTION, PARENTH1, PARENTH2, CURR and OWNER.

You can also add new members, new properties, and assign dimension formulas that calculate and store information based on member values. In addition, please note the following:

- If you are starting from the sample application set, ApShell, and editing a dimension member for the first time, you must delete all of the sample members. You can do this using Excel functionality.
- The dimension member sheet is contained in an Excel worksheet that you can modify using Excel functionality. Therefore, it is possible to change the name of the sheet. However, you must not change the name of the first sheet from *Member*, or it will not work correctly.
- Some properties are required, depending on the dimension you are editing. See Adding properties to dimensions.
- Special characters and spaces are not supported in member IDs. Underscores (\_) and periods (.) are allowed.
- The following are reserved names and cannot be used as member names:
  - AUX
  - COM1, COM2, COM3, COM4
  - CONS
  - LPT1, LPT2, LPT3, LPT4
  - PRN

- The member ID field is limited to 20 characters.

### Adding members to dimensions

You can add a single member or multiple members to a dimension. Typically, when setting up your system, you load bulk members into a dimension by entering the metadata from an existing source. For example, using Excel, you can open an existing spreadsheet, and use copy/paste to transfer the members from the source file to the member sheet. With subsequent additions, you can type a member and its properties in the appropriate row.

This topic describes the process of adding a single member at a time.

For a list of reserved member names, see [About maintaining dimension members](#).

### To add members to dimensions

1. From the Admin Console, select Dimension Library under the appropriate application set.
2. Select the dimension for which you want to add a member, then select Maintain dimension members.
3. In the first empty row, type the name of the member and properties in the appropriate columns, or paste the contents of the clipboard into the sheet if doing a bulk load.
4. After you have entered the desired members, click Process Dimension. Make sure that Process members from member sheet is selected.
5. If you made changes to any rules in the dimension, such as added a new formula, you must validate and save those rules.

### Changing member names

You can rename members using the NEWID property. NEWID is a required property in all dimensions. Note that if you type the new name in place of the existing name in the ID column, you will receive errors. You must use the NEW ID column.

For a list of reserved member names, see [About maintaining dimension elements](#).

### To change a member name

1. From the Admin Console, select Dimension Library under the appropriate application set.
2. Select the dimension for which you want to add a member, then select Maintain dimension elements.
3. Find the member you want to rename, and enter the new member name in the NEWID field. (Remember that the ID field is limited to 20 characters.)
4. After you enter the new member names, validate and process those members. See [Validating and processing members](#).

### Validating and processing members

After you create or make changes to members in a dimension, you must validate and process the members in order to have the changes reflected in your application and throughout BPC. The validation process reviews the members for errors, and allows you to build each dimension and process the database (the Analysis Services application).

### To validate and process members

1. From the Admin Console, select the dimension (under Dimension Library) that contains the member you modified, and want to validate and process.
2. From the Manage Dimensions action pane, select Process dimension.
3. In the Process Dimensions dialog box, select the Process members from member sheet check box.
4. Select the Full Process check box to run a full process on the dimension(s), then click OK. (If you do not select this and a full process is required, BPC detects it and runs the full process anyway.) See [Full vs. incremental processing](#).

### Full vs. incremental processing

When processing dimensions and members, BPC decides what type of processing (Full vs. Incremental) to perform based on the changes that have been made to the dimension. At any time, you can override the system-detected process option by forcing a full process of a dimension.

**Full:** The system performs a full process of a dimension when a property is added to the dimension, a member is inserted in the middle of the worksheet, and when a change is made to the parent/child relationships (if the dimension has a hierarchy).

**Incremental:** The system performs an incremental process of the dimension when changes are made that do not affect the structure of the dimension, like editing a formula, adding a property value, or adding base-level members to the end of the worksheet. If any of the applications in the Process Application list are selected, and the dimension only needs incremental processing, applications WILL NOT be processed if they do not need processing.

### Viewing member reports

Member reports show your hierarchies in a tree structure. A member report is added as a new sheet named Print in your dimension workbook. After you generate a report, your workbook is placed in print preview mode for the member report.

For more information about member reports, see [Adding Members to Dimensions](#).

### To view a member report

1. From the Admin Console, log on to the appropriate application set.
2. Under Dimension Library, select the dimension that contains the member for which you want to view a report.
3. Select Maintain Dimension Elements from the action pane.
4. Select the hierarchy or hierarchies you want in the report.
5. In the Select Description section, select which properties you want in the report.
6. Click the OK button to generate the member report.
7. Click the Print button on the print preview screen to print the report, or click the Close button. The new sheet containing the report, named Print, is visible in the workbook.

## Assigning dimensions to applications

After you add dimensions to the Dimension Library, they are available as shared dimensions in the application set, and can be assigned to applications. You assign them to applications in order to make the data from those dimensions available in your application.

### About assigning dimensions

After you create dimensions, they are available as shared dimensions in the application set, and can be assigned to applications.

Each dimension has a "type" associated with it. For example, type "A" is an Account-type dimension. Each application must have at least one of each of the following dimension types assigned to it: Account (A), Category (C), Entity (E), and Time (T). If you want to assign additional user-defined dimensions (type U) to an application, they are assigned sequential numbers: U1, U2, etc. For more information on dimension types, see [About managing dimensions](#).

Please note the following:

- Use caution when adding or removing dimensions from an application that already contains data. We recommend that you only assign dimensions to new applications that do not contain data. If you do add a dimension to an application with data, the system finds the first base member (alphabetically), and loads it into the fact tables. This means that all the application's data will be written to that member. (To move the data to a different member, you can run a DTS Move package. See [Data Manager Help](#).) Removing a dimension from an application with data will cause a loss of data.
- Before assigning a dimension to an application, make sure it has at least one member defined, and you have validated and processed the dimension. Otherwise, an error message is displayed when you attempt to save the application, stating that the dimension has no members. See [Adding members to dimensions](#)

- In the rare case that you have two application sets on the same server, and administrators attempt to save applications from multiple application sets at the same time, one or both of the administrators may receive an error message that they do not belong to the OLAP Administrator group. To work around this issue, try to save applications in one application set at a time.

### Assigning dimensions

Use this procedure to assign a dimension to an application. Note that each dimension has an associated *type*. Each application must have at least one dimension of the four required types, Account (A), Category (C), Entity (E), and Time (T). For more information on dimension types, see Adding dimensions.

You can also determine if dimensions are secured and/or read/write using this procedure.

#### To assign a dimension to an application

1. From the navigation pane in the Admin Console, select the application to which you want to assign dimensions.
2. Select Modify Application from the action pane.
3. The Shared Dimensions area shows all the dimensions for the application set. Select one or more of those dimensions and click one of the arrow buttons ('<', '>', '<<', or '>>') to move them to the Application Dimensions area.
4. For any given dimension assigned to the application, you can secure the dimension by clicking on the dimension and the clicking on the Secured button at the bottom of the window. A Y is displayed in the Secure column for secured dimensions. (Click the Secured button again to unsecure a selected dimension.)
5. For any given dimension assigned to the application, you can grant read/write access to the dimension by clicking on the dimension and clicking on the R/W button at the bottom of the window. If you grant read/write access, the dimension automatically becomes secured. If you have a secured dimension and Read/Write access, disabling Read/Write will not unsecure the dimension automatically; you will have to unsecure the dimension additionally.

### Removing dimensions from applications

You can remove dimensions from applications. This is not recommended if this application already contains data.

#### To remove a dimension from an application

1. From the navigation pane in the Admin Console, select the application to which you want to remove dimensions.
2. Select Modify Application from the action pane.
3. From the Application Dimensions area, select one or more dimensions to remove, then use the arrow buttons ('<', '>', '<<', or '>>') to move them to the Shared Dimensions area.
4. When done removing dimensions, click the Modify Application link from the action pane.

### Using dimension properties

Dimension properties are categories that are assigned to dimensions. Many of the categories are generic, such as *ID* and *EvDescription*, and others can be unique to a dimension, such as *Reviewer* and *Scale*. You use these categories to define the behavior of members within the dimension.

#### About dimension properties

Based on their assigned type, dimensions are assigned default properties, some of which are required. You can add more properties to further customize your dimension members. The following table describes several of the general dimension properties. These are required for each dimension.

Property	Description
ID	A user-defined ID for the property. Each ID must be unique within the dimension. Although you can use the same ID in different dimensions, we highly recommend



Property	Description
	<p>that you use unique IDs across dimensions. If you need to duplicate an ID in multiple dimensions, consider making it unique by prefacing the IDs with a letter representing its dimension.</p> <p>ID can be up to 20 characters and can not contain the following special characters:</p> <ul style="list-style-type: none"> <li>• Single quotation mark (')</li> <li>• Double quotation mark (")</li> <li>• Backslash (\)</li> </ul> <p>When renaming a dimension after data has been entered into the application, you must add a property called 'NEWID'. Simply typing the new name over the existing name in the ID column will result in errors. When you validate and process the dimension, the old ID is renamed to the NEWID, and data and Content Library documents from the original ID will be assigned to the NEWID. A log sheet is kept of the ID changes. Note: MDX formulas that have been defined for any dimensions are not updated with the NEWID; they must be updated manually.</p>
EVDESCRIPTION	<p>A user-defined description for the property. EVDESCRIPTION has the following constraints:</p> <ul style="list-style-type: none"> <li>• Can be up to 50 characters</li> <li>• Descriptions do not need to be unique</li> <li>• Cannot contain " (double quotation)</li> </ul>
GROUP	<p>A user-defined identifier for providing additional filtering options. By grouping members with your dimensions you can sort and filter with more flexibility. Values are optional.</p>
STYLE	<p>A user-defined identifier for denoting different formatting applied using Excel's conditional formatting. Values are optional. See Creative use of properties.</p>

---

**Note:** There are two system-generated properties: HLEVEL and CALC. HLEVEL denotes the member's hierarchical level. CALC indicates whether or not the account is calculated by means of a formula or is at a parent level. The property values for these items can be viewed in the Member Selector in BPC for Excel.

---

### Adding properties to dimensions

By assigning properties to dimensions, you can implement very powerful features in your reporting, member lookup, formulas, portable data selections, Data Manager selections, and so on.

You can filter on properties in many places in BPC. For example, if you want to be able to easily select entities by geographic region, you simply add a Region property and enter a region value for each entity. Then you can filter and sort by region, apply account logic by region, or define a report format based on region.

BPC requires various properties depending on the dimension, as described in the 'required properties' topics. You can also assign additional properties for your business needs.

The default number of characters for user defined properties is 10, but you can change the length to meet your needs. The length of properties has an impact on the number of additional properties and levels you can have in a dimension. By making property lengths accurately reflect the actual length you need, you can maximize the number of levels in the dimension or the size of other properties (such as Formula).

You can store additional properties in either the SQL or Analysis Services databases by selecting or deselecting the InApp checkbox while adding a property. Storing dimension properties in the SQL database gives you better performance. However, you cannot store properties in SQL that you will need in dynamic expansions or other MDX-type queries, such as from Dimension formulas. Properties that are referred to in MDX-type queries must exist in the application database. The system will not allow you to move required properties to SQL. For other properties, you need to know whether you will query the properties using MDX-type queries.

---

**Note:** Required properties must be stored within an application. Also, if you plan to query on a property from a dimension formula or from a dynamic expansion, you must store that property in the application (leave the InApp check box selected).

---

#### To add properties to dimensions

1. From the Admin Console, select Dimension Library.
2. Select the desired dimension.
3. Go to a blank line at the bottom of the property list and type the new property ID.
4. Select or deselect the InApp checkbox.
5. Click the Save button. While saving the dimension, BPC checks the length of the dimension records and the number of levels to ensure they are within the system maximums. Property names appear as column headings on the Member Sheet for a dimension. When you save the dimension after new properties have been added, columns are added to the Member Sheet for the new properties.

---

**Note:** You can keep columns on a member sheet that are not related to properties. These columns can be for your information only, and are ignored when the system validates and processes the member sheet.

---

#### Maintaining dimension properties

You maintain dimension properties by adding new properties to a dimension, and specifying whether a property is saved in the OLAP cube. After you add a property to a dimension, you can assign property values to members in the dimension. See Maintaining dimension members.

You can also delete dimension properties, as long as they are not required.

If a property is saved in the OLAP cube, you can write MDX queries using that property. This may increase the cube size significantly. So if you do not need to write MDX queries against a property, you should leave the InAPP check box unselected.

#### To add a dimension property (or change its InApp value)

1. From the navigation pane in the Admin Console, select Dimension Library, then select the dimension to which you want to add that property or change its InAPP value.
2. Select Maintain dimension properties from the action pane.
3. To add a new property, enter the name of the property in the empty field at the bottom of the list. In the adjacent text box, enter the maximum alpha-numeric size of the property values.
4. To store the property values in MDX, select the InApp check box for the appropriate property name. If you do not want to store the property values in MDX, deselect the InApp check box next to the appropriate property name.
5. When you are done with your changes, click the Modify Dimension Properties link in the action pane.

#### To delete a dimension property

1. From the navigation pane in the Admin Console, select Dimension Library, then select the dimension whose property you want to delete.
2. Select Maintain dimension properties from the action pane.
3. Delete the property name from the middle column.

#### Using the Owner property

The "Owner" property is used when you are using the work status feature. You add the property to the dimension that drives the work status flow. For example, if your business process dictates that an entity is the differing factor when it comes to entering data, then the Entity-type dimension is the work status driving dimension. If your business process dictates that a department name is the differing factor, then Department dimension would have the Owner property.

The dimension you select to drive work status must have more than one hierarchy.

The Owner property takes user and team names as values. You can enter multiple names and teams separated by commas. You must also include the domain or server name in the path.

The following example shows different user names and teams used in the Owner property of the Entity dimension. It demonstrates that HHATCHER is the owner of the Sales member, and can change the work status associated with Entity:Sales. However, the work state setting that is applied must also be 'controlled by' the owner. See Changing work status settings for applications.

	A	D	E	F	G
1	ID	PARENTH1	PARENTH2	CURRENCY	OWNER
2	Worldwide1			USD	BPC\Administrator
3	Sales	Worldwide1		USD	BPC \HHATCHER
4	RD	Worldwide1		USD	BPC \TJONES
5	Manufacturing	Worldwide1		USD	BPC \RSMITH
6	CorpCenters	Worldwide1		USD	BPC \CorpTeam1
7	CorpCenters2		OtherRegions	USD	BPC \CorpTeam2

'Managers' are the owners of the parent member. Since Worldwide1 is the parent of Sales and the Administrator user is the owner of Worldwide1, the Administrator user can change the work status setting for both Worldwide1 (only when all the children have been changed) and Sales. However, the work state setting that is applied must also be 'controlled by' the manager. See Changing work status settings for applications.

#### Account dimension required properties

The Account-type dimension defines the chart of accounts for your application, and how those accounts are calculated and aggregated. Any dimension that is assigned the type 'A' is considered an Account-type dimension. Each application must have one (and only one) Account-type dimension.

An Account-type dimension has the following required properties:

Property name	Description
ID	A user-defined member ID. See About dimension properties
NEWID	Used when you need to rename an existing member ID. See About dimension properties
EVDESCRIPTION	A user-defined description for the member. See About dimension properties
PARENTH1	Parent, Hierarchy 1. PARENTH1 is only required if you want to define a hierarchy. You can add additional PARENT properties to define parents for additional hierarchies, such as PARENTH2, PARENTH3, etc. We recommend that you use hierarchies rather than formulas to define subaccounts. See About dimension properties
GROUP	A user-defined identifier for providing additional filtering options. See About dimension properties
FORMULA	If you have a hierarchy of accounts and children are calculated using a formula, the parent must have a formula which defines the aggregation of its children in

Property name	Description
	order to aggregate properly.  You can include library files for use in dimension formulas on the Options sheet of the account dimension. ApShell includes the system functions in MDXLIB.LGL and the CONSTANTS.LGL file.
ACCTYPE	The account type: can be INC for Income, EXP for Expense, AST for Asset, LEQ for Liabilities & Equity
SCALING	Scaling options are Y or N. Used by EvDRE and live reporting. Value is optional, but if value is not defined, scaling is not available for the associated member ID.
RATETYPE	Used by the currency conversion business rules. Value is optional.  See Currency conversion business rules
ELIMACCT	Valid account names.  See About intercompany eliminations

#### Optional properties

- FINSTMT is a user-defined property in ApShell used to sort expansions by financial statement, such as Income Statement vs Balance Sheet. This property is optional.
- TEMPLATE is a user-defined property in ApShell used to sort input schedules. This property is optional.

#### Category dimension required properties

The Category dimension defines the 'buckets' in which you store information in your application. Typical categories would be Budget, Actual, Forecast, and so on. Any dimension that is assigned the type 'C' is a Category dimension. Each application must have one (and only one) Category-type dimension.

A Category type dimension has the following required properties:

Property name	Description
ID	A user-defined member ID.  See About dimension properties
NEWID	Used when you need to rename an existing member ID.  See About dimension properties
EVDESCRIPTION	A user-defined description for the member.  See About dimension properties
HLEVEL	A system-generated property. It denotes the member's hierarchical level. The property values can be viewed in the Member Lookup in BPC for Excel.
CALC	A system-generated property. It indicates whether or not the account is calculated by means of a formula or is at a parent level. The property values can be viewed in the Member Lookup in BPC for Excel.
STYLE	A user-defined identifier for denoting different formatting applied using Excel's conditional formatting.  See About dimension properties
YEAR	Used to assign a YEAR to the category, to be used with the EVGET and EVTIM functions in reporting. See the BPC for Excel for more information.

### Currency dimension required properties

The currency type dimension is required if your company reports on local currency and translated values. These dimensions store the reporting and input currencies for your organization. Any dimension that is assigned the type 'R' is a Currency-type dimension.

The following table describes a Currency dimension's required properties. If you are utilizing BPC's legal consolidation functionality, there are additional properties required for the Currency-type dimension. See [Setting up a legal consolidation application set](#).

Property name	Description
ID	A user-defined member ID. See <a href="#">About dimension properties</a>
NEWID	Used when you need to rename an existing member ID. See <a href="#">About dimension properties</a>
EVDESCRIPTION	A user-defined description for the member. See <a href="#">About dimension properties</a>
PARENTH1	Parent, Hierarchy 1. PARENTH1 is only required if you want to define a hierarchy. You can add additional PARENT properties to define parents for additional hierarchies, such as PARENTH2, PARENTH3, etc. We recommend that you use hierarchies rather than formulas to define subaccounts. See <a href="#">About dimension properties</a>
REPORTING	This property is used to specify your reporting currencies. If Y, this member is used for reporting purposes.
GROUP	A user-defined identifier for providing additional filtering options. See <a href="#">About dimension properties</a>
STYLE	A user-defined identifier for denoting different formatting applied using Excel's conditional formatting. See <a href="#">About dimension properties</a>
MD	MD is a property of the INPUTCURRENCY dimension from the Rate application. It is used to determine how the rate is calculated.  M: Multiply D: Divide
SCALE	The number of digits to the right of the decimal point that should be displayed for a currency.

### Entity dimension required properties

The Entity dimension defines the business units for your application, and how those units aggregate. Any dimension that is assigned the type 'E' is a Entity dimension. Each application must have one (and only one) Entity-type dimension.

The Entity dimension defines the organizational structure, which drives the process of submitting and approving data.

For information about required properties for all dimensions, see [Adding dimensions](#).

An Entity dimension has the following required properties:

Property name	Description
ID	A user-defined member ID. See <a href="#">About dimension properties</a>

Property name	Description
NEWID	Used when you need to rename an existing member ID. See About dimension properties
EVDESCRIPTION	A user-defined description for the member. See About dimension properties
PARENTH1	Parent, Hierarchy 1. PARENTH1 is only required if you want to define a hierarchy. You can add additional PARENT properties to define parents for additional hierarchies, such as PARENTH2, PARENTH3, etc. We recommend that you use hierarchies rather than formulas to define subaccounts. See About dimension properties
CURR	The currency used by the entity. See About currency conversions
HLEVEL	A system-generated property. It denotes the member's hierarchical level. The property values can be viewed in the Member Lookup in BPC for Excel.
STYLE	A user-defined identifier for denoting different formatting applied using Excel's conditional formatting. See About dimension properties
CALC	A system-generated property. It indicates whether or not the account is calculated by means of a formula or is at a parent level. The property values can be viewed in the Member Lookup in BPC for Excel.
SCALE	The number of digits to the right of the decimal point that should be displayed for a currency.

#### Intercompany dimension required properties

An Intercompany dimension has the following required properties:

Property name	Description
ID	A user-defined member ID. See About dimension properties
NEWID	Used when you need to rename an existing member ID. See About dimension properties
EVDESCRIPTION	A user-defined description for the member. See About dimension properties
HLEVEL	A system-generated property. It denotes the member's hierarchical level. The property values can be viewed in the Member Lookup in BPC for Excel.
STYLE	A user-defined identifier for denoting different formatting applied using Excel's conditional formatting. See About dimension properties
CALC	A system-generated property. It indicates whether or not the account is calculated by means of a formula or is at a parent level. The property values can be viewed in the Member Lookup in BPC for Excel.
ENTITY	This is a 20 character field that can either be left blank or contain a valid

Property name	Description
	member name of the Entity dimension associated to the current application. The ENTITY property is validated against the Entity dimension, and blank fields are allowed.

#### Time dimension required properties

The Time dimension defines the units of time for your application, and how those units aggregate. Any dimension that is assigned the type 'T' is a Time dimension. Each application must have one (and only one) Time type dimension.

A Time dimension has the following required properties:

Property name	Description
ID	A user-defined member ID. Its format is YEAR.PERIOD, for example, 2001.JAN or 2001.Q1.  See About dimension properties
NEWID	Used when you need to rename an existing member ID.  See About dimension properties
EVDESCRIPTION	A user-defined description for the member.  See About dimension properties
TIMEID	Needed to change the ID members into the time format required for Analysis Services.
USERTIMEID	USERTIMEID is used to define a new 'user defined' TIMEID. The TIMEID is an internally assigned ID that converts the ID into an all numeric format for use in the SQL database.  You can define your own TIMEID by specifying a USERTIMEID. If a USERTIMEID is specified, BPC will assign it to the TIMEID. Note that the new TIMEID must follow a logical naming convention so that the months, years, etc. are identified correctly.  The property column must follow the TIMEID property column or you get an error message when the system processes the Time dimension.
ParentH1	PARENTH1 is used for defining the parent of each member in Hierarchy 1. Having hierarchies in the Time dimension allows you to define how you want time periods to aggregate.  You can have multiple hierarchies within a Time dimension. You do this by adding additional PARENTHn columns to and then defining the parents for the additional hierarchies.
YEAR	The YEAR property allows you to do filtering, sorting, and reporting based on the year. The YEARS should be placed in chronological order in the file, in order for the EVTIM function in BPC for Excel to give offsets correctly.
PERIOD	The PERIOD property allows you to do filtering, sorting, and reporting based on the period.

Property name	Description
LEVEL	<p>Time can be a year, quarter, month, week or day. As described above, the PARENTH1 property is used to define the aggregation of the time periods.</p> <p>The LEVEL property is also very important in defining your time periods. You must have the correct Level (Year, halfyear, quarter, month, etc.) for each member. You must follow the chronological format throughout the Time dimension. The correct format is:</p> <ul style="list-style-type: none"> <li>• YEAR</li> <li>• HALFYEAR</li> <li>• QUARTER</li> <li>• MONTH</li> <li>• WEEK</li> <li>• DAY</li> </ul>
ISBEGINNING	This value should be set to 1 for periods that correspond to the beginning of a year (such as Q1 or January, for instance) in all Time dimensions that incorporate YTD calculations. Set this value to 0 for all other periods.
HLEVEL	A system-generated property. It denotes the member's hierarchical level. The property values can be viewed in the Member Lookup in BPC for Excel.

#### Creative use of properties

This section demonstrates how you can use properties to help format your reports and input schedules. Set up a report this way:

- Add a Style property to Accounts.
- For input-level accounts, enter a Style value of N (no style).
- For the highest level totals, enter a Style of T1. For the next level of totals, enter a Style of T2. You can have as many levels as you want: T3, etc.
- In reports, for each account, retrieve the Style property in the report definition area, using the EVPRO function.
- Use Excel's conditional formatting to define formatting that varies depending on the Style property retrieved for that row.

#### Special optional properties

The following optional properties are available for use in any dimension:

- Formula — Allows you to define formulas for dimensions.
- SolveOrder — Use this property to define the order in which calculated members are solved in the case of intersection with other calculated members. Zero is the highest priority. Solve Order determines the order in which dimensions, members, calculated members, custom rollups, and calculated cells are evaluated, and the order in which they are calculated. The member with the highest solve order is evaluated first, but calculated last.
- UnaryOperator — Use this property to control how member values are rolled up to their parent's values. The following table describes each operator.

Operator	Description
+	The value of the member is added to the aggregate value of the preceding sibling members.
-	The value of the member is subtracted from the aggregate value of the preceding sibling members.



*	The value of the member is multiplied by the aggregate value of the preceding sibling members.
/	The value of the member is divided by the aggregate value of the preceding sibling members.
~	The value of the member is ignored.

## Managing Applications

An application in BPC is a functional unit used for a specific purpose, such as a Budgeting application or a Management Reporting application.

### About working with applications

Applications can share some dimensions with other applications within the same application set, and can have some dimensions that are unique to that application. In Microsoft Analysis Services, an application equates to a cube; that is, there is one Analysis Services cube for each application.

You can create new applications, save applications, delete applications, and change application descriptions.

### Adding new applications

You add a new application to an application set by copying the structure of an existing application. The structure includes an application's dimensions, data, and templates. When creating a new application, you must choose an application type, which tells the system which properties to associate with the application. An application is either a reporting or non-reporting application.

#### Reporting applications

The following table describes the different types of reporting applications available.

Type	Description
Financial	<p>Financial-type applications allow you to perform management consolidation functions. They support data translations from local currencies to one or more reporting currencies, intercompany elimination calculations, and other calculations.</p> <p>When you create a Financial-type application, you must choose an associated Rate-type application (see below). You can also choose to set up the following business rules tables:</p> <ul style="list-style-type: none"> <li>• Currency conversion</li> <li>• Account transformation</li> <li>• Intercompany Bookings</li> <li>• US Eliminations</li> <li>• Carry-forward rules</li> <li>• Validations</li> </ul>
Consolidation	<p>The Consolidation-type application allows you to perform legal consolidation functions. It is similar to a Financial-type application, but with legal consolidation rules instead of management consolidation rules.</p> <p>Consolidation-type applications must reference an Ownership-type application and a Rate-type application.</p> <p>You can also choose to set up the following business rules tables:</p> <ul style="list-style-type: none"> <li>• Currency conversion</li> <li>• Account transformation</li> <li>• Intercompany Bookings</li> <li>• US Eliminations</li> <li>• Carry-forward rules</li> <li>• Validations</li> <li>• Automatic Adjustments</li> </ul>
Generic	<p>A generic-type application has no special requirements (other than to include the four minimally required dimensions). Generic applications have no out-of-the-box business intelligence, so if you want to apply logic, you must create it using BPC 's script logic.</p>

### Non-Reporting applications

Non-reporting applications are designed to support reporting applications or to simply hold data, such as price or rate information. You can report on non-reporting application data, but you cannot view its data in an Insight dashboard, or assign work status codes to the data. In addition, you cannot define business rules to these application types. As with Reporting applications, non-reporting applications require four dimension-types: Entity, Account, Time, and Category.

The following table describes the available non-reporting type applications.

Type	Description
Rate	<p>A rate application is a supporting application for one or more Financial or Consolidation reporting-type applications. It is used to store exchange rates that support currency conversion in financial applications.</p> <p>This application must include a Currency-type dimension detailing the exchange rates by each individual input currency.</p>
Ownership	<p>The ownership application is a supporting application for a Consolidation reporting-type application. It stores information such as the consolidation methods, ownership percentages, and group rollup information used for legal consolidation.</p>
Generic	<p>A generic application has no special requirements (other than to include the four minimally required dimensions). Generic applications have no out-of-the-box business intelligence, so if you want to apply logic, you must create it using BPC's script logic.</p>

### To create a new application

1. From the Admin Console, select Applications.
2. Select Add new application from the action pane.
3. Enter a name and description, and then click Go to Step 2 of 4.
4. Select the application type, then click Go to Step 3 of 4. (See table above for descriptions.)
5. Select a source application from the list. If you selected Financial or Consolidation as the type, select the corresponding Rate and Ownership applications and the business rules tables you want to use.
6. Select what to copy from the source application. You can choose to copy all options from the source application, or just some of the options. Options you can copy include: Dimensions, Application publications, Private Publications, Content Library, Reports, Team tasks, Data Manager Packages, and Journals. Make your selection(s), and then click Add New application.

**Note:** The report templates used when you use "New" to create an application are located on the server in the following directory: \BPC\WebFolders\<AppSet>\<AppName>\AppTemplate\Excel

7. If you deselected the Dimensions option, move the dimensions you want to copy from the source application on the left side to the right side. Click Add a new application.

**Note:** You must select one of each of the required dimension types before continuing. Required dimensions types include Account (A), Category (C), Entity (E), and Time (T). See Required dimensions in applications.

8. When processing is complete, click OK.

### About the Rate application

A rate application is a supporting application for one or more Financial or Consolidation reporting-type applications. It is used to store exchange rates that support currency conversions in those applications.

A Rate application has the following characteristics:

- The RATE application contains the usual required dimensions (Category, Time, Entity, and Account), and one Currency dimension.

- The Category and Time dimensions must be identical to the dimensions used by the applications using this application to store their foreign currency exchange rates.
- The Account dimension must include the GROUP property.
- The Currency dimension does NOT need to have the REPORTING property.

## Copying applications

You can copy an application as a quick way of creating a new application. When you copy an application, its dimensions, data, and templates get copied from the source application. You can modify the application later.

To copy an application

1. From the Admin Console, select Application.
2. Select Copy an application from the action pane.
3. Select the name of the source application, and then enter the name of the new application.
4. Select Copy an application.
5. When the progress indicator completes, click OK.

## Changing work status settings for applications

You can change the work status settings for each application. Changing the work status settings involves identifying three to five 'work status dimensions,' and then defining specific members for the remaining non-work status dimensions used for validation purposes. You must determine which account you will use for validation of the data at the time the work status is changed. The validation account must be "0" at the intersection of the 3-5 variable members and the members designated for the non-work status dimensions. If the account is not "0" then the owner/manager cannot set the work status.

The dimensions you select as the work status dimensions are the variables in your business process. For example, the entity, category, and time might change based on who is submitting data, but the account, data source, reporting currency, etc., remains static. (Typically, Time is a work status dimension since data is usually segregated based on time.)

For example, let's say that you set Entity, Category, and Time as your work status dimensions for a given application. You then set your other current view members to the following: Account: Validation; DataSrc: TotalAdj; Intco: All\_Intco; and RptCurrency: LC. In addition to assigning work status dimensions, you also specify which dimension is the 'owner dimension.' The owner dimension includes the Owner property. The owner property determines who can edit a work status setting. The following figure shows this setup, where Entity is the Owner dimension.

AppSet Dim Name	Work State	Member Validation
Account	No	VALIDATION
Category	Yes	
DataSrc	No	TotalAdj
Entity	Owner	
Intco	No	All_InterCo
RptCurrency	No	LC
Time	Yes	

A user attempts to post data to the current view shown in the following table. The system checks the 'validation' account to make sure the intersection equals zero (0). If so, the data is posted, and a success message is displayed. The user can now set the work state to 'submitted' on that intersection. Subsequent

submissions to that exact intersection will be rejected. Users can only send data to the same intersection if the Entity, Category, or Time member changes.

Account	<All>
Category	Actual
DataSrc	<All>
Entity	SalesNE
Intco	<All>
RptCurrency	<All>
Time	Feb.2007

To change work status settings for applications

1. From the Admin Console, expand the Application node.
2. Expand the application for which you want to change the work status setting.
3. Select Work Status Settings.
4. In the Work State column, select Yes for each dimension you want to use to control the work status settings. Select Owner for the dimension that contains the Owner property.
5. In the Member Validation column, select a member for each non-work status dimensions. (Use the browse button to open the Member Lookup.)
6. From the action pane, select Save Work Status Settings.

## Setting concurrent locks

You can define a concurrent lock setting for each application. Concurrent locks prevent users from sending data to the same data intersection at the same time. You define a concurrent lock by selecting three or more dimensions. When multiple users try to submit data to the same members for those dimensions, the data sent first is accepted. Any data sent concurrently will be rejected.

How do you determine which dimensions define the concurrent lock? Lets say that you set concurrent locks on Entity, Category, and Time. Your company has one individual that is responsible for the P&L data for an entity, and one who is responsible for the Balance Sheet data for the same entity. If both users attempt to write to the database at the same time, one is locked out. (Data submission is based on a first come first serve basis.) In this case, you would want to set your concurrent locks on Entity, Category, Time and Account. That way, there would be no data submission conflicts between the two individuals.

To set a concurrent lock

1. From the Admin Console, expand the Application node.
2. Expand the application for which you want to define a concurrent lock setting.
3. Select Concurrent Lock.
4. Select Yes for each dimension you want to use to define the intersection of data upon which a concurrent lock setting is applied.
5. Select the name of the source application, and then enter the name of the new application.

## Optimizing applications

Optimization cleans up data storage which improves the responsiveness of the system. For more information about optimization options, see the *BPC Operations Guide*.

### To optimize an application

1. From the Admin Console, select Application, and then select any application.
2. From the Manage Applications action pane, select Select Optimize application.
3. Select one or more applications to optimize.
4. Select Lite, Incremental, or Full Optimize. See the descriptions above for more information. The Compress Database option sums multiple entries for the same current view into one entry so that data storage space is minimized.

---

**Note:** Note: As a rule, lite optimizations are generally most effective when run at intervals measured in minutes (rather than days or hours). As a result, you may choose to schedule lite optimizations as part of a Data Manager package. See the *BPC Data Manager Guide* for more information.

---

5. Click Optimize Applications. (Depending on the size of your applications and the options selected, this process can take a long time.)

### Required dimensions in applications

Every application is required to have one dimension of each of the following types: Account, Category, Entity, Time, Currency, and Intercompany. While a currency type dimension is required to be present in all application sets, it does not have to be part of any applications within the application set. But it must be present in the application set for validation purposes.

Note that an application set can have multiple dimensions of one type, but each application within the AppSet can have only one of each of the required types. For example, the application set might have entity type dimensions named *EntityB* and *EntityF*, with *EntityB* used in the Budgeting application and *EntityF* in the Forecasting application.

Dimension type	Type identifier	Definition
Account	A	Contains your chart of accounts.
Category	C	Contains the types of data you are going to track, such as Actual, Budget, Forecast, etc. You can set up categories to store versions, such as BudgetV1, BudgetV2.
Entity	E	Contains the business units that are used to drive the business process. Depending on your application design, the Entity type can be an operating unit, a cost center, a geographic entity, etc. This dimension will be used to supply the members that are used in the Status Board approval process.
Time	T	Contains the time periods for which you want to store data.
Currency	R	Contains the currency rates for all currencies in which your company does business. Required in the application set, but not in each application. See the note, below.
Intercompany	I	Contains accounts that are used in Automatic Adjustment calculations. Required only if your application is set to Single- or Multi-currency type and you choose the Intercompany option. See Adding new applications.

### About the Measures dimension

In addition to the dimensions listed above, BPC also requires a dimension called Measures. This dimension is automatically included in all BPC application sets. It is not listed as a dimension in the Manage Dimensions task, but is displayed on the CurrentView bar in the BPC for Excel module. Measures allows you to change the view of your data. You can view Periodic, Quarter-to-date (QTD), Year-to-date (YTD), Half-year-to-date (HYTD), Month-to-date (MTD), and Week-to-date (WTD) views of your data using the Measures dimension.

## Managing data audit

You can set up data auditing to record an audit trail of BPC activity. You can set it up by application and category.

To manage audit data

1. Start BPC Administration, and select Manage data audit from the Administration Configuration action pane.
2. From the Select Application field, select the application for which you want to set properties.
3. Select Enable Data Activity to turn on auditing for the selected application.

---

**Note:** Click the View Audit Settings for All Categories link from the action pane to see a report that shows which categories and tasks are enabled for the selected application.

---

4. Select a category, then select one or more tasks to audit. Select the Select All button to select all of the task check boxes, or select the Clear All button to clear all check boxes. You can select from the following tasks:
  - BPC for Excel
  - Data Manager Import
  - Data Manager Clear
  - Logic Execution
  - Live Report
  - Journals
5. Under the Set Log Limits section, do one of the following:
  - If you want to keep all audit data for the selected category in the database, leave the default selected, Never purge audit data.
  - If you want to purge audit records for the selected category after a specified number of days, select Set purge frequency, and enter the number of days for which to keep audit records in the database. In addition, you must run the DTS Purge package to clear the database for everything beyond the specified number of days. See Data Manager Help.
6. Under the Schedule Synchronization section, set up a synchronization schedule by selecting Enable schedule, and entering the desired time frequency. By default, audit records are synchronized every 12 hours, if enabled.
7. Click the Save Audit Data Now link to run the synchronization process right away.
8. Click the Publish audit report link to send a data audit report to the System Reports page.
9. Click Update to save your changes.

## Creating a YTD storage application

Since most general ledger and other source systems store balances on a periodic basis, BPC's default data storage on applications is based on periodic time intervals. With this method, all calculations are performed on periodic balances. The balances are then accumulated for year-to-date reporting.

In some business cases, calculations should occur on a year-to-date basis, like in an application with foreign currency translation. If year-to-date basis is required, you can set your applications to store data on a year-to-date basis, so they accept data entry in the YTD format. When data is entered into YTD, its periodic values, used for reporting purposes, are derived by calculating the difference between the current period and the last period, as in the following example. (This example depicts INC or EXP accounts. There is no change in behavior for AST or LEQ accounts.)

	January	February	March
Periodic	100	200	0
YTD	100	300	300

Both periodic and year-to-date storage methods support daily, weekly, monthly, quarterly and year-to-date reporting requirements.

To create a YTD storage application

1. Start BPC Administration and click the Set Application Parameters link from the Administration Configuration page.
2. If you are using multiple time hierarchies, specify the time hierarchy to be used in this application in the *YTDInputTimeHir* field (H1 is the default).
3. Set the YTDINPUT parameter to 1, then click the Update button on the bottom of the page. If you need to add the YTDINPUT parameter, enter YTDINPUT in the New field at the bottom of the screen. Click Update at the bottom of the page.

## Deleting applications

An administrator can delete applications.

To delete an application

1. From the Admin console, select the application you want to delete.
2. Select Delete application, then select Yes in the confirmation dialog box.



## Managing Security

Managing users' access to sensitive information is a fundamental part of the BPC solution. BPC provides an easy-to-use interface for managing security, and provides two major levels of protection: authentication (through Microsoft Windows) and authorization (through Microsoft SQL Server and Analysis Services).

Authentication is the component of the security model that describes who can access the system, and is controlled by prompting users to enter credentials when accessing sensitive areas. This applies to users that are assigned a user name, and their status is 'active.' Authentication is controlled through the Windows security system, but is tightly integrated with BPC.

Authorization is the component that describes which data sets authenticated users may access, and is determined by a user's assigned task and member access profiles. This is controlled primarily through Analysis Services, and is also tightly integrated with BPC.

### About security

Security in BPC is based on task profiles and member access profiles. The default, that is when BPC is first installed, is no access to all tasks and all secured dimensions. This means that if you do not specifically assign task profiles to users or teams, no one will have access to any BPC tasks. Similarly, if you do not define access to members of a secured dimension to users or teams, no one will have access to that dimension. There is one default task profile and one default member access profile available in ApShell that you can use to get started quickly.

Defining security involves the following steps.

1. Name each BPC user. See Adding users
2. Assign users to teams. See Adding teams
3. Assign task profiles to users or teams. See Adding task profiles
4. Assign member access to users or teams. See Adding member access profiles

### Using SSL (HTTPS) Web Site Security

BPC recommends that you set up Secure Socket Layer (SSL) encryption on your BPC Web servers.

For information about setting up SSL on Internet Information Server (IIS), contact a consultant, or see these Microsoft Knowledge Base articles, in the order they are presented:

- 228821— Generating a Certificate Request File Using the Certificate Wizard in IIS 5.0 (<http://support.microsoft.com/default.aspx?sd=msdn&scid=kb;en-us;228821>)
- 228836— Installing a New Certificate with Certificate Wizard for Use in SSL/TLS (<http://support.microsoft.com/default.aspx?sd=msdn&scid=kb;en-us;228836>)
- 324069— HOW TO: Set Up an HTTPS Service in IIS (<http://support.microsoft.com/default.aspx?sd=msdn&scid=kb;en-us;324069>)

### Troubleshooting HTTPS

In certain situations, you can have issues when you set up HTTPS. The following list contains rules to follow and answers to issues that might arise after installing SSL on your BPC servers.

#### Rules for running SSL with BPC

Your BPC IIS Web Server must have only the SSL Web server active. If you leave both the HTTP and HTTPS servers running at the same time, BPC cannot function correctly.

If you run BPC in Multi-server mode, each server must have SSL installed and running. Again, you cannot have both non-SSL and SSL servers running at the same time on any of the servers.

#### Correcting Administration Client Issues

If you have followed the rules above, but have upgraded from a previous version of BPC, you might still have problems connecting through SSL from BPC Administrator clients.

If your Administrator clients get errors connecting to SSL servers, check the following registry key:

HKEY\_CURRENT\_USER\Software\VB and VBA Program Settings\BPC for  
Excel\LATEST\WEBSERVERPROTOCOL

The WEBSERVERPROTOCOL key's value should be set to 1 (one) for HTTPS. In certain situations, you may have to create this key and assign 1 as the value. Normally, this key's value is set when you log on to a server, but there are cases where it might not be set properly during an upgrade.

## Setting up users

You set up users so that they can use BPC. When setting up new users, you need to think about several topics:

- Is the user a domain user or a local user on the BPC server?
- Do you want BPC to automatically create Windows user IDs for local users?
- Do you want to set up teams so you can more easily assign privileges to users?
- Which users are administrators and which are not?
- To which Entity members will users have access?

### About adding users

There are two ways to add users to the system, and the method you use depends on your network infrastructure. If your security is controlled on a domain and maintained centrally through Active Directory Services, you must add users to the system with the associated domain name. If your security system is maintained through BPC, you can add users with only a user ID.

### About using domain user IDs

This method requires that each user is a valid user in the domain before adding them to BPC. When a user ID is added to the system with a domain name (for example, BPC\hsmith), the system assumes the user ID is being maintained outside of BPC within Active Directory Services. When the user logs on, the system validates the password against Active Directory Services.

There is a new user interface in Server Manager that allows server administrators to specify specific domains that are being used for BPC users. In addition, filters can be applied to those domains in order to select specific users from them. See Server Manager Help for more information.

When you are adding new users from a domain to BPC, you have the ability to select one of the user-defined groups, and customize it further, if required.

### About using BPC user IDs

If not on a domain, users must be valid Windows users. You must set users up on Windows first, and then add them to BPC's security.

You add BPC user IDs in the following format: hsmith. Behind the scenes, if only the user ID is entered, the system assumes that security is maintained on the BPC server, and, internally, the system automatically assigns a domain name.

The assigned internal domain name is determined by the NETMODEL the system is configured to use. If the system is configured to use NETMODEL=WORKGROUP, then the server machine name becomes the domain name. If the system is configured to use NETMODEL=DOMAIN, then the Domain parameter from the Appserver.ini file becomes the domain name.

### Adding users

You can add new users in BPC and can assign them to teams, task profiles and member access profiles.

If you are not using the default task or member access profiles and have not set them up yet, you might want to define them before adding users. You might also want to create teams, so you can assign the newly added users to the appropriate teams.

Alternatively, when you define the teams and profiles, you can assign users to them at that time.

### To add users

1. From the Admin Console, expand Security, expand Users, then expand the domain name. The Manage Users action pane is displayed.
2. Select Add New User. The Add New Users assistant is displayed.
3. In Step 1 of the Add New Users assistant select the users to which you want to grant access to BPC:

- a. Use the drop down menu in the Available domain field to specify the domain for which you want to add a new user. If multiple system user groups have been defined on the system, the associated custom filter is displayed upon selection of one of those groups. You can modify the custom filter, as needed, by clicking the Custom Filter button.
- b. Highlight the desired users from the Available from the domain window, and click the Add (>) button to move the selected user to the Selected window. To move all users from the Available to the Selected window, click the Add All button (>>).

---

**Note:** Use the Find field and Search button to find a specific user.

---

4. When Step 1 of the Add New Users assistant is completed click Next.
5. In Step 2 of the Add New Users assistant, enter an email address for each new user and click Next. (An email address is required if this user is a business process flow Owner or Reviewer.)
6. In Step 3 of the Add New Users assistant assign new users to teams, task profiles, and member access profiles:
  - a. In the A. Assign to Teams tab, select the desired team from the Team field. Highlight the desired users from the Available window and click the Add (>) button to move the selected users into the Selected window for the specified team.
  - b. Select the B. Assign to Task profiles tab. Select the desired profile from the Task profile field. Highlight the desired users from the Available window and click the Add (>) button to move the selected users into the Selected window for the specified profile.
  - c. Select the C. Assign to Member Access profiles tab. Select the desired profile from the Member access profile field. Highlight the desired users from the Available window and click the Add (>) button to move the selected users into the Selected window for the specified profile.
7. When Step 3 A, B, and C of the Add New Users assistant is completed click Next.
8. In Step 4 of the Add New Users assistant, review the information, then click Apply to process the new users.

### Modifying users

You can modify users in BPC by changing domain, Task profiles and or Member Access profiles.

#### To modify a user definition

1. From the Admin Console, expand Security, then select Users. The Users action pane is displayed.
2. In the hierarchical list at the left of the window, navigate to the desired user and select it.
3. From Manage Users Options task pane select Modify the selected user's definition. The Modify User assistant is displayed. Follow the prompts in the assistant.
4. Complete the Modify Profile assistant, and click Apply. See Adding users

### Setting up teams

You can set up and maintain teams of users. When you assign security to teams, the security works collectively on the team members. This allows you to easily maintain security for many users at the same time.

#### Adding teams

You can define teams in BPC to assign security rules to a set of users, rather than assigning security rules to each individual user. Teams are not required in order to successfully process security.

Each team can have one team leader. The team leader is the only user that has write access to the team folder. This means that only the user designated as the team leader can save, delete, and modify documents in the team folder.

#### To add a team

1. From the Admin Console, expand Security, then select Teams. The Manage Team Options action pane is displayed.
2. Select Add New Team. The New Team assistant is displayed.
3. In Step 1 of the New Team assistant enter the team name and description and click Next.
4. In Step 2 of the New Team assistant select the members for the new team:
  - a. Use the drop down menu in the Show field to specify the information that you want to show in the Available window below. You can specify Users or Teams (if you want to create a team that includes another team).
  - b. In the Available window, highlight the desired users (or teams) and click the Add (>) button to move the selections into the Selected window.

---

**Note:** Use the Find field and Search button to find a specific user.

---

- c. To move all users from the Available to the Selected window, click the Add All button (>>). Designate a single team leader by clicking on the box to the left of the leader's name in the Selected window.
4. When Step 2 is completed click Next.
5. In Step 3 of the New Team assistant assign task and member access profiles:
  - a. Use the drop down menu in the Task Profile field to select the task profile for which you want to specify one or more member access profiles. The combination of task/member access profiles determines access to BPC.
  - b. In the Member Access Profile window, select the desired profile(s).
  - c. When Step 3 is completed click Next.
6. In Step 4 of the New Team assistant, review the information, then click Apply to process the new teams.

#### Modifying teams

You can modify the definition of an existing team in BPC. When modifying a team, you can change everything except the team name. If you want to reuse most of an existing team, but give it a new name, refer to Copying a team.

#### To modify a team definition

1. From the Admin Console, expand Security, then select Teams. The Manage Team action pane is displayed.
2. In the hierarchical list at the left of the window, navigate to the desired team and select it.
3. From Manage Team action pane select Modify the selected team's definition. The Modify Team assistant is displayed. Follow the prompts in the assistant to revise the team definition, revise selected team members, or assign different task and member access profiles.
4. Complete the Modify Team assistant, and click Apply. See Adding teams

### Setting up profiles

Profiles allow you to define categories of users with who perform common tasks and who have common access to BPC applications. You can set up and maintain two types of profiles for users: Task profiles and Member Access profiles.

The combination of these two profile types defines the way an individual can use BPC.

#### Adding task profiles

Task profiles determine what type of activity or roles the user can perform in BPC. You can add tasks as needed. After creating a task profile, you assign it to multiple users, as needed.

For information about the default task profiles, and descriptions of each profile, see Default task profiles.

#### To add task profiles

1. From the Admin Console, expand Security, then select Task Profiles.

2. Select Add a New Task Profile. The New Task Profile assistant is displayed.
3. In Step 1 of the New Task Profile assistant, enter the profile name and description. Check one or more functions that the profile will contain, then click Next.
4. In Step 2 of the New Task Profile assistant, select the tasks (associated with the selected functions) you want to enable for the profile. The Default function tasks field displays the tasks that are automatically included in the profile. See Default task profiles.
  - a. In the View tasks by interface field, use the drop down menu to select a BPC interface.
  - b. A list of associated tasks appears in the Available interface tasks field. Highlight the desired tasks and click the Add (>) button to move the selections into the Selected interface tasks window.
  - c. To move all tasks from the Available to the Selected window, click the Add All button (>>).
  - d. Repeat these sub steps for each BPC interface.
5. When Step 2 is completed click Next.
6. In step 3 of the New Task Profile assistant, assign the users for whom the profile will apply.
  - a. Use the drop down menu in the View by field to specify the information that you want to show in the Available window below. You can specify Users or Teams (if you want to apply the profile to all members of a team).
  - b. In the Available window, highlight the desired users (or teams) and click the Add (>) button to move the selections into the Selected window.
  - c. Use the Find field and Search button to find a specific user.
    - d. To move all users from the Available to the Selected window, click the Add All button (>>). Designate a single team leader by clicking on the box to the left of the leader's name in the Selected window.
7. When Step 3 is completed click Next.
8. In Step 4 of the New Task Profile, review the information, then click Apply to process the new task profiles.

#### Default task profiles

A System Administrator, by default, has the following task rights:

- Appset
- DefineSecurity
- DefineDrillThrough

A Primary Administrator, by default, has the following task rights:

- AdminDataLock
- Application
- BusinessRules
- DefineSecurity
- DefineSecurity
- Dimensions
- InsightAdmin
- Lockings
- ManageAudit
- ManageBook
- ManageBPF
- ManageComments
- ManageDistributor
- ManageLiveReport
- ManageTemplates

- UpdateToCompanyFolder
- A Secondary Administrator, by default, has the following task rights:
- Dimensions
- eAnalyze
- ManageBPF

#### Task profile descriptions

The following table describes the available tasks. (The shading is for visual purposes only.)

Interface	Task	Can be assigned to	Description
Administration	Application	Only the primary administrator (default)	Can create, modify, and delete applications in this application set, make changes to dimensions and add dimensions, and optimize applications.
	Appset	System administrator, by default, but can be assigned to primary administrator	Can enable and disable Insight, create new application sets, modify application sets, and set application set parameters (in Web Admin Tasks).
	Business Rules	Primary administrator, by default, but can be assigned to secondary administrator	Can define business rules.
	Dimension	Only primary and secondary administrators (default)	Create, modify, process, and delete dimensions and members.
	Lockings	Primary administrator, by default, but can be assigned to secondary administrator	Set up and edit concurrent locks, and define and edit work status codes.
	Misc	Primary administrator, by default, but can also be assigned to system and secondary administrators.	Can manage and validate custom menus and view application set status.
AnalysisCollection	eAnalyze	Anyone	Can access, manage and edit ad hoc and audit reports, and access and save to the report library.
	ManageTemplate	Primary administrator, by default, but can be assigned to secondary administrator	Can manage the company report library, access and save templates from the library, restrict workbook options, and manage

Interface	Task	Can be assigned to	Description
			custom menus.
	SubmitData	Anyone	Can access the schedule library, build input schedules, send data. Can use spread, weight, and trend options. Can post documents with application context to the Content Library.
Audit	ManageAudit	Only primary administrators (default)	Can manage activity and data auditing.
BusinessProcessFlow	BPFExecution	Anyone	Can run BPFs from BPC for Office or BPC Web.
	ManageBPF	Primary and secondary administrators only (default)	Can create and edit BPFs.
Collaboration	ManageDistributor	Only primary administrator (default)	This user or team can use the Offline Distributor.
	PublishOffline	Anyone	This user or team collects changes to offline input schedules and sends data to a database.
Comments	AddComment	Anyone	This user or team can add comments.
	ManageComments	Primary administrator (by default), but can be assigned to system and secondary administrators.	This user or team can remove comments.

Interface	Task	Can be assigned to	Description
DM	Execute	Anyone	<p>This user or team can manage Data Manager packages:</p> <ul style="list-style-type: none"> <li>• Data upload</li> <li>• Data download</li> <li>• Data Preview</li> <li>• Clear saved prompts</li> <li>• View status based on user ID</li> <li>• View schedule status based on user ID</li> <li>• Run Specific package</li> <li>• Run user package</li> <li>• Validate &amp; Process conversion files for company</li> <li>• Validate &amp; Process transformation files for company</li> <li>• Save Transformation</li> <li>• Save Transformation As</li> <li>• Maintain status based on user ID</li> <li>• View status</li> </ul>
	GeneralAdmin	<p>Primary, system and secondary administrators.</p> <p>(No default assignment)</p>	<p>This user or team can perform tasks such as:</p> <ul style="list-style-type: none"> <li>• New Transformation</li> <li>• Test transformation with data</li> <li>• New Conversion</li> <li>• New Conversion Sheet</li> <li>• Save Conversion</li> <li>• Save Conversion As</li> </ul>
	PrimaryAdmin	<p>Primary, system and secondary administrators.</p> <p>(No default assignment)</p>	<p>Can perform default PrimaryAdmin tasks such as:</p> <ul style="list-style-type: none"> <li>• Manage transformation files for company and Validate &amp; Process</li> <li>• Manage conversion files for company and Validate &amp; Process</li> <li>• Packages that against the fact table directly are limited to admin</li> <li>• Manage team package access</li> <li>• Organize package list</li> <li>• Maintain status regardless of user ID</li> <li>• Run admin package</li> </ul>



Interface	Task	Can be assigned to	Description
	TeamLeadAdmin	Primary, system and secondary administrators. (No default assignment)	Can: <ul style="list-style-type: none"> <li>• Manage Transformation for non-company files and Validate &amp; Process</li> <li>• Manage Conversion for non-company files and Validate &amp; Process</li> <li>• Data Preview team folder</li> <li>• Validate &amp; Process conversion files for team</li> <li>• Validate &amp; Process transformation files for team</li> <li>• Data upload team folder</li> <li>• Data download team folder</li> </ul>
FileAccess	UpdateToCompanyFolder	Secondary administrator, by default, but can be assigned to primary administrators.	Can add files to the Company folder.
Insight	Analysis	Anyone	Has the following access rights to Insight: <ul style="list-style-type: none"> <li>• View Dashboard</li> <li>• Define KPI</li> <li>• View KPI Variance Analysis</li> <li>• Define KPI Alerts</li> <li>• Design KPI Charts</li> <li>• View KPI Radar</li> <li>• View KPI Predictions</li> <li>• Create KPI report (flash)</li> <li>• Perform KPI on-demand predictions</li> <li>• Comment viewing based on variance context results</li> <li>• Action Manager viewing</li> <li>• Add new and update actions based on owner</li> <li>• Edit actions regardless of owner</li> <li>• Insert KPI into Word/PowerPoint/Excel</li> </ul>
	InsightAdmin	Primary administrator (by default), but can be assigned to secondary administrators.	Can administer Insight.

Interface	Task	Can be assigned to	Description
Journal	AdminJournal	Primary, system and secondary administrators. (No default assignment)	Can manage journals: <ul style="list-style-type: none"> <li>Create and maintain journal templates</li> <li>Clear journal tables</li> <li>Create Journal</li> </ul>
	CreateJournal	Anyone	Can create or modify journal entries.
	PostJournals	Anyone	Can post journals.
	ReviewJournals	Anyone	Can review journals
	UnpostJournals	Anyone	Can unpost journal entries.
Publish	ManageBook	Primary administrator (No default assignment)	This user or team can delete books in BPC Administration.
	PublishBook	Primary administrator (No default assignment)	This user or team can publish a book of reports.
	PublishFile	Primary administrator (No default assignment)	Can post files to the Content Library or in BPC Web.
Security	DefineSecurity	Only system and primary administrators (by default).	Can manage users, task and member access profiles.
ViewSystemReport	AuditReport	Anyone	This user or team can create audit reports.
	BPFReport	Anyone	This user or team can define Business Process Flow reports.
	CommentReport	Anyone	This user or team can run a comment report.
	JournalReport	Anyone	This user or team can run a journal report.
	Workstatus report	Anyone	This user or team can run a work status report.

Interface	Task	Can be assigned to	Description
WorkStatus	SetWorkStatus	Anyone	This user or team creates work status on a data region.
ZFP	AccessContentLib	Anyone	This user or team can access, filter, and sort, and add pages to the Content Library in BPC Web.
	CreateWebPage	Anyone	This user or team can create new web pages in BPC Web.
	LiveReport	Anyone	This user or team can access live reports in BPC Web.
	ManageContentLib	Primary administrator (by default), but can be assigned to system and secondary administrators.	Can manage all items in the Content Library.
	ManageLiveReport	Primary administrator (by default), but can be assigned to secondary administrators.	This user or team allows you to manage live reports using drag & drop in BPC Web.
	WebAdmin	Primary administrator (by default), but can be assigned to secondary administrators.	Can do the following in Web Admin Tasks: <ul style="list-style-type: none"> <li>Edit drill through tables</li> <li>Set application parameters</li> <li>Manage dimensions (make changes to existing dimensions based on dimension)</li> <li>Publish Reports</li> <li>Manage document types and subtypes</li> <li>Use Bulk Collaboration</li> <li>Publish Non-BPC reports</li> </ul>

#### Adding member access profiles

Member Access profiles determine the specific applications to which users have Read access, Write access or no access. In addition to application you can restrict access for the profile by dimension and member. After creating a Member Access profile, you assign it to multiple users, as needed.

When defining access to a secured dimension that has one or more hierarchies defined, security is applied to the member and all its children. For example, if you grant access to a member that has 10 children, users with access to the parent member also have access to all ten children. In addition, access is applied across levels in BPC. This means if you grant access to a member that is at level two (2) in the hierarchy, that user also has access to all members at level two of the hierarchy for branches on which they are granted access at all.

#### To add member access profiles

1. From the Admin Console, expand Security, then select Member Access Profiles. The Member Access Profiles Options action pane is displayed.
2. Select Add a New Member Access Profile. The New Member Access Profile assistant is displayed.
3. In Step 1 of the New Member Access Profile assistant, enter a name for the profile and a description, then click Next.
4. In Step 2 of the New Member Access Profile assistant, use the table to define dimension and member access in each BPC application. (Application names are on the tabs.) Do the following:
  - a. Select the appropriate application tab.
  - b. On a row in the Dimension column, use the drop down menu to select a dimension. (Only secured dimensions will display in the drop-down.)
  - c. On the same row, click the lookup button (...) in the Member column to select a member using the Member Lookup dialog. See Using the Member Lookup.
  - d. On the same row in the Access column use the drop down menu to specify Read, Write (Read & Write), or No access.
  - e. Repeat a - d for each access/dimension/member rule you want to define.
  - f. When satisfied with your selections, click Next.
5. In Step 3 of the New Member Access Profile assistant, assign the users for whom the profile will apply by doing the following:
  - a. Use the drop down menu in the View by field to specify the information that you want to show in the Available window below. You can specify Users or Teams (if you want to apply the profile to all members of a team).
  - b. In the Available window, highlight the desired users (or teams) and click the Add (>) button to move the selections into the Selected window. To move all users from the Available to the Selected window, click the Add All button (>>).

---

**Note:** Use the Find field and Search button to find a specific user.

---

- c. Designate a single team leader by clicking on the box to the left of the leader's name in the Selected window.
6. When Step 3 is completed click Next.
7. In Step 4 of the New Member Access Profile assistant, review the information, then click Apply to process the new member access profiles.

#### Modifying profiles

You can modify the definition of an existing Task or Member Access profile in BPC.

#### To modify a profile definition

1. From the Admin Console, expand Security.
2. Select Manage Task Profiles or Manage Member Access Profiles. The selected Manage Profile Options task pane is displayed.
3. In the hierarchical list at the left of the window, navigate to the desired profile and select it.
4. From the Manage Profile Options task pane, select Modify the selected profile definition. The Modify Profile assistant is displayed. Follow the prompts in the assistant.
5. Complete the Modify Profile assistant, and click Apply. See Adding task profiles or Adding member access profiles

#### Viewing security reports

You can view security reports from the Admin Console and from BPC Web. In order to view (and publish) security reports, you must have the 'SecurityReport' task assigned to you.

### Viewing security reports from BPC Web

Before security reports can be viewed in BPC Web, or to refresh a security report if there have been changes to security settings, you must publish the desired

type of report. The following procedure describes how to publish, then view reports in BPC Web. You must have the 'SecurityReport' task assigned to you.

#### To view security reports from BPC Web

1. From the BPC Launch page, select BPC Administration.
2. From the action pane, select Publish Reports from the Web Admin Tasks section.
3. Select one or more reports that you want to publish.
4. Click the OK button.
5. From the action pane navigation bar, click the Home button.
6. From the Getting Started Options action pane, select Launch BPC System Reports.
7. Select Security reports.
8. Select the type of report you want to view: By User, By Team, By Member access profile, By Task profile, or BPF Security Report.
9. From the report, use the toolbar at the top to do any of the following:
  - Flip through pages
  - Increase or decrease the display size
  - Search for specific text in the report using full-text search
  - Export the report to a selected output format
  - Refresh the report
  - Get help on the report

### Viewing security reports from the Admin Console

#### To view security reports from the Admin Console

1. From the Admin Console, select Security.
2. From the action pane, select Security reports.
3. Select the type of report you want to view: By User, By Team, By Member access profile, By Task profile, or BPF Security Report.
4. From the report, use the toolbar at the top to do any of the following:
  - Flip through pages
  - Increase or decrease the display size
  - Search for specific text in the report using full-text search
  - Export the report to a selected output format
  - Refresh the report
  - Get help on the report

## Managing Business Process Flows

You manage business process flows (BPF) by adding new ones for use in an application sets, copying existing ones, and resetting instances of BPFs. You can also report on BPFs.

### About business process flows

Business Process Flows allow you to pre-package and sequence application tasks for different departments within your organization. Any of the available features within BPC Web, BPC for Office, and BPC Administration can be utilized by a business process flow.

Using the intelligence of BPC data regions, you can set up a single business process flow that can be used across applications within an application set. A data region serves as a key for opening a single instance of a business process flow. For example, an 'Allocate Expense' business process flow could be running for the same entity and time, but different categories.

Business process flows can also be integrated with work states to offer an additional level of intelligence and integration with your company's regulatory processes.

Using an Explorer-like interface that is accessible from BPC Web and BPC for Excel, end-users are guided through pre-defined sequential tasks that span across the different areas of BPC.

### Copying a Business Process Flows

Copy an existing Business Process Flow (BPF) when you want to create a new one that is largely similar to the existing one.

To copy a Business Process Flow

1. From the Admin Console, expand Business Process Flows. The list of existing Business Process Flows is displayed in the hierarchy.
2. Select the BPF that you want to copy.
3. From the action pane, select Save as business process flow.
4. Enter a name and description for the BPF.
5. Click Save as BPF.

### Resetting Business Process Flows

You can reset BPF instances back to step one. We recommend using with caution, however, because when you reset the BPF, you are resetting all instances of the BPF for all users.

To reset a business process flow

1. From the Admin Console, expand Business Process Flows. The list of existing Business Process Flows is displayed in the hierarchy.
2. Select the business process flow that you want to reset.
3. From the Manage Business Processes action pane, select Reset Process Flow Instances.
4. From the confirmation message, click OK.

### Reporting on Business Process Flows

You can generate two different types of BPF reports: a standard BPF report and a BPF step report. Both reports allow you to specify the business process flow, time frame, associated data region, and page orientation.

The standard BPF report lists the steps and sub-steps in the right column, and the status of each step or sub-step in the left column.

The BPF step report lists the entities for which the report is run in the rows, and the status of each step and sub-step in the columns.

---

**Note:** The following procedure describes how to prepare a report from BPC Web. From the Admin Console, you can prepare a standard-type report only. To prepare a standard BPF report from the Admin Console, expand Business Process Flows, select the BPF for which you want to report on, and select BPF status report. See steps 4 and 5 below to complete and display the report.

---

To generate a BPF report

1. From the BPC Web, Getting Started Options action pane, select Launch BPC System Reports.
2. Select BPF reports.
3. Select the desired BPF from the BPF Reports category if you want to prepare a standard report, or select the BPF from the BPF Step Reports category if you want to prepare a BPF Step report.
4. Select the business process flow for which you want to run that report.
5. From the action pane, select BPF status report.
6. Specify the start date, end date, orientation and data region to report on.
7. Click the OK button to generate the report for the selected business process flow. The report appears in a new browser window.

## Adding new business process flows

You can add new business process flows to an application set.

Add a new business process flow

You add a new business process flow by performing the following tasks:

- Defining the BPF, which involves naming it, giving it a description and controlling application, and identifying an owner
- Defining the data region, which involves defining which dimensions define the BPF data region
- Defining the timing, which involves setting the recurrence pattern, activation date and time, and notification time
- Setting up access rights, which allows you to specify which users or teams will be able to access the BPF
- Adding steps and substeps, which describe the general framework for the BPF
- Define actions, which involves identifying the tasks associated with each step and sub-step
- Enabling the BPF for the users who have been given access rights

To add a new Business Process Flow

1. From the Admin Console, select Business Process Flows from the navigation pane.
2. From the action pane, select Add a new business process. The Add a New Business Process Flow assistant is displayed.
3. Define the new Business Process Flow. See Defining a Business Process Flow.

### Defining a Business Process Flow

The Define BPF page allows you to name the BPF, give it a description and controlling application, and identify an owner.

The controlling application is the application whose Time dimension will govern the BPF. For example, if you select the Finance application, whose Time dimension is in Monthly increments, the BPF will use the monthly time grain.

The BPF owner is a user who will be notified when a step is completed.

#### To define a BPF

1. From the Setup BPF > A. Define BPF page of the Add a New Business Process Flow assistant, enter the Business Process Flow name and description in the first two fields.
2. In the What is the controlling application? field, enter the name of the controlling application.
3. In the BPF Owner field, enter a valid user name. You can click the lookup button to display a list of users. Select the desired user name, then click OK.

---

**Note:** The user must have an email address assigned to them in order for them to own this BPF.

---

4. Click Save on the bottom of the window, then click Continue to Setup Task B.
5. See Define Data Region.

#### Define the data region

The Define Data Region page displays the dimensions associated with the controlling application (specified in the Define BPF page).

The data region defines the "base" region in which the BPF is designed. However, end users who run the BPF may select a different data region, as needed, to perform their process flow tasks.

The 'drive' dimension is a dimension that is common to all data regions that users might use in completing their BPF tasks. For example, if you select the Finance application, the drive dimension might be Category or Entity. The drive dimension must have the 'Reviewer' property defined.

The 'identity' dimensions are all the dimensions that will be used in a BPF. The Time dimension, used in all BPFs, is automatically selected.

#### To define the data region

1. From the Setup BPF > B. Define the data region page of the Add a New Business Process Flow assistant, select the Drive dimension. The only options available are those that have a 'Reviewer' property defined for the dimension.
2. In the Identity Dimensions column, select all the dimensions that will be used in the BPF.
3. Click Save on the bottom of the New Business Process Assistant window, then click Continue to Setup Task C.
4. See Define timing.

#### Define timing

The Define Timing page is comprised of three sections: Recurrence pattern, activation date and time, and notification time.

#### To define BPF timing

1. From the Setup BPF > C. Define timing page of the Add a New Business Process Flow assistant, select the desired frequency of recurrence in the Recurrence field.

---

**Note:** Currently the only option available is Once. Additional options (Daily, Weekly, Monthly, Quarterly, and Yearly) will be available in a later release.

---

- Once - To set a BPF to run once and not recur. The BPF will become inactive after it runs once and will not appear on the end user's action pane. However, it will remain on the Administrator's list of BPFs and may be edited for future use.
2. In the BPF activation date section, use the drop down menus to specify the date and time the BPF will become active. Prior to this time, it will not appear on the end user's action pane.
  3. Automatic e-mail messages can be sent to BPF users to remind them to perform their appointed BPF tasks. In the When should users be notified section, use the two drop down menus to specify a number of days prior to the start of, or after the start of the BPF frequency you specified in Step 1.
  4. Click Save on the bottom of the New Business Process Assistant window.
  5. Click Continue to Setup Task D.
  6. See Setting access.



## Setting access

The Set Access page allows you to specify which users or teams will be able to access the BPF. Users not specified in this step will not see the BPF on their action panes.

### To set access

1. From the Setup BPF > D. Set Access page of the Add a New Business Process Flow assistant, in the View by field, select Teams to display team names or Users to display user names in the Available window.
2. From the Available window, highlight the desired users (or teams) and use the arrow buttons to move your selections into the Selected window.
3. Click Save on the bottom of the window.
4. Click Next to continue adding the BPF, or click Close to close and save the partial BPF for future editing.
5. See Defining steps and sub-steps.

## Adding a new step

Steps are general business actions you take in completing the business process. Each Business Process Flow (BPF) must contain at least one step.

Here are some properties of steps:

- Steps may or may not have sub-steps associated with them
- If a step has no sub-steps, it must have an action associated with it
- If a step does have one or more sub-steps, there can be no action associated with the step. Instead, an action is associated with each sub-step.
- Steps may be subject to approval before a BPF can be completed
- Steps must be completed in sequential order

### To add a step

1. From the Define Steps/Sub-Steps task in the Add a New Business Process Flow assistant, click Add. The Add New window is displayed.
2. In the What do you want to add field select Add a new process step.
3. In the Name field, enter the name of the step.
4. In the Instruction field, enter text describing the step. This instruction is displayed on the action pane for this BPF. It should give users a general description of this step.
5. In the Properties fields, use the check boxes to indicate if you want to:
  - Enable alerts. Reserved for future use.
  - Enable completion criteria. Completion criteria may be enabled if Work Status has been enabled in the controlling application of the BPF. If defined here, the selected combinations of members must be set to the specified work status before the step can be considered complete. See Managing work status.
  - Enable reviewers. This allows reviewers (as defined in the 'Reviewer' property of the drive dimension) to review the step. If you want to define an action that the reviewer must perform in order to review the step, select Set/Modify Custom Review Actions. See Defining actions.
  - Send email to reviewers. This option becomes active when you choose Enable reviewers. When you choose this option, an email is sent to the step's reviewers when a step is identified by the user as complete. The email subject line communicates the following: "*<Step\_Name> step of <BPF\_Name> has been completed by <User>*". The email uses the name of the person who completed the step as the email's sender. To set the text for the body of the email, choose Define email message. Note that if you do not define the custom email message, the above-mentioned email subject line is repeated in the body of the email.
6. If you selected to enable completion criteria, complete the Completion criteria section. For each dimension:

- Select the current view type: Member lookup, Inherit from CV, or Inherit from data region.
  - If you selected Member, use the Member Lookup dialog to select a member value for the Member field.
  - Use the Work status drop-down menu to select an existing work status designation. See Managing work status.
7. Click OK. The new process step is displayed in the hierarchy.
  8. Repeat the above steps as needed to add all the desired steps. To change the order of the steps, highlight one and use the up or down arrow button (in the top right above the hierarchy window). You can also use the Delete button to delete a step.
  9. Click Save to save the steps. To add sub-steps for a given step, highlight it and click Add. See Adding a new sub-step.
  10. Click Next to display the Define Actions screen. See Defining actions.

#### Adding a new substep

Sub-steps are used when you want to define one or more actions under a single step. Here are some properties of sub-steps:

- Each sub-step has an associated action
- Sub-steps are not subject to approval in the BPF
- Sub-steps are not subject to completion criteria
- Sub-steps need not be performed in sequential order
- They may be assigned to one or more users.

#### To add a sub-step

1. From the Define Steps/Sub-steps screen of the New Business Process Assistant window, highlight the step for which you want to create a sub-step. and click Add. The Add New window is displayed.
2. In the What do you want to add field select Add a new sub-step.
  - a. In the Name field enter the name of the sub-step.
  - b. In the Instruction field enter text describing the sub-step. The instruction will be displayed in the action pane shown to the user when the sub-step is selected.
3. Click OK. You are returned to the New Business Process Assistant window. The new sub-step appears in the hierarchy.
4. Repeat the above steps as needed to add all the desired sub-steps. To change the order of the sub-steps beneath a step, highlight one and use the up or down arrow button (in the top right above the hierarchy window). Use this for organizational purposes only; the order of the sub-steps does not affect the processing of the BPF. Multiple sub-steps may be performed concurrently. Sub-steps are not sequential. You can use the Delete button to delete a sub-step.
5. Click Save to save the sub-steps.
6. Click Next to display the Define Actions screen. See Defining actions.

#### Defining actions

An Action is an individual task performed in BPC as part of a BPF. Actions can be assigned to the following items:

- A step that does not contain sub-steps
- A sub-step
- A custom review action (this allows the reviewer to perform an action required to review a specific step)

An action could be something like opening a schedule or publishing a report. In previous versions of BPC, actions were driven by EV functions or MNU functions.

Actions are available in the action pane for all users who have access to the BPF. See Setting access.

#### To define an action

1. From the Define Steps/Sub-steps screen of the New Business Process Flow assistant, highlight the sub-step for which you want add an action. You may also highlight a step if it has no sub-steps. Click Next. The Define Actions page is displayed.
2. In the Action Name section, the sub-step (or step) name appears as the default action name. Click Add to add additional action names. The Add New Action window is displayed. Enter the action name and click OK. You are returned to the Define Actions page. The first action in the list is the one that will automatically display when the user selects the action.
3. In the Detail section of the page, use the drop-down menus to specify the BPC interface and task to associate with the action. In the Parameter Value field, if any, enter valid parameters for the specified interface/task. Certain parameters allow you to browse the file system for a parameter name.
4. In the Current View section, use the drop-down menus to specify the application associated with the specified action. Use the drop-down menus in the Member column to specify the source of the values used for each dimension. Alternatively, you may click the look button to display the Member Lookup dialog.
5. Click Save to save the actions.
6. Click Back to return to the Define Steps/Sub-steps screen, if you want to create more steps or sub-steps. Click Next to progress to the BPF Finish screen.

#### Finishing a Business Process Flow

You can finish a BPF after the first three general steps have been completed. The Finish page shows a summary of the BPF for your review, and allows you to make the BPF available to users immediately, or hold it aside for future use. If you make it available, the BPF will appear on the action panes of the specified users.

#### To finish a BPF

1. From the Setup BPF > D. Finish page of the Add a New Business Process Flow assistant, review the BPF information.
2. To enable the BPF for immediate use, select the Enable this BPF for users box in the lower left corner of the screen. Deselect the check box if you do not want the BPF to be available at the present time.
3. Click Save to finish the BPF, then click OK.
4. Click Close to close the assistant.

#### Deleting a Business Process Flow

You can delete a Business Process Flow (BPF) when you want to remove it from the application set entirely.

---

**Note:** As an alternative to deleting, you can disable a BPF. A disabled BPF does not appear in any end-user action panes but does remain on the administrator's list. It remains available to copy and modify in the future. For detail on disabling a BPF, see [Finishing a Business Process Flow](#).

---

#### To delete a Business Process Flow

1. From the Admin Console, select Business Process Flows. The list of existing Business Process Flows is displayed in the hierarchy. Also the Manage Business Processes action pane is displayed.
2. From the hierarchy, select the BPF that you want to delete.
3. Under the Process Flow Tasks section select Delete business process flow. A confirmation message appears. Click OK.

## Managing Work States

Work states allow submitted data to be tracked, approved and locked using customizable work state definitions that suite your business needs.

### About managing work status

Managing work status involves specifying who can make changes to data in the database and who can change the work state on a data set. Default work states are Unlocked, Submitted, and Locked. You can add new work states if you need more flexibility.

After your work states are defined and your ownership dimension is set up, end users can use the work states to apply a label to a specific current view intersection for the purpose of locking data so it can be reviewed, approved, etc. For example, your month-end close business process requires that a specific set of data is locked down so that accurate month-end reports can be created. After a data submission, the owner sets the work state to 'Submitted.' This locks the data intersection from subsequent submissions.

You can define one set of specific current view values for each application. See [Changing work status settings for applications](#).

The following rules describe work status behavior:

- The default method for managing work status is bottom-up. That is, the status of a parent cannot be higher than the status of its children. You can set work status to top-down in the TOPDOWN field on Setting Application Parameters page. See [Setting application parameters](#).
- For bottom-up behavior, the maximum state a parent can be set to is the lowest state of its immediate children.
- The minimum state a child can be set to is the state of its immediate parent. For example, if the parent state is Submitted, the child state must be at least Submitted.
- If the status of a parent is set to Locked, you cannot unlock the children.
- The owner of an entity can set the work state to any state designated as an Owner state. See [Adding new work states](#).
- The manager of an entity can set the work state to any state designated as a Manager state. See [Adding new work states](#).
- A manager is the owner of a parent-level member. The owner of a parent level member is the manager of all its descendants.

To use work state tracking you must specify the hierarchy (H1, H2, H3, ..., Hn) within the owner dimension for which you want to use work status.

For information on editing the work status information, see [Adding new work states](#).

### Adding new work states

Default work states are Unlocked, Submitted, and Locked. You can add new work states if you need more flexibility.

When you add a new work state, you define who can edit data, what area of BPC can be edited, and who can change the work state.

The levels of security are:

- All - All users (with the appropriate member access rights) can change data
- Locked - No one can change the data.
- Manager - Only Managers (parents of owners) can change data
- Owner - Only owners can change data

The areas of BPC for which you can control the level of security are:

- Data Manager - Controls data input from running a Copy, Import, or Move package
- Journals - Controls data input from posting journal entries
- Manual BPC for Office data entry - Controls data submissions from reports and input schedules in BPC for Office
- BPC Comments - Controls data input from posting comments (unstructured data)

- Documents - Controls posting documents with application context to the Content Library (unstructured data)

To add a new work state

1. From the Admin Console, select Work Status.
2. From the action pane, select Add a new work state. The Add a New Work State Step 1 task pane is displayed.
3. Enter the new work state name (up to 25 characters) and a description, then select Add a New Work State Step 2 of 2 in the task pane.
4. For each area (Data Manager, Journals, etc.), select the desired level of security: All, Locked, Manager, or Owner.
5. In the Controlled By field, select who can set the work status the new work state. From the drop down menu, select Owner, Manager, or Both.
6. Select Add a New Work State.

Editing work states

Settings that appear in the Work Status information table can be edited directly.

To edit a work state

1. From the Admin Console, select Work Status.
2. From the table in the center pane click in the cell that contains a value you want to change. For example, click in any cell within the Work State column, then type directly into the cell to change the work state description. Or click any cell within the other columns, then use the drop-down list box to select from the available options.
3. Under the action pane heading Work Status Tasks, click Update work state to save your changes to the database.

Editing work status descriptions

You can edit the descriptions for existing work states. This is useful in further clarifying the purpose of a work state.

To edit work state descriptions

1. From the Admin Console, select Work Status.
2. Select a single Work State in the table displayed in the center pane.
3. Edit the work state description, as desired, then click Edit description at work state.

Reordering work states

The order of the work status follows your business processes. For the ApShell application set, the order is Unlocked, Submitted, and Approved by default. While this may be a natural order based on the code names, you can change the order if it suits your needs.

The top of the list has the lowest level of security. The bottom of the list has the highest level of security.

To reorder work status

1. From the Admin Console, select Work Status. The center pane displays a table showing the current order of work status. The Manage Work Status action pane is displayed.
2. Under the action pane heading Work Status Tasks, click Reorder work states. The Reorder Work State action pane is displayed.
3. Select a desired work state on the action pane. Use the Up or Down arrows to reposition it. When the desired order appears in the Work States table, click Reorder Work States.

## Reporting on work states

The BPC Reporting Console allows you to report on and track work status for submitted data by application.

### To report on work status

1. From 'Getting Started' mode within BPC Web select Launch BPC System Reports from the action pane.
2. Select Work Status Report from the Application Reports section of the action pane.
3. Specify the report parameters, which include start date & time, end date & time, member values for the dimensions that track work status, and page orientation.
4. Click the OK icon in the action pane to generate the report.

## Deleting work states

You may delete a work status that is not currently in use.

### To delete a work status

1. From the Admin Console, select Work Status. The center pane displays a table showing the current order of work states. The Manage Work Status action pane is displayed.
2. Under the action pane heading Work Status Tasks, click Delete a new work state. The Delete Work Status action pane is displayed.
3. Select a desired work state(s) in the Data State table on the action pane and click Delete Work Status Options.
4. Click Yes to delete the selected work state and reset the work status settings for the application. Click No to retain the selected work status and return to the Delete Work Status action pane.

## Managing Business Rules

Business rules provide the mathematical foundation for your BPC applications.

### About business rules

Business rules allow you to customize large data manipulation tasks, such as bulk data imports with currency translations, as well as smaller tasks, such as submitting input data to the database. BPC's business rules support both management and legal consolidation reporting.

You can modify business rules using table-based logic or script-based files. Table-based logic provides the features that were available in the UCON accelerator (Consolidation Engine). Future releases will continue to provide standard calculations with a consistent UI for business rule modifications. Script-based files can be customized using an MDX and SQL syntax.

These are some of the activities for which you can define business rules:

- Initialization of beginning balances when a new fiscal cycle starts. See Carry-forward rules.
- Validation of input data. See Validation rules.
- Conversion of local currency data in the desired reporting currencies. See Currency conversion rules.
- Matching of inter-company transactions. See Intercompany booking rules.
- Generation of all the consolidation entries for the desired groups of entities (eliminations, adjustments, re-classifications, minority calculations, etc.) See Automatic adjustments.
- Other calculations

### Setting up a legal consolidation application set

This section describes how to set up a legal application set. BPC provides several consolidation business rules. The use of these rules depends heavily on the design of the application set.

The first step in setting up a legal consolidation application set is to ensure the dimensions are set up properly. Each application must contain some required dimensions, while some other dimensions are optional.

The dimensions discussed here are based on the standards used in the business rules. Other dimensions can co-exist in a reporting application but do not impact the business rule function.

All applications must contain the four required ENTITY, CATEGORY, TIME and ACCOUNT dimensions (but can be named as desired). The remaining dimensions have the following rules:

- The CURRENCY/GROUP dimension is required for the consolidation and/or currency business rules
- The INTCO dimension used for matching inter-company activity
- The DATASRC dimension is required for elimination and/or consolidation business rules
- The SUBTABLE (flow) dimension is optional, and based on your requirements

### About currency conversions

Your application sets do not need to be set up for currency conversions if every monetary amount is represented in one currency only. If this is the case, the application does not need a Currency-type dimension or a Rate application, because no currency conversions will be performed.

If the majority of your data is entered in just one currency, you do not need a Currency-type dimension. However, a few exceptions might exist, and these can be handled by duplicating a few members in the entity dimension. The classic case is an American corporation having most of its operations in the US, but just a few subsidiaries operating in foreign countries, for example, one in Mexico and one in Canada. This type of scenario can be dealt with by just duplicating the few entities representing the foreign operations, to separate the local currency amounts from the translated amounts. This application would require a Rate application to store the exchange rates needed for the automatic translation, and a property in the Entity dimension, indicating which one is the entity in local currency and which one is the one storing the translated amounts for that entity.

### Simple conversion requirements

The following list provides a summary of the application setup required. (These options are provided by default, with ApShell.)

- The application set must include a Rate application where exchange rates are stored.
- The Entity dimension must include the property CURRENCY (whose values are valid InputCurrencies)
- The Account dimension must include the property RATETYPE (whose values are valid Account members in the RATE application)
- The appropriate FXTRANS logic must be available
- The DEFAULT logic must include a call to the FXTRANS logic, if the translation is to be performed whenever data is entered.
- The Entity dimension must include the property TRANSLATE\_TO (whose value is a valid Entity). For example, the system reads all entities with a non-blank TRANSLATE\_TO property, and translates their values into the entities specified by the TRANSLATE\_TO property.

### Complex conversion requirements

A more complex scenario exists when you need to store data in the local currency as well as in one or more reporting currencies for any member in the Entity dimension. An application of this type will also require the ability to access a Rate application to store the exchange rates used for the automatic translation.

#### Requirements

The following list provides a summary of the application setup required.

- The application set must include a Rate application where exchange rates are stored.
- The application must include a Currency-type dimension
- The Currency dimension must include the property REPORTING (whose values are Y or blank)
- The Entity dimension must include the property CURRENCY (whose values are valid InputCurrencies)
- The Account dimension must include the property RATETYPE (whose values are valid Accounts in the Rate application)
- The appropriate FXTRANS logic must be available
- The DEFAULT logic must include a call to the FXTRANS logic, if the translation is to be performed whenever data is entered

The default translation reads all values in local currency (Currency = LC), applies the correct exchange rate, and writes the results in the appropriate reporting currency (USD, EURO, etc.).

#### Selecting the correct rate

For the selection of the correct rate, the following rules apply to both simple and complex options. The source currency is derived by the property CURRENCY of the entity being translated.

The type of rate (AVG, END, etc.) is derived by the property RATETYPE of the account being translated.

The valid rate types are those corresponding to an account of the RATE application belonging to the GROUP 'FX Rate'.

Any Account with a RATETYPE that is not part of the GROUP 'FX Rate' will be translated with a factor of 1. A special case is the reserved rate named NOTRANS, which will cause the account to be ignored during the translation. This rate does not need to exist in the list of rates in the RATE application.

#### Notes:

The default currency translation supplied with the product for multi-currency applications performs a cross-rate translation, i.e., it multiplies the amount in local currency by the ratio between the rate of the destination currency and the rate of the source currency. This allows the application to use only one table of rates for translating any source currency into any destination currency.



Other types of default translations can be defined by using the business rules tables to support these types of currency translations:

- Ability to use different tables of rates by reporting (destination) currency
- Ability to distinguish between "Multiply" currencies and "Divide" currencies

### About intercompany eliminations

The Intercompany dimension type (I) is required in an application if you want to perform currency conversions and intercompany eliminations. The following are required to support an application to perform intercompany eliminations.

- The application must include a dimension of type "I" (intercompany).
- The intercompany dimension must include the property ENTITY (whose values are Entity names)
- The Account dimension must include the property ELIMACC (whose values are Account names)
- The Entity dimension must include the property ELIM (whose values are Y or blank)
- The appropriate business rule table must be set up
- A DTS package executing the intercompany logic must be available

All the above will not only make sure that intercompany details can be entered for any account, but it will also support an automatic elimination-by-level for all desired accounts.

The default elimination logic, to do its job, will be driven by the values entered in the following properties:

Dimension	Property	Length	Content
Account	ELIMACC	20 chars	A valid account in this dimension
Entity	ELIM	1 char	Y or blank
Intercompany	ENTITY	20 chars	The entity ID corresponding to this intercompany member
Currency	REPORTING	1char	Y or blank

The default elimination logic does the following:

- Scans all base level non-elimination entities (entities having the property ELIM <> Y).
- In case the application has a currency dimension, restrict its action to all reporting currencies only (currencies having the property REPORTING=Y - Data in local currency cannot be eliminated because they are in different currencies).
- Eliminate all values of the accounts to be eliminated (accounts having property ELIMACC<>blank) into the desired plug account (the account specified by the ELIMACC property itself).
- The elimination will be performed in the "elimination entity" below the first "common parent" (\*).  
(\*) The "common parent" is derived as follows:

The system identifies the 2 entities for which a common parent must be found. The first entity is the current entity member. The second entity is the entity corresponding to the current intercompany member. This entity is obtained reading the content of the property ENTITY of the current intercompany member.

The system searches (in a selected entity hierarchy) the first member that has both entities as descendants. This is the "common parent".

Then the system searches, in the immediate descendants of such common parent, a valid "elimination entity" (an entity having the property ELIM=Y).

This is the entity where the results of the elimination will be stored.

Remarks:

- The default elimination logic does it searches in the first organization of the entity dimension. This can be modified to have the elimination performed in all hierarchies existing in the entity dimension
- If no common parent is found, no elimination will take place
- If no elimination entity is found below the first common parent, the next common parent will be searched.

## Adding business rules tables to applications

By default, applications are predefined with a set of business rules tables. For example, the Finance application comes with the Currency Conversions table, and the LegalApp application comes with all the available tables: Calculations, Currency Conversions, Intercompany Bookings, Automatic Adjustments, Carry-Forward, US Eliminations, and Validations.

You can add available tables to an application, or remove tables from an application.

To add business rules tables to applications

1. From the Admin Console, select Application.
2. Select the application for which you want to add the business rule table.
3. From the Manage Applications action pane, select Modify application.
4. Select Change Application Type.
5. Select Modify Application.
6. Under Select Application Option, select the check boxes next to the business rules tables you want to add to the application. Deselect the check boxes next to the business rules you want to remove from the application.
7. Select Modify Application.
8. When the progress indicator completes, click OK.

## Defining rules

You can define rules for the following types of business processes.

### Adding business rules

You can add new business rules to any application's business rule table. After adding a rule, you should validate and save it.

Upon validation, rules files are converted to LGL files. The BPC rules engine processes the LGL files and converts them to LGX files, which contain database code ready to be run by the SQL or MDX engines.

To add a business rule

1. From the Admin Console, select the application for which you would like to create a business rule, then click the Business Rules folder.
2. Select the business rule table for which you would like to add a rule.
3. Enter the appropriate parameters for the rule. See Defining rules.
4. If you would like to save the rule without validating the data entries, select Save without validation from the action pane, or if you would like to validate and save the rule, select Validate <business rule table> rule table from the action pane.

### Running business rules

There are several ways to run a business rule.

- Perform a data send. When data is sent through BPC for Office or BPC Web, the system runs the default logic (default.lgf). The default logic typically runs the appropriate currency translation business rules. While you can add other logic to the default script, we do not recommend it, as performance may decrease whenever you perform a send task.
- Run a Data Manager package (through DTS or SSIS) that contains a reference to a logic file (\*.LGF). Logic can be applied to a specified region of data that is stored in the database using any type of logic. You can create new logic files or modify existing ones. For example, an Import package runs against the default logic unless the Import package is modified to include other logic files. Each logic file contains one or more stored procedures that runs a specific set of logic commands.

- Post a journal entry. Upon posting a journal, the system runs logic that is specific to journal processes. To overwrite the default journal logic, you can create a new script logic file called Journal.lgf.

#### Account transformation rules

You can create a new logic file to run an account transformation rule. You run this rule by calling the stored procedure SPRUNCALCACCOUNT in any logic file.

This procedure is launched using the logic statement:

```
*RUN_STORED_PROCEDURE=SPRUNCALCACCOUNT(parameters list)
```

For example:

```
*RUN_STORED_PROCEDURE=SPRUNCALCACCOUNT([LEGALAPP], [ACTUAL], [USD],
[%SCOPETABLE%], [%LOGTABLE%],[Transformation group],[MAXSTATUS]
)
```

The following table describes the fields for the account transformation rules table.

Field Name	Description
Transformation group	The identifier for a group of calculations.
Source account	The name of the source account. You can also select values from TYPELIM* and DIMLIST*.
Source flow	The source flow member. You can also select from DIMLIST*.
Source data source	The name of the source DataSrc member. You can also select from DIMLIST*.
Destination account	The name of the destination account member. This must be a base member.
Destination flow	The destination flow member. This must be a base member.
Destination data source	The name of the destination DataSrc member. This must be a base member.
Reverse sign	If selected, the value of the amount is reversed.
Source period	Enter the month to process. It can be an absolute or a relative amount. Leave blank to use the current month.
Source year	Enter a year to process. It can be an absolute or a relative amount. Leave blank for the current year.
Apply to YTD	If selected, the YTD value will be used in the calculation in a PERIODIC application.
Remark	A description for the rule.

#### Currency conversion business rules

Currency conversion typically runs by default when default logic runs. You can also create a new logic file to run a currency conversion.

You run this rule by calling the stored procedure SPRUNCONVERSION from any logic file.

```
*RUN_STORED_PROCEDURE=SPRUNCONVERSION('APPLICATION','CATEGORY_SET%',
'%CURRENCY_SET%', 'GLOBAL', '%SCOPETABLE%', '%LOGTABLE%', impact, destination,
ratedimension, maxstatus)
*COMMIT
```

Parameter	Description
Application	The application ID.
Category	The Category ID.
Currency	The currency, Yes , or Multi.
Rateentity	The default Entity member for the Rate application.
%SCOPETABLE%	% SCOPETABLE% is a variable used by the system to define the name of a logic table.
%LOGTABLE%	% LOGTABLE% is a variable used by the system to define the name of a Log table.
Impact	Not used, but you must enter "N".
Destination	1 = WB; 2 = Fac2
Ratedimension	The member of the simulation dimension, if used.
Maxstatus	The maximum status level allowing write access.

The stored procedure SPRUNCONVERSION scans all records found in the selected region of data and translates them according to the RATETYPE property assigned to the ACCOUNT specified in each record, based on the following mechanism:

- All ACCOUNTS with no RATETYPE (ratetype = blank) will be translated with a factor of 1
- All ACCOUNTS with the reserved RATETYPE = NOTRANS will not be translated
- All other ACCOUNTS will be translated according to the definitions contained in the table of parameters called clcFXTRANS.

For information on required properties of the Category dimension, see Setting up a legal consolidation application set. For information on required properties of the Data Source dimension, see Setting up a legal consolidation application set.

The following table describes the fields for the currency conversion rules table.

Field name	Description
Account rate type	<p>The currency conversion type, for example, AVG, END, HIST, taken from the Account dimension's RATETYPE property. This is the main driving field, controlling the translation rule to apply to a given account.</p> <p>Note that one RATETYPE may generate more than one translated value. This can be defined by entering more than one entry with the same RATETYPE in the business rule interface for currency rules.</p>
Source flow	<p>This field, combined with the 'Account rate type' field, completes the definition of the criteria that drive the applicability of a given rule. In other words, the instruction can be read as follows: "If the account has such RATETYPE and the FLOW is such and such, then apply this rule."</p> <p>This field may contain the following values:</p> <ul style="list-style-type: none"> <li>• A valid base level or parent member ID from the FLOW dimension</li> </ul>

Field name	Description
	<ul style="list-style-type: none"> <li>A list of members of the FLOW dimension, as defined filtering the members using a value of the DIMLIST property (or any property whose name begins with DIMLIST)</li> </ul> <p>For information about the Flow dimension, see Setting up a legal consolidation application set.</p>
Destination account	The account that stores the specific conversion. If empty, it is the same as the source account's member.
Destination flow	The specific Accdetail member where translations are stored. If empty, then it is the same as the source subtable's member.
Formula	This field can contain any arithmetic expression combining any defined RATE as per the ACCOUNT dimension of the RATE application. All rates with the RATETYPE property value of FX TRANS can be used.
Force closing	If checked, force the generation of an additional record where the member of the destination FLOW is the closing balance. The closing balance member of the FLOW dimension is identified as the member with the property FLOW_TYPE = CLOSING.
Apply to periodic	<p>This field should only be used in a YTD application for the cases in which the currency conversion should be performed on the PERIODIC values and not on the YTD values.</p> <p>If the box is checked, the engine calculates the difference between current period and prior period amounts, and it applies to it the rate specified in RATE_FORMULA field. At the end, the result is added back to prior period's value as written in current period.</p>
Entity FX type	This field can be used to enforce a given set of rules to only apply to a desired set of ENTITIES. If this field has a value, the rule will only be applied to the entities having a matching value in a similarly named property (FX_TYPE) of the ENTITY dimension.
Remarks	A short description for this rule.

#### Formula field

In the FORMULA field, the rates must be enclosed in square brackets:

Examples:

```
[END]
[END] - [AVG]
```

The OPENING value of any rate can also be specified adding the prefix "OPE" to the rate itself.

Examples:

```
[OPEEND] - [END]
[OPEAVG]
```

These OPENING rates do not need to exist in the RATE cube. For example, if there is an [END] rate, the currency translation will also automatically recognize a rate called [OPEEND], which simply corresponds to the [END] rate of the OPENING period (typically last period of last year).

In addition, the RATE\_FORMULA field supports the following keywords:

- [AS\_IS] Leave untouched a value already existing in the destination currency. This keyword cannot be combined with other rates in the same line. The only valid exception is the format [AS\_IS]\*-1

- [COPYLC] This correspond to applying a rate of 1 in the translation

#### How the RATE table is selected

While most customers require a single table of rates, there are situations when more than one set of rates is required. In this situation, the translation procedure uses the RateEntity dimension to select the correct table of rates to use.

Whenever a destination currency is selected, the procedure searches for a RateEntity member flagged with this currency in the Currency property. For example, if translating into USD, the system uses the RateEntity member that has the Currency property set to USD.

If there is no RateEntity flagged as the destination currency, the system will use the RateEntity with Currency = " (blank).

In addition to this, some exceptions by ENTITY can be applied. For example, some entities just entering in the consolidation perimeter may need to be converted at their own specific set of rates. These entities may have a corresponding RateEntity member in the RATE cube. All ENTITIES having a corresponding RateEntity member in the RATE cube will use that member as rate table. For example, if there is a RateEntity member named like the ENTITY USOps, the RateEntity member USOps will be used to translate the values of entity ENTITY USOps.

The RateEntity member, when representing an ENTITY, may be any of the following:

- A valid base level or parent member ID from the ENTITY dimension of the main cube
- A list of members of the ENTITY dimension, as defined filtering the members using a value of the DIMLIST property (or any property whose name begins with "DIMLIST") of such dimension.

#### Carry-forward rules

Carry-forward rules give you the ability to generate the Opening Balance of any category based on two main properties:

- Flow\_Type in the Flow dimension
- DataSrc\_Type in the DataSrc dimension

This procedure can be used to initialize a new reporting period with the closing balances of the last period from the previous year into the opening balances of the current period.

In a legal consolidation application, such balances are usually identified as members of the FLOW dimension. In simpler applications, however, it is also possible to store them as additional accounts in the ACCOUNT dimension.

Important remark: Currently this procedure is only limited to the copy of the opening balances as found in the DATASRC dimension members flagged as "I" and "M" in the DATASRC\_TYPE property. This means that the procedure will only copy the "Input" balances and their related "Manual" adjustments. The balances generated "Automatically" by the consolidation procedure (DATASRC members flagged as "A") are taken care of during the consolidation process by the consolidation procedure itself.

The copy-opening process is handled by a stored procedure called SPCOPYOPENING. This procedure is launched using the logic statement:

```
*RUN_STORED_PROCEDURE=SPCOPYOPENING(parameters list)
```

Example:

```
*RUN_STORED_PROCEDURE=SPCOPYOPENING([LEGALAPP], [ACTUAL], [USD],
[%SCOPETABLE%], [%LOGTABLE%] )
```

For information on required properties of the Category dimension, see Setting up a legal consolidation application set.

The following table describes the values you can enter for the carry-forward rules.

Field name	Description
Source account	This is the main driving field, controlling the rule to apply to a given account. This field may contain the following values:

Field name	Description
	<ul style="list-style-type: none"> <li>A valid base level or parent level member ID from the ACCOUNT dimension</li> <li>A list of members of the ACCOUNT dimension, as defined filtering the members using a value of the DIMLIST property (or any property whose name begins with DIMLIST) in such dimension</li> </ul>
Source flow	This field controls the applicability of the rule to a given member of the FLOW dimension. Typically this field will contain the ID of the closing balance member. Valid values for this field are identical to the source account above.
Destination account	This is the destination account for the rule. Valid values for this field are identical to Source Account above. If left blank, the destination account will be the same as the source account.
Destination flow	This field controls the ID of the destination FLOW. If left blank the FLOW will be the same as the source FLOW. Valid values for this field are identical to the source account above.
Reverse sign	If selected, the system reverses the value of the amount.
Data source type	Input Only, Manual Only, or ALL.
Same period	If selected, the source TIME period is the same as the destination.
Apply to YTD	If selected, the YTD value to copy will be calculated if its a PERIODIC application.
Remark	A description for this rule.

### Intercompany booking rules

Intercompany booking rules are split into two independent procedures:

- **SPICDATA:** This procedure can be used to copy the declarations of all entities versus a given entity by intercompany account. Essentially, it concentrates into each single entity the declarations of all other entities versus each entity. This mechanism allows the owners of an entity to run a report matching all its declarations against what the rest of the world has declared against the entity, without the need to assign to each owner read permits into other entities.
- **SPICBOOKING:** This procedure can be used to automatically generate the bookings which will make the intercompany declarations to match. The stored procedure SPICBOOKING is driven by the table called clcBOOKING\_{app}.

For information on the intercompany application structure, see Setting up a legal consolidation application set.

The following table describes the options for setting up an intercompany booking rule.

Field Name	Description
Remark	A description for this rule.
Type	Seller rule, Buyer rule, or Greatest amount.
Parent matching account	A parent account member from the Account dimension or DIMLIST.
Other destination members	Blank or force destination. Used to force a target destination member.
Booking destination data source	A DataSrc member.



Field Name	Description
Max booking amount	The maximum amount authorized to book.
Debit account	An Account member.
Debit flow	A Flow member.
Debit intco	An Intco member or blank.
Credit account	An Account member.
Credit flow	A Flow member.
Credit intco	An Intco member or blank.

#### US Elimination rules

A US Elimination rule controls where in an Entity dimension member the results of eliminations are stored.

You run this rule by calling the stored procedure SPRUNELIM. This stored procedure performs the eliminations driven by the hierarchies in the Entity dimension and writes the elimination in an Entity member (property ELIM=Y).

The following table describes the fields to define in the US Elimination table.

Field Name	Description
Source data source	The data source member from which the amount is taken.
Destination data source	The data source member from which the amount is applied.
Remark	A user-defined description.

#### Automatic adjustments

The automatic adjustments process is handled by a stored procedure called SPRUNCONSO. This procedure is typically launched using the logic statement:

```
*RUN_STORED_PROCEDURE=SPRUNCONSO(parameters list)
```

Example:

```
*RUN_STORED_PROCEDURE = SPRUNCONSO([LEGALAPP], [ACTUAL], [2006.MAR],
[GROUP1], [%LOGTABLE%])
```

The driving elements of this procedure are:

- The consolidation METHOD by which each ENTITY is consolidated into a GROUP
- The consolidation RULE assigned to each ACCOUNT to consolidate
- The ELIMINATION mechanism, as controlled by each RULE (in turn driving the behavior of each ACCOUNT)

Basically, the whole process can be explained as follows:

Each source ACCOUNT is assigned a set of DESTINATION ACCOUNTS and a RULE of behavior through the definitions entered in the Automatic Adjustments business rule table.

For information on required properties of the Data Source dimension, see Setting up a legal consolidation application set.

The following table describes the values you can enter for the automatic adjustment rules.

Field Name	Description
------------	-------------

Field Name	Description
Adjustment ID	The identifier for the adjustment rule.
Source data source	<p>This field may contain the following values:</p> <ul style="list-style-type: none"> <li>Blank (all DATASRC members)</li> <li>A valid base level or parent member ID from the DATASRC dimension</li> <li>A list of members of the DATASRC dimension, as defined by filtering the members using a value of the DIMLIST property (or any property whose name begins with "DIMLIST") in such dimension.</li> </ul>
Destination data source	This field must contain a valid base level member ID of the DATASRC dimension having the property DATASRC_TYPE = A ("Automatic elimination")
Group type filter	<p>The CONSO_TYPE property value from the Groups (Currency) dimension.</p> <p>If, for example, you have the GROUP GROUP1 with conso_type = A, all the Automatic Eliminations with Blank are available for all groups.</p> <p>The Automatic Eliminations with Group_filter member = A are available for GROUP1 and all groups with Group filter member = A.</p>
Adjustment type	<p>This field may contain the following values:</p> <ul style="list-style-type: none"> <li>Blank: A generic ELIM rule</li> <li>E: "Equity" - This value indicates that this rule is used on the entities using an Equity method (i.e., having a method of type "E")</li> <li>P: "Proportional" - This value indicates that this rule is used on the entities using a Proportional method (i.e. having a method of type "P")</li> <li>N: "New" - This value indicates that this rule is used on the entities just entering into the consolidation perimeter (i.e., having a method of type "N")</li> <li>L: "Leaving" - This value indicates that this rule is used on the entities that are leaving the consolidation perimeter (i.e. having a method of type "L")</li> </ul> <p>In case this field has a non-blank value, there are never details associated with it in the clcELIM table.</p>
Adjustment level	This field must contain an integer value ranging from 0 to 99 and can be used to order the sequence of execution of the various ELIM rules, if desired.
Entity property filter	This field is only utilized for intercompany eliminations. If this field contains a value, the system will look for the existence of a corresponding PROPERTY in the ENTITY dimension, and will only perform the elimination if the ENTITY and the INTCO partner have the same value in this property.

Field Name	Description
Other dimension filter	<p>This field can be used to define a filter controlling the region where the elimination can be applied. It can contain the following values:</p> <ul style="list-style-type: none"> <li>Blank (No Filter)</li> <li>An expression defining one or more filters. These can be applied to any combination of the following information: <ul style="list-style-type: none"> <li>Any dimension of the MAIN cube (example: CATEGORY=BUDGET)</li> <li>The SIGNEDDATA amount (example: SIGNEDDATA&gt; 10)</li> <li>Any ACCOUNT of the OWNERSHIP cube (example: METHOD = 85)</li> <li>Any ACCOUNT of the OWNERSHIP cube as defined in Prior period (example: PPOWN &lt;&gt;0, where PPOWN is Prior period's POWN account)</li> <li>Any ACCOUNT of the OWNERSHIP cube as assigned to the INTER_COMPANY partner (example: I_POWN &lt; POWN)</li> </ul> </li> </ul> <p>The filters can be concatenated with the "AND" operand. Alternatively, any valid SQL expression filtering the values in any desired way can be used. In this case the syntax must strictly comply with SQL requirements (single quotes around the members IDs, etc.)</p>
Forced destination members	<p>This field can be used to enforce the destination member of any dimension existing in the MAIN cube.</p> <p>This field may contain the following values:</p> <ul style="list-style-type: none"> <li>Blank (No Change)</li> <li>An expression enforcing the value for one or more dimensions of the MAIN cube. Each dimension must be separated by a comma (example: CATEGORY=BUDGET, ENTITY= NewYorkBranch)</li> </ul>
Local currency	<p>This field can be used to control the reporting currency for which the elimination is to be performed. It may contain either the ID of a valid reporting currency or it may be left blank, to indicate that the GROUP currency must be used.</p>
Remark	A description of the rule.

#### Automatic adjustments detail

The automatic adjustments detail table defines the details of the automatic adjustment rules and their behavior. These rules control the association of each automatic adjustment rule with the existing ACCOUNTS and, among other things, controls the ID of the ACCOUNTS into which a given account is transformed.

Field Name	Description
Adjustment ID	The identifier for the adjustment rule.
Source account	<p>This field may contain the following values:</p> <ul style="list-style-type: none"> <li>A valid base level or parent member ID from the ACCOUNT dimension</li> <li>A list of members of the ACCOUNT dimension, as defined filtering the members using a value of the DIMLIST property (or any property whose name begins with "DIMLIST") in such dimension.</li> <li>A list of members of the ACCOUNT dimension, as defined filtering the members using a value of the TYPELIM property (or any property whose name begins with "TYPELIM") in such dimension.</li> </ul>
Reverse sign	If "Y" the value of the SIGNEDDATA amount is reversed.

Field Name	Description
Destination "All" account	<p>This field defines the ID of the first of four possible destination ACCOUNTs. It is linked to the "ALL" formula field of the Consolidation Rules Formulas table, which defines how the amount is generated.</p> <p>This field may contain the following values:</p> <ul style="list-style-type: none"> <li>Blank (the destination Account is equal to the source Account)</li> <li>A valid base level or parent member ID of the ACCOUNT dimension</li> <li>A list of members of the ACCOUNT dimension, as defined by filtering the members using a value of the DIMLIST or the TYPELIM property (or any property whose name begins with "DIMLIST" or "TYPELIM") in the dimension.</li> <li>A list of members of the ACCOUNT dimension, as defined by filtering the members using a value of the DIMLIST property (or any property whose name begins with "DIMLIST") in the dimension.</li> <li>The expression PROP(PropertyName), where PropertyName is the name of a valid PROPERTY in the ACCOUNT dimension. In this case, the destination account is the value of the property for the current source ACCOUNT.</li> </ul> <p>Use caution when the destination represents a list of accounts, as multiple records will be generated.</p>
Destination "Group" account	<p>This field defines the ID of the second of four possible destination ACCOUNTs. It is linked to the Group formula field from the Consolidation Rules Formulas table, which defines how the amount is generated.</p> <p>This field may contain the same elements as defined in the Destination "All account field. However, if left blank, it will NOT default to the source account, but it will generate NO record.</p>
Destination "Minority" account	<p>This field defines the ID of the third of four possible destination ACCOUNTs. It is linked to the Minority formula field from the Consolidation Rules Formulas table, which defines how the amount is generated.</p> <p>This field may contain the same elements as defined in the Destination "All" account field. However, if left blank, it will NOT default to the source account, but will generate NO record.</p>
Destination "Equity" account	<p>This field defines the ID of the fourth of four possible destination ACCOUNTs, and has a behavior of its own. The value is linked to the "ALL" formula field of the from the Consolidation Rules Formulas table, which defines how the amount is generated. However, it applies only to the Equity methods.</p> <p>The account defined in this field will replace the value of the Destination "All" account field, in case the current ENTITY is being consolidated using a METHOD of type "E" (Equity).</p> <p>This field may contain the same elements as defined in the Destination "All account field. If left blank, the definitions of the "ALL" formula field will apply.</p>
Rule ID	<p>This field must contain the name of a valid Rule ID from the Consolidation Rules table. This rule is used to define how to calculate the amount of the different destination ACCOUNTS.</p>

Field Name	Description
Source flow	<p>This field may be used to restrict the applicability of the current Elimination to some members of the FLOW dimension. It may contain:</p> <ul style="list-style-type: none"> <li>Blank (any FLOW members)</li> <li>A valid base level or parent member ID from the FLOW dimension</li> <li>A list of members of the FLOW dimension, as defined filtering the members using a value of the DIMLIST property (or any property whose name begins with "DIMLIST") in such dimension.</li> </ul>
Destination flow	<p>This field can be used to enforce the ID of the member of the FLOW dimension. It only supports an explicit FLOW ID or can be left blank to leave the member unchanged.</p>
Force closing	<p>This field can be set to Y to force the generation of an additional record where the member of the destination FLOW is the closing balance. The closing balance member of the FLOW dimension is identified as the member with the property FLOW_TYPE = CLOSING.</p>
Forced Intco member	<p>This field can be used to enforce the ID of the member of the INTCO dimension. It only supports an explicit INTCO ID or can be left blank to leave the member unchanged.</p>
Swap Entity - Intco	<p>This field may contain the following values:</p> <ul style="list-style-type: none"> <li>Blank or N: Nothing should be changed</li> <li>Y: The ID of the Entity dimension member is swapped with the ID of the INTCO dimension member.</li> </ul>
Apply to periodic	<p>This field should only be used in a YTD application for the cases in which the ownership percentage should be applied on the PERIODIC values and not on the YTD values.</p> <p>If this field is set to Y, the engine calculates the difference between current period and prior period amounts, and it applies to it the Ownership percentage specified in the "ALL" formula, Group formula, and Minority formula fields of the Consolidation Rules table. At the end, the result is added back to prior period's value as written in current period.</p>
Remark	<p>A description of the rule.</p>

### Validation rules

Validation rules allow you to check the integrity and correctness of entered values, before signing off such data as "approved." The action of this procedure is limited to the comparison of several accounts (or sets of accounts) and to post the difference, if any, into some "error" account. The purpose should be to have zeros in all these "error" accounts.

The validation process is handled by a stored procedure called SPRUNVALID.

This procedure is launched using the logic statement:

```
*RUN_STORED_PROCEDURE=SPRUNVALID( parameters list)
```

For example:

```
*RUN_STORED_PROCEDURE=SPRUNVALID([LEGALAPP], [ACTUAL], [USD],
[%SCOPETABLE%], [%LOGTABLE%])
```

The following table describes the fields to define in the validation table.

Field Name	Description
Validation account	The member ID of the 'error' account.
Remark	A description for the rule.
Validation operand	<p>=, &lt;, &gt;, &gt;=, &lt;=</p> <p>If the sign is =, the total of the left account must be equal to the total of the right account *-1</p> <p>If the sign is &gt;, the total of the left account must be greater than the total of the right account *-1</p> <p>If the sign is &lt;, the total of the left account must be smaller than the total of the right account *-1</p>
Other source dimensions	Blank or a filter criteria for the original data extraction. Used to limit the selection on one dimension.
Other destination dimensions	Blank or a forced destination. Used to force a target destination's dimension.
Applicable periods	Blank or one or more time periods.
Validation tolerance	Used to determine a limitation in the Value.

All validations compare the Left part with the right part (ACCOUNT\_L with ACCOUNT\_R). You can have more than one account in the left or in the right. The system calculates all the accounts on the left and compares it with all the accounts on the right.

Here are some examples:

- TOTAS00000 Total Asset = 1000
- TOTLI00000 Total Liabilities = 990

If the validation sign is =, 10 is generated in the validation account because 1000 is not equal to 990.

If the validation sign is <, 10 is generated in the validation account because 1000 is not smaller than 990.

If the validation sign is >, nothing is generated in the validation account because 1000 is greater than 990.

#### Validation rules detail

In the Fact table, the validation compares the left side (1) with the right side (2).

ACCOUNT1	FLOW1	SIGN1	ACCOUNT2	FLOW2	SIGN2
A0001	F_CLO	+	A1000	F_CLO	+
A0002	F_CLO	+			

In this example, we valid the following:

A0001 (F\_CLO) + A0002 (F\_CLO) = A1000 (F\_CLO)

The following table describes the fields to define in the Validation rules detail table.

Field Name	Description
Validation account	The member ID of the 'error' account.

Field Name	Description
Account 1	The member ID of the "left side" account.
Flow 1	Blank or the "left side" member of the FLOW dimension.
Sign 1	The operator used in the Left part of calculation ( + or - )
Account 2	The member ID of the "right side" account.
Flow 2	Blank or the "right side" member of the FLOW dimension
Sign 2	The operator used in the Right part of calculation ( + or - )
Remark	A description for the rule.

## Using the Business Rule Library

The business rule library allows you to set up global business rules that process legal consolidation data at the application set level.

### About the Business Rule Library

The business rule library allows you to set up global business rules for processing legal consolidation data at the application set level. Once the headers, methods and rules are defined, they apply to all related business rules tables within an application set.

You populate the following library tables:

- Consolidation rules
- Consolidation methods
- Consolidation rules formulas

### Adding business rule library entries

You can add a new business rule to the business rule library. Once defined, the rule can be used in any application in the application set.

#### To add a business rule library entry

1. From the Admin console, click the Business Rule Library folder.
2. Select Consolidation Headers from the navigation pane, then define a rule name, description, and type. See Consolidation Headers.
3. Click Save without validation or Validate Consolidation Rules rule table from the action pane.
4. Select Consolidation Methods from the navigation pane, then add a new method for the rule, if required. See Consolidation Methods.
5. Click Save without validation or Validate Consolidation Rules rule table from the action pane.
6. Select Consolidation Rules from the navigation pane, then define the parameters for the rule. See Consolidation Rules.
7. Click Save without validation or Validate Consolidation Rules rule table from the action pane.

### Consolidation rules

This simple table defines the list of business rules that can be used in the application set. The only extra thing it does (other than giving them an ID and a description), is restrict their applicability, if desired, to just some type of consolidation METHOD.

You define a name and description, and a consolidation method. The consolidation method (Rule type) limits the use of the rule to the specified type of consolidation method.

Field Name	Description
Rule ID	The identifier for this rule. For example, RULE01.
Rule description	A description for this rule. For example, Equity, 100% Minority Part, Dividends, Stock Holder Equities, or IC elimination.
Rule type	The consolidation method: <ul style="list-style-type: none"> <li>• Proportional - Uses Proportional method elimination rules</li> <li>• Equity - Uses Equity method elimination rules</li> <li>• ALL (or blank)</li> </ul>

### Consolidation methods

The Consolidation Method table describes the Entity methods used in the consolidation rules.

Field Name	Description
Method code	The method code. The value can be a single value (10) or a list (10,20,25,30).  99 - All Intco (even if the Entity is not in the group)
Method name	The description for this method. For example, Leaving, Equity, Proportional, Global, or Holding.
Method type	The Entity method type. For example, New, Holding, Global, Proportional, Equity, Leaving (End of Year), Leaving (During the Year)

### Consolidation rules formulas

The Consolidation Rules Formula table controls how the amounts or the destination accounts should be calculated. The behavior is controlled by the rule being used, the consolidation method assigned to the current entity, and its INTERCOMPANY partner, if applicable.

Field Name	Description
Rule ID	The name of the rule. The drop-down list contains the rules defined in the Consolidation Rules table.  See Consolidation Rules
Entity Method	A valid Entity method, as defined in the Consolidation Methods table, or a list of entity methods separated by commas.  See Consolidation Methods
IntCo Method	A valid Entity Method, as defined in the Consolidation Method table, a list of entity methods separated by commas, or 99 for all methods.  See Consolidation Methods
"All" Formula	An expression that represents the percentage (or formula) to apply to the Destination "All" account property from the Automatic Adjustment Detail table. The value can be an arithmetic expression combining any defined percentage in the Account dimension of the Ownership application. All percentages where the property 'IS_INPUT' is equal to 'Y' can be used.



Field Name	Description
	<p>Here are some guidelines:</p> <ul style="list-style-type: none"> <li>• The members must be enclosed in square brackets. For example, [POWN], [PCTRL], [POWN]</li> <li>• Add the prefix 'P' to the percentage to use the Prior value. For example, [PPOWN], [PPVOTE]</li> <li>• Add the prefix 'I_' to the percentage to add the Intco value. For example, [I_POWN]</li> <li>• The syntax of the prior value can be combined with the syntax of the INTCO value as follows: [I_PPOWN]</li> </ul>
Group formula	<p>An expression that represents the percentage (or formula) to apply to the Destination "Group" account from the Automatic Adjustment Detail table.</p> <p>The value can be an arithmetic expression combining any defined percentage as per the Account dimension of the Ownership application. All percentages where the property 'IS_INPUT' is equal to 'Y' can be used. The percentage must be enclosed in square brackets. (See the "All" formula field, above, for further details).</p>
Minority formula	<p>An expression that represents the percentage (or formula) to apply to the Destination "Minority" account from the Automatic Adjustment Detail table.</p> <p>An arithmetic expression combining any defined percentage as per the Account dimension of the Ownership application. All percentages where the property 'IS_INPUT' is equal to 'Y' can be used. The percentage must be enclosed in square brackets. (See the "All" formula field, above, for further details).</p>
Remark	Text description for this rule.

## Using Script Logic

Script logic allows you to write your own logic files using an MDX or SQL scripting language.

### Logic Assistant

You can use the Logic Assistant for logic functions; for example, you can look up functions, choose parameters, and then paste a formatted statement into your logic file.

You can look up both Logic module statements and user-defined functions. Use the UDF (user-defined functions) tab of the logic assistant dialog box to look up and use UDF functions. The UDF tab displays both system-supplied functions and user-defined functions.

To use the Logic Assistant

1. Create a new logic file or open an existing logic file.
2. Click the Logic Assistant menu on the action pane.
3. Select the UDF tab to look up a UDF function or the Logic tab to look up a logic statement.
4. Select a statement category and then choose a statement or function name.
  - o A short description of the statement or function is displayed in the dialog box. To see a detailed description select the More info button.
5. Select the Next button. The Logic Assistant displays a statement or UDF function wizard. Depending on the type of statement or function, the system guides you to creating the new logic statement. After completing each screen, select Next to continue.
6. When you reach the final screen, select the Paste button to add the statement to your logic file.
7. The logic assistant stays open so that you can create a new statement. Select the Close button when you are finished adding logic statements.

### Adding UDF functions to the Logic Assistant

You can add UDF functions to the Logic Assistant by creating your own logic libraries and placing them into the Logic Library directory on the server. The Logic Assistant automatically reads the Logic Library directory and adds your UDF functions to the UDF tab.

To add UDF functions to Logic Assistant, take the following steps:

1. Create your logic library file and save it with an .LGL extension.
2. Copy the logic library file to the <BPC>\Webfolders\<AppSet>\SystemLibrary\Logic Library folder. Where <BPC> is the root directory for BPC on the server and <AppSet> is the application set name. For example:  
C:\BPC\Webfolders\ApShell\SystemLibrary\Logic Library
3. Click the Logic Assistant menu on the action pane.
4. Select the UDF tab, and then select the file you created in the Statement category list to display your UDF functions in the Statement name list on the right.

## Using the Logic Debugger

The Logic Debugger allows you to test and run your logic files and to debug logic before you run it against live data. This applies to both default logic and other logic files; default logic files are in the default.xls workbook for each application and other logic is in workbooks with user-defined names.

- You can run logic against live data while using the debugger by using the Simulation option on the General tab.

The debugger has four tabs that define how you want to run your logic files.

- General

The general tab determines if you want to simulate expected results where changes do not affect the database, to send differences to the database, and to view log file setup.

- Region

This tab allows you to set up the member set that the logic is run against. You can choose all members or allow the logic system to base the region on the members used in a data file. If you choose members, you can leave a dimension blank to select all members, but you must choose at least one member of one dimension.

- Dynamic Formula

This tab allows you to enter a logic formula that is combined, at the beginning, with the logic file you are testing. This is helpful if you want to set up certain conditions, but do not want to save those conditions in the logic file after testing.

- Optimize option

This tab allows you to set advanced logic optimization options. You can set the memory management and the MDX Query type. There are three types of MDX queries that can be generated by the logic engine:

- Multit-axis - Builds the MDX query based on the number of dimensions in the member set that have more than one member. An axis is assigned to each dimension that has more than one member in the member set. The sub-cube for the query therefore can have many axes depending on the member set or region of data against which the query is run.
- 2-axis, crossjoin - Queries that are "flattened" so that they have no more than 2 axes by nesting multiple dimensions in the rows of the query. For the crossjoin type, the query first checks that a value exists in each cell in the application (MSAS) database. If data exists in any of the cells referenced by the query, those cells are included in the query, otherwise they are ignored.
- 2-axis, nonemptycrossjoin - Queries that are "flattened" so that they have no more than 2 axes by nesting multiple dimensions in the rows of the query. For the nonemptycrossjoin type, the query checks that a value exists both in the fact table (SQL) and application (MSAS). If data exists in both the fact table AND the application database, those cells are included in the query, otherwise they are ignored.

#### To use the logic debugger

1. Open Admin Console and go to the Script Logic menu and select the logic file. You can then see the Logic Debugger menu on the action pane.

Note: If you run the Logic Debugger without a logic file open, you are asked to open a logic file first.

2. Select each tab and choose the options you want. On the Dynamic formula tab, you can choose to use the Logic Assistant to paste logic functions in the Dynamic Formula area.
3. Select the Run Logic button.
4. When finished, select the Exit button.

## The BPC MDX library

BPC provides several MDX functions that you can use in your dimension rule formulas. You can use some of these MDX functions in advance rule formulas as well.

A majority of the MDX functions define industry-standard financial ratios. You can use ratios to evaluate the performance of your business and identify potential problems. Ratios expose factors such as the earning power, solvency, efficiency, and debt load of your business.

This section also discusses user-defined functions. The term 'user-defined functions' comes from Microsoft Analysis Services. BPC supplies user-defined functions, but you can create your own user-defined functions as well.

#### Common MDX functions

The following table describes some of the more common MDX functions. For additional MDX functions, see the Analysis Services Manager Help and look for help on MDX expressions and the function list.

MDX Function	Description
Ancestor	Returns the ancestor of a member at a specified level.
ClosingPeriod	Returns the last sibling among the descendants of a member at a level.
Cousin	Returns the member with the same relative position under a member as the member specified.
Current Member	Returns the current member along a dimension during an iteration.
Default Member	Returns the default member of a dimension.
FirstChild	Returns the first child of a member.
FirstSibling	Returns the first child of the parent of a member.
IsEmpty	Determines if an expression evaluates to the empty cell value.
Item	Returns a member from a tuple.
Lag	Returns a member prior to the specified member along the member's dimension.
LastChild	Returns the last child of a member.
LastSibling	Returns the last child of the parent of a member.
Lead	Returns a member further along the specified member's dimension.
Members	Returns the member whose name is specified by a string expression.
NextMember	Returns the next member in the level that contains a specified member.
OpeningPeriod	Returns the first sibling among the descendants of a member at a level.
ParallelPeriod	Returns a member from a prior period in the same relative position as a specified member.
Parent	Returns the parent of a member.
PrevMember	Returns the previous member in the level that contains a specified member.
Aggregate	Returns a calculated value using the appropriate aggregate function, based on the context of the function.
Avg	Returns the average value of a numeric expression evaluated over a set.
CoalesceEmpty	Coalesces an empty cell value to a number.
Correlation	Returns the correlation of two series evaluated over a set.
Count	Returns the number of tuples in a set, empty cells included unless the optional EXCLUDEEMPTY flag is used.
Covariance	Returns the covariance of two series evaluated over a set (biased).
CovarianceN	Returns the covariance of two series evaluated over a set (unbiased).
IIf	Returns one of two values determined by a logical test.
LinRegIntercept	Calculates the linear regression of a set and returns the value of b in the regression line $y = ax + b$ .

MDX Function	Description
LinRegPoint	Calculates the linear regression of a set and returns the value of y in the regression line $y = ax + b$ .
LinRegR2	Calculates the linear regression of a set and returns R2 (the coefficient of determination).
LinRegSlope	Calculates the linear regression of a set and returns the value of a in the regression line $y = ax + b$ .
LinRegVariance	Calculates the linear regression of a set and returns the variance associated with the regression line $y = ax + b$ .
Max	Returns the maximum value of a numeric expression evaluated over a set.
Median	Returns the median value of a numeric expression evaluated over a set.
Min	Returns the minimum value of a numeric expression evaluated over a set.
Stdev	Returns the standard deviation of a numeric expression evaluated over a set (unbiased).
StdevP	Returns the standard deviation of a numeric expression evaluated over a set (biased).
Sum	Returns the sum of a numeric expression evaluated over a set.

#### MDX formula syntax

BPC dimension and advanced rule formulas are based on the multi-dimensional expression language used by Microsoft SQL Analysis Services, called MDX. For details on the syntax and usage of this powerful language, you should consult the Microsoft SQL Analysis Services help.

Advanced rules formulas can also be based on Structured Query Language (SQL). Please consult the appropriate reference guides for more information and examples of using SQL.

#### Syntax basics

- The calculated member, which is the member into which the calculated values are written in the application, must be preceded with a pound (#) sign everywhere in the formula.
- When the dimension name is omitted from the left side of the formula, the ACCOUNT dimension is assumed. If it is omitted from the right side of the formula the dimension is automatically derived by Analysis Services.
- The right side expression is not enclosed in single quotes.
- If the calculated member is used on the right side of the formula, it must be enclosed in square brackets [].
- Remarks are denoted by double forward slashes (//). Any text to the right of double slashes is ignored during validation. This way, you can put remarks in the rules file to improve readability.
- Brackets are required by SQL when passing parameters that contain delimiters like "." or other special characters, to let the rules engine better understand where the parameter begins and where it ends. This is also true when a parameter contains a set of comma-delimited members as in the following:  
[2001.JAN, 2001.FEB]

Note: We recommend that you use brackets around parameters to be sure that they are parsed correctly.

#### MDX syntax exceptions

There are two exceptions to MDX syntax with BPC:

- Replace the keyword AS with an equal (=) sign
- Do not use single quotes around expressions

For example:

[ACCOUNT].[#GROSSSALES] = -[ACCOUNT].[UNITS]\*[ACCOUNT].[INPUTPRICE]

[ACCOUNT].[#COST] = -[ACCOUNT].[#GROSSSALES]\*80/100

Note that this structure:

{member} = {expression}[, solve\_order = n]

...is the normal syntax for calculated members required by MDX queries. The only exceptions are the equal sign ("=") in place of the "AS" keyword and the lack of single quotes around the expression. The above formula can also be written in a simpler format where the dimension name is omitted, for example:

#GROSSSALES = -UNITS\*INPUTPRICE

#COST = -[#GROSSSALES]\*80/100

#### Building dimension rule formulas

BPC gives you the ability to define very powerful rule formulas for calculating dimension members. BPC uses the MDX (multi-dimensional expression) language used by Microsoft Analysis Services. You can use any Microsoft MDX functions to create dimension rule formulas. You enter formulas using the MDX syntax, in the Formula column in a member sheet.

**Note:** For more information about Microsoft's MDX reference, see MDX Reference ([http://msdn.microsoft.com/library/default.asp?url=/library/en-us/olapdmd/agmdxbasics\\_04qg.asp?frame=true](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/olapdmd/agmdxbasics_04qg.asp?frame=true)) at the Microsoft Web site. For a complete list of Microsoft MDX functions, see the MDX function list ([http://msdn.microsoft.com/library/default.asp?url=/library/en-us/olapdmd/agmdxfnctions\\_9elw.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/olapdmd/agmdxfnctions_9elw.asp)).

#### Dimension files

You define your dimension rules formulas in dimension Excel files that you can access from the eAdmin > Manage Dimensions dialog box. When you validate these files, the dimension rules formulas are gathered together and placed in dimension-specific rules files.

**Note:** In order to use formulas in a dimension you must add a property called 'Formula' to the member sheet using the Manage Dimensions task in the Admin module. 'Formula' is a user-defined property, so you must define the length of the field. Enter a field length that is at least equal to the length of your longest formula.

#### Simple rule formula examples

Following are some examples of simple MDX formulas that can be used in the Formula column for an account.

Task	Syntax and example
Adding two accounts	Syntax: [dimension].[account1] + [dimension].[account2] Example (PreTax Income): [Account].[Operating Income] + [Account].[OtherExp]
Adding a range of accounts	Syntax: [dimension].[account1]:[dimension].[account2] Example (Total Personnel Exp): SUM([Account].[Salaries]:[Account].[Commission])
Calculating a percentage	Syntax: [dimension].[account1] / [dimension].[account2] or (to prevent division by zero): Iif([dimension].[account1] = 0, Null, [dimension].[account2]/[dimension].[account1]) Example (Gross Margin Pct): Iif([Account].[Revenue] = 0, Null, [Account].[Gross Margin]/[Account].[Revenue])
Multiplying by a factor	Syntax: [dimension].[account1]*[dimension].[account2] Example (Taxes): [Account].[Pretax Income] * -0.35

Task	Syntax and example
Calculating a YTD value	Syntax: [dimension].[account1],[Measures].[YTD]  Example (Current Year Net Income): ([Account].[Net Income],[Measures].[YTD])

#### Advanced rule formula examples

Advanced rule formulas include movement calculations and allocating expenses based on entity type.

### Movement calculations

Movement calculations define the copying or "movement" of data from one time period to another. In the formula examples below, [AccRec] is the accounts receivable account, and [AccPay] is the accounts payable account.

Account	Description	Formula
Mvmt Acc Rec	Movement in Accounts Receivable	<code>If([Time].CurrentMember.Children.Count=0, [Account].[AccRec] - ([Account].[AccRec], [Time].PrevMember), SUM(Descendants([Time].CurrentMember, [Time].[Month], LEAVES)))</code>
Mvmt Inventory	Movement in Inventory	<code>If([Time].CurrentMember.Children.Count=0, [Account].[Inventory] - ([Account].[Inventory], [Time].PrevMember), SUM(Descendants([Time].CurrentMember, [Time].[Month], LEAVES)))</code>
Mvmt Acc Pay	Movement in Accounts Payable	<code>If([Time].CurrentMember.Children.Count=0, [Account].[AccPay] - ([Account].[AccPay], [Time].PrevMember), SUM(Descendants([Time].CurrentMember, [Time].[Month], LEAVES)))</code>
Mvmt Work Cap	Movement in Working Capital	<code>SUM([Account].[Mvmt Acc Rec]:[Account].[Mvmt Acc Pay])</code>

### Allocating expenses based on entity type

The following examples assume there is an Entity property named "Function." If an Entity has the Function "SM", the "Total Dept Exp" is allocated to the "S&M Exp" account. If an Entity has the Function "RD", the "Total Dept Exp" is allocated to the "R&D Exp" account. If an Entity has the Function "CORP", the "Total Dept Exp" is allocated to the "G&A Exp" account.

Account	Description	Formula
S&M Exp	Sales & Marketing Expense	<code>If([Entity].CurrentMember.Children.Count = 0, If([Entity].CurrentMember.Properties("Function") = "SM", [Account].[Total Dept Exp], Null), SUM(Descendants([Entity].CurrentMember, [Entity].[LEV1], LEAVES)))</code>
R&D Exp	R&D Expense	<code>If([Entity].CurrentMember.Children.Count = 0, If([Entity].CurrentMember.Properties("Function") = "RD", [Account].[Total Dept Exp], Null), SUM(Descendants([Entity].CurrentMember, [Entity].[LEV1], LEAVES)))</code>

G&A Exp	G&A Expense	If([Entity].CurrentMember.Children.Count = 0, IIf([Entity].CurrentMember.Properties("Function") = "CORP", [Account].[Total Dept Exp], Null), SUM(Descendants([Entity].CurrentMember, [Entity].[LEV1], LEAVES)))
---------	-------------	---

#### Using the reserved function-EvCPN

The rules module supports an implicit function named EvCPN that can be used in formulas like any other user-defined function. EvCPN (Current Period Number) returns the number of the period being executed in the query. For example if you run a query for March and April, March is assigned the number 1 and April the number 2.

This feature can be used to decide whether a calculated value should be retrieved from the application or from memory. In this context "memory" means the result of the calculation performed by the query. For example, if you need to read a value from February, it can come from the application (so that February does not get re-calculated) and if you read it from March or April you must read the calculated result you have in memory.

Following is an example of how to use EvCPN:

```
[ACCOUNT].[~CPN]=EVCPN // dummy CPN account
*FUNCTION TLAG(%ACC%,%LAG%)
    iif([ACCOUNT].[~CPN]<(%LAG%+1),
        ([ACCOUNT].[%ACC%], [time].currentmember.lag(%LAG%)),
        ([ACCOUNT].[#%ACC%],[time].currentmember.lag(%LAG%)))
*ENDFUNCTION
[ACCOUNT].[#Z] = TLAG(Z,1) + X - Y
```

The above example enables you to correctly read the opening value of account Z without committing each period to the application in the process.

EvCPN supports a numeric parameter that can be used to restrict the function from calculating the period number beyond a certain number of periods. This could be used to make the calculation of CPN run faster. For example, if you need to test only whether you are in the first period or not (like in the above case), the function does not need to try and assign a period number to all processed periods. It is enough to assign the value of 2 to all periods beyond the first. To achieve this the rule can be written as follows:

```
[ACCOUNT].[~CPN]=EVCPN(1)
```

If you run the calculation from March, the function expands at run time into something like:

```
[ACCOUNT].[~CPN]=IIF([TIME].currentmember.name="2001.MAR",1,2)
```

All periods other than March have a period number greater than 1, which is all that the function call TLAG(Z,1) needs to know.

#### MDX Subroutines

You can use the MDX subroutines `Initialize_Elim` and `Eliminate_Org` in your dimension rules formulas.

##### Initialize\_Elim

Initializes an elimination organization with a specified value.

##### eliminate\_org

Perform an elimination on the given organization.

#### Using liquidity analysis ratios

Liquidity analysis ratios are ratios of assets versus liabilities. These ratios generally indicate a company's ability to cover short-term obligations. Liquidity analysis ratios include:

- CurrentRatio



- QuickRatio
- NetworkCapRatio

#### currentratio

CurrentRatio(% CURRENTASSET% ,% CURRENTLIAB% )

Calculates the current ratio. The current ratio is a general indicator of the business's ability to meet its short-term financial commitments. This ratio assumes that all current assets, if required, can be converted to cash immediately in order to meet all current liabilities immediately.

#### QuickRatio

QuickRatio(% CASH% ,% ACCREC% ,% CURRENTLIAB% )

Calculates a quick ratio, also called the acid test ratio. A quick ratio is a current ratio modified to provide a more prudent measure of short-term liquidity.

#### NetworkCapRatio

NetWorkCapRatio(% CURRENTASSET% ,% CURRENTLIAB% ,% TOTALASSET% )

Calculate Net working Capital Ratio. This is another indicator of a company's ability to satisfy short-term debt.

#### Using basic financial functions

This section contains definitions for two basic financial functions supplied with BPC:

- AvgBal - Average balance
- Growth

#### avgbal

AvgBal(% ACCOUNT% )

Calculates the account average. This result can be used in other functions such as ROA, ROE, and others.

#### Growth

Growth(% ACCOUNT% )

Calculates account growth rate. This result is used in sales growth, expense growth, and other functions.

#### Using profitability analysis ratios

Profitability analysis ratios provides tools that help you understand the profitability of your business.

Profitability ratios include:

- ROA — Return on assets
- ROE — Return on equity
- ROCE — Return on common equity
- CTS — Cost of goods sold to sales
- NPM — Net profit margin
- GPM — Gross profit margin
- SMGAEXPS — Sales, general and administrative expense to sales

#### ROA

ROA(% NETINCOME% ,% AVGTOTALASSET% )

Calculates Return on Assets (ROA). This ratio measures the ability of general management to use the total assets of the business in order to generate profits.

#### ROE

ROE(% NETINCOME% ,% AVGSTOCKEQT% )

Calculates Return on Equity (ROE). Measures the returns earned on both preferred and common stockholders' investments.

## ROCE

ROCE(% NETINCOME% ,%AVGCOMMONSTOCKEQT%)

Calculates Return on Common Equity (ROCE). Measures the return on common stockholders' investments only.

## cts

CTS(%COSTOFGOODS% ,%REVENUE%)

Calculates the cost of goods sold to sales (CTS).

## NPM

NPM(% NETINCOME% ,%REVENUE%)

Calculates the Net Profit Margin (NPM). The net profit margin, also known as the trading profit margin, measures trading profit relative to sales revenue. For example, a trading profit margin of 10% means that every \$1.00 of sales revenue generates \$0.10 in profit before interest and taxes. Some industries have low margins, which are compensated for by high volumes. Conversely, high margin industries can be low volume. Higher than average net profit margins for the industry can be an indicator of good management. The Net Profit Margin is calculated by Net Profit before Interest & taxes divided by Sales Revenue times 100 to give X%.

## GPM

GPM(%GROSSMARGIN% ,%REVENUE%)

Calculates Gross Profit Margin (GPM).

## SMGAEXPS

SMGAEXPS(%SMEXP% ,%GAEXP% ,%REVENUE%)

Calculates Sales, General and Administrative EXPense to Sales (SMGAEXPS).

## Using activity and efficiency analysis ratios

Activity and efficiency analysis ratios measure the efficiency of asset use. Activity and efficiency analysis ratios include:

- ATR — Asset turnover ratio
- ARTR — Accounts receivable turnover ratio
- ACP — Average collection period
- APTR — Accounts payable turnover ratio
- ITR — Inventory turnover ratio
- AAI — Average age of inventory
- STA — Sales to total asset ratio
- DAYSREC — Days in receivables
- DAYSPAY — Days in payable
- DAYSINV — Days in inventory

## ATR

ATR(%REVENUE% ,%AVGTOTALASSET%)

Calculates Asset Turnover Ratio.

## ARTR

ARTR(%REVENUE% ,%AVGACCREC%)

Calculates Accounts Receivable Turnover Ratio (ARTR).

## ACP

ACP(%ARTR% ,%DAYSOFYEAR%)

Calculates the Average Collection Period (ACP). The DAYSOFYEAR parameter can be either 365 or 360.

## APTR

APTR(%COSTOFGOODS% ,%AVGACCPAY%)

Calculates Accounts Payable Turnover Ratio (APTR).

ITR

$\text{ITR}(\% \text{COSTOFGOODS\%, \% AVG INVENTORY\%})$

Calculates Inventory Turnover Ratio (ITR).

aai

$\text{AAI}(\% \text{ITR\%, \% DAYSOFYEAR\%})$

Calculates Average Age of Inventory (AAI). The DAYSOFYEAR parameter can be either 365 or 360.

STA

$\text{STA}(\% \text{REVENUE\%, \% TOTALASSET\%})$

Calculates the Sales to Total Asset (STA) ratio.

DAYSREC

$\text{DAYSREC}(\% \text{ACCRC\%, \% REVENUE\%, \% DAYSOFYEAR\%})$

Calculates Days in Receivables (DAYSREC). The DAYSOFYEAR parameter can be either 365 or 360.

DAYSPAY

$\text{DAYSPAY}(\% \text{ACCPAY\%, \% COSTOFGOODS\%, \% DAYSOFYEAR\%})$

Calculates Days in Payable (DAYSPAY). The DAYSOFYEAR parameter can be either 365 or 360.

DAYSINV

$\text{DAYSINV}(\% \text{INVENTORY\%, \% COSTOFGOODS\%, \% DAYSOFYEAR\%})$

Calculates Days in Inventory (DAYSINV). The DAYSOFYEAR parameter can be either 365 or 360.

#### Using capital structure analysis ratios

Capital structure is the balance between debt and equity in a company. These ratios give you insight into the capital structure of your company. Capital structure analysis ratios include:

- DR — Debt ratio
- DTER — Debt to equity ratio
- ICR — Interest coverage ratio
- DCR — Debt coverage ratio

DR

$\text{DR}(\% \text{TOTALLIAB\%, \% TOTALASSET\%})$

Calculates the Debt Ratio (DR). Debt Ratio is total liabilities divided by total assets.

DTER

$\text{DTER}(\% \text{DEBT\%, \% EQUITY\%})$

Calculates Debt to Equity Ratio (DTER).

ICR

$\text{ICR}(\% \text{OPERINCOME\%, \% PAYINTEREST\%})$

Calculate Interest Coverage Ratio (ICR).

DCR

$\text{DCR}(\% \text{NETINCOME\%, \% NONCASHEXP\%, \% TOTALDEBT\%})$

Calculates the Debt Coverage Ratio (DCR).

#### Using capital market analysis ratios

Capital market analysis ratios indicate a company's ability to with the confidence of the stock market.

Capital market analysis ratios include:

- EPS — Earnings per share

- PER — Price earnings ratio
- BVPS — Book value per share
- MBR — Market to book ratio
- DY — Dividend yield
- DP — Dividend payout

#### EPS

EPS(% NetIncome%, % Shares%)

Calculates Earnings Per Share (EPS).

#### PER

PER(% PriceCommonShare%, % EPS%)

Calculates Price Earnings Ratio (PER)

#### bvps

BVPS(% TOTALSTOCKHOLDEREQUITY%, % PREFERREDSTOCK%, % SHARES%)

Calculates Book Value Per Share (BVPS).

#### MBR

MBR(% PriceCommonShare%, % BVPS%)

Calculates Market to Book Ratio (MBR)

#### DY

DY(% AnnualDividendPerCommonShare%, % PriceCommonShare%)

Calculates Dividend Yield (DY).

#### DP

DP(% AnnualDividendPerCommonShare%, % EPS%)

Calculates Dividend Payout (DP).

#### Using user-defined functions

User-defined functions are MDX functions that users of SQL Analysis Services create and register with the system. You can use the user-defined functions provided with BPC, or you can define your own.

User defined functions supplied by BPC include:

- MON — month
- PRO — property value
- MOVEMENT — month to month movement
- MOVEMENT2 — month to month movement on two accounts
- FIRSTPREV — opening balance, first month
- FXDIFFNetIncome — difference between balance sheet profit and profit/loss profit
- FUNCALLOC — allocate the full value of an account
- ENTITYALLOC — allocates value of account/entity based on ratio of account over total account
- OPENBALANCE — current period account balance
- PREVBALANCE — previous period account balance
- DynamicOpenBalance — previous period or previous year account balance

#### Building user-defined functions

In the context of the rules module, a user-defined function acts as a placeholder. Using the user-defined function syntax (explained below) you can assign a name to your rule formulas. When you want to use the formula in a rules file you need only to enter the user-defined function name and the rules module uses the formula it references when it creates the LGX file. Using user-defined functions makes maintenance of your rules formulas easier because the actual formula only resides in one location, so when you make a

change to a rule formula you only need to do it in one location. It also improves the readability of your rules files.

The definition for a user-defined function can be inserted anywhere in a rules file or in an included file.

**Note:** A good practice is to maintain a library of rules functions in a central file then use an INCLUDE statement to include the library file in your logic files. This way you only have to maintain one file. BPC comes with a predefined library of rules functions, called LogicFunctions.lgl, where you can add your own user-defined functions to this file.

### Syntax

For single line functions:

```
*FUNCTION {functionname}({Param1},{Param2}...) = {Function Text}
```

Parameter	Description
Functionname	The name of the function, you decide what name to give the function.
Param1, Param2, etc	(Optional) These parameters are used to dynamically modify the corresponding MDX string (the function formula). Parameters can be hard coded (i.e., *function FXRate(USD)) or variable (i.e. FXRate (%CURR%), that is, the value of the parameter is determined by the formula syntax itself.
Function Text	The function (in MDX syntax)

If you want to spread the formula across multiple lines you must follow this syntax:

```
*FUNCTION {functionname}({Param1},{Param2}...)
{Function text}
{Function text}
*ENDFUNCTION
```

### Using variable parameters

The following example shows how a variable parameter can be used in a user-defined function.

```
*function FXRate(%CURR%)
(
  LookupCube("RateCube", "([Entity2Dim].[RateEntDefault],[RateDim].[ "+[AccountDim].CurrentMember.Properties("RATETYPE")+"], [InputCurrencyDim].[ "+[EntityDim].CurrentMember.Properties("CURR")+ "], " +
  MemberToStr([CategoryDim].CurrentMember) + ", " +MemberToStr([TimeDim].CurrentMember) +
  " ) )/LookupCube("RateCube", "([Entity2Dim].[RateEntDefault],[RateDim].[ "+[AccountDim].CurrentMember.Properties("RATETYPE")+"], [InputCurrencyDim].[ %CURR% ], " +MemberToStr([CategoryDim].CurrentMember) + ", " +
  MemberToStr([TimeDim].CurrentMember) + " )" )
)
*endfunction
```

In this formula the variable %CURR% is determined by the Entity type dimension current member's CURR property value. Using variable parameters makes the formula dynamic in that it determines the variable value based on instructions found in the formula. In this case the value is determined by the value in the CURR property of the current entity member.

### Referring to a user-defined function

After you define your user-defined functions, you can refer to the name of the function in your rule formulas. If the definition of the user-defined function is in another file (such as a library file) you must use an INCLUDE statement to include that file in the LGF file you are working in. The following example

refers to the user-defined function FXRate. The reference to the user-defined function FXRate and the INCLUDE statement where the user-defined function is stored are highlighted.

For example, if the following function is defined in MdxLib.lgl:

```
//calculate CTS(Cost of goods sold To Sales)
*function CTS(%COSTOFGOODS%,%REVENUE%)
iif(%REVENUE%=0,Null,round(%COSTOFGOODS%/(-%REVENUE%),2))
*endfunction
```

This function can then be used in another rules file, such as FinStdAccount.lgl, as follows:

```
*include mdxlib.lgl
// Financial Standard Key Performance Indicators (KPI)
#KPI130 = CTS(FSA220,FSA100),SOLVE_ORDER=100
```

MON

Returns the MONTH member from the Time dimension.

PRO

PRO(%DIMENSIONNAME%,%PROPERTYNAME%)

Returns the value of a property of the current member of a dimension.

MOVEMENT

MOVEMENT(%ACCOUNT%,%TIMEDIM%)

Return the movement on an account from one month to the next month.

MOVEMENT2

MOVEMENT2(%ACC1%,%ACC2%)

Return the movement on two accounts combined from one month to the next month.

FIRSTPREV

FIRSTPREV(%ACCOUNT%)

Return the opening balance, but only in first month of year.

FXDIFFNetIncome

FXDiffNetIncome(%CYNI%,%NETINC%)

Returns the difference between Balance Sheet profit and Profit/Loss profit. This difference occurs because balance sheet profit is calculated based on the currency exchange rate on a specific days (last day of the month), but PL profit is calculated on the average currency rate over a month.

If the difference is zero in local currency, then the function returns the FX difference in Report currency.

FUNCALLOC

FUNCALLOC(%PROPERTY%,%PROPVALUE%,%DIMENSION%,%SOURCEACCT%)

Allocate the full value from an account based on a property value of a member in a dimension.

ENTITYALLOC

ENTITYALLOC(%ENTITYSRC%,%ACCOUNTSRC%,%ACCOUNTWHT%,%ENTITYTOP%)

Allocate value of account/entity based on ratio of account over total account with automatic offset (based on Entity/Function).

OPENBALANCE

OpenBalance(%ACCOUNTSRC%)

Returns the current period account balance.

PREVBALANCE

PrevBalance(%ACCOUNTSRC%)

Returns the previous period account balance.

### DynamicOpenBalance

#### DynamicOpenBalance(%ACCOUNTSRC%)

Previous period or previous year account balance. This function requires a solve\_order=5. Any account derived from this one must use the same or higher solve\_order.

This function is only suitable for dimension rule formulas, it does not work if used in advanced rule formulas.

## The BPC SQL library

BPC supplies SQL functions that you can use in your advanced rules formulas, including:

- ROLLTOBS
- ACCUMULATE
- TRANSLATE\_LDI
- CALC\_MOVEMENT
- CALCULATE\_TOTAL\_AND\_COMMIT
- FX\_OVERRIDE
- FX\_OVERRIDE\_RATE
- FX\_OVERRIDE\_NOZERO
- TRANSLATE\_PROFIT\_AND\_COMMIT
- CARRYFORWARD\_AND\_COMMIT
- Derive\_Ytd

### Building advanced rules

Advanced rules enable you to do calculations that only need to be performed on base-level cells (cells where all members have no children in any dimension). The results are aggregated up the dimensional hierarchy intact, without being re-calculated at upper levels.

### When to use advanced rules

An example of an advanced rule formula is: units times price calculations. Such calculations cannot be defined in dimension rules because they must only be performed on base-level cells. You can define units times price calculations in worksheets where the calculation is applied to values retrieved from the BPC database but maintaining and controlling the formulas is difficult.

BPC provides a library of MDX and SQL formulas. In addition to the sample formulas, the ApShell application set is pre-configured with many functions which are included in all application sets you create from ApShell.

### How advanced rules work

BPC rules read a specific data selection from the BPC application, apply to it a set of user-defined formulas (stored in a rules file), derive the values and writes those values back to the BPC application. This process allows you to perform such calculations as units times price because the formulas are applied only to those members specified in the rule file.

The rules module can be run in any of three ways:

- After data is sent to the database - It can be automatically invoked each time data is sent to the database (EvSND or EvINP) using the default.lgf file. By using this method, the formulas are executed immediately after the data is sent and the results can be seen in BPC right away.
- After Journals data sends rules in either default.lgf, or, if present, journal.lgf, is run after Journal data sends.
- It can also be run from Data Manager for batch processing of formulas. Using Data Manager to execute rules module formulas is useful for calculations that do not need to be executed immediately. For example, as an administrator you can decide to wait until all the data has been entered in the local currency before generating the translated amounts in the reporting currencies.

To invoke advanced rules, you need the following:

- The name of the advanced rules file to execute

- The data selection against which the formulas must be executed

If the rules module is invoked by a Data Manager package, the package provides this information, usually from a prompt to the person running the Data Manager package.

If advanced rules has been invoked automatically by BPC, BPC follows these rules:

1. Runs the DEFAULT.lgf advanced rules or, if the Journal module is sending data, the system runs the journal.lgf advanced rules unless that file does not exist, in which case the system runs default.lgf.
2. Passes a data selection that is derived by the scope of the data that has just been entered into the cube.

In order to build the correct data selection, BPC relies on all occurrences of different members in all dimensions, excluding ACCOUNT, for which it assumes ALL members.

For example, assume that you have entered data for six cells in the application. The corresponding records could be the following:

CATEGORY	TIME	ENTITY	CURRENCY	ACCOUNT	PRODUCT	PERIODIC
ACTUAL	2004.JAN	SALESITALY	LC	UNITS	PRODUCTA1	12345
ACTUAL	2004.JAN	SALESITALY	LC	INPUTPRICE	PRODUCTA1	100
ACTUAL	2004.FEB	SALESITALY	LC	UNITS	PRODUCTA1	5456
ACTUAL	2004.FEB	SALESITALY	LC	INPUTPRICE	PRODUCTA1	200
ACTUAL	2004.FEB	SALESITALY	LC	UNITS	PRODUCTA2	300
ACTUAL	2004.FEB	SALESITALY	LC	INPUTPRICE	PRODUCTA2	8456

In this case, a send (for example, EvSND) passes a selection containing the category ACTUAL, the time periods 2004.JAN and 2004.FEB and the entity SALESITALY.

With the information specifying the scope of the data to process and the formulas found in the default logic, the rules module builds an MDX query that returns the appropriate set of calculated cells.

#### About advanced rule files

The rules module relies on Advanced logic files which contain the formulas that are to be applied to the specified calculated members (data selection). The rules module does not perform the calculations directly, but uses MDX or SQL queries that it passes to the Analysis Server to get from it the desired results. The formulas that it uses must be consequently written in the correct syntax.

The Advanced rules files are LGF (ASCII) files and contain the code for the formulas.

#### The compiled logic file

When the advanced formulas text file is saved it is actually converted into another file (with the extension LGX) that is an executable version of the LGF file.

#### The library file

Library files store a library of standard functions. (Currency translation and inter-company eliminations, for example.) A library file has the extension LGL.

You can use the #INCLUDE function in your LGF file to call the library file at validation. The rules module scans the library file for the appropriate formulas to use based on the information in the LGF file.



## The dimension constants file

This is a rules file but it behaves a little differently. It maps your dimension names for the application to the standard BPC dynamic logic. By updating the dimension constants file with your dimensions, you avoid having to change or rewrite any of the standard functions that are included with BPC. The dimension constants file is located in:

<BPC>\Webfolders\<AppSet>\SystemLibrary\System\_Constants.LGT

```
// application constants
//-----
*FUNCTION CATEGORYDIM =%CATEGORY_DIM%
*FUNCTION TIMEDIM =%TIME_DIM%
*FUNCTION CURRENCYDIM =%CURRENCY_DIM%
*FUNCTION ENTITYDIM =%ENTITY_DIM%
*FUNCTION ACCOUNTDIM =%ACCOUNT_DIM%
*FUNCTION INTCODIM =%INTERCOMPANY_DIM%
// This part is needed when a RATE application
// is associated to the application
// (FX = single or multi currency)
//-----
*FUNCTION RATEAPP =%RATE_APP%
*FUNCTION RATEENTITYDIM =%RATEENTITY_DIM%
*FUNCTION RATEACCOUNTDIM =%RATEACCOUNT_DIM%
*FUNCTION INPUTCURRENCYDIM =%RATECURRENCY_DIM%
*FUNCTION RATEENTITYMBR =%RATEENTITY_DEFAULTMBR%
*FUNCTION RATESRCCALCMBR =RATECALC
```

where <BPC> is the drive where the BPC software is located and <AppSet> is the name of the application set.

For example, if your category type dimension is named "MYCATEGORY", you would change:

```
*FUNCTION CATEGORYDIM =%CATEGORY_DIM%
```

to:

```
*FUNCTION MYCATEGORYDIM =%CATEGORY_DIM%
```

You can also include application constants and member constants in the dimension constants file. For example, you would do this if your advanced formulas reference another application where values to be used in the formula were stored.

### Running rules after data sends

In this case the Logic module determines the data selection (the selection of members upon which the rules module performs calculations) based on all occurrences of the Category, Time and Entity dimension that are found in the posted data.

All rules in default.xls (default.lgl) are run after data sends.

### Running rules after Journal updates

If you are using the Journals feature, you can have special advanced logic for Journals, or Journals can run default advanced logic. Default advance rules are the same as the default advance rules run during BPC for Excel data sends.

To use special advanced rules for Journals, create an advanced logic worksheet named Journal.xls and save it and validate it. This creates Journal.lgl and Journal.lgx. After this file is created, every time users post data using a Journal the special journal advanced rules are run. If there is no Journal.lgl file, the system runs the standard default advanced formula (default.lgl) after Journal postings.

### Performing intercompany eliminations

The elimination of the intercompany values is based on the hierarchical relationships defined in the entity dimension. This type of relationship can be evaluated using MDX-based rules, because the MDX language understands hierarchies. But there is a special SQL-based function that allows the system to evaluate

hierarchies and understand parent-child relationships in a tree, to enable the elimination of intercompany values.

## Syntax

This function is named CPE (Common Parent Elimination) and supports the following syntax:

```
CPE( {Entity1} , {Entity2} [ , {Organization} [ , {ElimProperty} [ , {ElimDim} ] ] ] )
```

This function returns the name of the "Elimination entity" immediately below the first "common parent" of {Entity1} and {Entity2}, as found in the specified {Organization}. The "Elimination entity" is an immediate dependant of the common parent having the property {ElimProperty} set to "Y".

Parameter	Description
Entity1 and Entity2	<p>This parameters Entity1 and Entity2 are required and can be:</p> <ul style="list-style-type: none"> <li>• A specific name enclosed in double quotes ( like "Italy" )</li> <li>• A dimension name (like ENTITY, meaning the current member of the ENTITY dimension)</li> <li>• A dimension name and property combination (like INTCO.ENTITYPROP, meaning the value of the ENTITYPROP property of the current member of the INTCO dimension)</li> </ul> <p>For example, the following returns the name of the elimination entity below EUROPE, for example E_EUROPE.</p> <pre>• CPE( " ITALY " , "FRANCE" )</pre> <p>The following returns the name of the elimination entity below WORLD, for example E_WORLD.</p> <pre>• CPE( " ITALY " , "US " )</pre>
Organization	<p>This parameter is optional, and must be in the format Hn where n is the organization number (for example H1 or H2, etc., to indicate PARENTH1 or PARENTH2, etc.). If omitted, the value H1 is assumed.</p>
ElimProperty	<p>This parameter is optional, and can be used to indicate the name of the entity property that defines the "elimination" entities (i.e., the entities where the amounts should be eliminated). If omitted, the value ELIM is assumed.</p>
ElimDim	<p>This parameter can be used in case the dimension for which the elimination must be performed is NOT a dimension of type ENTITY. In the following example the user is defining the elimination between Business Units in the BU dimension (here the Inter-CO dimension is an inter-BU dimension).</p> <pre>*WHEN CPE(I_BU.BU,BU,H1,ELIM,BU)      // note the additional parameter *IS &lt;&gt;" "       *REC(BU=%CPE%,FACTOR=-1)       *REC(BU=%CPE%,ACCOUNT=ACCOUNT.ELIMACC) *ENDWHEN</pre>

## Examples

The CPE( ) function is only valid in a \*WHEN statement. The subsequent \*IS statement should verify that a valid elimination entity has been returned. (\*IS <> ""). The returned elimination entity can then be

referenced in a subsequent \*REC statements using the keyword %CPE%. The following example shows a correct use of the function:

```
*WHEN CPE(ENTITY,INTCO.ENTITY)
*IS <>" "
*REC( ENTITY = %CPE%, FACTOR = -1 ) // eliminate the original account...
*REC( ENTITY = %CPE%, ACCOUNT = "PLUG") //...into the account plug
*ENDWHEN
```

Note: Note how the name of Entity1 is derived from the current ENTITY member and the name of Entity2 is derived from the ENTITY property of the current INTCO member.

accumulate

ACCUMULATE(%src\_flowacc%, %dst\_bsacc%)

Accumulate periodic values from a flow account into a bs (balance statement) account. This is a complete commit section.

Parameters:

%src\_flowacc% The flow account that contains the periodic values that need to be accumulated.

%dst\_bsacc% The destination balance sheet account into which the values are accumulated.

CALCULATE\_TOTAL\_AND\_COMMIT

CALCULATE\_TOTAL\_AND\_COMMIT(%add\_property%)

This formula can be used to store the total of a set of accounts defined by a common property.

Parameters:

%add\_property% The name of the property defining the accounts to add

To use this formula, add a property to the account dimension and enter "Y" or "A" (for "Add") in all accounts that should be added to the total, enter "S" in all accounts that should be subtracted from the total, Enter "T" in all accounts where the total should be stored.

Enter a line in the logic invoking this formula with the appropriate property name, as in the following example:

```
CALCULATE_TOTAL_AND_COMMIT(ADD1)
```

CALC\_MOVEMENT

CALC\_MOVEMENT(%Acc%,%MovAcc%)

Calculates the net change between periods. The time periods are defined by the Current and Prior properties in the Time dimension.

Parameters:

%Acc% The account for which you want to calculate the net change.

%MovAcc% The account in which to place the results.

CARRYFORWARD\_AND\_COMMIT

CARRYFORWARD\_AND\_COMMIT()

This formula performs a carry-forward of last year closing value into current period.

CARRYFORWARD\_AND\_COMMIT requires the property OPE in the Account dimension (ID of the opening account). This formula must be executed from a DTS package in LGF format (prompting for category and time). For this reason it is not suitable to be used in a default formula sheet.

To use this formula, add a property called OPE to the account dimension, and enter the ID of the accounts where the opening value should be stored into the appropriate accounts.

Define a logic containing the line:

```
CARRYFORWARD_AND_COMMIT()
```

Then run this logic from a DTS package that calls it using explicitly the extension LGF.

## Derive\_Ytd

```
DERIVE_YTD(%ytd_acc%, %flow_acc%)
```

This is the only formula in the library that generates an MDX logic query and, as such, must be used on its own or as part of an MDX logic section.

This formula can be used to read a YTD amount from a PERIODIC appset, and store it in another account. Typically, it can create the net profit in balance sheet from the net profit in profit and loss.

## Parameters:

%ytd\_acc% The account to write in YTD view (LEQ)

%flow\_acc% The account to read in YTD view (INC)

Define a rule invoking this sub, as in the following example:

```
DERIVE_YTD(CYNI, NetIncome)
```

## FX\_OVERRIDE

```
FX_OVERRIDE(%from_ovr_acc%,
            %to_ovr_acc%,
            %fx_diff_acc%,
            %CURR%)
```

This formula can be used to override the default translation of a given account. The difference from the default translation value is posted into a translation difference account.

## Parameters:

%from\_ovr\_acc% The account from which to read the overriding amount

%to\_ovr\_acc% The account to override

%fx\_diff\_acc% The account used to post the difference from default

%CURR% The reporting currency of the overriding amount

## General remarks:

- This formula does not require a lookup.
- It can be invoked multiple times before a commit.
- It MUST be executed AFTER the default translation.
- The section MUST NOT contain any clear\_destination instruction.

To use this formula, enter one or more lines in the logic that call this formula with the appropriate parameter values, as in the following example:

```
FX_OVERRIDE(OVR_ACC1_USD, ACC1, FXDIF_ACC1_USD, USD)
FX_OVERRIDE(OVR_ACC2_USD, ACC2, FXDIF_ACC2_USD, USD)
*COMMIT
```

## FX\_OVERRIDE\_RATE

```
FX_OVERRIDE_RATE(%ovr_rate_acc%,
                %ovr_acc%,
                %fx_diff_acc%,
                %CURR%)
```

This formula is similar to FX\_OVERRIDE, but the first parameter is the name of an account where the translation rate is entered in place of the translated amount. See FX\_OVERRIDE.

## Parameters:

% ovr\_rate\_acc% The account from which to read the overriding rate

%to\_ovr\_acc% The account to override

%fx\_diff\_acc% The account used to post the difference from default

%CURR% The reporting currency of the overriding rate

General remarks:

- This formula does not require a lookup.
- It can be invoked multiple times before a commit.
- It MUST be executed AFTER the default translation.
- The section MUST NOT contain any clear\_destination instruction.

To use this formula, enter one or more lines in the logic that call this formula with the appropriate parameter values, as in the following example:

```
FX_OVERRIDE_RATE(OVR_ACC3_USD_RATE, ACC3, FXDIF_ACC3_USD, USD)
FX_OVERRIDE_RATE(OVR_ACC4_USD_RATE, ACC4, FXDIF_ACC4_USD, USD)
*COMMIT
```

FX\_OVERRIDE\_NOZERO

```
FX_OVERRIDE_NOZERO(%from_ovr_acc%,
    %to_ovr_acc%,
    %fx_diff_acc%,
    %CURR%)
```

This formula is equivalent to the FX\_OVERRIDE formula, and it takes the exact same parameters. However, when using this formula, the override mechanism only takes place if an overriding amount is entered. If not (or if set to zero), the default translation prevails.

Parameters:

%from\_ovr\_acc% The account from which to read the optional overriding amount

%to\_ovr\_acc% The account to override

%fx\_diff\_acc% The account used to post the difference from default

%CURR% The reporting currency of the overriding amount

General remarks:

- This formula does not require a lookup.
- It can be invoked multiple times before a commit.
- It MUST be executed AFTER the default translation.
- The section MUST NOT contain any CLEAR\_DESTINATION instruction.

Enter one or more lines in the logic invoking this sub with the appropriate parameters values like in the following example:

```
FX_OVERRIDE_NOZERO(OVR_ACC1_USD, ACC1, FXDIF_ACC1_USD, USD)
FX_OVERRIDE_NOZERO(OVR_ACC2_USD, ACC2, FXDIF_ACC2_USD, USD)
*COMMIT
```

ROLLTOBS

```
SUB ROLLTOBS(%bsaccount%,
    %workaccount%,
    %acctproperty%,
    %propertyvalue%,
    %LastPeriod%)
```

Rolls a periodic PL account to a BS account. Requires a property to identify the PL accounts to be totaled and requires a temporary work account to calculate the total of the PL account.

Parameters:

%bsaccount% Balance statement account name.

%workaccount% Temporary working account name.

%acctproperty% Property that identifies the PL accounts to be totaled.

%propertyvalue% The value of the property to total.

%LastPeriod% Identifies the last time period in which to roll to the balance sheet.

#### TRANSLATE\_LDI

TRANSLATE\_LDI (%bs\_acc%, %bs\_acc\_fxdiff%, %avg\_rate%, %end\_rate%)

This function translates a balance based on the current value and compares it against the prior balance. The current translated rate is applied to the difference from the current and prior values. For example:

January balance: 100  
exchange rate: .5

February balance: 200  
exchange rate: .75

The final balance is:  $(100 * .5) + (100 * .75) = 125$ , given that the difference between the two time periods is 100.

The local difference in the bs\_acc account is translated at the avg\_rate and added to the prior period translated value with the translation difference posted to the bs\_acc\_fxdiff account. This function has a complete commit section. Requires the properties PRIOR and CURRENT in the Time dimension.

Parameters:

%bs\_acc% Balance statement account.

%bs\_acc\_fxdiff% Balance statement account to which the local differences are posted.

%avg\_rate% The ID of the average rate (normally AVG).

%end\_rate% The ID of the end of period rate (normally END).

TRANSLATE\_LDI ( CYN1 , FXCYN1 , AVG , END )

#### TRANSLATE\_PROFIT\_AND\_COMMIT

TRANSLATE\_PROFIT\_AND\_COMMIT(%bs\_profit\_acc%,  
                                  %bs\_profit\_acc\_fxdiff%,  
                                  %avg\_rate%,  
                                  %end\_rate%)

The balance statement (BS) account is translated at AVG rate and the difference from END rate is posted into FXDIFF account. This function defines a complete commit section.

## Rules keyword reference

The reference contains descriptions of all the rule statements you can use. You can also use the Rules Assistant to help create rules statements.

\*ADD\_DIM - more information

Allowed uses: By Commit, MDX, SQL

\*ADD\_DIM {dimension name}={value}[,{dimension name}={value},...]

Works in conjunction with the \*DESTINATION\_APP instruction to post results of rules calculations to a different application. See the \*DESTINATION\_APP topic for more information.

\*ADD / \*ENDADD - more information

Allowed uses: By Commit, MDX

```
*ADD {variable} = {set}
{formula}
*ENDADD
```

This structure allows you to automatically add a set of members as specified in a comma-delimited range to a set of calculated members. The range can be dynamically derived using a \*SELECT( ) instruction.

**Note:** ADD/ENDADD loops executed on empty lists of elements do not cause the logic execution to fail.

Example 1:

```
*ADD %ACC%=A,B,C,D
#MYSUM = %ACC%
*ENDADD
```

This expands into:

```
#MYSUM = A+B+C+D
```

Example 2:

The expression to the right of the equals (=) sign can be a more complex expression as in the following example:

```
#MYSUM = (-[ACCOUNT].[%ACC%])
```

Note that the above format enables you to SUBTRACT the set of members from the calculated one.

Example 3:

Another possibility is to include a fixed part in the sum as follows:

```
#MYSUM = [ACCOUNT].[FIXED]+ [%ACC%]
```

The program uses the rightmost plus (+) sign as delimiter between the fixed portion and the portion to add. The above example expands as follows:

```
#MYSUM=[ACCOUNT].[FIXED]+ [A]+ [B]+ [C]+ [D]
```

Example 4:

The structure supports multiple formulas to be expanded simultaneously, for example:

```
*ADD %ACC%=A,B,C,D
#MYFIRSTSUM = %ACC%
#MYSECONDSUM = %ACC%
*ENDADD
```

\*BEGIN / \*END - more information

Allowed uses: By Commit, MDX

```
*BEGIN {multiple lines} *END
```

You can write MDX formulas that span multiple lines by enclosing the formulas between the two keywords \*BEGIN and \*END.

Example:

```
*BEGIN
[ACCOUNT].[#GROSSSALES] =
-[ACCOUNT].[UNITS]*
[ACCOUNT].[INPUTPRICE]
*END
```

Writing MDX formulas this way increases readability of your logic code.

\*CALC\_ORG - more information

Allowed uses: By Commit/Global, MDX, SQL

A special instruction can be inserted between WHEN / ENDWHEN structures, to calculate the parent members of a selected hierarchy. Its syntax and behavior are exactly the same as the STORE\_ORG instruction. The main difference from STORE\_ORG is that CALC\_ORG does not require that it is written between COMMIT instructions, and does not enforce a posting of the data to the database, until a subsequent COMMIT is found.

Its syntax is:

```
*CALC_ORG {DimensionName} = {OrgProperty}
```

Example:

```
*WHEN...
// some calculation...
*ENDWHEN
*CALC_ORG ACCOUNT = PARENTH1
```

In the above example the results of the preceding calculations are included in the calculation of the parents of PARENTH1 org.

The CALC\_ORG instruction automatically enforces a double GO instruction (before and after its execution). For this reason, no other instruction needs to be inserted between it and the preceding WHEN / ENDWHEN structures, nor the following ones.

#### Calculating orgs in memory

If the calculated members generated by a CALC\_ORG instruction do not need to be posted to the database, you can use a variation of the CALC\_ORG instruction that does not add these extra records to the database, even when the COMMIT instruction is reached.

The instruction is:

```
*CALC_DUMMY_ORG {DimensionName} = {OrgProperty}
```

A valid alternative syntax is:

```
*CALC_ORG {DimensionName} = {OrgProperty}, DUMMY
```

This instruction automatically adds a pound sign ("#") in front of the generated member IDs. As a result these records are considered memory variables that are not posted to the database.

The dummy members generated with this technique do not exist in any dimension, even if generated from real members. Because of this, it is not possible to use in the logic any property assigned to them. They can only be referenced using their ID.

#### CALC\_ORG and CALC\_DUMMY\_ORG Multi-line syntax

The instructions CALC\_ORG and CALC\_DUMMY\_ORG now support a multi-line syntax that gives you better control of the scope of the calculation. Though the original single-line syntax is still supported, the following new syntax can also be used:

```
*CALC_DUMMY_ORG
*ORG {dimension}={property}
[*WHERE {dimension} = {member set}]
[*WHERE {dimension} = {member set}]
...
*ENDCALC
```

For example:

```
*CALC_DUMMY_ORG
*ORG ENTITY=PARENTH1
  *WHERE ACCOUNT = CASH, ACCREC
  *WHERE INTCO = NON_INTERCO
*ENDCALC
```

In the above example the calculation of the hierarchy PARENTH1 for the ENTITY dimension will only be performed for accounts CASH and ACCREC and for the intercompany member NON\_INTERCO, even if all accounts or intercompany members are in memory.

This feature can be extremely helpful in controlling the number of records that will be created by the CALC\_ORG or CALC\_DUMMY\_ORG instructions, in all cases where only a few elements are actually needed for some dimension.



## Notes:

- The instruction still works only on the records that have been pulled in memory by the rules query, as defined by the XDIM\_MEMBERSET or similar instructions. The WHERE clause in the CALC\_ORG or CALC\_DUMMY\_ORG will NOT modify the region accessed by the query, but will only apply an additional filter to the region.
- The WHERE clause only supports the EQUAL ("=") sign. Other operands like "<>" are not currently supported.
- In addition, the member set at the right of the equal sign in the WHERE clause must be EXPLICIT. Other syntaxes (like WHERE ACCOUNT = ACCOUNT.GROUP="Assets" or similar) are currently not supported.

## Using \*XDIM\_NOSCAN and \*NOSCAN instructions

The instructions \*XDIM\_NOSCAN and \*NOSCAN in \*CALC\_DUMMY\_ORG allows you to load in memory information that is only needed in a \*GET( ) statement and never used in a \*IS statement. For example, in a Units \* Price calculation the logic might read as follows:

```
*XDIM_MEMBERSET ACCOUNT=UNITS, PRICE
*WHEN ACCOUNT
*IS UNITS
*REC(FACTOR=GET(ACCOUNT="PRICE"),ACCOUNT="REVENUE")
*ENDWHEN
```

In such situations, you can now instruct the logic to ignore the PRICE by simply saying:

```
*XDIM_MEMBERSET ACCOUNT=UNITS, PRICE
*XDIM_NOSCAN ACCOUNT=PRICE
*WHEN *
*IS *
*REC(FACTOR=GET(ACCOUNT="PRICE"),ACCOUNT="REVENUE")
*ENDWHEN
```

This makes the scanning of the record set loaded in memory somewhat faster, as all those with account PRICE will be skipped very efficiently.

Similarly, a \*NOSCAN instruction can be added to a \*CALC\_DUMMY\_ORG structure as follows.

```
*CALC_DUMMY_ORG
*ORG= {dimension name}
*NOSCAN
*ENDCALC
```

This enables the scanning of the record set loaded in memory somewhat faster, as all parent members will be skipped very efficiently in any subsequent WHEN evaluation.

**\*CALC\_EACH\_PERIOD** - more information

Allowed uses: By Commit/Global, MDX, SQL

The instruction CALC\_EACH\_PERIOD can be inserted within a commit section, to enforce an orderly calculation of the members of the time dimension. When this instruction is present, the source records are scanned one period at a time, in ascending order from the oldest to the youngest, and the results generated by each time period are merged with the rest of the record set, before the next period is calculated.

With this technique we can more easily and efficiently handle the typical calculations of financial reporting applications, where the results of one period are the inputs for the calculation of the next period.

Example:

```
//-----
*CALC_EACH_PERIOD
*WHEN ACCOUNT
*IS OPEN_BALANCE, MOVEMENTS
*REC (ACCOUNT=CLOSING_BALANCE)
*REC (ACCOUNT=OPEN_BALANCE, TIME=NEXT)
*ENDWHEN
//-----
```

The above sample rule performs a carry-forward of the closing balance of each period into the opening balance of next period.

\*CALCULATE\_DIFFERENCE - more information

Allowed uses: By Commit, MDX, SQL

```
*CALCULATE_DIFFERENCE = 0 | 1
```

The database always stores the difference between the new value and the old value. This option defines where the calculation is performed. By default (when \*CALCULATE\_DIFFERENCE = 1), the logic automatically calculates the difference and sends only that value to the posting engine ("BPC "), instructing it that what it's receiving is already the difference, and no other action is required. Since the calculation of the difference takes time, if the calculation is performed directly by the Logic module at the time of executing the rules, the subsequent posting time may be reduced.

If you set the calculate\_difference value to zero (0), the logic does not perform the calculation, and sends to the posting engine the desired final value. In this case, it tells the posting engine that the difference is still to be calculated, and the posting engine will take over the job. The main reason for using this instruction is so the debug file shows the values exactly as you expect them to look in the application.

\*CLEAR\_DESTINATION - more information

Allowed uses: By Commit, SQL

```
*CLEAR_DESTINATION
*DESTINATION {DimensionName1}={MemberSet1}
*DESTINATION {DimensionName2}={MemberSet2}
```

Clears all records in the destination region, defined by one or more \*DESTINATION instructions.

In most cases there is no need to perform a clear of the destination area when re-executing a modeling logic that uses SQL queries. SQL logics, in fact, base their computation on the existence of records in the fact table, regardless of the value assigned to these records. Since values that have been re-set to zero maintain records in the fact table, the execution of the SQL logic ensures the correct handling of any value, including those that have been set to zero.

This may not be true in all cases. For example, if a value is set to zero and, before the SQL rules are executed, an administrator performs a compression of the fact table, the zeroed-out records are lost. This could lead to a situation where the rule does not clear the records it generated in a prior pass, until you re-enter some value in the values that were set to zero. This situation is unlikely to happen when the rule is a default rule (which is executed at the same time data is entered), but it could happen with rules that are executed in a batch mode (like eliminations, allocations or consolidations).

To control these situations, the administrator can make use of a couple of instructions that enforces a clear of all records existing in the destination region at the time the logic is executed.

---

**Note:** These instructions may lead to deletion of the input data, if used incorrectly. A good understanding of their behavior is required, in order to avoid the risk of serious losses of data in the database.

---

The instructions to use are:

```
*CLEAR_DESTINATION
```

```
*DESTINATION {DimensionName1}={MemberSet1}
*DESTINATION {DimensionName2}={MemberSet2}
```

...

The instruction CLEAR\_DESTINATION activates the clear mechanism. If not present, the rules do not try to perform any clear.

The second instruction, the DESTINATION instruction, is optional. However, when CLEAR\_DESTINATION is used, the DESTINATION instruction MUST be used for ALL the dimensions for which you want to be SURE that the correct region is cleared. If not used, the program tries to automatically decide what to clear in each dimension, and this, in some cases, might be incorrect. Following is an explanation that clarifies what could happen:

Case 1: The source and destination regions are the same.

For example, the category is ACTUAL for both the source and the destination regions. The category the program clears is definitely ACTUAL. For this dimension, there is no need to specify anything.

Case 2: The source and destination regions are different, as specified by a XDIM\_MEMBER instruction.

```
Example: *XDIM_MEMBER DATASRC=LC TO ELIM
```

In this case the logic knows that the destination Datasrc member can only be ELIM. The Datasrc the program clears is definitely ELIM. In this case too, there is no need to specify anything.

Case 3: The source and destination members are different, as defined by one or more \*REC( ) instructions.

This is the case where the rules might have an issue in deciding what to clear. The REC( ) instruction, in fact, has a great deal of power in deciding where to write its output, and could do it on multiple dimensions at the same time. It could say something like:

```
*REC(CURRENCY="EURO")
*REC(ACCOUNT=ACCOUNT.PLUGACCOUNT)
*REC(ENTITY="IC_%ENTITY%", CURRENCY=ENTITY.CURR)
```

As a result, for all the dimensions where the destination region is defined by one or more \*REC instructions, it is MANDATORY for you to explicitly restrict the region that are to be cleared.

For example, if a translation rule looks like this:

```
*WHEN ACCOUNT.RATETYPE
*IS "AVG", "END"
*REC(FACTOR=LOOKUP(EURO)/LOOKUP(SOURCECURR), CURRENCY="EURO")
*ELSE
*REC(CURRENCY="EURO")
*ENDWHEN
```

The instructions to use are:

```
*CLEAR_DESTINATION
*DESTINATION CURRENCY=EURO
```

If the DESTINATION for the currency dimension is not specified, the destination region for the currency is: LC,EURO

which would result in the loss of all input data.

The members specified in the DESTINATION region can be a list of comma-delimited members (example: \*DESTINATION CURRENCY=EURO,USD) or a member set defined with an MDX expression, for example:

```
*DESTINATION
CURRENCY=filter([CURRENCY].members,[CURRENCY].properties("REPORTING"="Y")
```

A destination region is created by the rules even when no CLEAR\_DESTINATION instruction is used. This is done when the CALCULATE\_DIFFERENCE option is active, to calculate the difference between the newly calculated values and the values existing in the database. In this situation, having a destination region that could be broader than needed cannot do any harm, and you do not need to worry. However, the instruction DESTINATION could still be used, even if no clear\_destination instruction is used, simply to optimize the size of the destination region to query, with some benefit to the performance and memory footprint of the logic execution.

The instruction \*DESTINATION also supports the "not equal to" operator with the syntax:

```
*DESTINATION<>{MemberSet}
```

This operator can be handy to pass to the SQL query smaller lists of valid members, which is more efficiently parsed by the Microsoft SQL engine.

\*COMMIT\_EACH\_LEVEL - more information

Allowed uses: By Commit, MDX, SQL

```
*COMMIT_EACH_LEVEL={dimname}
```

This instruction sorts and group members of the selected dimension and enforces a rules execution from the bottom to the top of the structure, with a commit between each level. This feature can be used to ensure that the eliminations performed on each level take into account the results of the eliminations performed on all lower levels.

\*COMMIT\_EACH\_MEMBER - more information

Allowed uses: By Commit, MDX, SQL

```
*COMMIT_EACH_MEMBER={dimname}
```

This instruction enforces a commit for each member of the selected dimension. If the dimension is of type TIME, the members are also sorted in ascending sequence, so that older periods are processed first. In addition, the rule is executed for all periods between the oldest and the youngest, filling any gaps existing in the range of selected periods. This can be useful for formulas performing a carry-forward of prior periods values as in this example.

```
#ClosingBalance= (ClosingBalance, lag([TIME].currentmember,1)) + Changes
```

---

Note: The lag() function is an MDX function. Please see your Microsoft MDX reference for more information.

---

\*COMMIT\_MAX\_MEMBERS - more information

Allowed uses: By Commit, MDX

```
*COMMIT_MAXMEMBERS
```

Allows you to modify the behavior of the XDIM\_MAXMEMBERS option so that commits are performed after each individual query rather than all at once at the end of the loop of queries.

When the instruction XDIM\_MAXMEMBERS is used in MDX-type rules, the rules query is broken is as many queries as required to accommodate all members to process. However, all resulting records are committed to the database in one lump at the end of the loop of queries, and not after each query. There may be cases where this is not desired (for example for memory limitations), and it may be preferable to perform a commit to the database after each individual query.

You must insert this instruction inside an \*XDIM\_MAXMEMBERS instruction.

\*COMMIT - more information

Allowed uses: By Commit, MDX, SQL

```
*COMMIT
```

By default, the rules engine performs calculation on values in memory. You use the \*COMMIT instruction to post those values to the database. Using the \*COMMIT instruction defines a "COMMIT section." Certain instructions can be used only once per COMMIT section rather than globally.

It may happen that a logic file contains formulas that depend on the result of calculations performed by the application, and that these calculations in turn depend on the results of some other formulas in the same rules.

Take this example:

```
[Account].[#1] = {expression}
[Account].[#2] = ([ENTITY].currentmember.parent,[Account].[#1])
```

In this example account 2 depends on the calculation of the parent entity values performed by the application, and this calculation in turn depends on the calculation of account 1.

This logic, if written in the above format, does not work correctly, because account 1 cannot be retrieved from the parent of the current entity until its result has been posted to the application. To get the right results, account 1 must be calculated AND stored in the application. THEN, the 'calculated' result can be retrieved from the parent and be used to calculate account 2.

In order to force a write back of the result of the calculation of account 1 into the application, the instruction "\*COMMIT" can be inserted between the two calculations, to enforce a write back of account 1 before the calculation of account 2. The logic works if written as follows:

```
[Account].[#1] = {expression}
*COMMIT
[Account].[#2] = ([ENTITY].currentmember.parent,[Account].[1])
```

Note that in this case account 1 in the second formula does not have the pound sign (#), because it is a 'stored' amount read from the application.

---

Note: Any number of commit instructions can be entered in a rules file. However, the number of commit instructions should be kept to a minimum, because they have a negative impact on the overall performance of the rules execution.

---

\*DESTINATION - more information

Allowed uses: By Commit, SQL

\*DESTINATION {DimensionName}={MemberSet}

This instruction is used in conjunction with \*CLEAR\_DESTINATION. Please see the \*CLEAR\_DESTINATION rules for more information.

\*DESTINATION\_APP - more information

Allowed uses: By Commit, MDX, SQL

```
*DESTINATION_APP = {app name}
*SKIP_DIM= {dimension name}[,{dimension name},...]
*ADD_DIM {dimension name}={value}[,{dimension name}={value},...]
*RENAME_DIM {dimension name}={value}[,{dimension name}={value},...]
```

Allows you to write the results of calculations to a different application. Dimensions can be added, removed, or renamed in order to conform with the destination application.

For example, when some data is entered into a divisional application, some of the data may need to also be posted into a central application that consolidates the results of different divisional applications.

Example:

```
*DESTINATION_APP = CentralApplication
```

Often, the destination application shares only some of the dimensions of the original application. In this case the missing dimensions can be dropped from the original records with the instruction:

```
*SKIP_DIM= {dimension name}[,{dimension name},...]
```

Multiple dimension names can be supplied to the instruction separated by commas, or multiple SKIP\_DIM instructions can be entered in separate lines.

If the destination application has dimensions that do not exist in the original application, these can be added to the passed records, using the instruction:

```
*ADD_DIM {dimension name}={value}[,{dimension name}={value},...]
```

Multiple dimension names and values can be supplied to the instruction separated by commas, or multiple ADD\_DIM instructions can be entered on separate lines.

In addition to the instructions ADD\_DIM and SKIP\_DIM, the keyword RENAME\_DIM can be used, to change name of one or more dimensions. The syntax is:

```
*RENAME_DIM {dimension name}={value}[,{dimension name}={value},...]
```

This instruction can be used when data is to be written into an application where a dimension is named with a different ID.

Multiple dimension names and values can be supplied to the instruction separated by commas, or multiple RENAME\_DIM instructions can be entered on separate lines.

Example:

```
*RENAME_DIM ACCOUNT_FLASH= ACCOUNT_MAIN
```

Here is a more complete example:

```
*DESTINATION_APP = CentralApplication
*SKIP_DIM= PRODUCT,MARKET
*ADD_DIM DATASRC=INPUT
*ADD_DIM CURRENCY=LC
*RENAME_DIM ACCOUNTPM=ACCOUNTMAIN
```

In this example some calculated values are transferred into a central application that is not detailed by product and market, but contains two extra dimensions (datasrc and currency). For these two dimensions the members input and lc are used. Also, the chart of account is defined in dimension accountmain.

\*FIRST\_PERIOD more information

Allowed uses: By Commit, MDX, SQL

```
*FIRST_PERIOD
```

This instruction comes as an integration of the \*PRIOR instruction and can be used whenever the switch between the current category and prior category should not happen at year end, but in a different, user-defined period.

The syntax is:

```
*FIRST_PERIOD = {period}
```

For example, say that a rolling forecast is entered in the 12 periods 2005.APR through 2006.MAR, but the preceding values for the months of 2005.JAN through 2005.MAR should be read from a different category.

This result can be achieved combining the following two instructions:

```
*PRIOR CATEGORY = "ACTUAL" // quotes are optional
*FIRST_PERIOD = "APR"
```

An alternative syntax allows the dynamic retrieval of the name of first period by reading a property of the current category, like in this example.

```
*FIRST_PERIOD = CATEGORY.FIRSTPERIOD
```

The search for the correct date will be performed scanning all periods preceding the first modified period moving backwards until a time period with the property PERIOD = {period} is met. For example, if {period} is "MAR" and the first modified date is 2005.FEB, the first date will be 2004.MAR, which is the first date where PERIOD="MAR" going backwards from 2005.FEB.

\*FLAG\_PERIOD more information

Allowed uses: By Commit, MDX, SQL

```
*FLAG_PERIOD= {period}
```

..where period must be a valid value of the PERIOD property of the TIME dimension. (For example the date 2005.MAR would typically have PERIOD="MAR").

The value of {period} can also be retrieved from a property with the following alternative syntax:

```
*FLAG_PERIOD= {dimension }.{property}
```

For example:

```
*FLAG_PERIOD = CATEGORY.FLAGPERIOD
```

This instruction creates a value for the reserved keyword %FLAG\_PERIOD%, which can be used as a replacement string inside the POS( ) keyword as follows:

```
*WHEN POS(TIME)
*IS >=POS(%PREFIX%.%FLAG_PERIOD%)
```

Example:

```
//-----
*FLAG_PERIOD=CATEGORY.FIRSTPERIOD
*WHEN POS(TIME)
*IS >= POS(%PREFIX%.%FLAG_PERIOD%)
*WHEN ACCOUNT
*IS UNITS
*REC(FACTOR=GET(ACCOUNT="PRICE"), ACCOUNT= "REVENUE")
*ENDWHEN
*ELSE
*REC // this may be needed if clear_destination is used
*ENDWHEN
//-----
```

Remarks:

Similar to the \*PRIOR instruction, the \*FLAG\_PERIOD instruction will work dynamically only if just ONE member of the selected dimension (in this example CATEGORY again) is being processed by the logic.

On the other hand, this instruction can also be used in a logic definition where \*CALC\_EACH\_PERIOD is not required.

\*FOR/ \*NEXT - more information

Allowed uses: Global, MDX, SQL

```
*FOR {variable1} = {set1} [ AND {variable2}={set2}]
{text}
{text}
...
*NEXT
```

You can use the \*FOR / \*NEXT structure to define loops over one or more lists of members. In addition, the rules module supports any number and nesting of FOR/NEXT loops in the body of the rules files.

Single For/Next loop

Say, for example, in the default translation rules you want to repeat a calculation for each reporting currency. You can write the following logic:

```
*FOR %CURR%=USD,EURO
//Average Rate for currency %CURR%
```

```
[measures].[!Avg_%CURR%] = {expression}
*NEXT
```

This works the same as the following commands:

```
//Average Rate for currency USD
[measures].[!Avg_USD] = {expression}
//Average Rate for currency EURO
[measures].[!Avg_EURO] = {expression}
```

In addition, \*FOR/NEXT loops support up to two variables iterating on two independent sets of members:

```
*FOR %ACC1%=ThisA,ThisB,ThisC AND %ACC2%= ThatA, ThatB, ThatC
  [ACCOUNT].[#%ACC1%] = ([ACCOUNT].[ %ACC2%],[TIME].currentmember.lag(1))
*NEXT
```

The correct number of members is driven by the set of the first variable. If the first variable has less values than the second variable, the extra values of the second variable is ignored. If the first variable has more values than the second variable, the missing values of the second variable is assumed to be null. Note that this is not a nested loop. It is just one loop on two sets of variables.

FOR/NEXT loops executed on empty lists of elements do not cause the logic execution to fail. Instead, it is skipped without generating an error message.

#### Nested \*FOR / \*NEXT loops

The rules module supports any number of nested levels of FOR/NEXT loops in the body of the rules files. Following is an example of valid syntax:

```
*WHEN TIME
*IS <>TOT.INP
  *WHEN ACCOUNT
    *IS PERCENT.ALLOC
      *FOR %YEAR%=2003,2004,2005
        *FOR
%MONTH%=JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
          *REC(FACTOR=GET(ACCOUNT="TOT.OVRHEAD",TIME="TOT.INP")
/100,TIME="%YEAR%.%MONTH%")
        *NEXT
      *NEXT
    *NEXT
  *ENDWHEN
*ENDWHEN
```

Note that, in case of a single-level loop and the set of elements is empty, the rules still validate and execute correctly. However, in case of nested For/Next loops, none of the loops can contain an empty set of elements, otherwise the rules do not validate.

Nested loops, similar to single-level loops, can handle up to two sets of "parallel" variables, with the syntax:

```
*FOR %VariableOne%=FIRSTSET AND %VariableTwo%=SECONDSET
```

For example, these are two nested loops both using parallel variables:

```
*FOR %X%=1,2,3 AND %Y%=A,B,C
  *FOR %M%=4,5 AND %N%=E,F
  //...
```



```
*NEXT
*NEXT
```

Using a passed set of members at run-time

You can use the passed members of a dimension in a For/Next loop, when the validation is performed at run time.

```
*FOR %MYTIME% = %TIME_SET%
// logic content
*NEXT
```

When the dimension is Time, you can have the passed members ordered in the correct sequence (that is, from earliest to latest). You can do this by inserting the keyword ORDER\_TIME after the \*FOR instruction, as follows:

```
*FOR ORDER_TIME %MYTIME% = %TIME_SET%
```

The ordered set of members will also automatically fill the gaps in the time sequence. For example, if the members to order are 2005.JUN, 2005.FEB, the ordered set will be:

```
2005.FEB, 2005.MAR, 2005.APR, 2005.MAY, 2005.JUN
```

---

Note: The set of items of the FOR loop must represent members of the TIME dimension, otherwise an error message will be returned.

---

\*FUNCTION / \*ENDFUNCTION- more information

Allowed uses: Global, MDX, SQL

This statement has two forms: a single-line format and a multiple-line format:

Single-line format:

```
*FUNCTION {functionname}({Param1}[,{Param2}...]) = {Function Text}
```

Multiple-line format:

```
*FUNCTION {functionname}({Param1}[,{Param2}...])
{Function text}
{Function text}
*ENDFUNCTION
```

The \*FUNCTION statement defines a user-defined function. A function can be inserted into formulas in place of corresponding MDX statements. This can greatly improve the readability of a logic statement.

The following list contains additional rules about the \*FUNCTION statement:

- The definitions of the logic functions can be inserted anywhere in a logic file or in an included file.
- An unlimited number of functions can be defined.
- An unlimited number of parameters can be passed to a function in order dynamically modify the corresponding MDX string.
- Functions can be nested (a function can invoke another function), and any level of nesting is supported.
- The position of the functions in the logic file is irrelevant.
- If multiple instances of the same function are entered in the logic file, the FIRST occurrence prevails on the subsequent ones. This behavior can be used to redefine a function using the "add formula" text box in the TDSRunLogic task.
- The values of the passed parameters are replaced in the function text without any validation, even if they are embedded in longer words, like in this example:

```
*FUNCTION TEST(Param1,Param2)
AParam1Param2D
*ENDFUNCTION
```

The following logic line, calling the above function:

```
[ #123 ] = [ TEST ( B , C ) ]
```

expands into:

```
[ #123 ] = [ ABCD ]
```

For the above reason some caution should be used in defining the names of the parameters, to avoid the risk of conflicts with MDX reserved words and in general with the text surrounding them in logic. One good practice could be to always surround the name of the parameters with some delimiter, like in this example:

```
*FUNCTION GET_PROPERTY ( %DIMNAME% , %PROPERTYNAME% ) = ...
```

- Some characters are invalid in logic functions names. These characters are all those listed below, plus the blank character:

```
+ - / * ^ % > < = ( ) [ ] { } , . ; ' : & \ | # ~ "
```

Invalid characters are trapped during logic validation.

\*GO - more information

Allowed uses: By Commit, SQL

The \*GO instruction is used like a \*COMMIT statement between \*WHEN/\*ENDWHEN structures. The \*GO instruction defines the end of a logic section, much like the \*COMMIT instruction. Unlike a \*COMMIT, data is not posted to the database. Instead, all generated results are merged with the original set of source records and the logic restarts from the beginning of the recordset for the next \*WHEN/\*ENDWHEN structure.

There is no limit to the number of \*GO instructions that can be inserted within a \*COMMIT section. However, since each \*GO generates a little bit of overhead, and requires an additional scan of the record set, their number should be kept to the minimum.

It might appear that the \*GO instruction can be used in place of a \*COMMIT instruction, but this is not quite true. All instructions that are \*COMMIT-specific (for example XDIM\_MEMBERSET) are still \*COMMIT-specific and not \*GO-specific. In other words you cannot redefine the data region to process for each \*GO instruction, but only for each \*COMMIT instruction. The \*GO instruction only sets a stop-and-go point between \*WHEN/\*ENDWHEN structures of the same \*COMMIT section, i.e. of the same data region.

Example of why \*GO was created

In general, multiple sequential WHEN / ENDWHEN structures are processed in sequence for each record found in the source region. As a result, the following example might not work correctly:

```
//-----
*WHEN ACCOUNT
*IS UnitsSold
*REC(FACTOR=GET(ACCOUNT="Price"), ACCOUNT=Sales)
*ENDWHEN
*WHEN ACCOUNT
*IS Sales
*REC(FACTOR=.08, ACCOUNT=SaleTaxes)
*ENDWHEN
//-----
```

The reason this logic does not work is that the value of account Sales held in memory is not the newly calculated one, but the value found in the database before the new value is calculated. Sales, once calculated, must be posted to the database with a \*COMMIT instruction, and then retrieved for the subsequent calculation of Sales Taxes, as follows:

```
//-----
*WHEN ACCOUNT
```

```

*IS UnitsSold
*REC(FACTOR=GET(ACCOUNT="Price"), ACCOUNT=Sales)
*ENDWHEN
*COMMIT
*WHEN ACCOUNT
*IS Sales
*REC(FACTOR=.08, ACCOUNT=SalesTaxes)
*ENDWHEN
//-----

```

A better way to write this logic would be to calculate SalesTaxes at the same time Sales are calculated, as shown here:

```

//-----
*WHEN ACCOUNT
*IS UnitsSold
*REC(FACTOR=GET(ACCOUNT="Price"), ACCOUNT=Sales)
*REC(FACTOR=GET(ACCOUNT="Price") * .08, ACCOUNT=SalesTaxes)
*ENDWHEN
//-----

```

However, there are situations where this is not a practical approach. For these situations, in order to avoid the inefficiencies of a \*COMMIT step, the instruction \*GO can be used. Here is how the above logic would look using \*GO:

```

//-----
*WHEN ACCOUNT
*IS UnitsSold
*REC(FACTOR=GET(ACCOUNT="Price"), ACCOUNT=Sales)
*ENDWHEN
*GO
*WHEN ACCOUNT
*IS Sales
*REC(FACTOR=.08, ACCOUNT=SalesTaxes)
*ENDWHEN
//-----

```

\*INCLUDE - more information

Allowed uses: Global, MDX, SQL

\*INCLUDE {includedfile}(Param1,Param2,...)

Allows you to include other logic files in the current rules file. If no file name extension is given, .LGF is assumed. Can be used anywhere inside a rules file.

The benefit of using included files instead of multi-line functions is that this method permits you to include entire logic sections, containing any type of instructions like \*COMMIT, \*XDIM\_MEMBER, etc., while logic functions can only perform string substitutions on single line statements. On the other hand, functions do not need to be written in separate files.

Parameters

You can pass parameters to the included logic. The parameters are referenced as %P1%, %P2%, etc.

For example:

```
*INCLUDE MyModule.LGF (REVENUE, COST)
```

The content of the file MyModule.LGF could be:

```
#GROSSPROFIT= [ACCOUNT].[%P1%] - [ACCOUNT].[%P2%]
#GROSSMARGIN= (([ACCOUNT].[%P1%] -
[ACCOUNT].[%P2%])/[ACCOUNT].[%P1%])*100
```

Note: You can get similar functionality using \*SUB procedures. SUB procedures can have user-named parameters, and do not need to be stored in individual external files.

\*JOIN - more information

Allowed uses: By Commit, SQL

\*JOIN( {tablename} , {dimension.property} [, {tablefield}] [, {selected fields}])

Where...	Is...
{tablename}	The name of the user-defined table.
{dimension.property}	The dimension and property that should be used to join the table with the BPC data.
{tablefield}	The field in the selected tables on which the dimension property is joined.
{selected fields}	The fields in the table that are of interest in the current logic. If more than one, the list of fields must be enclosed in double quotes.

With this instruction the rules can be linked to the values entered in some field of a user-defined table.

Example:

```
*JOIN(RulesTable, Account.Rule, Rule, "Destacc, Factor")
```

The above instruction tells the rules to join the BPC data with the data contained in a table called Consolidation Rules, performing a join of the property Rule of the account dimension and the field Rule of the RulesTable table. The fields to read in the RulesTable are DestAcc and Factor. Behind the scenes, the generated SQL query looks more or less as follows:

select

```
... , mbrACCOUNT.[RULE] AS [ACCOUNT.RULE],
RULESTABLE.[DestAcc] AS [RULESTABLE.DestAcc],
RULESTABLE.[Factor] AS [RULESTABLE.Factor]
INNER JOIN RULESTABLE on mbrACCOUNT.[RULE] = RULESTABLE.[RULE]
```

The parameters {tablefield} and {selected fields} are optional. If omitted, the {tablefield} is assumed to be the same as the property of the joined dimension, and the {selected fields} are assumed to be all fields in the table. In other words, the above example could have been written as follows:

```
*JOIN(RulesTable, Account.Rule)
```

With the above technique, an account could be assigned a rule of behavior using a regular property. A separate table of Rules could then be maintained, defining in a generic way what a rule should imply in the execution of a rule.

Once the join has been defined, all fields in the joined table can be used anywhere in the WHEN/ENDWHEN statement to perform the appropriate calculations.

Example:

```
*JOIN(AccountRules,account.rule)
*WHEN *
*IS *
*REC(FACTOR= AccountRules.Factor, ACCOUNT=AccountRules.DestAcc)
```

### \*ENDWHEN

By nature a join only accepts the values of the selected BPC region of data that have in the selected property a value corresponding to the joined field in the joined table. In other words, still referring to the above example, all records not having a valid Rule is skipped, even if included in the selected region. This could also be considered as a way to filter the members of a dimension without using a XDIM\_MEMBERSET instruction.

\*LAST\_MEMBER - more information

Allowed uses: By Commit, MDX

\*LAST\_MEMBER {Dimension name} = {Member}

With this instruction, the rule is run for the indicated dimension from the first passed member to the member specified in the instruction itself. For example, if the instruction says:

\*LAST\_MEMBER TIME=2002.DEC

...the logic runs from the first modified period up to period 2002.DEC.

The year can be made dynamic using the keyword %PREFIX% as follows:

\*LAST\_MEMBER TIME=%PREFIX%.DEC

In this case, the year is the year of the last modified period.

The instruction can also interpret some MDX functions in the passed parameters. These can be used, for example, to define an offset from a given period. The following syntax:

\*LAST\_MEMBER TIME=[%PREFIX%.DEC].LEAD(6)

enables the rules to run from the first selected period until June of the following year.

The member name must be enclosed in brackets ([]) to be recognized as an MDX call. The dimension name must be omitted from the expression.

\*LOCAL\_CURRENCY - more information

Allowed uses: Global, MDX, SQL

\*LOCAL\_CURRENCY= {local currency member}

Changes the default local currency member from "LC" to the specified member.

Example:

\*LOCAL\_CURRENCY= ValutaLocale // this is Italian!

This instruction must be used in all applications where the local currency member of the currency dimension is named differently from "LC", even if the local currency member is not explicitly mentioned in your logic formulas, because there are several cases where the Logic engine creates queries that make automatic reference to it behind the scenes.

This instruction can be entered just once in the whole logic and does not need to be repeated in each commit section. For a cleaner design, a good practice is to include this instruction in the application constants library APP\_CONSTANTS.LGL.

\*LOGIC\_BY - more information

Allowed uses: Global, MDX, SQL

\*LOGIC\_BY = {dimensions list}

(Only valid for \*LOGIC\_MODE = 2)

Overrides the default dimensions to scan in order to build the names of logic files to execute when in logic mode 2.

In logic mode 2 (multiple logics), the program builds the names of the logics to execute reading the content of the user-defined property named "LOGIC" in all members being processed for the dimensions CATEGORY and ENTITY.

For example, if the category and entity being processed are ACTUAL and SALESITALY respectively, and these members have, as "LOGIC" property, the values 'ACT' and '1' respectively, the logic being executed is ACT1.LGX.

The dimensions to scan in order to build the name of the logic to execute are by default category and entity, but this default can be overridden with the instruction:

```
*LOGIC_BY = {dimensions list}
```

For example, the instruction could say:

```
*LOGIC_BY = TIME, DATASRC
```

This means that the logic varies by time period and by member of the DATASRC dimension.

The order in which the dimensions are written in the instructions controls the sequence in which the content of the 'LOGIC' properties are concatenated (the module reads the dimension names from left to right). Any number of dimensions can be specified.

Blank values for the 'LOGIC' property are acceptable, as long as at least one of the dimensions listed in the instruction has a non-blank value (otherwise the resulting logic name is blank).

LOGIC\_MODE is automatically set to 2 in case the instruction \*LOGIC\_BY is used.

The Logic engine does not set any limit to the number of dimensions by which the logic may vary. However, we recommend to not use more than two dimensions for this purpose, as this could lead to a very high number of logic files to define and maintain and excessive fragmentation of the logic execution in many small queries, resulting in unsatisfactory performance.

\*LOGIC\_MODE - more information

Allowed uses: By Commit, MDX

```
*LOGIC_MODE = 0 | 1 | 2
```

The Logic module can select the logic to execute according to three different "logic modes":

- Mode 0: run the default logic DEFAULT.LGX.
- Mode 1: override the default logic, and use a different one.
- Mode 2: override the default logic, and use one or more logics, identified using some special criteria.

By default, the Logic module runs in mode 0 (use the default logic), but the mode could be overridden with the logic instruction:

```
*LOGIC_MODE = 0 | 1 | 2
```

(In reality this instruction makes only sense in the form: \*LOGIC\_MODE = 2, as the former two settings would be useless inside a logic file. Their values are controlled by the UI of the EvDTSRunLogic task).

The instruction:

```
*LOGIC_MODE = 2
```

tells the program NOT to run the default logic or any other given logic, but to run one or more logics, to be identified according to some specific criteria. In this mode, the Logic module scans the data file for all members of dimensions listed in the \*LOGIC\_BY instruction, searches for a property named "LOGIC", and uses that property to build the logic file name.

For example, if you enter "ENTITY" and the data file contains records for SalesItaly, and SalesItaly has "LOGIC\_XYZ" in the LOGIC field, the logic file used against the data of entity SalesItaly is LOGIC\_XYZ.LGX.

---

**Note:** If not set up correctly, setting the logic mode to two (2) can cause a serious degradation of performance. Plan carefully if you think you need to use this feature.

---

LOGIC\_MODE is automatically set to 2 in case the instruction \*LOGIC\_BY is used and this instruction generally never has to be specifically invoked by someone writing logic files.

\*LOGIC\_PROPERTY - more information

Allowed uses: Global, MDX, SQL

```
*LOGIC_PROPERTY = {property name}
```

This instruction works in conjunction with \*LOGIC\_BY. This instruction is used to override the default name of the property driving the selection of the logic to use.

This feature can be useful when the default property name "LOGIC" is already taken by another set of logics. For example the DEFAULT logic could vary by category using the default property LOGIC while a consolidation logic could vary by category using the property CONSOL\_LOGIC.

Example:

```
// Content of  DEFAULT.LGF
//-----
*LOGIC_BY = CATEGORY
// Content of  CONSOL_LOGIC.LGF
//-----
*LOGIC_BY = CATEGORY
*LOGIC_PROPERTY = CONSOL_LOGIC
Content of CATEGORY.XLS:
ID LOGIC CONSOL_LOGIC
ACTUAL Default1 ConsLogic1
BUDGET Default2 ConsLogic2
```

\*LOOKUP / \*ENDLOOKUP - more information

Allowed uses: By Commit, SQL

This set of instructions can be used in conjunction with a WHEN/ENDWHEN structure to retrieve (lookup) some other values that may be needed either in the calculation of the new value or to define some criteria to be evaluated. The lookup can be performed in the current application or into a different application.

The lookup mechanism essentially defines a relationship between the current record being processed and another record in a corresponding user-defined record set. For example, in a currency translation, you may want to identify, in the RATE application, the value of the rate for the current entity, category and period.

The syntax is:

```
*LOOKUP {App}

*DIM [{LookupID}:] {DimensionName}="Value" |
{CallingDimensionName}[.{Property}]

[*DIM ...]

*ENDLOOKUP
```

Where...	Is...
{App}	The name of the application from which the amounts are searched
{DimensionName}	A dimension in the lookup app
{CallingDimensionName}	A dimension in the current application
{LookupID}	An optional identifier of the "looked-up" amount. This is only required when multiple values must be retrieved.

Example:

```
*LOOKUP RATE
*DIM ENTITY2="DEFAULT"
*DIM SOURCECURR: INPUTCURRENCY=ENTITY.CURR
*DIM DESTCURR1: INPUTCURRENCY="USD"
```

```
*DIM DESTCURR2: INPUTCURRENCY=" EURO "
*DIM RATE=ACCOUNT.RATETYPE
*ENDLOOKUP
```

In the above example, three different values are retrieved from the INPUTCURRENCY dimension (the rate of the currency of the current entity, the rate of the currency EURO and the rate of the currency USD). Each of these values has been assigned a specific identifier (SOURCECURRE, DESTCURR1 and DESTCURR2) that are used somewhere in the WHEN/ENDWHEN structure.

Any dimension not specified in the lookup instruction is assumed to match with a corresponding dimension in the source application. In the above example, the following instructions have been omitted, because redundant:

```
*DIM CATEGORY=CATEGORY
*DIM TIME=TIME

In the following example, a currency translation in the two reporting
currencies USD, and EURO is performed.

// ----- Get the rates
*LOOKUP RATE
*DIM ENTITY2="DEFAULT"
*DIM RATE=ACCOUNT.RATETYPE
*DIM SOURCECURRE: INPUTCURRENCY=ENTITY.CURR
*DIM DESTCURR1: INPUTCURRENCY="USD"
*DIM DESTCURR2: INPUTCURRENCY="EURO"
*ENDLOOKUP

// ----- Translate
*WHEN ACCOUNT.RATETYPE
    *IS "AVG", "END"
*REC(FACTOR=LOOKUP(DESTCURR1)/LOOKUP(SOURCECURRE), CURRENCY="USD")
*REC(FACTOR=LOOKUP(DESTCURR2)/LOOKUP(SOURCECURRE), CURRENCY="EURO")
    *ELSE
*REC(CURRENCY="USD")
*REC(CURRENCY="EURO")
*ENDWHEN
*COMMIT

// -----
-----
```

Below is a different example of how a LOOKUP amount can be used to define a WHEN criteria. In this case, what is being tested is an amount (corresponding to a consolidation METHOD) in the lookup application. (The logic is not very meaningful, but it is a simplified version of a real one. Take it only as an example of valid syntax)

```
// -----Get the methods and percent consol
*LOOKUP OWNERSHIP
*DIM INTCO="IC_NONE"
*DIM PARENT="MYPARENT"
*DIM MY_METHOD: ACCOUNTOWN="METHOD"
*DIM IC_METHOD: ACCOUNTOWN="METHOD"
```



```

*DIM PCON:                ACCOUNTOWN="PCON"
*DIM MY_METHOD:ENTITY=ENTITY
*DIM IC_METHOD:  ENTITY=INTCO.ENTITY
*DIM PCON:                ENTITY=ENTITY
*ENDLOOKUP
*WHEN LOOKUP(MY_METHOD) // check my method
    *IS 1,2,3
*WHEN LOOKUP(IC_METHOD) // check the method of the partner
    *IS 1,2,3
        *REC(FACTOR=LOOKUP(PCON), PARENT ="MYPARENT")
    *ENDWHEN
*ENDWHEN
// -----

```

Finally, a LOOKUP keyword can also be used as part of an \*IS statement, as shown in the following example:

```

// ----- Get the percent consols
*LOOKUP OWNERSHIP
*DIM INTCO="IC_NONE"
*DIM PARENT="MYPARENT"
*DIM PCON:        ACCOUNTOWN="PCON"
*DIM IC_PCON:ACCOUNTOWN="PCON"
*DIM PCON:        ENTITY=ENTITY
*DIM IC_PCON:ENTITY=INTCO.ENTITY
*ENDLOOKUP
*WHEN LOOKUP(PCON)
*IS <= LOOKUP(IC_PCON)
*REC(FACTOR=-1, PARENT ="MYPARENT",DATASRC="ELIM")
*ENDWHEN
// -----

```

The \*WHEN instruction can also take a property as a parameter of one of the dimensions of the application against which a \*LOOKUP has been performed, even if such dimension does not exist in the current application.

In the following example, a currency translation checks for the MD field of the source currency, in order to decide what formula to apply to the rate (Multiply or Divide):

```

//-----
// load the rates from the RATE application
//-----
*LOOKUP RATECUBE
*DIM RATEENTITY="GLOBAL"
*DIM RATE=ACCOUNT.RATETYPE
*DIM SOURCECRR:INPUTCURRENCY=ENTITY.CURR

```

```

*DIM USD:INPUTCURRENCY="USD"
*NEXT
*ENDLOOKUP

//=====
// define the translation rule
//=====

*WHEN ACCOUNT.RATETYPE
*IS "AVG","END"
// check the multiply or divide property of the currency
*WHEN INPUTCURRENCY.MD
*IS "D"
    *REC(FACTOR=LOOKUP(USD)/LOOKUP(SOURCECURR),CURRENCY="USD")
*ELSE
    *REC(FACTOR=LOOKUP(SOURCECURR)/LOOKUP(USD),CURRENCY="USD")
*ENDWHEN
*ELSE
*REC(CURRENCY="USD")
*ENDWHEN

```

The LOOKUP instructions must define the link between the dimension referenced to in the WHEN statement and one of the dimensions of the current application. In the above example the logic understands automatically that it needs to evaluate the MD field of the InputCurrency matching the Currency of the current Entity.

#### MDX-based LOOKUP

The \*LOOKUP / \*ENDLOOKUP structure normally generates an SQL query, and, as such, is unable to return values calculated by the OLAP application. To overcome this limitation a different version of the LOOKUP instruction has been implemented. This instruction generates an MDX query and, as a result, allows the retrieval of any value available in the application.

The syntax is:

```
*OLAPLOOKUP [{application name}]
```

All other instructions specified inside the structure remain the same as a regular LOOKUP instruction.

Example:

```

*OLAPLOOKUP FINANCE
*DIM ENTITY="SALESEUROPE"
*DIM ACCOUNT="REVENUE"
*ENDLOOKUP

```

The above example is able to retrieve the REVENUE account of entity SALESEUROPE, even if one or both these members are parent or otherwise calculated by the application.

This feature, although less efficient than an SQL-based LOOKUP, can be particularly useful for accessing aggregated data that need to be used in the factor of an allocation logic. Below is an example of allocation logic where the expense HUMAN\_RES\_EXP incurred by the entity CORP\_SERVICES is allocated to all children of EUROPE based on their number of employees (HEADCOUNT):

```

//-----
-----
*XDIM_MEMBERSET ENTITY=[ENTITY].[EUROPE].children
*OLAPLOOKUP FINANCE

```

```

*DIM TOT_HC:ENTITY="EUROPE"
*DIM TOT_HC:ACCOUNT="HEADCOUNT"
*DIM TOT_HR:ENTITY="CORP_SERVICES"
*DIM TOT_HR:ACCOUNT="HUMAN_RES_EXP"
*ENDLOOKUP
*WHEN ACCOUNT
*IS HEADCOUNT
*REC(FACTOR=LOOKUP(TOT_HR)/LOOKUP(TOT_HC),ACCOUNT="ALLOCATED_HR_EXP")
*ENDWHEN
//-----
-----

```

\*MEASURES - more information

Allowed uses: By Commit, MDX

\*MEASURES = {dimension}

Use this instruction to tell the Logic module that the results of the query in the measures dimension contain information from another dimension.

This functionality can be used whenever a logic calculates members using the measures dimension, for efficiency.

\*MEMBERSET - more information

Allowed uses: By Commit, MDX, SQL

```
*MEMBERSET({variable}, {member set in MDX format})
```

Allows you to retrieve a list of elements from a dimension and save it in a user-defined variable for later use anywhere else in the logic.

For example, with this instruction:

```
*MEMBERSET((%REPORTING_CURRENCIES%, "filter{[CURRENCY].members,
[currency].properties("GROUP")="REP"")
```

You can fill the variable %REPORTING\_CURRENCIES% with the list of reporting currencies defined in the current application. The content of the resulting variable can then be used anywhere in the logic, as in this example:

```
*XDIM_MEMBER_SET CURRENCY=%REPORTING_CURRENCIES%
```

Important remarks:

1. The MEMBERSET statement generates an MDX query, and can be preferred to the SELECT statement to perform complex hierarchical selections or in general whenever an MDX query is more appropriate than a SQL one. If you want to select a list of elements from a dimension using a SQL query, use the \*SELECT statement.
2. In case any parameter contains embedded commas the entire parameter must be enclosed in an extra set of double quotes.
3. The \*MEMBERSET statement is executed at the time the logic is validated, and the expanded result is written in the LGX file. This means that if the related dimension is modified, it may be necessary to re-validate the logic.
4. Statements returning no members do not necessarily cause the validation of the logic to fail. In this case a warning is entered in the validation log.
5. These instructions are not specific to a given logic section, but they can be written once anywhere in the logic and used across multiple commit sections.

Memory variables

It is possible to create intermediate result and assign them to dummy members (like dummy accounts or dummy members of any other dimension). These members can be used as placeholders to store

intermediate results that can be used as inputs for subsequent calculations. These values are automatically skipped at commit time.

Dummy members must be identified with a leading pound (#) sign. For example:

```
*REC (ACCOUNT = #TEMP)
```

Account #TEMP does not exist in the account dimension. The generated record can be used somewhere else in the rules, but its value is not stored in the database.

Example:

```
*WHEN ACCOUNT.FLAG
*IS = Y
*REC (ACCOUNT=#TEMP)
*ENDWHEN
*GO
*WHEN ACCOUNT
*IS #TEMP
*REC (FACTOR=GET (ACCOUNT=MULTIPLIER) ,ACCOUNT=FINAL)
*ENDWHEN
```

The above technique could be used in an allocation procedure, requiring the calculation of the total value of the coefficient to use in the allocation process. Alternatively, you could calculate an opening balance amount that does not need to be stored in the database.

The dummy members generated with this technique do not exist in any dimension. Because of this, it is not possible to assign to them properties that could be used in the rules. They can only be referenced using their ID.

#### Using memory variables in WHEN / ENDWHEN instructions

When a memory variable represents a real member of a dimension, you can now access its properties in a WHEN / ENDWHEN structure.

The logic may use the properties of a valid memory variable in evaluating a WHEN criteria as well as in the definition of a destination member.

The following example uses the property of a memory variable to read an aggregated value from a parent and transfer it into a base level member whose ID is stored in a parent's property:

```
//-----
// create the memory variables (in this case the parents in H1, for
// example #SALES, #WORLDWIDE1)
*CALC_DUMMY_ORG ENTITY=PARENTH1
// Some parent might have a corresponding input member specified in a
// property
*WHEN ENTITY.INPUTMEMBER
*IS<>" "
*REC (ENTITY=ENTITY.INPUTMEMBER)
*ENDWHEN
//-----
```

Similarly, it is now possible to access the value of a memory variable from a GET instruction, using the syntax shown in the following example:

```
//-----
*XDIM_MEMBER ACCOUNT=SQ_FEET
// create the memory variables
*CALC_DUMMY_ORG ENTITY=PARENTH1
// Calculate the allocation factor of each base member (note # used to
identify the dummy parent)
*WHEN ENTITY.ISBASEMEM
*IS="Y"
    *REC(FACTOR=1/GET(ENTITY=# + ENTITY.PARENTH1) * 100,
ACCOUNT=PCT_SQ_FEET)
*ENDWHEN
//-----
```

If the base-level members have many levels of parents, it is also possible to specify the number of levels to ascend, in the search for the value of a "parent" member. The syntax is:

```
GET(dimension=dimension.property (number of levels) )
```

The example below is a search performed in a parent that is 4 levels above current member. Note that this example does not use memory variables, because the parents in the stored hierarchy PARENTS1 are actually stored (in fact the function used is CALC\_ORG and not CALC\_DUMMY\_ORG).

```
//-----
*XDIM_MEMBER ACCOUNT=SQ_FEET
// Calculate the parents to store
*CALC_ ORG ENTITY=PARENTS1
// Calculate the allocation factor of each base member having 4 levels of
parents above
*WHEN ENTITY.PARENTS1
*IS<>" "
    *REC(FACTOR=1/GET(ENTITY=ENTITY.PARENTS1(4)) * 100,
ACCOUNT=PCT_SQ_FEET)
*ENDWHEN
//-----
```

\*NO\_PARALLEL\_QUERY - more information

Allowed uses: By Commit, SQL

\*NO\_PARALLEL\_QUERY

This instruction suppresses the generation of parallel queries on the server.

SQL server automatically tries to run multiple queries in parallel, if possible. In case many k2logic executions are requested to the server simultaneously, this mechanism may end up using too much memory on the server, ultimately slowing down the process, or even generating an "out of memory" error.

### \*PRIOR more information

It may sometimes happen that two different data categories need to be linked in time for some logic calculation. A good example is a BUDGET category that must retrieve the opening balances of the balance sheet from category ACTUAL.

This link can now be easily established with the following instruction:

```
*PRIOR {dimension name} = {member ID}
```

Example:

```
*PRIOR CATEGORY = ACTUAL
```

This instruction automatically forces the retrieval of prior period's values from a different category (in this case ACTUAL).

An alternative syntax allows the dynamic retrieval of the name of the prior category reading a property of the current category, like in this example.

```
*PRIOR CATEGORY = CATEGORY.PRIORCAT
```

Note that this syntax only makes sense in a case where the logic needs to go backwards in time to retrieve values from past periods. For example, assume you want to load in memory the three months preceding the first modified month. This will be normally achieved with the instruction:

```
*XDIM_MEMBERSET TIME = PRIOR(3),%SET%
```

However, if the first modified period is FEB, going backwards three months will cross the year-end boundary, and NOV and DEC will be retrieved from last year. Now, if last year's data are not stored in the current category but in a different one, the above instruction will do the job.

It is important to realize a few things:

1. Prior category does not need to be included in the set of categories to load in memory: its data will be automatically loaded in memory if needed (according to the number of periods the logic goes backwards).
2. The values of prior category will be available for processing by the logic, but it is very likely that these should not be modified (last year data might even be locked) and the logic will have to skip them. However, the way to control this is NOT to check the name of the category with something like:

```
// This will not work!!!
*WHEN CATEGORY
*IS ACTUAL
```

The good news is that in reality what needs to be skipped are not the values coming from prior category, but the values coming in ANY of the prior periods, irrespective of whether they belong to ACTUAL or BUDGET (if the first modified period is FEB, all three periods NOV DEC and JAN must not be changed). This condition can be easily tested with these instructions:

```
// This will work!!!
*WHEN TIME
*IS PRIOR
```

3. The logic will always work as if all periods come from current category, even while reading records that come from prior category. For this reason the destination category does not need to be specified in the REC instructions.

```
// This will write in ACTUAL, even if reading a record coming from BUDGET
*REC(ACCOUNT="#TEMP",TIME=NEXT(3))
```

4. In lead and lag calculations the most common (and sometimes difficult) decision to take is whether one should get values from the past or push values into the future. This may sometimes depend on the type of formula. In most cases it is more practical to push values into the future using the existence of values in the past as triggers.

```
// Example
*WHEN ACCOUNT
```

```
*IS COLLECTIBLES_AT_THREE_MONTHS
*REC(ACCOUNT=#COLLECTIONS_AT_THREE_MONTHS, TIME=NEXT(3))
*ENDWHEN
```

Note that this logic should only write in valid periods (<>PRIOR). This is why the logic stores the value in a temporary variable (using a leading pound sign #). The temporary values can then be written only if the period is correct.

```
*WHEN TIME
*IS <> PRIOR
*WHEN ACCOUNT
*IS #COLLECTIONS_AT_THREE_MONTHS
*REC(ACCOUNT=COLLECTIONS_AT_THREE_MONTHS)
*ENDWHEN
*ENDWHEN
```

5. This instruction will only work as desired if only ONE CATEGORY at a time is modified by the user during data entry (or processed from a Data Manager task). The administrator should somehow make sure that a logic containing this instruction is used correctly.
6. It is important to remember to use the above techniques in conjunction with the \*CALC\_EACH\_PERIOD option. This will enforce the calculation of all periods in the correct sequence.

#### \*PROCESS\_EACH\_MEMBER - more information

Allowed uses: Global, MDX, SQL

```
*PROCESS_EACH_MEMBER={dimname1}[ , {dimname2}, ...]
```

This instruction enforces a commit for each member of the selected dimension in the whole logic file, not just in a specific \*COMMIT section. This instruction is similar to \*COMMIT\_EACH\_MEMBER with the following important differences:

- It applies to the entire rule and not to a specific COMMIT section
- It is processed separately, before the rule is interpreted. This makes the rules behave exactly as if it had been called multiple times (once for each member in the dimensions listed in the instruction).
- The member set to process for the named dimensions must be explicitly passed to the rules call, as if such dimension sets were required (see instruction XDIM\_REQUIRED).
- It applies to both MDX logic as well as SQL logics (while currently COMMIT\_EACH\_MEMBER only works on MDX logic)

Similar to the COMMIT\_EACH\_MEMBER instruction, this instruction processes the TIME dimension in a special way. It sorts the time members from the oldest to the newest and fills all gaps in the range. For example, if the members passed are:

```
2001.Mar, 2001.Jan
```

Then the resulting set whose members are processed individually is:

```
2001.Jan, 2001.Feb, 2001.Mar
```

#### \*PROCESS\_FAC2 - more information

Allowed uses: By Commit, MDX, SQL

#### \*PROCESS\_FAC2

This instruction can be used to trigger the processing of the FAC2 partition of BPC cubes after a commit directly into the FAC2 partition has been performed. The instruction applies to a single commit section.

Example:

```
*WRITE_TO_FAC2
```

```
*PROCESS_FAC2
WHEN *
*IS *
    *REC(... )
*ENDWHEN
*COMMIT
```

\*PUT - more information

Allowed uses: By Commit, MDX, SQL

This instruction can be used to directly write values in a selected region. The syntax is:

```
*PUT({dimension}={member}[ , {dimension}={member} ]
[EXPRESSION={expression}... )
```

Example:

```
*PUT(ACCOUNT=FLAG, INTCO=NONE, CURRENCY=LC, EXPRESSION=123)
```

This instruction writes the value 123 in account=flag, intco=none and currency=lc for all entities, categories and time periods passed in the selected region.

The instruction can be used to flag a region of data with a value indicating the status of the data. For example, whenever data is entered in a certain entity, the default logic could flag that entity as impacted. Later on, a currency conversion could be executed just for the impacted entities. At the end of its execution, the translation logic could de-impact these entities, by setting their flag account back to zero.

Important remarks:

The parameters can only take explicit values and cannot be derived from a dimension property (unless passing through a SELECT instruction).

The scope of the records to write is by default defined as all the members passed for the dimensions category, time and entity. This default scope can be modified with the instruction \*PUTSCOPE\_BY, which is similar to the SCOPE\_BY instruction, but is specific to the PUT instruction.

Example:

```
*PUTSCOPE_BY=CATEGORY, TIME, DATASRC
```

The destination member(s) must be specified for all dimensions. The dimensions defined in the scope derive the correct members from the passed region. All other dimensions must have one member specified in the PUT parameters or by a XDIM\_MEMBERSET or similar instruction.

Example:

```
*XDIM_MEMBERSET INTCO=NONE
*XDIM_MEMBERSET CURRENCY=LC
*PUT(ACCOUNT=FLAG, EXPRESSION=123)
```

The EXPRESSION parameter is optional. If omitted it defaults to 1. This example generates a value of 1 for account flag:

```
*PUT(ACCOUNT=FLAG)
```

Multiple PUT instructions can be specified in the same commit section. Example:

```
*XDIM_MEMBERSET INTCO=NONE
*XDIM_MEMBERSET ACCOUNT=FLAG
*PUT(CURRENCY=USD)
*PUT(CURRENCY=EURO)
```

The PUT instruction MUST be defined in a STANDALONE COMMIT section. It cannot be mixed with MDX formulas or with WHEN/ENDWHEN structures.

The PUT instruction always and automatically enforces a \*calculate\_difference=0



The PUT instruction supports the ability to write into an application different from the source. The following is a valid example of logic that runs from a RATE application and writes in the FINANCE application.

```
*destinationapp=finance
*skipdim=inputcurrency
*skipdim=rate
*skipdim=entity2
*adddim intco=non_interco
*Adddim datasrc=%DATASRC% // this is needed if the datasrc is re-directed
in the PUT
*Adddim account=flag
*adddim currency=lc
*adddim entity=dummy

*PUT(datasrc=input)
*PUT(datasrc=adjustment)
```

\*QUERY\_FILTER - more information

Allowed uses: By Commit, MDX

\*QUERY\_FILTER={member}

This instruction enforces a \*QUERY\_TYPE=2 (nonemptycrossjoin) in the MDX rules execution, and causes the system to search for existing values to be performed on the passed member.

Example:

```
*QUERY_FILTER=NetProfit
```

The rules module assumes the member to be an account if no dimension is specified. Otherwise, the correct dimension can be explicitly named in the usual MDX format, for example, as follows:

```
*QUERY_FILTER=[CATEGORY].[ACTUAL]
```

\*QUERY\_SIZE - more information

Allowed uses: By Commit, MDX

\*QUERY\_SIZE = 0 | 1 | 2

When the module runs in multi-rule logic mode (\*LOGIC\_MODE = 2), it automatically tries to group the regions of data sharing the same logic to execute and tries to run the largest possible queries against a given logic, in order to maximize the speed of execution. In many cases, however, the resulting queries may be too large to fit in memory, and the performance of large queries can actually deteriorate significantly, instead of improving.

To prevent this situation, the administrator can use the \*QUERY\_SIZE instruction with the following values:

- 2 is the (default) largest size of queries
- 1 is an intermediate size, and
- 0 is the smallest practical size

---

**Note:** Note that the effects of this instruction combine with those of the instruction XDIM\_MAXMEMBERS, in defining the scope of the queries ultimately being run. The appropriate combination of these two settings should be identified, in order to identify the best compromise between performance and memory usage.

---

Breaking the query in multiple queries (when scanning a data file)

When the rules module scans a data file to build a region against which to run, it breaks the query into many smaller queries, according to the different combinations of members it finds in the file, for the dimension defining the scope. See \*SCOPE\_BY. In this situation, similar to the case of multiple rules

(logic\_mode = 2), the program can try to maximize the size of the query, according to the setting of the QUERY\_SIZE parameter.

\*QUERY\_TYPE - more information

Allowed uses: By Commit, MDX

You can control the type of MDX queries generated by the rules engine. Set this instruction to 0 (default multi-axis), 1 (row/column/multiple crossjoins in rows), or 2 (one nonemptycrossjoin).

While the rules module by default generates a multi-axis query, the format of the MDX query can be controlled by the logic using the instruction:

\*QUERY\_TYPE=0 | 1 | 2

Where...	Is...
0	The default multi-axis type
1	A row/column type, with multiple crossjoins in rows
2	A row/column type, with one nonemptycrossjoin in rows

The query type 1 (row/column crossjoin) can be useful when you want to check the generated query using Microsoft MDX Samples program. This product, in fact, does not support multi-axis queries. A query type 1 can simply be copy/pasted from the debug file into the MDX sample UI and executed.

While the type 2 (nonemptycrossjoin) is by far the fastest format, it only gives the desired results in selected cases, and it must be used with care. One example of its use can be found in the Default translation logic in Apshell. This type of query can be also controlled using the instruction

\*QUERY\_FILTER.

\*RENAME\_DIM - more information

Allowed uses: By Commit, MDX, SQL

\*RENAME\_DIM {dimension name}={value}[, {dimension name}={value},...]

Works in conjunction with the \*DESTINATION\_APP instruction to post results of logic calculations to a different application.

\*RUN\_STORED\_PROCEDURE - more information

Allowed uses: By Commit

Used to invoke the execution of an SQL stored procedure.

```
*RUN_STORED_PROCEDURE={stored procedure name}(' {params}')
```

Example:

```
*RUN_STORED_PROCEDURE=spEliminate(ACTUAL, '2001.JAN')
```

Parameters containing delimiter characters (like 2001.JAN) should be passed between single quotation marks (like '2001.JAN').

To pass multiple values in a single parameter, it may be appropriate to enclose them between single quotes. However, it is up to the stored procedure code to support the un-packing of that parameter into its individual members, if appropriate. See this example:

```
*RUN_STORED_PROCEDURE=spCompare('2001.JAN,2001.FEB','ITALY,FRANCE')
```

Stored procedures must be written in their own commit section, i.e., they cannot coexist with a WHEN/ENDWHEN structure, nor can be part of any MDX rules. On the other hand, multiple stored procedures can be executed from the same commit section, like in the following example:

```
*RUN_STORED_PROCEDURE=FirstProcedure('%TIME_SET%')
*RUN_STORED_PROCEDURE=SecondProcedure('%TIME_SET%')
*COMMIT
[account].[#SALES]=Units * Price
```

#### \*COMMIT

The instruction RUN\_STORED\_PROCEDURE supports the keyword %APP% as the current application name. This is a common requirement for stored procedures that need to work on different applications.

#### Support of a log table in a stored procedure

The parameters passed to a stored procedure can include the name of a log table that the stored procedure can fill with whatever information is appropriate. The name of the table is generated automatically by the rules engine and passed to the stored procedure, once the table has been successfully created. This table contains a single field named "MSG" that can be filled with any length of text (it is of type NTEXT).

Once the stored procedure has completed execution and has written messages in the log table, the Logic engine reads its content and merges it into the normal rules log file, then the table is automatically dropped.

To activate this feature you must include in the list of parameters passed to the stored procedure the keyword %LOGTABLE%. The rules engine replaces it with the appropriate table name.

Example:

```
*RUN_STORED_PROCEDURE=spEliminate(' %APP%', 'ACTUAL', '2001.JAN',
'%LOGTABLE%')
```

#### Support of blank parameters passed to a stored procedure

A stored procedure fails if one of the required parameters is blank (null string). This situation may occur, for example, if one of the parameters is a set of members that was left blank in a DTS prompt, indicating that ALL members be processed.

The rules engine automatically traps this situation by replacing any null parameter (") with the <NULL> keyword ('<NULL>'). The stored procedure has to check for parameters with a <NULL> value and take the appropriate action.

Example:

This instruction:

```
*RUN_STORED_PROCEDURE=spEliminate( ' ', '2001.JAN')
```

is converted into:

```
*RUN_STORED_PROCEDURE=spEliminate( ' <NULL>', '2001.JAN')
```

where <NULL> must be interpreted by the stored procedure as (for example) ALL categories.

#### Passing the selected region to a stored procedure using a table

The parameters passed to a stored procedure can now include the name of a temporary table that the logic engine will use to store the members of the selected region for which it was invoked. The name of the table is generated automatically by the rules engine and passed to the stored procedure, once the table has been successfully created and populated with the appropriate information. This table will contain two fields named DIMENSION and MEMBER, respectively, and will be populated by the rules engine with one record per dimension/member combination that has been passed to it by the calling program.

Once the stored procedure has completed execution, the table is automatically dropped by the rules engine.

To activate this feature, you must include the keyword %SCOPETABLE% in the list of parameters passed to the stored procedure. The logic engine will replace it with the appropriate table name.

Example:

```
*RUN_STORED_PROCEDURE=spEliminate( [%SCOPETABLE%])
```

\*RUNLOGIC - more information

Allowed uses: Global, MDX, SQL

Allows you to redirect the execution of a logic file towards a different data region than the one for which the logic was originally executed. Used in conjunction with \*APPSET, \*APP, \*LOGIC, and \*DIMENSION.

These instructions can be used to redefine:

- The application set

- The application
- The selection of dimension members
- The logic to run

Possible uses:

- To enter data in one period and trigger an allocation across all periods of the year
- To enter data in one entity and trigger the elimination logic for the elim-entities
- To modify an exchange rate in the rate application and trigger a translation in the main application

The instructions must be written between a \*RUNLOGIC and a \*ENDRUNLOGIC command. Here is the full list of supported instructions:

```
*RUNLOGIC
*APPSET= {AppSet} //optional
*APP = {App} //optional
*LOGIC = {logicname} //required
*DIMENSION {dimname} = {member set} //optional (one per dim allowed)
*ENDRUNLOGIC
```

All instructions are optional except \*LOGIC.

The following list is a summary of rules regarding the \*RUNLOGIC instruction:

- All logic properties that are not redefined with one of these instructions retain the values they have in the calling logic (for example, if the application is not redefined, the rules push is performed against the original application).
- The RUNLOGIC sections contained in a logic file is executed at the END of the entire rules execution, regardless of their position in the rules relative to all other statements.
- Multiple RUNLOGIC sections can be entered in the same rules file. They are executed in the order in which they are encountered.
- RUNLOGIC sections contained in rules called by a RUNLOGIC statement are ignored (In other words RUNLOGIC calls cannot be nested).
- The instruction DIMENSION can be used to redefine the scope of execution in a given dimension. Multiple DIMENSION instructions can be used to redefine the member sets of multiple dimensions in the current data region.

Examples:

```
//execute eliminations in elim entities
//-----
*RUNLOGIC
*DIMENSION ENTITY=filter([entity].members,[entity].property('ELIM')='Y')
*LOGIC=Intco
*ENDRUNLOGIC

//execute currency translation in main app
//-----
*RUNLOGIC
*APP=MAIN
*DIMENSION ENTITY2= //blank out scope of invalid dims
*LOGIC=DefaultTranslation
*ENDRUNLOGIC
```

Special Keywords in the DIMENSION instruction

The strings passed to the instruction DIMENSION can contain the keyword:

%{dimname}%

For example, if the main logic is being run for entity EUROPE, the following logic push is executed for all children of EUROPE:

```
*RUNLOGIC
*DIMENSION ENTITY = [ENTITY].[%ENTITY%].children
*LOGIC=SomeLogic
*ENDRUNLOGIC
```

Also, if the main rules was run for more than one entity (example: EUROPE and US), the logic push is performed for the children of each of them, using the following selection in the rules query:

```
{[ENTITY].[EUROPE].children, [ENTITY].[US].children}
```

**Note:** This works if the calling selection enumerates the members individually as follows:

```
DIMENSION:ENTITY
EUROPE, US
```

In case the calling selection contains an MDX statement, this is treated as one member. The following example would not work in the above context

```
DIMENSION:ENTITY
[WORLD].children
```

#### Members sub-names

The keyword %DIMNAME% can be adjusted to return only the prefix or the suffix of a member name.

For example, if the passed member set for the TIME dimension contains the member:

2001.JAN

The keywords return:

```
%TIME% 2001.JAN
%TIME_PREFIX% 2001
%TIME_SUFFIX% JAN
```

Here is an example that uses the prefix keyword to perform an allocation:

```
*RUNLOGIC
*DIMENSION TIME=descendants([time].[%TIME_PREFIX%.total],99,leaves)
*LOGIC=AllocationLogic
*ENDRUNLOGIC
```

If the user enters data for more than one month for the SAME year, the rules push is smart enough to process that year only once.

\*SCOPE\_BY - more information

Allowed uses: Global, MDX, SQL

\*SCOPE\_BY = {dimensions list}

By default, the data region is restricted to only the members found in an existing data file or values from an Excel sheet for all dimensions except the account dimension. This instruction allows you to expand the scope of the data region.

Using this instruction can be extremely useful to expand the scope of a logic execution, when data is entered through BPC for Excel or from an import file. If, for example, the formulas to execute should span across all products, regardless of what products have been modified, it is possible to expand the scope of the execution to all products by redefining the "SCOPE\_BY" with a list of dimensions that does not include the product. In this case the instruction could be:

```
*SCOPE_BY = CATEGORY, TIME, ENTITY
```

To be precise, the logic runs for all non-calculated products.

In many cases it may be more practical to modify the scope of the logic execution using the instruction \*XDIM\_MEMBERSET. This instruction allows the administrator to specify what members to process for a given dimension, in effect, overriding the default scope with greater precision.

\*SELECT - more information

Allowed uses: Global, MDX, SQL

\*SELECT ({variable}, {What}, {From}, {Where})

Allows you to retrieve a list of elements from a dimension and save it in a user-defined variable for later use anywhere else in the rules.

For example, with this instruction:

```
*SELECT(%REPORTING_CURRENCIES%, "ID", "CURRENCY", "[GROUP] = 'REP'")
```

In this case, you retrieve the ID of all members in the CURRENCY dimension where the property GROUP has the value REP. Running this example fills the variable %REPORTING\_CURRENCIES% with the list of reporting currencies defined in the current application.

#### Important information

1. The SELECT statement generates a SQL query, and NOT an MDX query. This implies that it can be executed against any SQL table existing in the application set database, and not just against the properties of a dimension in the application. The prefix "mbr" is automatically added to any name entered in the table parameter. Otherwise, the name of the table is taken as is. If you want to select a list of elements from a dimension using an MDX query, see the \*MEMBERSET statement.
2. In case any parameter contains embedded commas the entire parameter must be enclosed in an extra set of double quotes.
3. The \*SELECT statement is executed at the time the logic is validated, and the expanded result is written in the LGX file. This means that if the related dimension is modified, it may be necessary to re-validate the rules.
4. Statements returning no members do not necessarily cause the validation of the rules to fail. In this case a warning is entered in the validation log.
5. These instructions are not specific to a given logic section, but they can be written once anywhere in the rules and used across multiple commit sections. The following example works correctly.

```
//example -----
-----
*SELECT(%INCACC%, "[ID]", "ACCOUNT", "ACCTYPE='INC'")
*XDIM_MEMBERSET ACCOUNT=%INCACC%
[category].[#realistic]=[category].[actual] *1. 2
*COMMIT
*XDIM_MEMBERSET ACCOUNT=%INCACC%
[category].[#optimistic]=[category].[actual] *1. 3
//end of example -----
-----
```

Note also that, since the \*SELECT statement is expanded first, it can be placed anywhere in the logic. The above example would work even if the select statement was the LAST line in the logic, like this:

```
//example -----
-----
*XDIM_MEMBERSET ACCOUNT=%EXPACC%
[category].[#realistic]=[category].[actual] * 1.2
*COMMIT
```

```
*XDIM_MEMBERSET ACCOUNT=%EXPACC%
[category].[#optimistic]=[category].[actual] *1. 3
*SELECT(%EXPACC%, "[ID]", "ACCOUNT", "ACCTYPE='INC' ")
//end of example -----
-----
```

The special format **\*\*SELECT( )**

The instruction **SELECT** can also be invoked with a double leading asterisk. This triggers the execution of the instruction before any expansion is performed on the logic file (like function substitutions or file inclusions). This feature can be useful in case the result of the **SELECT** instruction is needed to control other tasks like **INCLUDE** or **RUNLOGIC**, which could be dependant on the result of the **SELECT** itself.

Example:

```
//-----
**SELECT(%MYFILE%, [filename], MyTable, "[CATEGORY]='%CATEGORY_SET%' AND
[ENTITY]='%ENTITY_SET%'")
*INCLUDE %MYFILE%.LGF
//-----
```

In the above example an included rules file is derived from an entry in a special table using the passed category and entity as key. Note that in this case the keywords **%CATEGORY\_SET%** or **%ENTITY\_SET%** must contain only one member each.

---

**Note:** A rule containing a **\*\*SELECT** instruction can only be executed in LGF format. Such rules might also refuse to validate if the result of the instruction is a function of the region passed at runtime, like in the above example.

---

**\*SELECTCASE / \*ENDSELECT** - more information

Allowed uses: By Commit, MDX

```
*SELECTCASE {expression}
  *CASE {value1}[,{value2},...]
    {formulas}
  [*CASE {value1}[,{value2},...]
    {formulas}
  [*CASEELSE]
    {formulas}
*ENDSELECT
```

where:

{expression} is the condition to be evaluated

{value1},... is the range of comma-delimited results that satisfy the condition for the current case

This structure replaces the need for multiple, nested **\*IIF** statements and helps make your rules code more readable.

**\*SKIP\_DIM** - more information

Allowed uses: By Commit, MDX, SQL

**\*SKIP\_DIM=** {dimension name}[,{dimension name},...]

Works in conjunction with the **\*DESTINATION\_APP** instruction to post results of rules calculations to a different application. See the **\*DESTINATION\_APP** topic for more information.

**\*STORE\_ORG** - more information

Allowed uses: ByCommit, SQL

Sometimes it happens that a section of SQL-based logic needs to read the value of a parent member, to perform some subsequent calculation. (For example it may need to store a profit and loss net profit into a balance sheet account). This instruction allows you to store in the writeback table all parent members of a given dimension as specified in a property like **PARENTH1**, **PARENTH2**, etc.

The syntax is:

```
*STORE_ORG {dimension name} = {property name}
```

Example:

```
*STORE_ORG ACCOUNT = PARENTH1
```

The STORE\_ORG instruction must be executed standalone in its own COMMIT section.

There is no particular drawback to storing records in the writeback table that OLAP never sees (parent members are recalculated on the fly in the OLAP cube), except that this may increase the size of the writeback table unnecessarily, especially if the selected organization has many intermediate parents.

While it may work, duplicating the values of OLAP parents in the database is not a recommended way to use this feature. A much more practiced approach is to use this instruction to store the values of "parent-like" members, defining some generic property such as MYPARENTH1 or MYSUBTOTALS, and using it to store members that for the application are NOT parents in any real hierarchy.

This also permits you to store only the appropriate parents, skipping all non-required intermediate parents that a true organization may include for other reasons. For example you could store just the value of Net Income in a dummy organization defined in property MYORG, skipping all other parents like Gross Profit, NPBT, etc. In this case, the logic statement is:

```
*STORE_ORG ACCOUNT = MYORG
```

\*SUB / \*ENDSUB - more information

Allowed uses: Global, MDX, SQL

```
*SUB {SubName} ({Param1} [, {Param2} ...])
```

```
    {body text}
```

```
    {body text}
```

```
    [...]
```

```
*ENDSUB
```

A SUB structure allows you to define reusable logic sections that can be invoked anywhere in the body of the rules to make the rules easier to read and maintain. When a SUB is then invoked somewhere else in the rules, its body lines are inserted in the rules with all the values passed to its parameters appropriately replaced.

SUBs behave similarly to included files to which any number of parameters can be passed. When the logic is validated, the invoked subs are inserted in the body of the logic as if they were included files invoked with an \*INCLUDE instruction. However, to invoke a SUB structure, no special keyword is required. A SUB is simply called inserting a line with the name of the SUB, followed by the values assigned to its parameter enclosed in brackets. The other important difference from included files is that a SUB does not need to be written in a file of its own, but can be written in any part of the logic, more similarly to a FUNCTION.

Example:

```
// Here the sub is defined
//-----
*SUB MYSUB(Param1,Param2,Param3,Param4)
[AccountDim].[#Param1]= [AccountDim].[Param2]+ [AccountDim].[Param3]
[AccountDim].[#Param4]= [AccountDim].[#Param1]*
[AccountDim].[Factor_Param4]
*ENDSUB

// Here the sub is used
//-----
MySub(A1,B1,C1,D1)
MySub(A2,B2,C2,D2)
MySub(A3,B3,C3,D3)
```

Similar to FUNCTIONS, SUBs are not position-sensitive, and can be defined anywhere in a logic, and, if desired, stored in separate library files that must then be merged with the logic using an INCLUDE



instruction. Also, SUBs can be invoked in any commit section of the logic without the need to be “re-defined” in each section.

SUBs without parameters are supported, but they must always be followed by brackets when invoked, like in this example:

```
// Here the sub is defined
//-----
*SUB SetTheAppropriateRegionToClear
*CLEAR_DESTINATION
*XDIM_MEMBERSET INTCO=NonIntco
*XDIM_MEMBERSET PRODUCT=NoProduct
*XDIM_CURRENCY=%REPORTING_CURRENCIES%
*ENDSUB
// Here the sub is used
//-----
SetTheAppropriateRegionToClear( )
```

\*SYSLIB - more information

Allowed uses: Global, MDX, SQL

\*SYSLIB {Includedfile} (Param1,Param2,...)

This instruction works like the \*INCLUDE instruction, however the file to be included is searched for in a reserved folder that is only intended to contain logic libraries provided by BPC that should not be modified. This folder is located below the Appset folder with the name "SystemLibrary\Logic Library".

Parameters

You can pass parameters to the included logic. The parameters are referenced as %P1%, %P2%, etc.

For example:

```
*SYSLIB sys_logic.LGF (REVENUE, COST)
```

The content of the file sys\_logic.LGF could be:

```
#GROSSPROFIT= [ACCOUNT].[%P1%] - [ACCOUNT].[%P2%]
#GROSSMARGIN= (([ACCOUNT].[%P1%] -
[ACCOUNT].[%P2%])/[ACCOUNT].[%P1%])*100
```

Time-shift instructions

To simplify the calculation of leads and lags in financial reporting applications, the following instructions are available for SQL-based logic:

- PRIOR
- NEXT
- BASE
- FIRST

The instructions PRIOR and NEXT support an optional numeric parameter. This parameter represents the number of time periods by which the current period must be shifted. If omitted, the function assumes a time shift of 1 period (forward or backwards). Negative values are accepted (A negative value for a NEXT function corresponds to a positive value for a PRIOR function and vice-versa).

Examples:

```
TIME=NEXT // In a monthly application this means next month
TIME=PRIOR(3) // Three periods backwards
TIME=NEXT(-3) // Same as PRIOR(3)
```

The keyword BASE always represents the last period of the prior fiscal year. When the fiscal year is a normal calendar year and the frequency is monthly, the base period of 2004.JUN is 2003.DEC.

The instruction BASE can be useful in YTD applications, where the opening balances need to be retrieved from the last period of the prior year.

The keyword FIRST always represents the first period of the current fiscal year. When the fiscal year is a typical calendar year and the frequency is monthly, the base period of 2004.JUN is 2004.JAN.

In case the time shift goes past the boundaries of the TIME dimension, these time shift functions return no period.

These functions can be used in four ways:

- 1: To redirect the destination period in a \*REC statement

Example 1: \*REC(TIME=NEXT)

Example 2: \*REC(TIME=BASE)

- 2: To retrieve a value from a different period in a \*REC statement

Example 1: \*REC(FACTOR=GET(TIME=PRIOR(3)))

Example 2: \*REC(FACTOR=GET(TIME=BASE))

- 3: To add periods to the selected data region in a XDIM\_MEMBERSET statement

Example: \*XDIM\_MEMBERSET TIME=PRIOR, %TIME\_SET%

In this example, if the first modified period is 2004.APR, the instruction PRIOR adds 2004.MAR to the region to process.

- 4: When the keywords PRIOR, FIRST or BASE are added to a XDIM\_MEMBERSET instruction, the time period PRIOR, FIRST or BASE can be also evaluated in a WHEN / ENDWHEN structure, as in the following example:

```
*WHEN TIME
*IS PRIOR
// ignore
*ELSE
*REC(...)
*ENDWHEN
```

In the presence of an XDIM\_MEMBERSET containing the PRIOR keyword, like in the above example, the WHEN structure recognizes 2004.MAR as PRIOR period.

\*USE - more information

Allowed uses: By Commit, MDX, SQL

\*USE {usedfile} (Param1,Param2,...)

The USE instruction behaves like the \*INCLUDE instruction, but only the members that are needed by the calling logic are included. This generates smaller, faster to execute LGX files.

The instruction \*USE must be called again after each \*COMMIT instruction. This rule does not apply to \*INCLUDED files.

Files included with the \*USE keyword take the same default path and extension of the included files.

All special instructions (like \*COMMIT and other instructions) are ignored, if found in a "USEd" file. We can say that when a rules file is "USEd" by another file, the only thing the calling file is interested in is the list of calculated members the "USEd" file contains. This restriction does not apply to "INCLUDED" files.

All temporary accounts contained in USEd files should always be named with a leading exclamation mark, even if this is not technically required. They could also begin with a pound sign (#), but, in this case, the rules module tries to write their resulting values in the application.

Example

```
'-----
```

```
*USE TranslationMembersFile
[account].[#cost] = -
([currency].[lc],[account].[grosssales])/[account].[!End]

[account].[#revenue] = -
([currency].[lc],[account].[cost])/[account].[!Avg_End]

'-----'
```

The file TranslationMembersFile may contain the definitions of many accounts, but only those for [account].[!End] and [account].[!Avg\_End] is included in the calling logic.

#### WHEN/ENDWHEN

This set of instructions triggers the use of SQL syntax rather than MDX syntax. This structure works in the same way as the SELECTCASE / ENDSELECT structure, but the instructions that it processes are not the usual MDX formulas of a modeling logic, but some \*REC( ) statements that generate new records.

\*WHEN / \*ENDWHEN - more information

Allowed uses: By Commit, SQL

This set of instructions triggers the use of SQL syntax rather than MDX syntax. This structure works in the same way as the SELECTCASE / ENDSELECT structure, but the instructions that it processes are not the usual MDX formulas of a modeling logic, but some \*REC( ) statements that generate new records.

The syntax is:

```
//-----
-----

*WHEN {criteria}
*IS {valid condition1}[,{valid condition2},...]
*REC([ [FACTOR|EXPRESSION={Expression}[, {dim1}={member},{dim2}=...]])
[ *REC([ [FACTOR|EXPRESSION={Expression}[, {dim1}={member},{dim2}=...]])]
...
[ *ELSE]
...
...
*ENDWHEN

//-----
-----
```

Where...	Is...
{criteria}	<p>What to test. Typically, this is a property of the current member of a dimension. The syntax is:</p> <pre>DimensionName.Property</pre> <p>Example:</p> <pre>*WHEN ACCOUNT.RATETYPE</pre> <p>{criteria} can also use the reserved keyword LOOKUP( ).</p>
{ValidCondition}	<p>One or more values that meet the criteria. They can be enclosed in double quotes if they need to be treated as strings. If they represent numeric values, the quotes should be omitted.</p> <p>Examples:</p> <pre>*IS "AVG", "END"</pre>

```
*IS 10,20,30
```

If no operator is specified, the \*IS clause assumes the presence of an equal sign (\*IS = "AVG", "END"). Other types of comparisons are also supported. The following examples represent valid conditions:

```
*IS > 2
```

```
*IS <= 7
```

```
*IS <> "ABC"
```

{ValidCondition} can also be a dimension property specified with the simple format:

```
*IS dimension.property
```

#### Notes:

- Multiple operators (like "<>" or ">=") must not be separated by any space. One or more blanks can be inserted between the operators and the value.
- If any operator is used, only one value can be passed. (This syntax is invalid: \*IS >2,3,4)
- Other operators like AND, OR and NOT are not currently supported.

The following are valid examples of using \*WHEN:

```
*WHEN ENTITY
*IS INTCO.ENTITY
...
*WHEN ACCOUNT.SCALE
*IS <>CURRENCY.SCALE
```

### WHEN criteria special case

A special case of \*WHEN criteria is:

```
*WHEN *
```

This criteria can be used when there is actually no criteria to test. In this case, the \*WHEN statement is only needed to trigger the SQL mode. The correct syntax is:

```
*WHEN *
    *IS *
        *REC(...)
*ENDWHEN
```

### Improved recognition of PRIOR(n) periods in \* IS statements

In logic like the following...:

```
*CALC_EACH_PERIOD
*XDIM_MEMBERSET TIME = PRIOR(3), %SET%
*WHEN TIME
*IS PRIOR
//.....
*ENDWHEN
```

...all 3 periods of data preceding the first period in %SET% will be recognized as PRIOR in the statement \*IS PRIOR

If, for example, the user modifies 2005.MAR and 2005.MAY, the three period 2004.DEC, 2005.JAN, and 2005.FEB will meet the criteria stated in the \*IS line.

This allows the logic to recognize periods that may have been loaded in memory only with the purpose of calculating some lags in the time dimension. A fair example of how to use this feature is represented by the following logic:

```
*XDIM_MEMBERSET ACCOUNT=REVENUE,PRICE,PAYMENTS
*CALC_EACH_PERIOD
*XDIM_MEMBERSET TIME=PRIOR(3),%SET%,%PREFIX%.DEC
*WHEN ACCOUNT
*IS REVENUE
    *WHEN GET(ACCOUNT="PRICE")
    *IS 0
        *REC(ACCOUNT=#PAYMENTS0,TIME=NEXT(0))
    *IS 1
        *REC(ACCOUNT=#PAYMENTS1,TIME=NEXT(1))
    *IS 2
        *REC(ACCOUNT=#PAYMENTS2,TIME=NEXT(2))
    *IS 3
        *REC(ACCOUNT=#PAYMENTS3,TIME=NEXT(3))
    *ENDWHEN
*ENDWHEN
*GO*WHEN TIME
    *IS<>PRIOR // prior here means any period before first in %SET%
    *WHEN ACCOUNT
    *IS #PAYMENTS0,#PAYMENTS1,#PAYMENTS2,#PAYMENTS3
        *REC(ACCOUNT=PAYMENTS)
    *ENDWHEN
*ENDWHEN
```

#### Nesting of \*WHEN/\*ENDWHEN

WHEN / ENDWHEN structures can be nested by as many levels as desired, and in any sequence. For example, the following structure could be a valid one:

```
*WHEN xxx
    *IS "A"
        *REC(...)
        *REC(...)
    *IS "B"
        *REC(...)
    *WHEN yyy
        *IS "C","D","E"
            *REC(...)
    *ELSE
        *REC(...)
```

```
*ENDWHEN
*ENDWHEN
```

Note: Indentation is used just for readability purposes, and is not required by the syntax.

The \*REC instruction

The \*REC( ) instruction tells the program what to do once a criteria has been met. Each REC instruction generates ONE new record to be posted to the database. Each source record can generate as many records as desired, even pointing to the same destination cell.

The parameters of the REC( ) function specify what to modify of the original record. Any dimension member can be modified with the syntax:

{DimensionName}={member}

```
Example: *REC(CURRENCY="USD", ENTITY="SALESITALY")
```

{member} must be enclosed between double quotes and can contain the name of any dimension, enclosed between the percent sign (For example: ENTITY="IC\_%ENTITY%"). In this case, the dimension name is replaced with the value of the current member for that dimension, and not with just the dimension name.

An alternative syntax allows the rules to retrieve the member name from the value of a property of any dimension. In the following example, the entity name is read from the "ENTITY" property of the current member of the INTCO dimension:

```
*REC(FACTOR=-1, ENTITY=INTCO.ENTITY)
```

The option NOADD for the \*REC( ) instruction

There are cases when multiple values found in a source region should generate a certain fixed value for a given destination record and such values should be the same, regardless of how many source records have been encountered. An example could be the need to assign a value of 1 to a "flag" account, if any account of the selected region has a value. This is done in a SQL logic by inserting the option NOADD anywhere in the REC statement.

Example:

```
*WHEN ACCOUNT.TYPE
*IS "AST"
*REC(EXPRESSION=1, NOADD, ACCOUNT=" FLAG_AST")
*ENDWHEN
```

## Using the LOOKUP value in dimension redirection

The REC() statement can use the value of a LOOKUP to define a destination member. For example this could be a valid syntax:

```
*REC(ACCOUNT = LOOKUP(LK1))
```

This generates a numeric account ID corresponding to the value retrieved by the LOOKUP.

This syntax also support a (limited) string concatenation functionality, like in this example:

```
*REC(ACCOUNT = ACC_ + LOOKUP(LK1))
```

If the value of the lookup is 20, the resulting destination account is ACC\_20.

The FACTOR and EXPRESSION instructions

The amount to assign to the new record can be derived from the original amount with either the instruction FACTOR or the instruction EXPRESSION.

The FACTOR instruction defines a factor by which the retrieved amount is to be multiplied.

```
*REC(FACTOR=1/1.55)
```

The EXPRESSION instruction defines any formula that results in the new value to post. The formula can include regular arithmetic operators, fixed values and the keyword %VALUE%, representing the original retrieved value.

```
*REC(EXPRESSION=%VALUE% + 1000)
```

Both FACTOR and EXPRESSION can also contain the reserved keyword LOOKUP( ), as described later.

The FLD( ) instruction inside a FACTOR or EXPRESSION

In a FACTOR or EXPRESSION it is also possible to specify a dimension property using the special format:

```
FLD(dimension.property)
```

Here is a valid example:

```
*WHEN ENTITY.SCALE
*IS <> " "
*REC(EXPRESSION=%VALUE%*FLD( ENTITY.SCALE ))
*ENDWHEN
```

This syntax is a slight departure from the usual format where the dimension property does not need to be enclosed inside a FLD( ) clause. This format has been adopted in order to (1) retain good performance in the logic execution, (2) simplify the logic validation and (3) allow for future extensions of logic functionality.

All the syntaxes that are supported by the FACTOR and EXPRESSION instructions can be combined inside a calculation expression, like in the following (meaningless) example:

```
*REC(EXPRESSION=%VALUE%*FLD( ENTITY.SCALE )+
GET( ACCOUNT=ACCOUNT.MYACC ) *5*LOOKUP( XYZ ) )
```

The GET( ) instruction

You can assign a FACTOR or an EXPRESSION to a source value in order to calculate a new value inside a \*REC( ) statement in SQL rules. The factor and the expression can include formulas using hard-coded values (factor=1.3) or values retrieved using the LOOKUP function (factor=lookup(avg)).

In addition to the above, you can use the keyword GET( ), which allows you to use values from some other record within the selected region.

The syntax is:

```
GET( {dimension}={member}[, {dimension}={member}]...)
```

Where...	Is...
{dimension}	A valid dimension name
{member}	A valid dimension member. This can be an explicit member name like "ABC" enclosed in double quotes, or it can be derived by reading the property of the current member of any dimension.

Valid examples include:

```
GET( ACCOUNT="ABC" )
GET( ACCOUNT=ACCOUNT.MYPROPERTY )
GET( ACCOUNT=ACCOUNT.ELIMACC, ENTITY=INTCO.ENTITY )
```

For example, assume that you want to calculate the account Revenue as follows:

```
#Revenue = Units * Price //this is the MDX format
```

The SQL rules formula is:

```
*WHEN ACCOUNT
    *IS "UNITS"
        *REC (ACCOUNT="REVENUE" FACTOR=GET (ACCOUNT="PRICE" )
*ENDWHEN
```

The get statements also supports the concatenation of properties with trailing fixed strings as follows:

```
GET(dimension=dimension.property + string )
```

Example:

```
GET (ACCOUNT=ACCOUNT.ID + .INP )
```

**Note:** While using the Get() instruction in SQL provides the ability to perform much more complex calculations than the system previously allowed in SQL rules, it has some limitations that might make it preferable to use MDX logic. Here is a further explanation of the limitations to using the SQL logic GET() instruction:

- The value to retrieve is not queried from the database, but it is searched for in the currently selected record set. (If the value is not found, it is assumed to have a value of zero). This implies that all values required by the logic must be included in the region to process. In the previously described example, the logic does not work correctly unless the account PRICE has been included in the region to scan, with something like:

```
*XDIM_MEMBERSET ACCOUNT = Units, Price
```

This is fundamentally different from the way any MDX rules works: MDX can automatically retrieve from the application all necessary information, even if not included in the queried region.



- The logic is not able to use the results of a previously calculated value. The following example does not work:

```
*WHEN ACCOUNT
    *IS "UNITS"
        *REC(ACCOUNT="REVENUE", FACTOR=GET(ACCOUNT="PRICE"))
    *IS "REVENUE"
        *REC(ACCOUNT="TAXES", FACTOR=.5)
*ENDWHEN
```

...Unless the two calculations are separated by a commit statement:

```
*WHEN ACCOUNT
    *IS "UNITS"
        *REC(ACCOUNT="REVENUE", FACTOR=GET(ACCOUNT="PRICE"))
    *ENDWHEN
*COMMIT
*WHEN ACCOUNT
    *IS "REVENUE"
        *REC(ACCOUNT="TAXES", FACTOR=.5)
*ENDWHEN
```

\*TEST\_WHEN - more information

Allowed uses: By Go/Commit, SQL

When you have conditions in your logic that are not dependant on the records being scanned, you might only want to test those conditions once. For example, you might want to test that there is a specific keyword or that a certain cell of the application has a specific value in the selected set of members for a given dimension. Rather than applying the condition to the entire set of records to scan, the condition can be evaluated before the WHEN / ENDWHEN loop, with the following instruction:

```
*TEST_WHEN( {condition} )
```

The {condition} is a string that the logic engine passes to a Visual Basic script for evaluation. If the returned value is TRUE, the subsequent WHEN / ENDWHEN loop is processed. Otherwise the entire loop is skipped.

Example:

```
// skip the loop if Budget is not among the passed categories
*TEST_WHEN(instr("%CATEGORY_SET%", "BUDGET")>0)
*WHEN *
//...
*ENDWHEN
*GO

// skip the loop if account FLAG is zero for a certain time,intco
combination
*TEST_WHEN(GET(ACCOUNT="FLAG",INTCO="NON_INTERCO",TIME="2004.JAN")<>0)
*WHEN *
//...

```

#### \*ENDWHEN

As shown in the above example, the instruction supports the use of the `%{dim}_SET%` keyword. It also supports the GET instruction, to retrieve a value from the recordset. When one or more GET instructions are used in the evaluation of the condition, you must remember to specify all required dimensions. The non-specified dimensions default to the values they have in the first record of the source record set. In other words, you can only omit dimensions that do not vary in the recordset being scanned.

Another syntax supported anywhere in this instruction is the `FLD()` keyword. Here is a valid example:

```
*CALC_EACH_PERIOD // handle periods one by one
*XDIM_MEMBERSET TIME=PRIOR, %TIME_SET% // include prior period to
selected dates
// This tests that the evaluated period is not the prior period
*TEST_WHEN(instr("%TIME_SET%", "FLD(TIME.ID)")>0)
*WHEN...
```

The TEST\_WHEN instruction is specific to the current GO section. If the script does not contain a GO section, it is specific to the current COMMIT section.

#### POS() keyword

The POS() keyword can be used in a WHEN / ENDWHEN structure when you need to compare the position of the current time period relative to a different one. This situation may arise when a logic must change behavior when passing beyond a certain date. For example, the first three months of a year may contain actual data, where some accounts are input, and the other months contain budget data, where the same accounts are calculated.

The new keyword comes in two formats:

```
POS(TIME) // to use in a *WHEN instruction
```

And

```
POS({time}) // to use in a *IS instruction
```

The correct structure is:

```
*WHEN POS(TIME)
*IS <=> POS({time})
//...
```

...where {time} must be an explicit date. For example:

```
*WHEN POS(TIME)
*IS <=> POS("2005.MAR")
//.....
```

To make the date more dynamic, you can use the \*FLAG\_PERIOD instruction. See \*FLAG\_PERIOD.

#### \*WRITE\_TO\_FAC2 - more information

Allowed uses: By Commit, MDX, SQL

#### \*WRITE\_TO\_FAC2

This instruction triggers the posting of the results directly to the short-term data storage table (tblFAC2) of the AppSet.

If this instruction is found in a given COMMIT section, the records to be posted end up directly in the FAC2 table rather than real-time data storage (the write back table).

This mechanism can be helpful when a rules execution may generate a significant amount of new records, leading to a very fast increase in the content of the write back table, which in turn would cause a significant degradation of performance in all I/O activities.

Note: At the end of the rules execution a processing of the FAC2 partition is required, in order to make the results of the logic available to the application. This can be done adding an appropriate task at the end of the DTS package that executed this type of rule. Alternatively the \*PROCESS\_FAC2 instruction can be used.

\*WRITE\_TO\_FILE - more information

Allowed uses: Global, MDX, SQL

This instruction writes a copy of all records to be posted into the specified file in a textual, comma-delimited format.

This setting automatically suppresses the writing of the records to the log file, and the results are not written to the application, even if the simulation mode is off.

If no path is included in the file name, the Data Manager data files path is used. If no extension is included, a TXT extension is appended to the file name.

Note: This instruction is only supported in "overwrite formula" mode (\*logic\_mode=1), and can only be entered in the "add formula" text box of the DTS logic task (k2DTSRunlogic). If written in a logic file directly, it is ignored.

\*XDIM\_ADDMEMBERSET - more information

Allowed uses: By Commit, MDX, SQL

```
*XDIM_ADDMEMBERSET {dimension} = {members set}
```

The rules can merge a specific set of members with the members passed in the region for which the rules should be executed using this instruction.

This instruction is similar to the instruction \*XDIM\_MEMBERSET. The difference is that, while XDIM\_MEMBERSET redefines the region passed by you, XDIM\_ADDMEMBERSET adds the defined set to the passed region.

For example, if a user enters a value in entity SalesItaly, and the default rule says:

```
*XDIM_ADDMEMBERSET ENTITY=SalesFrance
```

...the logic is executed for SalesItaly AND SalesFrance.

On the other hand, if the rule simply said:

```
*XDIM_MEMBERSET ENTITY=SalesFrance
```

...the rule would have been executed for SalesFrance only, regardless of the entity where the data had been entered.

\*XDIM\_DEFAULT - more information

Allowed uses: Global, MDX

```
*XDIM_DEFAULT {Dimension name} = {Members Set}
```

You can redefine the default member set of a dimension using this command. Typically, when no member set is passed to a dimension, the rules module automatically queries all non-calculated members of that dimension (with the exception of the currency dimension, if existing, which defaults to the LC member).

The difference with XDIM\_MEMBERSET is that XDIM\_DEFAULT is only used if no selection is passed for the specified dimension, while XDIM\_MEMBERSET is used in any case. In practical terms, this only applies to logics called by a TDS package, because BPC for Excel always passes the selection for all dimensions (except the accounts).

\*XDIM\_FILTER - more information

Allowed uses: By Commit, MDX, SQL

```
*XDIM_FILTER {Dimension name} = {Members Set}
```

```
*XDIM_FILTER {Dimension name} < or > {Time member}
```

The member set used for a given dimension can be filtered using a user-defined criteria with this instruction. This instruction does not replace the passed set with a hard-coded set, but filters the passed set with a predefined criteria.

For example, if the members passed for the ACCOUNT dimension are Cash, Receivables and Payables, this instruction:

```
*XDIM_FILTER ACCOUNT = [account].properties("ACCTYPE")="AST"
```

...accepts only asset accounts, so that the resulting account set is limited to Cash and Receivables.

If no member is passed, the filter criteria will not apply to only the non-calculated members (the default members set for all dimensions except currency), but to all members in the dimension. This could also be used as a way to modify the default filtering criteria.

The instruction automatically removes duplicates from the filtered set. This could be helpful when a returned member set contains duplicates, a situation that can easily be encountered in the entity dimension where the double hierarchy returns the same entity twice.

A typical use of this feature is to filter a list of members against one or more properties. When the members must be filtered against some values in the application (for example, only the entities that have a value <: > 0 in a given account), the instruction to use is \*XDIM\_GETMEMBERSET.

The \*XDIM\_FILTER {Dimension name} < or > {Time member} syntax also supports the %PREFIX% and %FLAG\_PERIOD% keywords like in the following example:

```
*PROCESS_EACH_MEMBER=CATEGORY
*XDIM_FILTER CATEGORY=[CATEGORY].PROPERTIES("CALCULATE")="Y"
*FLAG_PERIOD=CATEGORY.FIRSTPERIOD
*XDIM_FILTER TIME = > %PREFIX%.%FLAG_PERIOD%
```

In such example, the selected categories are processed one by one. For each of them the property "CALCULATE" is evaluated and only those that require calculation are processed by the logic.

Furthermore, the logic reads the property FIRSTPERIOD of the processed category, to derive the starting month from which the periods must be processed. For example, if the FIRSTPERIOD is "APR", and the selected periods belong to year 2004, all periods preceding 2004.APR are ignored.

\*XDIM\_GETINPUTSET - more information

Allowed uses: By Commit, MDX, SQL

```
*XDIM_GETINPUTSET {dimension} [= {member set}]
[*APP={application}] //optional
[*XDIM_MEMBERSET {dimension} [= {member set}] //as many of these as needed
[*CRITERIA {expression}] //optional
*ENDXDIM
```

This instruction filters the members of the selected region for a given dimension according to their compliance with some user-defined criteria that must be met by the values in the database.

XDIM\_GETINPUTSET serves the same purpose of XDIM\_GETMEMBERSET, but it generates an SQL query instead of an MDX query behind the scenes. This implies that, while it can only be used to check for the value of input cells (i.e. having entries in the fact table), it will, in most cases, deliver much better performance and scalability.

Example:

```
*XDIM_GETINPUTSET ENTITY
*APP OWNERSHIP
  *XDIM_MEMBERSET ACCOUNTOWN=METHOD
*XDIM_MEMBERSET CURRENCYPARENT=C_GR_FIN
  *XDIM_MEMBERSET INTCO=TPNONE
*CRITERIA SIGNEDDATA>71
```

**\*ENDXDIM**

This filters all entities that in application OWNERSHIP have a value greater than 71 in account METHOD, for currencyparent=C\_GR\_FIN, etc.

For all the dimensions not specified in the instruction, the search is performed in the corresponding members of the selected region. For example, the category and period are those for which the logic was being executed.

Warning: the criteria instruction can be used to filter the members against an acceptable value range. However, currently the instruction is not able to take into account the stored sign of the selected account, and this has to be figured out manually. For example, if the instruction is supposed to filter a REVENUE account with a value greater than 100, the instruction should say:

```
*CRITERIA SIGNEDDATA<-100 // this means >100 for income accounts
```

...because REVENUE is stored with a negative sign.

Similarly to XDIM\_GETMEMBERSET, the XDIM\_GETINPUTSET instruction is specific to the COMMIT section it is written in.

Special case: Change of dimension name across applications

The instruction \*XDIM\_GETINPUTSET can redefine the name of the dimension being filtered, if, when querying a different application, such dimension has a different name than in the source application.

The syntax supports an optional "AS" statement as follows:

```
*XDIM_ GETINPUTSET {ThatDimension} [ AS {ThisDimension} ] [= {member set} ]
```

Example:

```
*XDIM_GETINPUTSET SOMEENTITY as ENTITY=[SOMEENTITY].members
*APP SOMEOWN
  *XDIM_MEMBERSET ACCOUNTOWN=METHOD
  *XDIM_MEMBERSET RPTCURRENCY=GROUP1
  *XDIM_MEMBERSET INTCO=Non_Interco
*CRITERIA SIGNEDDATA<>0
*ENDXDIM
```

In the above example the members of the ENTITY dimension are extracted from the members of the SOMEENTITY dimension existing in the SOMEOWN application.

\*XDIM\_GETMEMBERSET - more information

Allowed uses: By Commit, MDX, SQL

```
*XDIM_GETMEMBERSET {dimension} [= {member set} ]
[ *APP={application} ] //optional
[ *XDIM_MEMBERSET {dimension} [= {member set} ] //as many of these as needed
[ *QUERY_TYPE= 0 | 1 | 2 ] //optional
*CRITERIA {expression} //required
*ENDXDIM
```

This instruction filters the members of the selected region for a given dimension according to their compliance with some user-defined criteria that must be met by the values in the application.

Example:

```
*XDIM_GETMEMBERSET ENTITY=[ENTITY].[PARENT1].CHILDREN
*APP=OWNERSHIP
*XDIM_MEMBERSET INTCO=I_NONE
```

```
*CRITERIA [ACCOUNTOWN].[METHOD]>1
*ENDXDIM
```

This filters all children of entity PARENT1 to only those that in application OWNERSHIP have a value greater than 1 in account METHOD, intercompany I\_NONE.

For all the dimensions not specified in the instruction, the search is performed in the corresponding members of the selected region. For example, the category and period are those for which the logic was being executed.

This instruction can be very useful to restrict the scope of a logic execution where you have selected a large region of data to process. For example, a currency conversion selected for all entities could be automatically limited to only those entities that have a value in a specified account.

Another use of this feature could be to break a complex modeling logic into independent sections separated by multiple COMMIT instructions, and to only execute those for which some specific accounts have changed.

Another possibility is to trigger a different type of logic based on the value of an account. For example a FORECAST logic could only be executed for just those months where [account].[One\_If\_Forecast]=1.

Special case: Change of dimension name across applications

The instruction \*XDIM\_GETMEMBERSET can redefine the name of the dimension being filtered, if, when querying a different application, such dimension has a different name than in the source application.

The syntax supports an optional "AS" statement as follows:

```
*XDIM_ GETMEMBERSET {ThatDimension} [ AS {ThisDimension}] [= {member set}]
```

Example:

```
*XDIM_GETMEMBERSET SOMEENTITY as ENTITY=[SOMEENTITY].members
*APP OWNERSHIP
*XDIM_MEMBERSET RPTCURRENCY=GROUP1
  *XDIM_MEMBERSET INTCO=Non_Interco
*CRITERIA [ACCOUNTOWN].[METHOD]<>0
*ENDXDIM
```

In the above example the members of the ENTITY dimension are extracted from the members of the SOMEENTITY dimension existing in the SOMEOWN application.

\*XDIM\_NOSCAN

The instructions \*XDIM\_NOSCAN and \*NOSCAN in \*CALC\_DUMMY\_ORG allows you to load in memory information that is only needed in a \*GET( ) statement and never used in a \*IS statement. For example, in a Units \* Price calculation the logic might read as follows:

```
*XDIM_MEMBERSET ACCOUNT=UNITS, PRICE
*WHEN ACCOUNT
*IS UNITS
*REC(FACTOR=GET(ACCOUNT="PRICE"),ACCOUNT="REVENUE")
*ENDWHEN
```

In such situations, users can now instruct the logic to ignore the PRICE by simply saying:

```
*XDIM_MEMBERSET ACCOUNT=UNITS, PRICE
*XDIM_NOSCAN ACCOUNT=PRICE
*WHEN *
*IS *
*REC(FACTOR=GET(ACCOUNT="PRICE"),ACCOUNT="REVENUE")
*ENDWHEN
```

This makes the scanning of the record set loaded in memory somewhat faster, as all those with account PRICE will be skipped very efficiently.

Similarly, a \*NOSCAN instruction can be added to a \*CALC\_DUMMY\_ORG structure. See \*CALC\_ORG.

\*XDIM\_MAXMEMBERS - more information

Allowed uses: By Commit, MDX, SQL

```
*XDIM_MAXMEMBERS {dimension}= {max number of members}
```

This instruction breaks the query into multiple queries if the designated maximum number of dimension members is reached. Using this instruction can help performance if there are too many members in the scope of the query.

In most cases, the fastest results are obtained running the logic as one single query. However, if the number of members in a dimension is too big, the performance can deteriorate significantly. In this case it may be preferable to break the execution in multiple queries. This can be accomplished using the \*XDIM\_MAXMEMBERS instruction in the logic.

```
Example: *XDIM_MAXMEMBERS Entity = 50
```

This instruction breaks the query into multiple queries of no more than 50 entities each, in case the entities to process exceed the limit of 50 members.

The selected dimension can be any one, including a dimension explicitly mentioned in the passed region.

The default for this option is, in MDX rules, one entity per query. In other words, if you do not specify any value for this option, the rules module automatically generates one separate MDX query for each entity being processed.

To reset the default value of maxmembers to ANY number of entities in MDX logics, you can set the instruction to zero, as follows:

```
*XDIM_MAXMEMBERS Entity = 0 // unlimited number of entities
```

Important: the maximum number of members can be specified for up to TWO dimensions, like in these examples:

```
*XDIM_MAXMEMBERS Entity = 50
```

```
*XDIM_MAXMEMBERS Time = 1
```

Or..

```
*XDIM_MAXMEMBERS Entity = 50, PRODUCT=100
```

\*XDIM\_MEMBER - more information

Allowed uses: By Commit (Global in the Formula script of a DTS task), MDX, SQL

```
*XDIM_MEMBER {dimension}={member} [TO {member}]
```

This instruction is similar to the \*XDIM\_MEMBERSSET instruction, but, while it only supports ONE member to be passed for the specified dimension, it permits you to specify a different destination member into which the results of the rules execution must be written.

Example:

```
*XDIM_MEMBER CATEGORY=ACTUAL TO BUDGET
```

The above statement forces the logic to be executed reading the desired values from the ACTUAL category, but writes the results into the BUDGET category.

Multiple DIM\_MEMBER instructions can be entered in the same rule, like in this example:

```
*DIM_MEMBER DATASRC=INPUT TO ELIM
```

```
*DIM_MEMBER PARENTDIM=NONE TO GROUP1
```

This feature can be used to simplify some rules expressions. For example this rule...

```
#ACC1 = ([ACCOUNT].[X],[CATEGORY].[BUDGET])
```

```
#ACC2 = ([ACCOUNT].[Y],[CATEGORY].[BUDGET])
```

```
#ACC3 = ([ACCOUNT].[Z],[CATEGORY].[BUDGET])
```

...could become:

```
*DIM_MEMBER CATEGORY=ACTUAL TO BUDGET
#ACC1 = [ACCOUNT].[X]
#ACC2 = [ACCOUNT].[Y]
#ACC3 = [ACCOUNT].[Z]
```

This instruction can also be useful when the QUERY\_TYPE=2 is used. In such case, if multiple dimensions must be nested on rows, a NonEmptyCrossJoin query could result, and it may be important to run the query directly from the data region containing the source values, in order not to miss some of them in the calculation (see the MDX language documentation for an explanation of the NonEmptyCrossJoin function).

\*XDIM\_MEMBERSET - more information

Allowed uses: By Commit (Global in the Formula script of a DTS task), MDX, SQL

\*XDIM\_MEMBERSET {Dimension name} = {Members Set}

While the rules module automatically builds the set of members that must be included in the logic query for each dimension, this set can be also controlled by the rules itself using the \*XDIM\_MEMBERSET instruction.

For example, in the exception translation rule we can enforce the query to generate results for all reporting currencies with the instruction:

```
*XDIM_MEMBERSET CURRENCY=USD,EURO
```

The instruction \* XDIM\_MEMBERSET supports also the “not equal to” operator with the syntax:

```
* XDIM_MEMBERSET {Dimension}<>{MemberSet}
```

This operator is only supported for SQL logic (in MDX rules this feature is not needed, as the MDX syntax allows to build any sort of sets), and can be handy to pass to the SQL query smaller lists of valid members, that are more efficiently parsed by Microsoft SQL engine.

Example:

```
*XDIM_MEMBERSET INTCO<>NonInterco
```

This corresponds to passing the list of all intercompany members, excluding the NonInterco member.

Forcing a dimension to read all members

Using the All keyword, you can force a dimension to read all members. Previously, when you wanted to make sure that all members of a given dimension would be loaded, irrespective of what specified in the passed region, you would have written something like:

```
// a workaround
*XDIM_MEMBERSET INTCO<>INVALID
```

Today the rules engine supports a cleaner definition, through the keyword <ALL>. The above example can be written as follows:

```
// a better syntax
*XDIM_MEMBERSET INTCO = <ALL>
```

This improves the readability of the logic and also generates faster SQL queries.

Added recognition of NEXT(n) keyword

The instruction XDIM\_MEMBERSET, when applied to the TIME dimension, can now also handle the keyword NEXT(n), which allows you to extend in the future the set of passed periods to process. For example:

```
*XDIM_MEMBERSET TIME=PRIOR,%SET%,NEXT
// add 1 period before and 1 period after

*XDIM_MEMBERSET TIME= %SET%,NEXT(3)
// add 3 periods after
```



\*XDIM\_REQUIRED - more information

Allowed uses: Global, MDX, SQL

\*XDIM\_REQUIRED={dimname}[,{dimname}]

Sets dimensions that are required in order for the rules to run from a Data Manager package. If the specified dimension(s) are not passed to the rules from a package, then an error message is generated.

Example:

```
*XDIM_REQUIRED=CATEGORY, TIME
```

If no selection is passed for any of the required dimensions, the rule fails and displays an error message.

## Managing custom menus

Custom menus allow you to choose from an array of pre-defined business-process-oriented reports.

### Adding custom menus

After creating custom menus in BPC for Excel, you can define security for them and add them to the custom menu list in the BPC for Excel action pane. Once they are added to the list, they are available to the appropriate users for selection in the action pane upon opening BPC for Excel.

There are two types of custom menus: a default custom menu and a team custom menu. Users will see one type or the other, depending on their team affiliation and the custom menu setup. The default custom menu is defined on the EV\_DEFAULT tab on the custom menu workbook. If users are not part of a team, they will only be able to access the default custom menu.

You can define one or more team custom menus that are appropriate to specific BPC teams. Users assigned to a team will see those custom menus, and not the default custom menu.

The following rules apply when the system determines which custom menus to display in the action pane:

- If a user is not assigned to a specific team, they see the default custom menu.
- If a user is assigned to a team custom menu, they see the team's custom menu only. They do not see the default menu.
- If a user is assigned to multiple teams, the user has the option of selecting any one of the available team custom menus.

The following table describes the information you enter in the process.xls workbook. Enter one row of information for each custom menu that you want to make available for selection.

Column name	Description	Example
Team	One or more team names that have access to the custom menu specified in the Name field.	<ul style="list-style-type: none"> <li>• Admin, UserTeam</li> <li>• CORPUSERS</li> </ul>
ProcessName	A name used to identify the name of the process. For example, you can enter Budget if you are adding a custom menu representing the budgeting process.	<ul style="list-style-type: none"> <li>• Budget</li> <li>• Actual</li> <li>• Administrative</li> </ul>
Default Process	Users may have rights to see multiple custom menus. This field allows you to specify which custom menu is selected by default in the action pane. For example, if you specify Yes for a custom menu called ADMINTASKS, then ADMINTASKS will be selected upon logon.  Only specify Yes for one row per process.xls workbook.	<ul style="list-style-type: none"> <li>• Yes</li> <li>• (blank)</li> </ul>
Process Description	A description for the custom menu. The description is displayed when the custom menu is selected in the action pane and at the top of the custom menu.	<ul style="list-style-type: none"> <li>• Current Month Results</li> <li>• Administrative tasks</li> </ul>
CV	The default current view for the given custom menu. The format is: <dimension>=<member>, <dimension>=<member>, ...	<ul style="list-style-type: none"> <li>• Category=Actual,Time=2005.Apr</li> <li>• Category=Budget,Time=2006.Jan</li> </ul>
Application	The application in which the custom menu resides.  If you want users to access more than one application using the same template, you can enter multiple applications separated by commas. For example, if you want a single	<ul style="list-style-type: none"> <li>• Finance</li> <li>• HR</li> </ul>

Column name	Description	Example
	<p>template to use applications <i>Finance</i> and <i>Finance2</i>, enter Finance, Finance2. This allows you to define separate current views for each application. You use the <i>CVOOverride</i> field in the custom menu to define the different current views.</p> <p>If using more than one application in a custom menu template, you must also set up the file structure under each application folder to be identical on the server.</p>	
MenuFile	<p>The location of the custom menu. The root folder for custom menus is the custom menu folder in the Wizard directory in the application's WebExcel folder.</p> <p>On a client machine, the directory is located in the user's My Documents\BPC\ApplInfo\&lt;AppSet&gt;\&lt;application&gt;\eExcel\Reports\Wizard\custom menu Reports folder.</p>	<ul style="list-style-type: none"> <li>ProcessMenu\ProcessFunctions.xlt</li> <li>ProcessMenu\AdministratorTasks.xlt</li> </ul>
Message	<p>A short message to users about the custom menu. This message can be changed, as needed.</p> <p>For example: "Comments are due by..." The message is displayed in the action pane.</p>	<ul style="list-style-type: none"> <li>Due the 3rd business day of each month</li> </ul>

#### Assigning users to a team custom menu

This procedure describes how to assign users to a team custom menu.

#### To assign users to a team custom menu

1. Define teams. See Adding teams.
2. When creating the custom menu, name the worksheet using the same team ID as the team ID used in users.xls. For example, if the team ID is CORPUSERS, the custom menu's worksheet name would be CORPUSERS. (The worksheet name is case-sensitive, and must be identical to what was defined on users.xls.)
3. When adding a custom menu to process.xls (defined in the procedure below), associate the user team to a custom menu. Complete the rest of the fields, as described in the procedure below for Adding a custom menu in the action pane.

#### Adding a custom menu to the action pane

This procedure describes how to add custom menus to the action pane.

#### To add a custom menu

1. From the Admin Console, select Custom menus.
2. Define a row for each custom menu you want to add. Use the table above for descriptions.
3. Test the new information by selecting Validate Custom Menu from the action pane. If there are any errors, fix the errors by editing the Process.xls file. Perform this step until there are no errors.

## Managing Journals

You can use Journals to track changes and adjustments made to the database by many users, usually for the end of month or end of quarter process.

For example, you load general ledger information into your application using Data Manager. Afterwards, a number of users look at the data and make adjustments. Journals keeps track of who made which changes. It also allows you to generate reports on the changes by amount, date, user, and other properties.

The process companies follow at end of month generally has these main parts:

- 1) Data Manager Import
- 2) WebExcel data sends
- 3) Journal changes to data
- 4) Logic-based consolidation

Journals allow a company to capture an audit trail of the changes made to the database during the review and analysis step. You can formalize this process by setting up a special work status that locks changes from BPC for Excel data sends and only allows changes made by Journals. See Locking data for Journal input only.

### Setting up journals

Journals are available by default. All you need to do to use the Journals feature, is to set up a journal template for the application. After a template is created, users can create Journal Entries by filling out required information in the template, and then posting changes to the database. You create one journal template per application.

### Journal advanced formulas

You can set up advanced formulas that are specific to Journal postings, or you can run the default advanced formulas when journals get posted. See the \*Logic functions under Rules Keyword Reference.

### Glossary of journals terms

The following list of definitions of Journal terms is provided to help explain the concepts presented in this section.

Term	Description
Audit trail	The audit trail is a record of posted journal entries. You can create a Journal report that displays the audit trail.
Journal template	<p>A journal template is designed by the Administrator. Administrators can create a different template for each application in an AppSet. A Journal Template is used as a Journal Entry Form by a Journal User. Journal users fill out the form, then either save or post their journal entries, depending on their level of access.</p> <p>See Creating journal templates</p>
Journal entry	<p>Journal users create and post Journal entries by filling out the Journal Entry form. This form is based on the journal template created by an administrator.</p> <p>Users can save partially completed Journals. Incomplete Journal Entries must have at least one field in a detail line so they can be saved.</p>
Journal security	<p>You set up Journal security to enforce your Journal creation and posting policy.</p> <p>See Setting up Journal Security</p>
Detail dimension	Dimension that you set line item information for in the Journal Entry.
Header dimension	Dimension used as a "page key" for the Journal entry. No line item detail.

## Creating journal templates

You set up a journal template for each application. A journal template is an input form where users enter journal entries.

The template consists of input fields for Header dimensions, additional header items, and detail dimensions:

Field	Description
Header Dimensions	Fixed (constant) dimensions for all journals in an application.
Additional Header Items	Text fields or lists that are saved along with the Journal entry that contain clarifying information. Additional Header Items are not required. But if they are present, users must give them values before posting a Journal entry.  The maximum length of a Additional Header Item entry is 128 bytes.
Detail Dimensions	The dimensions that contain the data you need to change. They are set up in columns so that each row under the dimension name becomes a detail line. You fill in members for each of the dimensions and the debit or credit amount for that detail line.  The maximum number of detail lines is determined when you use the Journal assistant. See the procedure below for more information.  <div>Note: If you already created a Journal Template, creating a new template that changes the structure of the journal entries deletes the old template and all journal entries associated with that template. This removes your audit trail, even though changes made to the application data through posted Journal entries are maintained. If you recreate the Journal template, but do not change the structure of the template (keep all header and detail dimensions the same), then you have the option to keep the existing journal entries.</div>

For more information on the journal template, see BPC for Excel Help.

### To create a journal template

1. Log on the appropriate application set, and start the Admin console.
2. Select Application, and then select the application for which you want to create the journal template.
3. Select Journals, and from the action pane, select Journal wizard.
4. In the Select Header Dimensions step, select the dimensions you want in the header of the Journal template. The header dimensions become "page keys" for the journal entry. The following dimensions are required to be placed in the header: Category, Time, and RptCurrency (if defined in the application). Click the Next button to continue.
5. In the Set Header Dimension Order step, you can change the top-to-bottom order of the header dimensions. When you are finished, click the Next button.
6. The remaining dimensions in your application that have not been placed in the header become detail dimensions. These are the dimensions that you will add line items to for a journal entry. In the Set detail dimensions step, you can change the left-to-right order of the detail dimensions and you can set the maximum number of detail lines. When you are finished, click the Next button to continue.
7. On the Create additional header items step, you can create extra header text fields. You can also change the text fields to selection boxes by defining sub-items for one of the additional header items. To add items:
  - a. Click the left Add button, then type the name of the additional header item and its maximum length, then click Add. The dialog box remains on the screen so you can enter more items. The maximum length of an Additional Header Item is 128 bytes. Continue to add items, and then click OK.

- b. To change one of the items to a selection list, Click the item to select it, and then click the sub-item Add button. Type the name of the sub-item, and then click Add. Continue to add items, and then click OK.

---

Note: The word "Remark" is a restricted keyword and cannot be used as a header item name.

---

8. When you are finished creating additional header items, click the Next button.
9. Review your new journal template information on the Summary screen, and then click the Finish button to create the Journal Template.

---

Note: See the BPC for Excel documentation for information on using journal templates to create journal entries.

---

## Setting up journal security

You set up journal security to enforce your Journal creation and posting policy. Journal security is a component of task security, and uses application member access rights. For example, if you give users create access, but do not give them write access to a data region, those users will not be able to create journal entries for that data region.

Journal security involves defining users that can administer (AdminJournal), create (CreateJournal), review (ReviewJournal), post (PostJournal), or unpost (UnpostJournal) entries. Given 'AdminJournal' rights, the user can create templates and clear journal tables.

---

Note: This procedure describes how to set journal security only, but you can also combine journal security tasks with other tasks. For example, if you give one or more users or teams the default 'PrimaryAdmin' task profile rights, they can perform all journal tasks, as well as the others granted by that administrator-type.

---

### To set up journal security

1. From the Admin console, select Security, then Task Profiles.
2. Select Add new task profile, and then set up a Task Profile called "Journals," and give it a description. Click Next. (By not selecting any check boxes, you are creating a custom task profile that can apply to journals only.)
3. From the View tasks by interface drop-down, select Journals.
4. Select one or more of the journal tasks, then click Next.
5. Select one or more teams or users to assign the selected rights, then click Next.
6. Click Apply to save and process the task profile.
7. Click OK at the confirmation prompt.

## Clearing journal tables

Users with 'AdminJournal' rights can clear journal entries from an application database. This includes all journals and journal security setup. You might use this option to clear journals before you move from a development environment to a production environment.

Take caution when doing this, as it removes journals from an application. Any posted changes to the database will remain, but you will lose your audit trail.

### To clear journal tables

1. Log on the appropriate application set, and start the Admin console.
2. Select Application, and then select the application for which you want to clear journal information.
3. Select Journals, and from the action pane, select Clear journal table.
4. Select Yes at the confirmation prompt.

## Limiting journal dimension member lists

You can limit the number of dimension members that are available to users during journal entry. Normally, all base-level members that the user has access to are available when the user fills out journal template. By using the EnableJRN property for each dimension for which you want to limit members, you can control the available members when users double-click member cells in the journal template.

For example, you might want only some account members used for journal entries, regardless of whether the users have access to other members.

To limit journal dimension member lists

1. Add the property EnableJRN to the dimension for which you want to limit journal access. See [Adding properties to dimensions](#).
2. Open the dimension sheet for the dimension whose members you want to limit. See [About maintaining dimension elements](#).
3. For those base-level members for which you want to allow users to post journal entries, type a "Y" in the EnableJRN column.
4. Repeat steps 1-3 for any other dimensions whose members you want to limit. Leave each member sheet open.
5. Process the dimension. See [Validating and processing members](#).

## Locking data for journal input only

You can set up Work Status so that owners or managers can lock data from users making any changes except journal changes. This is useful if you want to separate the BPC for Excel input schedules used to send data from your Review & Analysis phase, where all data is sent using Journal changes.

To lock data for Journal input, take the following steps:

1. Set up a work status code, such as 'Journal Only'. (See [Changing or adding status codes](#) section for detailed information on setting up a new status code.)

For example, you can set up a code named 'Journal Only' in which the Entity owner can set status from *Unlocked* to *Submitted* and set it back to *Unlocked* if they want to. The Manager can change from *Submitted* to *Journal Only*, which locks the data from changes by BPC for Excel input schedules or data manager data loads, but allow changes by Journals. The manager can then set status to *Approved*, which locks changes from any source, including Journals.

For more information, see [Managing Work States](#).

2. As Owners or Managers set work status, entities can be set for Journal input only.

## Setting up Journals Application parameters

You use application parameters to define conditions for applications. There are five application parameters related to journals behavior for users. These are:

Application parameter	Description
JRN_BALANCE	<p>Defines whether journals are required to be balanced.</p> <p>"1" - (or "Y") Journals must be balanced when they are posted. Journal users can save an unbalanced journal but may not post it.</p> <p>"0" - (or "N") Journals can be balanced or unbalanced. Journal users can change this behavior. If they select Balanced, it works the same as option 1.</p> <p>"2" - Journals users are prompted to save the journal as balanced, but are not required to do so.</p>
JRN_MAXCOUNT	Maximum number of journal entries returned from a query in the Journal Manager. This parameter is useful if you have a large number of Journal Entries and want to protect a user from launching a very large query that

	returns many journals and takes a long time.
JRN_POST_OVERWRITE	Sets whether the system keeps the same Journal ID when saving journals that were previously set to Posted status, but changed to Unposted status. "Y" indicates that the Journal ID is preserved, while "N" indicated that a new ID is created when the unposted journal is saved.
JRN_DESC_MODE	Determines whether member IDs or Descriptions are shown in the Journal template. If "Y", member descriptions are shown as row and column headings, if "N," member IDs are shown as row and column headings.

To set up Journal application Parameters

1. Open BPC Web and navigate to the application for which you are defining variables.
2. Click the Administration link.
3. Click the Application parameters link.

If either Journal parameter is not listed, in the New row, enter the name of the variable. Variable names are case-sensitive.

4. In the value field enter numeric or text values as described in the above table.
5. Click the Update button.

## Defining journal validation rules

You can set up journal validation rules to prevent users from saving invalid journal entries. After journal validation rules are defined, users who try to submit a journal entry to an invalid member set will receive an error, and will not be able to save the entry until they enter valid members.

To define journal validation rules, you must do the following tasks:

- Define validation property names for a dimension, and assign property values to the members you want to validate.
- Define the previous dimension as your driver dimension, and associate it with another dimension, the driven dimension.
- Determine which driven dimension members are permitted for associated driver dimension property values.

To define journal validation rules

1. From the Admin Console, define a validation property for each dimension you want to use to identify as your driver dimension. For example, if you want *Account* as your driver dimension, you can create a new property for it called Validation. See Adding properties to dimensions.
2. Assign property values on the members you want the system to validate against. For example, under the property Validation, you can set up property values called ICRule, and NOIC. The following table shows how a sample member sheet would look for the Account dimension. There are four account members using the ICRule value and two using NOIC.

ID	Description	Validation
IICSales	Inter-company sales	ICRule
IICCost	Inter-company cost of sales	ICRule
IICAccRec	Inter-company Accounts Receivable	ICRule
IICAccPay	Inter-company Account Payable	ICRule
3rd party	3rd party product	NOIC
AccRec	Accounts Receivable	NOIC



3. Close the Admin Console, and open the BPC Administration action pane. (From the launch page, select BPC Administration.)
4. Under the Browser Admin Tasks category, select Journal Validation Rules, then click the Dimension dependencies link.
5. From the Driver dimension name field, select the dimension for which you assigned the validation property, and then a driven dimension to associate it with. For example, you might want to associate Account with Intco, and Account with Entity.
6. You then identify the validation property you defined for the driver dimension.
7. Repeat steps 5 and 6 for each dimension dependency you want to define, then click Update.
8. Select Member Filters.
9. In the Member Filter page, enter a driver dimension property value for which you want to define a rule in the Driver property value field.
10. In the Driven dimension name field, select a driven dimension to associate with the property value.
11. In the Driven dimension filtered values field, specify which members of the driven dimension to include (or exclude). You can click the adjacent button to help define the filter. Select one or more members, then select Finish, or select the <> radio button to prevent users from posting to the member(s) selected.

When you match a driven dimension filtered member list with the driver dimension property value, during validation, the system will look for a driver dimension member that has the validation property value assigned to it (ICRule, for example), and the driven dimension associated with it (IntCo, for example). For example, if you set up <>Non\_IntCo with the Intco driven dimension and ICRule property value, and a user tries to save a journal entry with Account member ICCost, it cannot be posted to IntCo dimension member Non\_IntCo.

12. Click Update.

## Managing Insight

Managing Insight involves enabling application sets to use Insight, and maintaining system information related to Insight.

### Setting up Insight

Before you can use Insight, it must be set up. These steps are required for each application set after a new version of BPC has been installed, or if you have added a new application set for which you want to enable Insight.

To set up Insight

1. From the Admin console, enable Insight on the active application set. To enable Insight, do the following:
  - a. Start the Admin console and log on to the desired application set.
  - b. Select Insight from the left navigation pane.
  - c. Select Enable Insight from the Insight action pane. (If Disable Insight is displayed on the action pane, the application set is already enabled for Insight.)
  - d. Click OK when the Successfully Finished message is displayed. (This may take a few moments, depending on the size of the application set.)
2. From Insight Administration, synchronize the Insight server with the BPC server. See [Synchronizing data](#).
3. You can define KPIs for users, or users can define KPIs themselves. See [Adding KPIs](#).

### Setting up the RootCauseEvent table

Root cause events are those which exist outside of the BPC application, yet might have an effect on the performance monitored by an individual KPI. For example, if you had a KPI called "Sales for Florida, December, 2004," the root cause events could include:

- Florida rainfall 3.2 inches, December 12, 2004
- Florida temperature 83 degrees, December 12, 2004
- Florida humidity 86 percent, December 12, 2004

Before root causes are displayed in a Variance view for a KPI, the root causes must be loaded from an external source into Insight. This is done using Data Manager.

To load the data using Data Manager, you import a text file into a SQL table using a sample package, called LoadRootCause DTS. After modifying and running the LoadRootCause.dts package, you import the data into Insight by synchronizing and Insight. See [Synchronizing data](#).

You set up one root cause table per application set. The root cause table must contain at least 12 months of data, prior to the current month. Insight needs 12 months of data because it scans all the previous data to find a linear correlation between all KPIs for the current time period. See [About the RootCause/KPI association](#).

The RootCauseEvent table has the following information:

Field	Description	Example
EventID	(Required) A unique number for the event.	00001
EventName	(Required) The name of the event.	Weather
MeasureType	(Required) The type, or category, of event.	Rainfall
Measure	(Required) The data value.	3.2
EventTime	(Required) The time of the event, in the format mm/dd/yyyy.	12/12/2004

Field	Description	Example
EventAttribute 1-n	One field is required for each dimension in the application set (except for the Time dimension). For example, if you have three dimensions, Entity, Product, and DataSrc, you must create three fields, one field for each dimension. The data values for these fields can be empty.	Entity, DataSrc, Product, etc.
MeasureUnit	The unit of measure related to the measure value. This is for display purposes only.	Inches
AggregateFunction	(Required) The method of aggregation used to calculate the root cause value. For example, if SUM, the daily values in the root cause table are added together in the same aggregation as used by the KPI. For example, if the KPI aggregates monthly, the root cause events are aggregated monthly.	SUM (default) or AVG

To set up a RootCauseEvent table

1. Create a text file with the events for an application set.
2. From Data Manager, add the sample package LoadRootCause.
3. Modify and run the package.
4. Synchronize Insight.
5. Perform this procedure for each application set for which you want to correlate root causes and KPIs.

## About the root cause/KPI association

In order for Insight to associate an event in the RootCauseEvent table with a KPI, the event must conform to the following rules:

- The EventTime value must fall within the previous time period specification of the KPI, as defined by the SampleSize.
- Each Dimension value of the event must be empty, a child of the KPI dimension constraint value, or equal to the KPI dimension constraint value. If, for example, a KPI has dimension constraints: `ACCOUNT.[NetIncome], DATASRC.[TotWithAlloc], DEPARTMENT.[SalesMkt], ENTITY.[NorthAmerica]`

...And the event dimension constraints for the event dimensions are:

```
ACCOUNT.[PreTaxIncome], DATASRC.[TotWithAlloc], (no DEPARTMENT
value), ENTITY.[Florida]
```

...All the other event dimensions that are not in the KPI are ignored, and this event becomes associated with the KPI.

- Next, Insight calculates the correlation between the associated event and the KPI by doing the following:
  - Aggregating the associated event Measure values by Time Period
  - Getting previous Time periods (SampleSize) actual data for the KPI
  - Calculating the correlation using Linear Regression (LR)
- Finally, if the correlation coefficient  $\geq$  the threshold, get the top three events (based on MeasureType) according to the correlation and performance band of the KPI variance. If the variance  $> 0$  and the LR slope  $> 0$  (or the variance  $< 0$  and the LR slope  $< 0$ ), get the top three events with maximum Measure value. If the variance  $< 0$  and the LR slope  $> 0$  (or the variance  $> 0$  and the LR slope  $< 0$ ), get the top three events with minimum Measure value.

## Managing users

From the User Management tab, you can view the BPC users that have access to the active application and application set. You can also manage user information, including KPIs.

### View user information

From the User Management tab, you can view the user name and the groups the user belongs to, and the user's email address. If there is no email address specified, you can enter one.

### To view user information

1. Start Insight, and click Insight Admin.
2. From the User Manager tab, select View User Information.
3. Enter an email address, if desired, then click Update.

### Adding KPIs

Each BPC user has their own custom view of Insight, so each user must have their own set of KPIs. This means that each user can only see the KPIs that have been specifically defined for (or by) them.

When you define a KPI, you define performance bands to represent different variances, and dimensional constraints. The following table describes the performance bands. Dimensional constraints are the dimension members (in addition to the account) to which the KPI applies.

**Note:** The following performance band specifications are for income accounts. To create a similar performance band for expense accounts, simply reverse the signs (+ -).

Performance bands	Default value	Description
Weak	(,-0.2)	The current value is greater than 20% below the value for the previous time period. Weak performers are shown in red in the Dashboard and Radar views.
Under	(-0.2,-0.05)	The current value is between 20% and 5% below the value for the previous time period. Underperformers are shown in red in the Dashboard and Radar views.
Normal	(-0.05,0.05)	The current value is between 5% below and 5% above the value for the previous time period. Values that are in the normal band do not have any formatting in the Dashboard and Radar views.
Over	(0.05,0.2)	The current value is between 5% and 20% above the value for the previous time period. Overperformers are shown in blue in the Dashboard and Radar views.
Strong	(0.2,)	The current value is greater than 20% above the value for the previous time period. Strong performers are shown in blue in the Dashboard and Radar views.

### To add a KPI

1. From BPC Web, start Insight, then select Maintain KPIs from the action pane.
2. Select Add new KPI from the action pane.
3. In the Step 1 – Define KPI tab, do the following:
  - a. From the Application field, select an application. The subsequent field choices reflect the specified application.
  - b. In the Account field, enter the account for which you want to define the KPI. Click the '...' link to select an account from a list of accounts in the application.
  - c. In the KPI Name field, accept the default value (the selected account) or enter a meaningful name for the KPI.

- d. In the Dimensional Constraints fields, accept the default members (which are the top-most members in the hierarchy to which you have access), or enter new ones. To select new members, select a member from the drop-down list, or click Browse to select one from a window that displays the hierarchy. RPTCURRENCY, INTCO, DATASRC and ENTITY.
  - e. In the Correlation Context field, specify how you want to calculate root cause and prediction values for this KPI: Base values or Variance.
4. Select the Step 2 – Define Basis tab, and do the following:
    - a. In the Performance Band fields, accept the default performance bands, or adjust them. See the table above for the default values. Notice that you only need to modify one end of a band. The adjacent band's range limit that touches the current one is adjusted accordingly. Also notice that it is not necessary to modify the range of the 'Normal' band directly as both ranges are determined by modifying the adjacent bands.
    - b. In the Algorithms fields, choose the default algorithm from the list, and select the other algorithms you would like to use to calculate the prediction. Available algorithms are:
      - Linearregression (default)
      - Nonlinearregression
      - Multiline regression
      - Peicewiseline regression
  5. Select the Step 3 – Select Options tab and do the following:
    - a. Under KPI Management Options, select to get notified of a favorable or unfavorable variance, and ability to modify or delete the KPI definitions.
    - b. Under Default Chart Options, select how you want to view KPI values in the dashboard. From the Chart Type field: Columns (default), Line Chart or Pie Chart.
    - c. Select the Include YTD data check box if the dashboard should display year-to-date values.
    - d. In the Scales field, select how you want to scale your dashboard data: 1 - 100,000.
    - e. In the Decimal field, select how many values you want to appear to the right of the decimal point desired when viewing values in the Dashboard.
  6. Select the green OK button, when you have completed all modifications.

### Importing KPIs

If you want to add several KPIs at once, you can create a text file that contains all the KPI data, and then import it. The following table describes the fields that you define per line, separated by the a delimiter (pipe | or caret ^). You include one line item for each KPI you want to define.

Field	Description
User Name	(Required) The name of the user for which you are defining the KPI. The syntax is <i>&lt;domain name&gt;\&lt;user name&gt;</i> . An example of a domain and user name is mydomain\hjacobs.
Application	(Required) The name of the application for which you are defining the KPI. An example of an application is Finance.
KPI Name	(Required) A meaningful name for the KPI. The name is displayed in the first column of the Dashboard view. An example KPI name is Total Revenue.
KPI Fact	(Required) The ID of the account for which you want to define the KPI. An example of a KPI fact is ACCOUNT.[TOTREV].
Constraint	(Optional) The dimension members for which you want to define the KPI. If left blank, the top-most members in the hierarchy to which you have access are used. Otherwise, enter the desired dimension members. An example of a constraint is ENTITY.[MCG1],PRODUCT.[TOP].
Notified	(Required) Y (default) if you want to be notified by email when the KPI is within the underperformer or weak range, or N if you do not want an email alert.

Field	Description
Normal Performer Band	(Optional) The value that defines the normal performer range. If left blank, the default is [-0.05,0.05].
Strong Performer Band	(Optional) The value that defines the strong performer range. If left blank, the default is [0.2,).].
Over Performer Band	(Optional) The value that defines the overperformer range. If left blank, the default is (0.05,0.2).].
Under Performer Band	(Optional) The value that defines the underperformer range. If left blank, the default is (-0.2,-0.05).].
Weak Performer Band	(Optional) The value that defines the weak performer range. If left blank, the default is (,-0.2].

Note: The default performance band specifications shown above are for income account KPIs. To create a similar performance band for expense accounts, simply reverse the signs (+ -).

The following figure shows a sample KPI text file:

If you are moving KPIs from one server to another, you can import the Insight configuration file to the new server. See Importing system configuration files.

#### To import KPIs

1. Start a text editor, such as Notepad.
2. Type the desired KPI data, as described in the table above.
3. Save the file to the web server.
4. Start Insight Administration, and select Import User KPIs from the User Management tab.
5. Enter the path to the file you saved in step 3.
6. Enter a pipe ( | ) or a caret ( ^ ).
7. Select Import.

#### Exporting KPIs

After defining one or more KPIs, you can export them to a file, so you can import those KPIs to other application sets.

After you perform the export, all KPIs in the active application set are exported to a text file. You can take an exported KPI file and import it to another application set using Insight's import user KPI feature. See Importing KPIs.

#### To export KPIs

1. Start Insight Administration, and select Export User KPIs from the User Management tab.
2. In the Export file name field, enter the desired file name. The file will be saved to a folder called Insight Files under the applicable application set. (You can enter any file extension, for example, .TXT, or you can omit the file extension. The file will be saved under the exact name you specify.)
3. Select a pipe ( | ) or a caret ( ^ ) to use as the delimiter in the exported file.
4. Click Export.

#### Editing KPIs

After you add KPIs by either adding them individually or importing them, you can edit them.

#### To edit a KPI

1. Start Insight, and click Insight Admin.

2. From the User Management tab, select Manage & Add KPIs, then click Update.
3. Select Edit next to the KPI you want to edit.
4. Edit the fields, as desired. For descriptions of each field, see Adding KPIs.

The dashboard is updated the next time the KPI is recalculated, which may be overnight or the next time the Insight data is synchronized.

#### Deleting KPIs

You can delete KPIs.

To delete a KPI

1. Start Insight, and click Insight Admin.
2. From the User Management tab, select Manage & Add KPIs.
3. Select Delete next to the KPI you want to delete.
4. Click Update.

## Importing/Exporting configurations

You can import or export system configuration files between application sets. System configuration files contain KPIs and default Insight settings.

#### Importing system configuration files

You can import an application set's system configuration files to another application set on the same server, or to a different server. Moving configuration files between servers is useful when you are moving files from, for example, a development server to a production server. When importing to a different server, the application set names and the user names must be identical.

To import a system configuration file

1. Start BPC Administration, and click Manage Insight from the Administration Configuration action pane.
2. Select Import/Export Configuration, then select Import.
3. Select Import from files to get the files from the specified location, or select Import from another appset, then select the desired application set.
4. Click the green check mark in the action pane.

#### Exporting system configuration files

You can export an application set's system configuration file to another application set.

To export a system configuration file

1. Start BPC Administration, and click Manage Insight from the Administration Configuration action pane.
2. Select Import/Export Configuration, then select Export.
3. Select Export to files to get the files from the specified location, or select Export to another appset, then select the desired application set.
4. Click the green check mark in the action pane.

## Setting system configurations

Using the System Configuration options, you can specify the current time period, set up your email server, synchronize your BPC and Insight data, and set up KPI variance categories for an application set.

#### Setting the current time period

The current time period is the period in which variances are calculated. It also determines the default view of the Dashboard. Only Administrators can change the current time period.

To set the current time period

1. Start BPC Administration, and click Manage Insight from the Administration Configuration action pane.
2. Select Configure System, then select Current Time Period.
3. For each application in the application set, select a default time period from the drop-down list.

---

**Note:** Click Browse to select a time period from a window that displays the available time periods.

---

4. Click Apply.

Viewing email server information

You can view the name of the email server configured for Insight. The email server is specified in the SMTPSERVER field on the System Parameters page in BPC Web. See Setting System Parameters.

To view email server information

1. Start BPC Administration, and click Manage Insight from the Administration Configuration action pane.
2. Select Configure System, then select Email Server.
3. View the information, then click Cancel to go back to the previous page.

Synchronizing data

The first time you use Insight on an application set, data must be synchronized between the BPC and Insight databases. You should also synchronize the data at regular intervals to ensure Insight always has updated data. The synchronization process does the following:

- Synchronizes the Insight application with data from the BPC application.
- Recalculate correlations for each KPI to determine relevant accounts.
- Refreshes the cache of KPI calculations such as root cause events, variance, and predictions.

You can run a synchronization at any time, or set up a data synchronization schedule.

To synchronize data

1. Start BPC Administration, and click Manage Insight from the Administration Configuration action pane.
2. Select Configure System, then select Synchronize.
3. Select Schedule Synchronization to set up a schedule. Select Daily, Weekly, or Monthly, then specify a time of day by selecting the time from the drop-down.
4. Enter your user name and password (twice).
5. Do one or both of the following:
  - Click Apply to enable the schedule.
  - Click Synchronize Now to start the synchronization process immediately.

Defining KPI variance categories

Variances are calculated by taking the actual amounts and comparing them to the budgeted amount. You can define the actual and budget categories that you want to use for these comparisons.

To define a KPI variance category

1. Start BPC Administration, and click Manage Insight from the Administration Configuration action pane.
2. Select Configure System, then select Variance Category.
3. From the Category 1 field, select the category you want to use for one of the values.



4. From the Category 2 field, select the category you want to use to compare against the other value.
5. Click Apply.

#### Setting the measures type

You can specify a default measures type and an aggregated measures type for each Insight application. The measures type is used to calculate KPI variance.

By default, the basic measure type is set to periodic, and the aggregate measures type is set to YTD.

#### To set the measures type

1. Start BPC Administration, and click Manage Insight from the Administration Configuration action pane.
2. Select Configure System, then select Measures Type.
3. For each Insight application, select a default measures type and an aggregate measures type.

#### Configuring Prediction Algorithms

In each application, you can set the parameters for the prediction algorithms to be used in determining the relationships between your KPIs and the underlying business drivers.

---

**Note:** The development and implementation of the algorithms is outside the scope of this topic. For more information about the general concepts underlying each algorithm type (linear, non-linear, multi-linear, and piece-wise linear), see the *BPC Web User's Guide*.

---

#### To set prediction algorithm parameters

1. Start BPC Administration, and click Manage Insight from the Administration Configuration action pane.
2. Select Configure System, then select Measures Type.
3. For each Insight application, set the following parameters:

Parameter	Default value	Description
Seasonal period	12	BPC can be configured to screen out the effects that seasonal variations might have on your data. For instance, in the case of a retail store, there is often a relationship between the number of employees and sales. BPC can be configured so that the prediction algorithm compensates for the effect a holiday shopping season would normally have on the sales volume, and allow you to gain an understanding of the true employees-to-sales relationship. The number of periods define the seasonality. The period (as a unit) is the base level of your Time dimension member. In many cases, the period will be a month; the default setting of 12 would tell BPC that it should create a seasonal adjustment for each month (12 months in 1 year). In a weekly Time dimension, you set 52 periods, with the period consisting of a week.
Prediction time lag	0	The number of periods by which you'd like to offset one variable which (may be) affected by another variable. For instance, you may wish to determine the effect that a quality improvement program has on customer satisfaction. In this case, you might need to adjust for the expected lag time (in the number of time periods) between the time you begin your quality program and the time it should take for the effects of that program to affect customer satisfaction.

Parameter	Default value	Description
Calculation periods used in predictions in the application	24	The total number of data periods included in the prediction analysis. For example, given the default setting and a time period of months, the algorithm would use the previous 24 months of data.
Minimum sample size	12	This is the minimum sample size (in terms of time periods) for your data. If any of the variables included in your sample sets do not meet this criterion, the analysis will not be performed.

## Viewing services and logs

Insight uses several services to manage its components. The Services & Logs tab shows the status of each Insight service, and allows you to view a log file that displays detail information about a specified service.

### Viewing services

The page displays all the Insight services, and their IP addresses, ports, and statuses.

#### To view the services

1. Start Insight, and click Insight Admin.
2. From the Service & Log tab, select View Services.

### Viewing service logs

You can use the View Service Logs page for troubleshooting your Insight services. The page allows you to select a log file, service, and severity.

#### To view a service log

1. Start Insight, and click Insight Admin.
2. From the Service & Log tab, select View Services Log.

## Maintaining a BPC server

This chapter presents tasks for maintaining a BPC Server.

### BPC Server Manager

BPC Server Manager helps you maintain your BPC server by providing you with status information on server, diagnostics on the status of the BPC installation as well as information on the application sets in BPC. These tools will come in handy when you need to access the overall status of the server.

When you open the Server Manager you are presented with the server information screen (example below).

To access BPC Server Manager, select Start > Programs > SAP > Server Manager.

Use the Help menu to get more information about this utility.

### Importing Non- BPC reports

You can import reports from other systems into BPC. Using BPC advanced mapping feature you have the ability to control the viewing of these reports with the current view. BPC security will also be applied to these reports.

The files to be imported must be arranged in a package. The package must consist of:

- An index file, this contains the instructions for importing the reports. It is read at run time by BPC Web
- Files to be imported

The index file and package folder must be placed in the following directory on the BPC server.  
C:/Everest/Webfolders/[AppSet]/[application]/REPORTIMPORT.

The package folder and the index file must both have the same name.

#### Mapping to BPC

BPC maps the reports to be imported to BPC dimensions based on the report's file name. Depending on the format of the file name you can have to either rename the files manually or use the TRANS feature. The TRANS feature is described below. You rename the file to a name BPC can understand. For example, a report from an external system maps to the Entity and Time dimensions in BPC and you want this report to be viewed by Entity and/or Time. You need to rename the file name for the Entity member and time member. For example, WORLDWIDE1,2001.JAN.HTML. With the file name in this format BPC is able to import the report and link it to the Entity and Time dimension.

In order for a non-BPC report to be controlled by the current view, you must map the reports to the proper dimensions in BPC . You do this in the index file. The index file is made up of sections, each section identifies a different set of instructions for BPC . The index file is described below.

#### The Index File

BPC gets its instructions from the index file based on the sections in the file. The sections are OPTION, MAP, FIXED, and TRANS, and they must be present in the index in this order.) There must be one blank line between sections but not more than one. The section headings must be in all CAPS and in brackets [ ].

OPTION - contains user definable options. They are:

Option	Description
Delimiter	Specifies the delimiter used in the file name. This field can be blank (no delimiter)
Usetrans	Specifies whether to translate member names (i.e. external to internal). accepts YES or NO value.
Useinitdir	Reserved for future use.
Initdir	Reserved for future use.
Report	The name of the report - any name you want to use. Cannot contain any special characters

Option	Description
Section	The name of the section in the book. Cannot contain any special characters
Book	The name of the book. Cannot contain any special characters

Report, Section and Book, the same as for Books published from BPC. BPC Web reads the Book parameter as the name of the book. When you publish to a Web Publication you choose the book name.

MAP - specifies the name of the dimension(s) that are represented in the file name. Remember that the file name is made up of member names. (Entity and Time from the example above) If there is no delimiter in the file name you can use the Start Point, Length feature. For example the file name for a report is SalesUS2000.jan. You can specify the mapping for this as follows.

```
[MAP]
ENTITY 1,7
TIME 8,7
```

The first number is the start point of the member name and the second number is the length of the name.

FIXED - any dimension specified here is fixed and cannot be changed by changing the current view

TRANS - used to translate member names from an external name to an internal name.

The dimension being translated must be entered in CAPs, the TRANS statement must be in brackets [] (ex. [TRANS:TIME]).

You can have more than one TRANS statement.

For example: The reports from the external system have similar names to BPC dimensions. You can specify a mapping that will translate the external name an internal name.

Here is an example of a translation:

```
[TRANS:TIME]
20010101=2001.JAN.W1
0010102=2001.JAN.W2
0010103=2001.JAN.W3
0010104=2001.JAN.W4
0010201=2001.FEB.W1
```

Here is an example of an index file:

```
[OPTIONS]
DELIMITER=
USETRANS=no
USEINITDIR=no
INITDIR=
REPORT=Growth Analysis
SECTION=SECTION1
BOOK= SAP Book 3
```

```
[MAP]
PRODUCT
TIME
```

```
[FIXED]
CATEGORY=BUDGET
```

```
[TRANS:TIME]
20010100=2001.JAN
20010200=2001.FEB
20010300=2001.MAR
20010400=2001.APR
20010500=2001.MAY
```

#### Creating a package

1. Create a folder named for the package in the C:/Everest/Webfolders/[AppSet]/[application]/REPORTIMPORT directory.
2. Copy the reports to be published into this folder
3. Rename the report files if necessary.
4. Create an Index file. (This file should reside below the REPORTIMPORT folder but outside the package folder.

#### Running a package

1. In the Company Desktop of BPC Web click the Administration link, and then click the Bulk Collaboration link.
2. Click on the package you want to run.
3. Click the Process This Package link.

### Editing default messages

As the administrator, you have the ability to modify the default messages that appear when Web publications on the desktop cannot be displayed for various reasons. The messages are contained in .htm files in the SystemLibrary directory for the application set.

The following table describes the files you can modify.

Filename	When Used	Sample Message
NoAccess_.htm	When a report cannot be displayed because the user does not have access to the necessary combination of category and entity.	You do not have access to this report.
NoTarget_.htm	When the collaboration file specified for a Web publication is missing.	Target is missing.
NoReport_.htm	When the specified report is not available for the current view.	No report is available.
NoBook_.htm	When the specified book is not available for the current view.	No book is available.

Filename	When Used	Sample Message
Default_.htm	Default text that appears in a new Web publication object.	Click on Edit button to select information to display here and to change the caption.

## Database specifications

The following specifications relate to Microsoft® SQL Server™ 2000 Analysis Services (referred to in this document as MS Analysis Services).

MS Analysis Services provides the online analytical processing (OLAP) and data mining capability behind BPC.

Item	Specification	BPC notes
Dimensions in a database	65,535 maximum, regardless of the number of cubes or whether dimensions are shared or private	There is no limit to the number of dimensions defined in an AppSet but an application cannot have more than 20 dimensions
Levels in a database	65,535 maximum	
Cubes in a virtual cube	64 maximum	No explicit support for virtual cubes
Measures in a cube	1,024 maximum	BPC uses one input measure, and uses calculated measures for frequencies.
Measures in a virtual cube	2,048 maximum	Not applicable.
Dimensions in a cube	128 maximum, including the Measures dimension	20 is the recommended maximum number of dimensions for BPC
Levels in a cube	256 maximum	
Levels in a dimension	64 maximum	The number of levels you can have in BPC is dependent on the number of properties in the dimension.
Members in a virtual dimension created in SQL Server version 7.0 OLAP Services	759 maximum	No explicit support for virtual dimension. Virtual dimensions can be manually created and used in applications.
Members in a parent	64,000 maximum	5,000 members per dimension supported for version 2000
Calculated members (server defined) in a cube	65,535 maximum	
Calculated members in a parent measure in session context	31,743 maximum	
Calculated members in a parent measure in query context	31,743 maximum	
Calculated members in a	759 maximum	

Item	Specification	BPC notes
parent dimension member in session context		
Calculated members in a parent dimension member in query context	759 maximum	
Aggregations per partition	65,535 maximum	
Cells returned by a query	$2^{31}-1 = 2,147,483,647$ cells maximum. Although cubes can be larger than this limit, a query that requests more than $2^{31}-1$ cells from a cube will fail	
Record size for source database table	8060 bytes maximum	
Length of object name (except dimension name)	50 characters maximum when using Analysis Services Manager. 24 characters maximum when using PivotTable® Service	Max. 50 characters for member names, and 50 characters for member descriptions
Length of dimension name	24 characters maximum	Max. 20 characters for dimension names, and 50 characters for dimension descriptions.
Length of aggregation prefix	50 characters maximum	

Note: The term "character" above refers to a UNICODE character.

## Web Admin tasks

The Web Admin tasks are available from a browser and allow you to customize the BPC system for your needs.

### Setting application set parameters

Administrators who have "Appset" task security rights can view and change application set parameters.

The parameters that require a value include 'Required' in the description. Parameters that do not require a value include 'Optional' in the description. If an application set parameter is required, you can leave it blank to accept the default, but if you delete the parameter, the system may not work correctly. If an application parameter is optional, you can leave it blank or delete it.

The following table describes the parameters you can set.

Key ID	Description
ALLOW_EXTENSIONS	Use this parameter to specify whether or not a given file type can be uploaded to any part of the system. Enter the file extension in the Value column.
ALLOWEXTENSIONS	Use this parameter to prevent certain file types from being displayed in the Business Planning and Consolidation web pages.
ALLOW_FILE_SIZE	You can enter a file size (MB) to allow uploading for files of this size and smaller.
APPROVALSTATUSMAIL	Defines whether owners and managers get email when one or the other changes a work status. Valid values are Yes (send email) and No (do not send email). (Optional)
APPROVALSTATUSMSG	<p>Allows you to define a custom email message that is sent when a work status code is changed. The message is applicable to all applications in the application set. You can customize the message using the following variables:</p> <ul style="list-style-type: none"> <li>• %USER% - name of user who changed the status</li> <li>• %ED% - Entity dimension</li> <li>• %EM% - Entity member</li> <li>• %CD% - Category dimension</li> <li>• %CM% - Category member</li> <li>• %TD% - Time dimension</li> <li>• %TM% - Time member</li> <li>• %STA% - Work status</li> <li>• %OWNER% - Entity owner</li> <li>• %TIME% - time of change</li> </ul> <p>For example, you can enter "This is to inform you that %USER% has updated the work status for %EM%, %CM%, %TM% on %TIME%". The message can be up to 255 characters, and there is no need for quotes or brackets around parameters.</p>
AVAILABLEFLAG	<p>Controls whether the system is offline or not. Yes means the system is online and available for sending data to the database. You can take the system offline by changing the value to No. (Required)</p> <p>See Setting application set status</p>



AVAILABLEMSG	<p>The message that is displayed to users who try to access an application that is offline (AVAILABLEFLAG = No). (Required)</p> <p>For example, the message might be "BPC is temporarily unavailable due to scheduled maintenance. Please try again later".</p> <p>See <a href="#">Setting_application set status</a></p>
AVAILABLEURL	<p>The name of the Web page to display to users who try to access an application that is offline (AVAILABLEFLAG = No). (Required)</p> <p>For example, the url might be: /osoft/NotAvailable.asp</p> <p>See <a href="#">Maintaining applications</a></p>
BPFSTEP_COMPLETE_MSG	<p>When a step in a business process flow is identified by the user as complete, an email is sent to that step's reviewers (when reviewing and email notification is enabled for the step). This is the text that is included in the body of that email message.</p> <p>This parameter is optional. If this field is blank, the body of the message (as well as the subject line) will communicate the following:  <i>"&lt;Step_Name&gt; step of &lt;BPF_Name&gt; has been completed by &lt;User&gt;".</i></p> <p>See <a href="#">Adding new business process flows</a></p>
COMPANY_LOGO	<p>Use this parameter to add your corporate logo to the default templates in BPC for Excel. Enter the file name for the logo image that you want to display.</p> <p>The logo image (for example, SAP.JPG) must be stored in the AppSet directory "[Server Install]\Data\WebFolders\[Appset]". You can use BMP, GIF, and JPG image types.</p>
DEFAULT_EXTENSIONS	<p>Use this parameter to add allowed file types (by extension name). By default, the system allows you to enter the file types listed below:</p> <p>XLS, XLT, DOC, DOT, PPT, POT, XML, MHT, MHTML, HTM, HTML, XLSX, XLSB, ZIP, PDF, PPTX, PTM, POTX, DOCX, DOCM, DOTX, TDM, PDM, JPG, PNG, GIF, CSS, MRC</p>
DTSSTATUSCHECK	<p>Use this parameter to hide to show the 'Refresh Status Every' checkbox in the Data Manager Status view.</p> <ul style="list-style-type: none"> <li>0 Hide the checkbox</li> <li>1 Show the checkbox</li> </ul>
FILESFOLDERDELIMITER	<p>When you create web-ready files in Excel, the system creates subfolders based on the native Excel 'Save as HTML' function. Since the naming rule of the subfolder differs for each Microsoft Office language, this parameter allows the system to find the subfolder containing the defined delimiter when selecting web-ready files in BPC Web.</p> <p>Separate multiple delimiters with a colon; for example, :,:_.</p> <p>For example, under a folder named 'Report.htm', the system creates the following subfolders:</p> <ul style="list-style-type: none"> <li>English: report_files</li> <li>Chinese: report.files</li> <li>French: report_fichiers</li> <li>German: report-Dateien</li> <li>Italian: report-file</li> <li>Korean: report.file</li> </ul>

JREPORTZOOM	This parameter allows you to set the default zoom magnification value on HTML journal reports. We recommend that you set the value to 75%. (Optional)
LANDINGPAGEITEM	To customize the Getting Started page on BPC Web, please contact your system administrator.
LIMITOFDIFFERENCE	Use this parameter to set the smallest value for processing logic. When the system processes logic, it ignores data with a smaller value.
LOPTZ_AVAILABLE	You can use this parameter to take the system offline during the Lite-Optimize application process. <ul style="list-style-type: none"> <li>1: Change to offline</li> <li>0: Do not change to offline</li> </ul>
MAXLRCOLUMNS	The maximum number of columns to display in a live report in BPC Web. The value includes header and data columns. For example, if you specify a value of 5, one heading column and four data columns are displayed.
MAXLRROWS	The maximum number of rows to display in a live report in BPC Web. The value includes header and data rows. For example, if you specify a value of 5, one heading row and four data rows are displayed.
MSNIMPassword	The password that the system uses to operate IM alerts in Insight.
MSNIMUser	The user name that the system uses to operate IM alerts in Insight.
MULTIBYTE_FORMULA	Use this parameter to support dimension formulas with member IDs that contain double-byte characters, such as those in Japanese, Chinese, Korean, and Russian. <ul style="list-style-type: none"> <li>0 No support for double-byte characters</li> <li>1 Support for double-byte characters</li> </ul>
RETRIEVE_ON_OFFLINE	By default, users can perform certain tasks when the application set's status is 'Unavailable'. You can use this parameter to prevent users from doing this. The tasks include: <ul style="list-style-type: none"> <li>Execute logic (script, business rules) from DTS package</li> <li>Run Export from fact table package</li> <li>Run Append into fact table package</li> <li>Add new comments</li> <li>Save data through DHE (Dynamic Hierarchy Editor)</li> </ul> The options include the following: <ul style="list-style-type: none"> <li>0 Do not allow users to perform tasks while offline</li> <li>1 Allow users to perform tasks while offline</li> </ul>
SESSIONTIME	You can use this parameter to define the session time in minutes. The session time appears in 'Who's Online' in BPC Administration. The default is 3000 minutes; you can enter another number of minutes.

SMTPAUTH	<p>The authentication method of the SMTP server. (Required)</p> <ul style="list-style-type: none"> <li>• 0 = Anonymous</li> <li>• 1 = Basic</li> <li>• 2 = NTLM</li> <li>•</li> </ul> <p>This setting does not change the method on the SMTP server, but must match the type of authentication enabled on it. Failure to set this appropriately can result in errors from the email server.</p>
SMTPPASSWORD	The password for the sending email user name. (Required)
SMTPPORT	Port number for your SMTP email server. The default is port 25, the default SMTP server port number. (Required)
SMTPSERVER	The name or TCP/IP address of the SMTP email server the system uses to send email. (Required)
SMTPUSER	The user name from which email from the system originates. (Required)
TEMPLATEVERSION	<p>Current version number of the dynamic templates in your application set. Whenever you add to or change your input schedule or report dynamic templates, you should increment this version number so that users will automatically get the new templates downloaded when they log on to this application set. (Required)</p> <p>You can also reset the template version from the Admin Console. See Setting template version.</p>
UPLOADTEMP	Temporary folder used to store Content Library documents. (Required)
USE_VARCHAR_FOR_DIM	<p>If you have a space restriction problem from SQL and OLAP, you can use this parameter to determine whether the columns in the Dimension table are Varchar type or Nvarchar type (one of the data types from the SQL table column).</p> <ul style="list-style-type: none"> <li>• 0 All columns are created as Nvarchar type</li> <li>• 1 All columns are created as Varchar type</li> </ul> <p>If you use a 2-byte character language, such as Korean, Japanese, Chinese, or Russian, we do not recommend using the Varchar data type.</p>

Note: When you design a Content Library page that contains a BPF object, an application set parameter starting with *B0000000000* (such as *B00000000002*) is automatically created by the system. This is an internal number only, and is reserved for future use.

To set application set parameters

1. Start BPC Administration, and from the Administration Configuration action pane, select Set AppSet parameters.
2. Modify the parameters, as desired, and then click Update.

## Setting application parameters

Administrators with Administration - Application task security rights can set application parameters. Application parameters control the way certain features behave in an application. The application parameters can be different for each application within an application set.

The following table describes the parameters you can set. If an application parameter is required, you can leave it blank to accept the default, but if you delete the parameter, the system may not work correctly. If an application parameter is optional, you can delete it if you want to.

Many of the journal (JRN\_) parameters below are used to enable Statutory Consolidation journal requirements. When these parameters are set, the system generates auto-generating closing values, data entry locking of opening balance, and automatic reversing of the sign of specific account details.

Key ID	Description
CUSTOMFACTTBLINDEX	Use this parameter to define a dimension list to create a custom index in the appropriate database table and accelerate import processing. Enter a list of dimensions separated by commas; for example, 'Account, Time, Entity, Category'. When you use a custom fact table index, we recommend that when you modify the application, you select the Reassign SQL Index option.
DimsForFactTblIndex	The Fact and Fac2 tables use a clustered index, and the write-back table uses a composite index. The default fields and order of the fields for the index is Category, Time, Entity, Account, RptCurrency.  If you need to change the field and order, enter them here.
DTSLOGPAGESIZE	Use this parameter to set the number of records that display in the Data manger Status View. The parameter defaults to 300.
JRN_ACCDETAIL_DIM	A special dimension name used to manage the Opening, Closing, and Reverse Sign codes. This field is used in conjunction with the JRN_CLOSING_CODE, JRN_OPENING_CODE, and JRN_REVSIGN_CODE fields.
JRN_BALANCE	Controls whether Journals are required to be balanced. 1=Yes, 0= No. (Optional)
JRN_CLOSING_CODE	The member name for the 'closing' code. This member is part of the dimension defined in the JRN_ACCDETAIL_DIM field.
JRN_DESC_MODE	Accept the default N when you want member IDs to be displayed in the application's journal template. The default also provides optimal readability.  Use the value Y to display the description in the journal template.
JRN_IS_STAT_APP	If the application is used for statutory consolidation, set this field to Y to enable the subsequent journal fields. Set to N to disable the subsequent fields.
JRN_MAXCOUNT	Maximum number of journal entries returned from a query in the Journal Manager. This parameter is useful if you have a large number of Journal Entries and want to protect a user from launching a very long query. (Optional)
JRN_OPENING_CODE	The member name for the 'opening' code. This member is part of the dimension defined in the JRN_ACCDETAIL_DIM field.
JRN_POST_OVERWRITE	Controls whether the system keeps the same Journal ID when saving journals that were previously set to Posted status, but changed to Unposted status. (Optional)  Y indicates that the Journal ID is preserved, while N indicated that a new ID is created when the unposted journal is saved.
JRN_REOPEN	Allows you to define the default for reopening journals: N = do not allow reopening of journals (default); or Y = allow the reopening of journals.

JRN_REOPEN_PROPERTY	<p>A custom Journal module assumes that a property named UB must be present in account dimension to further filter the Journals to reopen.</p> <p>The default is Group; when this is set you do not need to modify the account dimension.</p>
JRN_REVSIGN_CODE	<p>The member property name that is used to reverse the sign during posting and unposting. The property value must be set to Y to use this feature.</p> <p>This property is defined for the members defined in the 'JRN_ACCDETAIL_DIM' field.</p>
JRN_VALIDATION_SP	<p>Allows you to specify the SQL stored procedure name to be executed before posting/unposting data. This stored procedure performs custom validation on the entire RecordSet to be posted/unposted and gives back go/no-go to the posting engine.</p>
LOCKREPORT	<p>Use this parameter to launch a report when submission validation for a data region is not '0' in BPC for Excel. Store the template in the eExcel\Report\ folder.</p>
ORG_OWNERSHIPCUBE	<p>The default value is OWNERSHIP.</p>
ORG_INTCO	<p>The default value is I_NONE, which should also be a member ID in the INTCO dimension in the ownership application if using dynamic hierarchies.</p>
ORG_ACCOUNTOWN	<p>The default value is PGROUP.</p>
ORG_ACCOUNTLIST	<p>The default value is METHOD,POWN,PCON.</p>
ORG_PARENTPROPERTY	<p>This parameter is used with dynamic hierarchy statutory applications when defining fixed hierarchies. The value must match the value in the ParentProperty property value of entities in the statutory application's supporting ownership application.</p>
OWNERSHIP_APP	<p>The name of the Ownership application. If this parameter does not exist, the consolidation procedure will by default search for an application named OWNERSHIP.</p>
SEC_CACHE_EXPIRE	<p>Use this parameter to set the value of the secured profile cache expiration time. The default value is 24 hours; enter another value to change the number of hours.</p> <p>Note: If you change this parameter, you must reset IIS on the Application Server.</p>
SEND_SGTABLE_COUNT	<p>You can use this parameter so that the system can split the sgData[Application] table when it sends large amounts of data.</p> <p>The default value is 2.</p> <p>Note: After the parameter is added or modified, modify the application in the Administration console.</p>

SIGNED_DATA_FORMAT	<p>When you send data from the client side, the system inserts data into the Facttbl and the SignedData column. You can use this parameter to set the decimal length of precision and of scale for these elements.</p> <p>The default value is 25 for precision and 10 for scale (25,10). For example, if you send a 10 value from BPC for Excel, 10.0000000000 is inserted into the fact table.</p>
TOPDOWN	<p>Use this parameter to set work status approval based on whether or not sub-work status is approved.</p> <ul style="list-style-type: none"> <li>No - bottom up approval; work status can be approved only if all sub-work status is approved.</li> <li>Yes - top down approval; work status can be approved whether the sub-work status is approved or not. (Required)</li> </ul> <p>Note: When you change this parameter, all existing work status approval information is removed.</p>
VALIDATE_MBR_LOGIC	<p>Use this parameter to validate members when the system processes logic.</p> <ul style="list-style-type: none"> <li>1 The system filters records that would otherwise post to a parent member, to a member that no longer exists, or to a member-calculated dimension formula.</li> <li>0 The system bypasses this validation.</li> </ul>
WORKSTATUSVALIDATE	<p>Use this parameter to determine if the system validates submitted values for work status purposes.</p> <ul style="list-style-type: none"> <li>Yes - validate</li> <li>No – do not validate</li> </ul> <p>Note: To use work status validation, you must set up a validation account in the Administration console, See Changing work status settings for applications.</p>
YTDINPUT	<p>This parameter controls whether data is input in year-to-date format. Valid options are 1, which means ytd format; or 0, which means periodic format. (Optional)</p>
YTDInputTimeHir	<p>This parameter designates the time hierarchy that will be used by a YTD storage application. H1 is the default.</p>
YTD_NECJ_RETRIEVE	<p>You can choose a retrieval query type to improve the performance of data retrieval when retrieving YTD measures on Periodic applications. EVDRE, EVGET, EVGTS, EVSND, EVINP functions You can use this parameter is useful when the below condition are met:</p> <ul style="list-style-type: none"> <li>YTD / QTD measures data is retrieved on periodic application or Periodic / QTD measures data is retrieved on YTD application.</li> <li>The retrieval range is huge, but the portion of the data existing cells is small.</li> <li>ColKeyRange or RowKeyRange has over 2 dimensions.</li> <li>The retrieval range has no members having dimension formula.</li> </ul> <p>Enter one of the following options:</p> <ul style="list-style-type: none"> <li>1 To use NonEmptyCrossJoin query</li> <li>0 Not to use NonEmptyCrossJoin query</li> </ul>

To set application parameters

1. Start BPC Administration, and from the Administration Configuration action pane, select Set Application parameters.
2. Modify the parameters, as desired, and then click Update.

## Deleting books

You can delete books that have been published to BPC Web.

To delete a book

1. Start BPC Administration, and from the Administration Configuration action pane, select Manage Books.
2. Select the check box next to one or more books that you want to delete, then click the Delete button.
3. Select Yes to confirm the deletion.

## Enabling activity auditing

Activity auditing allows you to track the administrative tasks performed in the system. Once activity is recorded, you can run a report that shows system activity, based on specified criteria. The report shows when the task was performed, and by whom.

If enabled, BPC tracks activity for the following functional tasks:

- Application set and application setup
- User and team setup
- Member access and task profile setup
- Business Process Flow management
- Adding, deleting, and modifying business process flows
- Saving business process flows to new names
- Data audit maintenance and activation of data audits
- Document type and document sub-type setup
- Activation of activity audit

To enable activity auditing

1. Start BPC Administration, and from the Administration Configuration action pane, select Manage activity auditing.
2. Select Enable Admin Activity to record all administrative activity, and select Enable user activity to record all end user activity.
3. Click Update.

## Reporting on activity audit

The BPC Reporting Console allows you to report on and track activity audit within BPC.

To report on activity

1. From 'Getting Started' mode within BPC Web select Launch BPC reporting console from the action pane.
2. Select Audit Activity Report from the AppSet Reports section of the action pane.
3. Use the following table to specify the report parameters.

Option	Description
Start Date	The start date for the query
Start time	The start time for the query
End date	The end date for the query
End time	The end time for the query
Show activity for	AppSet - Report on activity at the application set level
	App - Report on activity at the application level
Show activity kind	All - Report on activity for all user types
	Admin - Report on activity for admin users
	User - Report on activity for general users
Show activity type	All - All activity types
	Add &ndash; Only add activity
	Change - Only activity that involves changes within the system
	Delete - Only delete activity
	Query - Only query activity
Report from	Active - Report on an active BPC system
Function task	Set AppSet Parameters - Report on application set information
	Set Application Parameters - Report on application information
	Modify user - Report on user setup information
	Modify team - Report on team setup information
	Modify member access profile - Report on member access profile information
	Modify task profile - Report on task profile information
	ManageBPF - Report on managing business process flows
	Add a new business Process - Report on business process flow information
	Delete a Process Flow - Report on deleted business process flows
	Modify Process Flow - Report on modifications to business process flows
	Save As business Process - Report saving business process flows to new names
	Manage Data Audit - Report audit information
	Enable Data Audit - Report on archiving audit information
	Manage Document Types - Report on document types
	Manage Document Sub Types - Report on document subtypes



Option	Description
	Enable Activity Audit - Report on activity audit settings
Source	This is the database field value to base your source on. For example, you can specify a source as User ID. Leave this field blank to report on all possible values.
Parameter	This is the value for the source specified. For example, if you specify a source as User ID, the parameter could be User1 (where User1 is the BPC Id of the user). Leave this field blank to report on all possible values.
Succeeded	All - Report on all successful and unsuccessful activity for the function task
	Yes - Only report on successful activity for the function task
	No - Only report on unsuccessful activity for the function task
Field	This is the database field value to specifically track activity on. Leave this field blank to report on all database fields.
Previous Value	This is the previous value for the field specified. Leave this field blank to report on all previous values for the field specified.
New Value	This is the new value for the field specified. Leave this field blank to report on all new values for the field specified.
Expand All	Activity audit reports default to display tasks collapsed. Selecting this check mark will force the report to appear with all tasks expanded.

- Click the Check Mark in the action pane to generate the report.

## Setting up drill-through

Drill-through lets you view data from an external database or from the Journals database by clicking the Drill-through button on a cell in a report for which a drill-through has been defined.

You can define a number of Drill-throughs, but you can only use one per dimension.

drill-through is available on any cell that contains an EvGTS, EvGET, or EvINP formula.

### Example for external database

You have a report in your BPC application that has a row defined for TotalUnitsSold and perhaps you want to see this number broken out by customer. As long as you know where that data resides, and it is an OLE DB database, you can retrieve it from that external database using Drill-through.

### Example for journal database

For the same report, you have Commission on a row. You can define a Drill-through to show the Journal entries that contributed to the final numbers for the Commission member.

To set up a drill-through you need to do the following:

- Create a DrillKey property — This must be present in the dimension for which you are setting up a drill-through. The value in the DrillKey column can be any alphanumeric name you would like. You specify the DrillKey value for the member(s) on which you want to drill-through.

To add properties to your dimension(s) see Assigning Properties to a dimension.

- Create a Database Query — (external database only) to create the query which will retrieve data from an external database.
- Set up the Drill-through table in BPC Web — You connect Drill-throughs to your application set using the BPC Web Administration > Edit Drill-through Table task.

### Adding drillkey properties to dimensions

Add a DrillKey property to the dimension for which you are creating the Drill Through. After adding the property to the dimension you must define a DrillKey value, this can be anything you want it to be. The DrillKey value should be entered for any member that you want to be able to access the Drill Through information.

#### To add drillkey properties to dimensions

1. Add a property named DrillKey to the dimension or dimensions for which you want to use drill-through. See Assigning properties to a dimension.
2. Devise a DrillKey value and assign that value to any member of the dimension for which you want to use drill-through. See Adding members to a dimension.

### Creating drill-through queries

If you are setting up drill-through for an external database, you need to create a drill-through query that gets the data from the external database.

---

**Note:** You do not need to do this if you are creating a Journal drill-through. In that case, a query is generated for you by BPC and is named Journal.dqy. This query is generated automatically when you use the Journal Wizard to create a Journal Template.

---

#### To create drill-through queries

1. Open the BPC for Excel Module.
2. Select Data > Get External Data > New Database Query.
3. On the Databases tab double click on <New Data source>.
4. Enter a name for the new data source. The name should have some relevance to the database you are querying.
5. Select the appropriate driver for the type of database to which you are connecting.
6. The screens that follow are going to differ based on the type of database to which you are connecting. Complete the screens as prompted by the query wizard.
7. On the last screen of the query wizard click on the Save Query button.
8. Name your query and select a destination for the query file. The destination must be the QueryFiles directory found under <BPC >/Webfolders/[ApplicationSet]/[application])

## Varying Dimensions by current view

While defining your query you can set the query filter to vary based on the current view of a dimension. You can do this on the Query Wizard — Filter Data screen. The syntax to base the dimension on the current view is '%DimName%', where DimName is the dimension name you are referencing- such as %Entity%.

### Editing the drill through table

You can set up drill-through by editing the drill-through table.

To edit the drill-through table

1. Click on the Edit drill through table link in the BPC Web Administration page.
2. Complete the following fields:

Field	Description
DrillKey	The value specified in the DrillKey property column for the dimension member on which you would like to Drill Through.
Title	The title appears at the top of the Drill Through page.
FileName	The name of your SQL Query file (.dqy). This file must reside in the QueryFiles directory under the application. For journals, a Journal.dqy file is automatically built for you. For all other external databases, you must create your own SQL Query file.
MaxRows	The maximum number of rows to display on one page
UserID and Password	A valid userID and Password on the database to which you are querying.

3. Click the Update button.

## Managing document types and subtypes

Document types and subtypes are used to categorize collaboration postings. This is useful if you want to use the filter option to filter collaboration postings to see only certain types. Subtypes can be managed by administrators or created by users "on the fly" as they post.

### Managing document types

You can use document types to categorize your bulletin board postings. The ApShell application set comes with several document types by default. You can add to or remove from this list using the Manage Document Types link on the Administration page.

To manage document types

1. Log on to BPC Web with an administrator ID.
2. Click the Administration link at the top of the page.
3. Click the Manage Document Types link.
4. To add a new document type, type the name in the New text box, click the Next button, then click the Yes button to confirm the new type.
5. To delete a document type:
  - a. Select the check box(s) next to the type(s) you want to remove, then click the Next button.
  - b. On the confirmation screen, select the type the system should change the old type to for each document type that you are deleting.
  - c. Click the Yes button to confirm your changes.

### Managing document subtypes

Document subtypes help you to further categorize documents when you post them on the collaboration forum bulletin board. You can define a list of subtypes for your users to choose from or users can type their own as needed. This procedure explains how to set up a list of subtypes from which users can select when they post collaboration documents.

#### To manage document subtypes

1. Log on to BPC Web with administrator ID.
2. Click the Administration link at the top of the page.
3. Click the Manage Subtypes link.
4. To add a new subtype, enter its name in the New text box, then click the Update button.
5. To delete subtypes, select the check box(s) next to the type(s) you want to delete, then click the Update button.

#### Changing the order of document subtypes

While document types always appear in an alphabetical list, document subtypes appear in the order in which they are input into the system. You can change the order of the subtypes by using the Change Order mode of the manage subtypes function.

#### To change the order of document subtypes

1. Log on to BPC Web with administrator ID.
2. Click the Administration link at the top of the page.
3. Click the Manage Subtypes link.
4. Click the Change Order icon at the top of the Order column.
5. Select a sequence number from the drop-down next to the subtype whose order you want to change.

---

**Note:** The sequence is automatically adjusted so that the new sequence number is swapped with the subtype that had that number. For example, if the original order is 1-Apr03, 2-Aug03, and 3-DEC03, and then you change Apr03 to 3 (three) in the sequence, DEC03 gets assigned 1 (one) in the sequence.

---

## Tips and Troubleshooting

This section describes some tips for using BPC Administration, and some troubleshooting ideas.

### Microsoft Office tips and tricks

#### Excel: Autofilter

Autofilter is very useful in editing Member Sheets. By turning on autofilter, you can filter the members that are displayed based on any combination of properties. This makes it easy to find and verify information on the sheet when you are working with large member sheets.

Highlight the top row of the Member Sheet, which contains the column headings (the property names). Select Tools > Autofilter > On.

#### Excel Editing Features

##### Moving or Copying a Range of Cells

1. Highlight the range of cells you want to copy.
2. Point to the border of the selection (place the cursor on the box around the cells -- the cursor will change from a cross to an arrow).
3. To move the cells, drag the selection to the upper-left cell of the paste area. Microsoft Excel replaces any existing data in the paste area. (To copy the cells, rather than move them, hold down CTRL as you drag.)
4. To insert the cells between existing cells, hold down SHIFT (if moving) or SHIFT+CTRL (if copying) as you drag.
5. To drag the selection to a different sheet, hold down ALT and drag over a sheet tab.

##### Fill in a Series of Numbers, Dates, or Other Items

This lets you easily create new rows or columns that automatically increment. For example, if just have a column heading "Jan", and use the fill technique below, you can automatically drag the cell to create Feb-Dec in the adjacent columns. Excel intelligently increments the cell values based on whether the cells contain numbers, dates, time, etc.

1. Select the first cell in the range you want to fill, and then enter the starting value for the series.
2. To increment the series by a specified amount, select the next cell in the range and enter the next item in the series. The difference between the two starting items determines the amount by which the series is incremented.
3. Select the cell or cells that contain the starting values.
4. Drag the 'fill handle' over the range you want to fill. The fill handle is the little square in the corner of the box that highlights the selected range.
5. To fill in increasing order, drag down or to the right.
6. To fill in decreasing order, drag up or to the left.

---

**Note:** To specify the type of series, use the right mouse button to drag the fill handle over the range, and then click the appropriate command on the shortcut menu. For example, if the starting value is the date JAN-2002, click Fill Months for the series FEB-2002, MAR-2002, and so on; or click Fill Years for the series JAN-2003, JAN-2004, and so on.

---

#### Conditional Formatting in Excel

Format / Conditional Formatting can be used to vary the appearance of a cell based on a value or result of a formula. This can be used in conjunction with properties in BPC, to automatically format rows or columns in a report based on a property of the members in the row or column.

For example, if the Accounts have a style property, the Style value for each account can be retrieved into the definition area of a report using the EVPRO function in BPC for Excel. You can give an account a style value based on whether it is base level input, summary level total, or some total level in between. Conditional formatting can be used to vary the formatting of a row based on the Style value retrieved. The format will then automatically be applied to a row based on the account in that row, so that if you change the account, the format will automatically change.

### Publishing Office Documents to BPC Web

You can publish any Microsoft Office document to your BPC Web desktop. To facilitate this publishing you should set up a Webfolder that points to the application set directory on the server.

To set up a Webfolder in Windows NT:

1. From the desktop, select My Computer.
2. Select Web Folders.
3. Select Add Web Folder.
4. Enter the server name, URL or IP address for your BPC Web server, followed by the application set directory (the application set name), for example:  
`http://BPC/Server/GlobalMotors`
5. You can now reference this Webfolder location when you publish to the Web from Microsoft Office products.

### Publishing from Microsoft Word

1. Select File > Save as Web Page.

---

**Note:** If you do not have this menu option, you can need to click the vv at the bottom of the menu to display additional commands. To set you menus so they always show all available commands, use Tools / Customize / Options and un-check "Menus show recently used commands first."

---

2. Enter the desired filename. The default extension will be .htm.
3. Select Web Folders (icon on left or drop-down at top).
4. Select the Web folder that points to your BPC application set location.
5. Select the appropriate directory. You should select one of the following directories, as these are the ones that are available from within the BPC Web when you add a Web publication to your desktop:  
  
AppSetPublications - use this for publications that apply to all applications within the AppSet, that is, publications that are general to the company.  
  
[application] - use an application directory, for example BUDGET2001, for publications that are specific to an application.  
  
\_private - underneath \_private are directories by user. Each of these directories is available only to that user.
6. Select Save.

### Publishing from Microsoft PowerPoint

1. Select File > Save as Web Page.  
  
(Note, if you do not have this menu option, you can need to click the vv at the bottom of the menu to display additional commands. To set you menus so they always show all available commands, use Tools / Customize / Options and un-check "Menus show recently used commands first.")
2. Enter the desired filename. The default extension will be .htm.
3. Select Web Folders (icon on left or drop-down at top).
4. Select the Webfolder that points to your BPC application set location.
5. If you want to select special PowerPoint publishing options, such as publishing only selected slides and other Web options, select Publish..., set the desired options, and then press Publish.
6. To save the entire PowerPoint file to the Web without special options, select the appropriate directory and select Save. You should select one of the following directories, as these are the ones that are available from within the BPC Web when you add a Web publication to your desktop:

AppSetPublications - use this for publications that apply to all applications within the AppSet, that is, publications that are general to the company.

[application] - use an application directory, for example BUDGET2001, for publications that are specific to an application.

[site] – use a site directory, for example HQ, for publications that are specific to a site.

\_private - underneath \_private are directories by user. Each of these directories is available only to that user.

## Setting client options

Client options are available for maintenance purposes. You can perform the following client maintenance tasks:

- Reset the current view bar
- Clear local application information
- Refresh dimensions
- Updating wizard templates

For information on setting member selector options, see Using the Member Selector.

### Resetting the current view bar

You can reset the current view bar so that for each dimension, the top hierarchical level member is displayed and the list of recently accessed members is cleared.

1. From BPC for Excel, select eTools > Client Options.
2. Click Clear current view bar.
3. When prompted to confirm, click Yes, then click OK.
4. Click Close.

### Clearing local application information

You can clear the files associated with an application from your client. After you perform this procedure, the next time you log on to this BPC for Excel application, you are prompted to download a new AppSet.

#### To clear local application information

1. From BPC for Excel, select eTools > Client Options.
2. Click Clear Local Application Information.
3. When prompted to confirm, click Yes, then click OK to close and restart BPC for Excel.

### Refreshing dimensions

You can refresh the dimension information saved on your client with the dimension and member information saved on the server.

#### To refresh your dimensions

1. From BPC for Excel, select eTools > Refresh dimension members.
2. In the Update complete message box, click OK.

### Updating wizard templates

You can update the wizard templates saved on your client with the templates saved on the server.

#### To update your wizard templates

1. From BPC for Excel, select eTools > Client Options.
2. Click the Refresh Wizard Templates button.
3. In the Update complete message box, click OK.

### Changing the local BPC folder

By default, the BPC client uses the MS Windows-defined MY FOLDER as the location in which to store cached files. If this folder is not in an optimal place for this function (on a remote server, for instance), you may wish to change the location.

To change the local BPC folder

1. From Business Planning and Consolidation for Excel, select eTools > Client Options.
2. Click the Set local folder for BPC button.
3. In the Local Folder message box, enter the path you'd like to use and click OK.

A message box appears, indicating that your changes take effect at your next logon.

### Updating application set information

When you log on to an application set whose structure has changed, you are prompted to update your client information. Structure changes that require updates are changes to dimensions and templates stored on the server.

You can also initiate this procedure manually during a BPC for Excel session.

---

**Note:** We recommend doing this only if instructed to do so by your administrator.

---

#### Refreshing dimensions

You can refresh the dimension information saved on your client with the dimension and member information saved on the server.

To refresh your dimensions

1. From BPC for Excel, select eTools > Refresh dimension members.
2. In the Update complete message box, click OK.

#### Refreshing templates

You can refresh the wizard templates saved on your client with the templates saved on the server.

To refresh your templates

1. From BPC for Excel, select eTools > Client Options.
2. Click the Refresh Wizard Templates button.
3. In the Update complete message box, click OK.

### Troubleshooting

This topic contains solutions to some common problems that might occur when using BPC. While this list is meant to capture as many issues/problems as possible, it is only updated with each new release of BPC. If you can not find an answer to your problem please contact BPC Support.

Issue/Problem	Solution
Protected VBA code in a workbook does not work when taking it offline using Park N Go.	<ol style="list-style-type: none"> <li>1) Open a workbook that has protected VBA code. (This can be an existing report or input schedule, or a template that users will access from the report or input schedule library or wizard in BPC for Excel.)</li> <li>2) Open the VBA editor and unprotect the code using the appropriate password.</li> <li>3) Using Park N Go, take the workbook offline, then online again.</li> <li>4) From the workbook, open the VBA editor and delete the hidden sheets.</li> <li>5) Protect the VBA code by entering the appropriate password.</li> </ol>
You cannot access an	You may not have security access to the application or your



Issue/Problem	Solution
application.	security profile for the application might be incomplete. Contact your application administrator to request access, or to check your profile for errors.
Your report and input schedule libraries do not open. An Internet Explorer browser opens instead.	The problem may be with your MS Office installation. Uninstall and reinstall Excel and Internet Explorer. Make sure the Webfolder support check box is selected on the reinstall.
You try to send data to the database and the data submission fails.	Check to make sure the current view is set to base level members only. Or check the work status of the entity for which you are submitting data. The status must be set to <i>Unlocked</i> in order to submit data to the database.
You cannot view a dimension member.	You may not have security access to view the dimension member. Check with your application administrator for more information on your security rights.
The eCollab Help item is disabled in the eCollab menu.	In order to view the eCollab Help item, you must connect to a server through the Connection Wizard when logging on to BPC for Excel.  If you need help with eCollab, see eCollab Help.

Error message	Solution
"Could not load object because it is not available on this machine."	The problem might be with your MS Office installation, and you may need to repair it. You can do this by accessing the Windows Control Panel, selecting Add/Remove programs, and selecting your MS office installation. Then select change and choose to repair your installation when asked by the wizard.
"OLE DB Provider for Analysis Services. The Operation requested failed due to Security problems - resources access error - access is denied."	You do not have security access to the application set. Contact your application administrator to request security access.
"Cannot log on to Analysis Server. It can be due to Network or Security Problems."	You do not have security access to the application set. Contact your application administrator to request security access.
"Run Time Error 1004"	Your application database needs to be processed. Contact your application administrator.

## Appendix A: Security Management in BPC

Security in BPC has been significantly improved since BPC 4.2x. The new security model allows for more flexibility and easier maintenance. One of the changes we have implemented is the ability to assign task profiles and member access profiles to users. The combination of these two profile types defines the way an individual can access and use different areas of BPC.

When BPC is first installed, by default, only the system administrator can perform administrative tasks. This means that no other users have access to tasks or secured dimension members until they are explicitly assigned.

### General rules for security profiles

BPC security is based on the following rules:

- By default, other than the System Administrator, no one has access. Access must be explicitly granted.
- A user can be assigned access individually and through team membership.
- Member access privileges flow down the hierarchy, from parent to child.
- When in conflict, the least restrictive member access profile is applied.
- In case of a conflict between individual and team member access, individual security overrules the security defined for a team to which the individual belongs.
- Denial of member access can only be set at the user level.

### About task profiles

The ability to assign task profiles is a new security feature in BPC. Task profiles determine what type of activity or role the user can perform in BPC. For your convenience, two default task profiles are available, they are: System Admin and Primary Admin. Depending on your requirements, both of these task profiles can be modified.

Any task profile that either already exists, or is newly created, can have the following default roles associated with them: System Admin, Primary Admin, and Secondary Admin.

### Tips for assigning task profiles

- The number of task profiles administrators can assign to a user is not limited. However, we recommend that you do not assign multiple task profiles to users because it may cause confusion in determining their ultimate access rights.

Task access security is cumulative, and tasks cannot be explicitly denied. As a result, assigning multiple task profiles can create a situation where users have access to tasks that you may not want them to have. For example, an administrator wants UserA to only retrieve data. If UserA belongs to a team that has data-send task rights, UserA can also send data.

- Administrators can assign multiple task profiles to a team. However, we recommend that you do not assign multiple task profiles to a team because it may cause confusion in determining the ultimate access rights of that team. This will be addressed in a future release so administrators will not be able to assign multiple task profiles to teams.

### About member access profiles

Member Access profiles determine the specific applications to which users have Read access, Read and Write access or 'Denied' (no access). In addition to granting or restricting access by application, you can grant or restrict access by dimension and member.

When defining access to a secured dimension that has one or more hierarchies defined, security is applied to the member and all its children. For example, if you grant access to SALESUS, users with access to SALESUS will also have access to all children of SALESUS.

You can restrict a child member of a parent with 'Read' or 'Read and Write' access by creating a separate member access profile and assigning the child 'Denied' access.

### Resolving member access profile conflicts

As you can define member access by individual users and by teams, there may be situations in which conflicts occur. This section describes some potential member access conflict scenarios and the rules the

system applies to resolve those conflicts. These scenarios are based on the assumption that the Entity dimension is a secured dimension and has the following hierarchical structure.

Hierarchy	Level			
	Level1	Level2	Level3	Level4
H1	WorldWide1	Sales	SalesAsia	SalesKorea SalesJapan ESalesAsia
			SalesEurope	SalesItaly SalesFrance ESalesEurope
H2	WorldWide2	Asia	Korea	SalesKorea
			Japan	SalesJapan
			eAsia	ESalesAsia
		Europe	Italy	SalesItaly
			France	SalesFrance
			eEurope	ESalesEurope

### Conflict between profiles

When there is a conflict between member access profiles, the least restrictive profile is always applied. This section describes three different scenarios where there are conflicts between profiles.

#### Scenario 1

- User1 belongs to Team1 and Team2.
- There are two member access profiles: ProfileA and ProfileB.
- ProfileA is assigned to Team1 and ProfileB is assigned to Team2.

The member access profiles are described in the table below.

Member access profile	Access	Dimension	Member
ProfileA	Read & Write	Entity	Sales
ProfileB	Read Only	Entity	SalesAsia

In this case, the least restrictive profile between the two, ProfileA (Read & Write), will be applied.

As a result, ProfileB will be ignored by the system, and User1 will be able to send data to both SalesKorea and SalesItaly.

#### Scenario 2

- User1 belongs to Team1 and Team2.
- There are two member access profiles: ProfileA and ProfileB.
- ProfileA is assigned to Team1 and ProfileB is assigned to Team2.

The member access profiles are described in the table below.

Member access profile	Access	Dimension	Member
ProfileA	Read Only	Entity	Sales
ProfileB	Read & Write	Entity	SalesAsia

In this case, the least restrictive profile between the two, ProfileB (Read & Write), will be applied for the child members of SalesAsia.

As a result, ProfileA will be ignored by the system, and User1 will be able to send data to SalesKorea, but not to SalesItaly.

#### Scenario 3

- User1 does not belong to any team.
- There are two member access profiles: ProfileA and ProfileB.
- Both the profiles are assigned to the user.

The member access profiles are described in the table below.

Member access profile	Access	Dimension	Member
ProfileA	Denied	Entity	SalesAsia
ProfileB	Read Only	Entity	Sales

In this case, the least restrictive profiles between the two, ProfileB (Read Only), will be applied.

As a result, ProfileA will be ignored by the system, and User1 will be able to retrieve data from both SalesKorea and SalesItaly.

#### Conflict between parent and child members

Authority always flows down the hierarchy, from parent to child. Child members always have the access level of their parents, unless otherwise specified.

#### Scenario 1

- User1 belongs to Team1 and ProfileA is assigned to Team1.
- Two levels of member access profiles are defined for ProfileA.

The member access profiles for the ProfileA are described in the table below.

Member access profile	Access	Dimension	Member
ProfileA	Read & Write	Entity	Sales
ProfileA	Read Only	Entity	SalesAsia

In this case, the Read & Write access of the Sales member flows down to its children. This flow is interrupted by assigning Read Only access to SalesAsia (a descendant of Sales), and SalesAsia's access flows down to its descendants.

As a result, User1 will be able to send data to SalesItaly, but not to SalesKorea.

#### Scenario 2

- User1 belongs to Team1 and ProfileA is assigned to Team1.
- ProfileA has two levels of member access profiles.

The member access profiles for the ProfileA are described in the table below.

Member access profile	Access	Dimension	Member
ProfileA	Read Only	Entity	Sales
ProfileA	Read & Write	Entity	SalesAsia

In this case, the Read Only access of the Sales member flows down to its children. This flow is interrupted by assigning Read Only access to SalesAsia (a descendant of Sales), and SalesAsia's access flows down to its descendants.

As a result, User1 will be able to send data to SalesKorea but not to SalesItaly.

#### Conflict between teams and individual users

When there is a conflict between individual user and team access, the individual user's access level overrules the security defined for a team that the individual belongs to.

## Scenario 1

- User1 belongs to Team1.
- There are two member access profiles: ProfileA and ProfileB.
- ProfileA is assigned to User1 and ProfileB is assigned to Team1.

The member access profiles are described in the table below.

Member access profile	Access	Dimension	Member
ProfileA	Read Only	Entity	Sales
ProfileB	Read & Write	Entity	SalesAsia

In this case, the user's profile, ProfileA will determine User1's access.

As a result, even if User1 belongs to Team1 (ProfileB) and the team members have data-send access to SalesAsia member, User1 will not be able to send data to either SalesKorea or SalesItaly.

## Scenario 2

- User1 belongs to Team1.
- There are two member access profiles: ProfileA and ProfileB.
- ProfileA is assigned to User1 and ProfileB is assigned to Team1.

The member access profiles are described in the table below.

Member access profile	Access	Dimension	Member
ProfileA	Denied	Entity	SalesAsia
ProfileB	Read Only	Entity	Sales

In this case, the user's profile, ProfileA will be applied.

As a result, even if User1 belongs to Team1 (ProfileB) and the team has data-retrieval access for the Sales member, User1 will be able to retrieve data from SalesItaly but not from SalesKorea.

## Conflict when the same member belongs to different hierarchies

When a member belongs to different hierarchies, and there is a conflict in member access, currently the most restrictive access is applied.

## Scenario 1

- ProfileA and ProfileB are assigned to User1.

The member access profiles are described in the table below.

Member access profile	Access	Dimension	Member
ProfileA	Read Only	Entity	WorldWide1
ProfileB	Read & Write	Entity	WorldWide2

In this case, ProfileB will determine User1's access.

As a result, User1 will be able to send data to SalesKorea, even if ProfileA denies User1 Write access to SalesKorea (in WorldWide1 hierarchy).

## Appendix B: Best Practices for Writing Logic

The BPC modeling logic engine has proven over time to be a very flexible and effective tool to perform all sorts of calculations within our database, delivering the desired results efficiently and accurately.

In many situations, however, this result has been obtained only if the logic had been written “correctly,” a condition that sometimes was not easy to achieve. In many cases we have been forced to review (and possibly rewrite) what had been originally written by our customers (or partners or consultants), because, while returning the correct values, the logic would simply not perform as desired in its first incarnation.

There are several reasons for this. The logic syntax is not easy to grasp at first, and the way the engine works (especially with the SQL-based instructions) is not very intuitive. Moreover, the flexibility we have built into the engine results in a multitude of different ways to skin the same cat, and some of these ways are better (or worse) than others, performance-wise.

In addition, there has been only limited training available on this subject for a long time, and the knowledge has not been spread around as it should have.

We have tried to make logic editing and debugging easier with the introduction of the Logic Assistant and the Logic Debugger in 4.2, but these tools do not help much if you do not know how to structure logic in the first place (more or less like trying to learn a language by reading a dictionary).

While a good knowledge of how to write efficient logic may only come from experience, with this document, we are trying to pass along many of the lessons we learned the hard way, while trying to help our users write more efficient logic.

A fair knowledge of our logic syntax and basic constructs is assumed. This document is not intended for beginners.

### The Golden Rules

Typically, the recommendations we give administrators when trying to optimize logic can be summarized into ten Golden Rules:

1. Stay away from MDX logic
2. Load in memory only the required data
3. Carefully select the “triggers” of your calculations
4. Keep the logic structure as compact as possible
5. Minimize the number of COMMITs
6. Minimize the number of GO statements
7. Keep in default logic only the calculations that are absolutely required to be performed in real time
8. Carefully check what the LOG files say
9. Avoid refresh after send in Excel
10. Run a stress test before going live

### Rule 1: Stay away from MDX logic

MDX logic has some appealing advantages: (1) the syntax is, in most cases, fairly intuitive and easy to master, and (2) it lets you access parent-level and other calculated values not stored in SQL.

Unfortunately, our experience has been that MDX queries may easily deliver very poor performance and do not scale well. In practically all cases where we replaced MDX logic with SQL logic, the improvements have been substantial, and the switch is definitely worth the effort. The latest versions of SQL logic provide methods to access parent-level

values and there are very few things (if any) written in MDX syntax that SQL logic cannot calculate. MDX formulas should only be used in "dimension formula members," and these should only be limited to ratio-analysis accounts, where the formula must be applied at all levels in the cube dimensions.

#### Accessing parent member values in SQL logic

Parent values are not natively stored in the fact tables, and as such, are not immediately available for SQL logic calculations. However our SQL logic syntax provides a set of instructions that can be used to generate such values in memory, and use them as input for other calculations. For the benefit of those who have not read the product documentation, here is a refresher:

```
// make sure you have all entities in memory
*XDIM_MEMBERSET ENTITY=<ALL>

// generate all parent values
*CALC_DUMMY_ORG ENTITY=PARENTH1

// use them as appropriate (note the # sign)
*WHEN ENTITY
    *IS #SALESEUROPE
        *REC(FACTOR=1/GET(ENTITY=#WORLDWIDE1), ACCOUNT="SomeRatio")
*ENDWHEN
```

Note that a CALC\_DUMMY\_ORG assumes an implied GO statement before the next WHEN section. Inserting a GO before the WHEN statement will do no harm, but it is redundant.

#### Net Profit to Balance Sheet

Another classical example of a calculation that can now be performed in SQL logic without recurring to MDX expressions is the carry-forward of net profit from the income statements into the balance sheet. The sample shown below demonstrates how to put together various techniques to (1) calculate parent values and (2) derive YTD values in SQL logic. Note in particular how the following features are used:

- The CALC\_EACH\_PERIOD instruction
- The CALC\_DUMMY\_ORG instruction
- The PRIOR keyword
- Memory variables (like #OPE\_CYN1)



```
//----- Net Income to BS
*CALC_EACH_PERIOD

*XDIM_MEMBERSET TIME=PRIOR,%TIME_SET%,%YEAR%.DEC

*CALC_DUMMY_ORG ACCOUNT=PARENTH1

*WHEN TIME
*IS PRIOR
  *WHEN ACCOUNT
  *IS CYNI
    *REC(ACCOUNT=#OPE_CYNI,TIME=NEXT)
  *ENDWHEN
*ELSE
  *WHEN ACCOUNT
  *IS #NETINCOME
    *REC(ACCOUNT=#OPE_CYNI,TIME=NEXT)
    *REC(ACCOUNT=CYNI)
  *IS #OPE_CYNI
    *WHEN TIME.PERIOD
    *IS<>JAN
      *REC(ACCOUNT=#OPE_CYNI,TIME=NEXT)
      *REC(ACCOUNT=CYNI)
    *ENDWHEN
  *ENDWHEN
*ENDWHEN
//-----
```

Refer to the product documentation for other advanced ways to use the above mentioned features.

## Allocations

Very often the driver of an allocation is a value (like Sales or SquareFootage or Employees) that needs to be compared with its total over a given region. For example, the allocation formula for LeaseCost could look as follows:

Allocated LeaseCost = LeaseCost \* SquareFootage / Total Square\_Footage

The challenge here is the need to derive the Total Square\_Footage, which is the sum of SquareFootage across all ENTITIES. While the temptation to write an (indeed quite simple) MDX formula can be hard to resist, you have plenty of ways to obtain the desired result without paying the performance penalty of MDX.

Here are some:

#### Approach 1: Use CALC\_DUMMY\_ORG

```
*XDIM_MEMBERSET ENTITY=<ALL>

*CALC_DUMMY_ORG ENTITY=PARENTH1

*WHEN ACCOUNT
*IS SQUAREFOOTAGE
  *REC(FACTOR=GET(ACCOUNT="LEASECOST", ENTITY="GLOBALOPS") /
  GET(ENTITY="#ALL_ENTITIES"), ACCOUNT="LEASECOST")
*ENDWHEN
```

#### Approach 2: Use a WHEN/ENDWHEN

```
*XDIM_MEMBERSET ENTITY=<ALL>

*WHEN ACCOUNT
*IS SQUAREFOOTAGE
  *REC(ENTITY="#ALL_ENTITIES")
*ENDWHEN

*GO

*WHEN ACCOUNT
*IS SQUAREFOOTAGE
  *REC(FACTOR=GET(ACCOUNT="LEASECOST", ENTITY="GLOBALOPS") /
  GET(ENTITY="#ALL_ENTITIES"), ACCOUNT="LEASECOST")
*ENDWHEN
```

#### Approach 3: Use the latest allocation instructions

```
* RUN_ALLOCATION
  *FACTOR USING/TOTAL
  *DIM ACCOUNT   WHAT= LeaseCost;           WHERE=<<<;      USING=
SquareFootage;   TOTAL= <<<
  *DIM ENTITY    WHAT= GLOBALOPS;           WHERE=>>>;      USING=
>>>;            TOTAL= BAS(ALL_ENTITIES)
*ENDALLOCATION
```

## Rule 2: Load in memory only the required data

### Why it's important

The way SQL-based logic works needs to be clearly understood. Once the mechanism is clear, how to apply it to your benefit becomes fairly straightforward.

The logic engine pulls into memory all records existing in the fact tables for the data region that has been specified either by the calling object (BPC for Excel or Data Manager) or by some instructions specified in the logic itself (for example, through the XDIM\_MEMBERSET instruction).

When the appropriate array of records is in memory, it is scanned sequentially from the first to the last one, in whatever order they come in, and the logic is applied to each record. If an individual record meets some user-defined criteria (\*WHEN something \*IS some value), some other record will be generated (generate \*REC such and such).

In contrast to MDX logic, SQL logic will only be able to access values that have been loaded in memory. For example, any GET statement pointing to something which is not in memory will return zero, even if the value exists in the database.

The first performance consequence of this mechanism is that, while you need to make sure that all required records are in memory, it is equally important to load in memory

as few records as possible, otherwise the time to load the records and scan them one by one may become unacceptably long.

For example, assume you need to perform only this calculation:

Account A = Account B + Account C

The logic to achieve this will look as follows:

```
*WHEN ACCOUNT
*IS B,C                                // if you find values
for these accounts...
  *REC(ACCOUNT=A)                      //... add them into account A
*ENDWHEN
```

This logic would, by default, load in memory all possible accounts. However, if this is the only calculation to perform, there is no need to load in memory all accounts, and the logic will run faster if it is written this way:

```
*XDIM_MEMBERSSET ACCOUNT=B, C

*WHEN *
*IS *
  *REC(ACCOUNT=A)
*ENDWHEN
```

(In this extreme case there is not even the need to test for any criteria, since all accounts in memory have the same behavior.)

#### Running logic from BPC for Excel or Data Manager

Speaking of the data region to load in memory, the following rule must be remembered: there is a difference in the way the region to process is passed to the logic engine when called by BPC for Excel and when called by a package in Data Manager.

BPC for Excel by default passes to the default logic the list of all members that have been sent to the DB for ALL dimensions EXCEPT the ACCOUNT dimension. This means that the default logic, if not instructed differently, will scan ALL accounts but only for the members of the Entity, Category, Period, Intco, Datasrc, etc. dimensions that have been modified via a send.

On the other hand, Data Manager will only pass to the logic engine the relevant members for the dimensions that are defined in the calling package via some hard-coded definition or via a run-time prompt to the user. For all other non-specified dimensions, the logic engine will assume ALL members (with the exception of the CURRENCY dimension, for which it will default to the LC member).

Knowing the above rule is important, as this may obviously influence the scope of the region being processed. We have had cases of people reporting that default logic would give different results when called from a data send in Excel or from Data Manager. The truth is that the logic behavior does not change. What may change is the scope of the data that it is instructed to process.

### Rule 3: Carefully select the "triggers" of your calculations

The fundamental principle around which SQL logic is based is that our database is sparse. This means that, while the potential number of cells in a given cube could be overwhelming, the actual number of cells containing a value is much, much smaller. As a result, while a given formula could apply to (say) a million cells, most of these cells are empty and applying that formula to all such empty cells would be a waste of time. So our approach is to first look at the (few) cells having a stored value; then we decide what formulas to execute. Here is an example.

Assume you want to calculate REVENUE = UNITS \* PRICE. The SQL logic expression for this could be written as follows:

```
*WHEN ACCOUNT
*IS PRICE
*REC(FACTOR=GET(ACCOUNT="UNITS"), ACCOUNT=REVENUE)
*ENDWHEN
```

However, your application may contain the price of thousands of different products, while only a few products are sold in a given period. So, the above logic will find many PRICE accounts that will be multiplied by UNITS accounts containing no value, simply returning a result of zero. A more correct structure of the logic would use the UNITS as a trigger, as follows:

```
*WHEN ACCOUNT
*IS UNITS
*REC(FACTOR=GET(ACCOUNT="PRICE"), ACCOUNT=REVENUE)
*ENDWHEN
```

In reality the above example is not the most common case of the use of an incorrect trigger, and is not even the worst. There are situations which lead to extremely slow executions, and they are actually fairly easy to spot. Quite often these logics contain the NOADD keyword, and/or REC statements containing an EXPRESSION which does not reference the current %VALUE%. Here is a simple example:

Assume that account B must be equal to account A, in case the category is ACTUAL. Some administrators would write a logic script like this:

```
*WHEN CATEGORY
*IS ACTUAL
*REC(EXPRESSION=GET(ACCOUNT="A"), ACCOUNT="B", NOADD)
*ENDWHEN
```

However this is clearly a poorly written logic: the logic is scanning many records belonging to category ACTUAL, so they all meet the WHEN criteria, and for each of them the value of A will be added to B. As a result, the administrator has been forced to use the NOADD keyword, to make sure account B receives the value of account A only once. (And besides, this is done even if A is null.)

Such a logic script may easily turn out being 100 (yes, 100!) times slower than this (more correct) one:

```
*WHEN ACCOUNT
*IS A
*WHEN CATEGORY
*IS ACTUAL
*REC(ACCOUNT="B")
*ENDWHEN
*ENDWHEN
```

#### Rule 4: Keep the logic structure as compact as possible

When the logic scans the records one by one trying to find if some calculation must be applied to them, it may need to go through the entire list of criteria defined in the logic to find the applicable case. For example, the logic may contain a long list of WHEN / ENDWHEN structures, more or less as follows:

```

*WHEN ACCOUNT
*IS A
    *REC (ACCOUNT=X)
*ENDWHEN
*WHEN ACCOUNT
*IS B
    *REC (ACCOUNT=X)
*ENDWHEN
*WHEN ACCOUNT
*IS C
    *REC (ACCOUNT=Y)
*ENDWHEN
*WHEN ACCOUNT
*IS D
    *REC (ACCOUNT=Y)
*ENDWHEN

```

In this example, the criteria is always the same: we are always testing for the ID of an ACCOUNT. If this is the case, the first thing to do is to turn the multiple WHEN / ENDWHEN structures into only one, as follows:

```

*WHEN ACCOUNT
*IS A
    *REC (ACCOUNT=X)
*IS B
    *REC (ACCOUNT=X)
*IS C
    *REC (ACCOUNT=Y)
*IS D
    *REC (ACCOUNT=Y)
*ENDWHEN

```

This will not only reduce the number of instructions the engine has to process, but it will make sure the remaining instructions in the structure are skipped, once an instruction meeting the criteria for the current record is found. For example, if the account in the current record is A, all lines starting from \*IS B to \*ENDWHEN will be skipped.

Another way to reduce the size of this logic is to combine the instructions for the accounts that have the same behavior, i.e., those which share similar REC statements. In this case the accounts A and B add up into the same account X, and the accounts C and D add up into the same account Y, so the logic could be written like this:

```

*WHEN ACCOUNT
*IS A,B
    *REC (ACCOUNT=X)
*IS C,D
    *REC (ACCOUNT=Y)
*ENDWHEN

```

Finally, should the logic still be very long, it might be worth making use of properties, to define what account must be added to what account in a more condensed way. While in the current simple example this may be overkill, our logic might be further reduced by defining a property SOMEPROPERTY where, for the appropriate accounts, you enter the ID of the destination account as follows:

	A	B
1	ID	SOMEPROPERTY
2	A	X
3	B	X
4	C	Y
5	D	Y

6	E	
7	F	
8	G	

This will have the benefit of reducing the length of the logic to the bare minimum, while simplifying its maintenance, as any new account will be directly controlled by the value of SOMEPROPERTY, avoiding the need to modify the logic. No matter how many accounts are involved in the process, the final logic will simply say:

```
*WHEN ACCOUNT.SOMEPROPERTY
*IS<>" " // i.e. different from blank
  *REC(ACCOUNT=ACCOUNT.SOMEPROPERTY)
*ENDWHEN
```

Another case of inefficient logic structure is the way some WHEN / ENDWHEN structures are nested. For example, take the following case:

```
*WHEN ACCOUNT
*IS A
  *WHEN INTCO
  *IS NONINTCO
    *REC(ACCOUNT=X)
  *ENDWHEN
*IS B
  *WHEN INTCO
  *IS NONINTCO
    *REC(ACCOUNT=Y)
  *ENDWHEN
*IS C
  *WHEN INTCO
  *IS NONINTCO
    *REC(ACCOUNT=Z)
  *ENDWHEN
*ENDWHEN
```

Here, several accounts generate a calculation only when the INTCO member is NONINTCO. The above structure may end up testing several account IDs before finding that the current INTCO member is not NONINTCO and the entire process might have been skipped right away. A more efficient way of writing this logic would be:

```
*WHEN INTCO
*IS NONINTCO
*WHEN ACCOUNT
*IS A
  *REC(ACCOUNT=X)
*IS B
  *REC(ACCOUNT=Y)
*IS C
  *REC(ACCOUNT=Z)
*ENDWHEN
*ENDWHEN
```

## Rule 5: Minimize the number of COMMITs

Quite often an entire logic cannot be executed in one single step, as different portions of the calculations may require different sets of input data to be loaded in memory.

As a result, logic is normally broken into separate COMMIT sections that get executed in sequence by the logic engine.

Each COMMIT section in logic triggers the following set of actions:

- Some input records are read from the db, issuing one or more SQL queries, and pulled into memory
- The input records are scanned one by one and some new records are generated
- The new records are sent to the db

These actions, if performed several times, make the logic slower, and it is a proven fact that logic broken into many small COMMIT sections will run much slower than a corresponding logic where all data are loaded, calculated and written at once, using one single COMMIT section. For this reason, while reducing an entire logic into one single COMMIT may not be practically feasible, it is standard practice to try and merge as many different COMMIT sections into as few as possible.

Here are a few examples.

#### Example 1

Assume you have some accounts that need to be calculated for  $INTCO = NONINTCO$ , and some others must be calculated for  $INTCO \neq NONINTCO$ .

This may be written splitting the logic in two COMMIT sections:

```
*XDIM_MEMBERSET INTCO=NONINTCO           // load only NONINTCO

*WHEN ACCOUNT
*IS A,B,C
  *REC(...)
*ENDWHEN

*COMMIT

*XDIM_MEMBERSET INTCO<>NONINTCO           // load all other INTCO
members

*WHEN ACCOUNT
*IS X,Y,Z
  *REC(...)
*ENDWHEN
```

The same logic will, however, run faster if the two COMMIT sections are merged into one as follows:

```
*XDIM_MEMBERSET INTCO=<ALL>               // load all INTCO members at
once

*WHEN INTCO
*IS NONINTCO
*WHEN ACCOUNT
*IS A,B,C
  *REC(...)
*ELSE
*WHEN ACCOUNT
*IS X,Y,Z
  *REC(...)
*ENDWHEN
*ENDWHEN
```

The tradeoff here is to load more data in one single pass and add a WHEN evaluation in the body of the instructions.

## Example 2

Assume you have logic that calculates these accounts:

```
REVENUE = UNITS * PRICE
VAT = VAT_RATE * REVENUE / 100
```

Since the result of the first calculation (REVENUE) is to be used as input for the second calculation, you need to split the logic in two steps. One way to achieve this is to separate the steps with a COMMIT instruction as follows:

```
*WHEN ACCOUNT
*IS UNITS
  *REC(FACTOR=GET(ACCOUNT="PRICE"), ACCOUNT=REVENUE)
*ENDWHEN

*COMMIT

*WHEN ACCOUNT
*IS REVENUE
  *REC(FACTOR=GET(ACCOUNT="VAT_RATE") / 100, ACCOUNT=VAT)
*ENDWHEN
```

By so doing, the latest value of the REVENUE account will be available for reading in the second part of the logic.

This technique however will make the logic run slower. A better solution is to replace the \*COMMIT instruction with a \*GO instruction as follows:

```
*WHEN ACCOUNT
*IS UNITS
  *REC(FACTOR=GET(ACCOUNT="PRICE"), ACCOUNT=REVENUE)
*ENDWHEN

*GO

*WHEN ACCOUNT
*IS REVENUE
  *REC(FACTOR=GET(ACCOUNT="VAT_RATE") / 100, ACCOUNT=VAT)
*ENDWHEN
```

Any GO statement will simply break the logic in separate sections, basically similar to a COMMIT statement, but with the fundamental difference that the results are not posted to the database but simply appended to the set of records existing in memory. After this, no new query is issued, and the subsequent GO section of instructions is applied to the same set of records held in memory, now containing also the results of the prior GO section. At the end of the process all values calculated by all the GO sections combined are written to the database in one single action.

## Example 3

Another situation where the result of a prior calculation must be used as inputs for a subsequent calculation is whenever the calculated closing balances of an account need to be used as opening balances for the next period calculation. In earlier versions of our product this mechanism could be activated using the instruction:

```
*PROCESS_EACH_MEMBER=TIME
```

This instruction would perform a logic execution (read / calculate / write) for each selected member of the defined dimension. If the dimension happened to be the TIME dimension, the execution would also automatically be performed in the correct sequence (from the past to the future) and also fill any gaps (for example, selecting JAN and MAR would automatically include FEB).

While this instruction would do the job, it would split the logic execution in multiple COMMIT actions, leading to slower performance.

Today this instruction can, in most situations, be replaced with a \*CALC\_EACH\_PERIOD, which is similar in nature, but calculates all periods in memory before writing all results at once to the database. In this respect, a CALC\_EACH\_PERIOD behaves like the GO



instruction, while the older PROCESS\_EACH\_MEMBER behaves like a COMMIT instruction.

---

**Note:** The efforts of reducing the number of COMMIT sections may force the user to violate rule number 2 (load in memory as few data as possible). In general, loading less data in memory is not as effective a performance improvement as reducing the number of COMMITs. In most cases the logic will run faster even if loading more data in memory, thanks to the benefit of having fewer COMMIT sections in the logic itself. This trade-off must be evaluated on a case-by-case basis.

---

## Rule 6: Minimize the number of GO statements

While using GO statements in place of COMMIT statements will definitely make the logic run faster, this is still not the ultimate solution in logic speed. After all, even if data are not re-loaded in memory, they are scanned multiple times (essentially once per GO statement), and this adds to the time of logic execution. While it may not always be possible, it is wise to avoid using too many GO statements within the same COMMIT section.

In the example shown above, a logic structure that runs faster will be one that calculates everything at once, within the same pass, avoiding the use of any COMMIT or GO statement. One way of writing it could be as follows:

```
*WHEN ACCOUNT
*IS UNITS
  *REC(FACTOR=GET(ACCOUNT="PRICE"), ACCOUNT=REVENUE)
  *REC(FACTOR= GET(ACCOUNT="PRICE") * GET(ACCOUNT="VAT_RATE") / 100,
ACCOUNT=VAT)
*ENDWHEN
```

## Rule 7: Keep in default logic only the calculations that are absolutely required to be performed in real time

No matter what you do, the time of execution of the default logic when data are entered from BPC for Excel may remain unacceptably slow. If this is the case, the only solution is to try and reduce the amount of data that is calculated and stored in real time during the data entry process. This depends very much on what you need to see immediately after entering the number in a schedule. Whenever possible, the generation of all calculated values that do not need to be shown as immediate feedback to the user should be postponed to a more appropriate time.

For example, the execution of the currency conversion may only need to be executed in a batch mode, while the input schedule only shows the values in local currency. Similarly, any inter-company elimination or allocation, which can only be executed when values from all entities have been entered (and translated into the reporting currency), can be delayed to a later phase.

Along these lines, some customers use the following approach: their default logic only contains some instructions which flag an account with a value of 1, to indicate that "This entity has been modified. The logic is to be executed on it." A separate process, triggered manually or scheduled to run every so many minutes (or even seconds), looks for the entities that need to be processed, and runs the appropriate logic on them, setting their flag account back to zero ("calculated") at the end of the process.

## Rule 8: Always review the LOG files

It is certainly redundant to mention that any logic should be carefully tested before releasing it in a production environment. What people sometimes do not realize, however, is the importance of analyzing the content of the log file. The log file is critical to figuring out what went wrong during a logic execution.

The log files generated by a logic execution contain an incredible amount of information, ranging from some statistical details (like the date and time of execution, the version of the DLL, the duration of the execution of each single step, etc.), to more specific

information (like the region passed to the logic, the queries used to read or calculate the values, the results of the calculations, etc.).

The log file is the starting point of any investigation concerning the behavior (or misbehavior) of any logic. For this reason, a copy of the log file should always be submitted to the support team, together with the LGF and LGX files of the logic, whenever an issue with logic is reported.

Moreover, while the log file is critical to fixing logic, it is also a best practice to review the log file when your logic runs without errors. This will ensure you understand the performance profile of your logic, and might point out areas for improvement (or simply solidify your understanding of well-written logic).

### Rule 9: Avoid refresh-after-send in Excel

In several situations, what is shown in the data entry schedules as a calculated value is only limited to some total (like the value of a parent member) or some validated amount (like total assets minus total liabilities).

In these cases, it may be appropriate to consider the possibility to perform such calculations directly in Excel (using Excel formulas), even if this is a repetition of what gets calculated in the db behind the scenes. This has the double advantage that (a) values are immediately updated on the screen as users type in their numbers, and (b) there is no need to perform a refresh of the workbook after every send as no calculated data will be retrieved. This will make the data entry process much lighter and faster.

### Rule 10: Run a stress test before going live

This too should go without saying, but we have experienced a very high number of cases where people are caught by surprise when some logic, which seemed to perform very efficiently in a development environment, became a real show stopper when put under the stress of a production environment, with many concurrent users running it simultaneously over large amounts of data.

While this has proven particularly true in case of MDX logic, to some extent the same situation has been experienced with SQL logic. In the latter case, it may not be SQL itself not scaling well, but it is the response time of the OLAP queries issued by the input schedules or reports being refreshed by many users while many data are sent to the database by many logic instances. This may expand beyond the most conservative expectations, as a result of the large amount of locks being applied to the cube by OLAP.

For this reason, we recommend that you allocate time to perform an adequate stress test on the application in the implementation plan, well before the moment to go live arrives.

## Appendix C: Script logic documentation updates

The following information is not included in the Administration documentation at this time, but will be included in a future release.

### Ability to control SQL time-out

The following new instruction can be used to modify the time out limit to SQL queries (currently defaulting to 600 seconds):

```
*TIMEOUT={number of seconds}
```

This instruction can be written anywhere in the logic and it applies to the entire logic.

Examples:

```
*TIMEOUT=300      // 5 minutes
*TIMEOUT=1200     // 20 minutes
*TIMEOUT=0        // unlimited time out
```

### Calling the same stored procedure multiple times

Now, multiple executions of the same stored procedure within the same COMMIT section (with or without the same parameters) are supported. Previously, the same stored procedure could only be invoked once, unless the executions were separated by a COMMIT statement.

### Running the last stored procedure within a given COMMIT section

Until now, if a stored procedure would return an error, all subsequent stored procedures (together with the rest of the logic) would be skipped. Now you can recur to an option to make sure the last stored procedure invoked within the current COMMIT section will still be executed, even in case of failure of a preceding one. This option can be turned on using the following syntax:

```
*RUN_STORED_PROCEDURE=ProcedureName(Parameters) ON_ERROR_RUN_LAST
```

If any stored procedure with the above setting (ON\_ERROR\_RUN\_LAST) fails (either because of an SQL error or an \*ERROR\* message returned in the log file), the logic will still run the LAST stored procedure in the COMMIT section, before aborting with the error message.

This option can be useful in case the user needs to make sure that some cleanup activity is performed, no matter what happens to some stored procedures being invoked by the logic.

Here is an example:

```
//-----
*RUN_STORED_PROCEDURE=CHECKSTATUS({Parameters},LOCK)
*RUN_STORED_PROCEDURE=SPRUNCONVERSION({Parameters})
ON_ERROR_RUN_LAST
*RUN_STORED_PROCEDURE=SPRUNCONSOLIDATION({Parameters})
ON_ERROR_RUN_LAST
*RUN_STORED_PROCEDURE=CHECKSTATUS({Parameters},UNLOCK)
*COMMIT
//-----
```

### Added support of %WHEN% keyword in \*REC instruction

It is now possible to retrieve within any part of a REC statement the value of the innermost WHEN instruction of the current WHEN / ENDWHEN structure

```
*WHEN TIME.TIMEID
*IS *
  *REC(EXPRESSION=%WHEN%/10000, ACCOUNT= # + ACCOUNT.ID + _ +
%WHEN%)
*ENDWHEN
```

The above example, when TIME is 2006.MAR and ACCOUNT is CASH, will return a record where the SIGNEDDATA is 2006.03 and the ACCOUNT is #CASH\_20060300.

### Ability to disable CALC\_EACH\_PERIOD in one GO section

When the CALC\_EACH\_PERIOD action is set, all GO sections within a given COMMIT section will be processed one period at a time. It is possible however that one of the GO sections could be processed for all periods at once. To allow for this (faster) processing, we now support a different syntax for the GO instruction, as follows:

```
*GO_ALL_PERIODS
```

This instruction is similar to a GO instruction, with the difference that, even if the CALC\_EACH\_PERIOD action is turned on for the current COMMIT, this individual GO section will be calculated for all periods at once.

### Ability to disable CALC\_EACH\_PERIOD in one CALC\_ORG or CALC\_DUMMY\_ORG section

When the CALC\_EACH\_PERIOD action is set, all CALC\_ORG or CALC\_DUMMY\_ORG instructions within a given COMMIT section will be processed one period at a time. It is possible however that one of the CALC\_ORG or CALC\_DUMMY\_ORG instruction could be processed for all periods at once. To allow for this (faster) processing, we now support a new setting for the CALC\_ORG and CALC\_DUMMY\_ORG instruction, as follows:

```
*CALC_DUMMY_ORG
*ORG {dim}={property}
*CALC_ALL_PERIODS
*ENDCALC
```

This setting, which can only be triggered from inside the multi-line version of the CALC\_ORG and CALC\_DUMMY\_ORG instructions (like the one used in the above example), will make so that, even if the CALC\_EACH\_PERIOD action is turned on for the current COMMIT, this individual CALC\_ORG or CALC\_DUMMY\_ORG action will be calculated for all periods at once.

## Index

### %

%CURRENCY_SET%	68
%LOGTABLE%	68
%SCOPETABLE%	68
%WHEN% keyword in *REC instruction	209
<b>*</b>	
**SELECT	132
*LOGIC	130
*RUN_ALLOCATION	199

### A

A - dimension type	38
Account	27, 38, 186
Account Transformation rules	68
activity audit, enabling	180
activity audit, reporting	180
Actuals	28, 38
ADD / *ENDADD	101
add drillkey	182
ADD_DIM	101, 108
adding business rules tables to applications	67
adding dimension properties	26
adding KPIs	163
adding properties to dimensions	25
adding steps	57
adding teams	43
adding users	42
adding work states	60
admin users	42
allocation	199
Appendix: best practices for writing logic	196
application parameters	177
application set parameters	175
Appset Not Available	11
AppSet, creating new	11
AppSetPublications	186
ApShell reports and input schedules	15
assigning dimensions to applications	24
automatic adjustment detail	75
automatic adjustments	73

### B

BASE	136
BEGIN	102
best practices, logic	196
Book Reports folder	15
Budget	28, 36, 38
business rules tables, adding to applications	67
business rules tables, deleting	67

### C

CALC	102
CALC_DUMMY_ORG	102
CALC_EACH_PERIOD	104, 134
CALC_ORG	102, 134
CALCULATE_DIFFERENCE	104
Calculations	68
Calling the same stored procedure	208
carry-forward rules	71
Category	22, 28, 38
CC property	32
CLEAR_DESTINATION	105
COMMIT	107
COMMIT_EACH_LEVEL	106
COMMIT_EACH_MEMBER	106
COMMIT_MAX_MEMBERS	107
COMMITs	203
Compress Fact Table	37
concurrent locks	37
consolidation methods	80
consolidation rules	80
consolidation rules formulas	80
controlling SQL time-out	208
CPE	96
creating dimensions	19
creating new application	36
Creating new application sets	11
Currency	29, 38
currency conversion rules	68
currency conversions	63
Currency translation	29
current view	37, 38, 182

custom actions for reviewers .....	57	ENDLOOKUP.....	117
<b>D</b>		ENDSELECT .....	134
database specifications .....	173	ENDSUB .....	135
default task profiles.....	45	ENDWHEN .....	137
defining teams.....	43	Entity .....	22, 29, 38, 172
defining user IDs .....	42	Entity Method.....	80
delete dimension property.....	26	Entity property, adding CC.....	32
deleting appsets .....	11	EQUITY .....	27, 80
deleting business rules tables from an application .....	67	EVDES.....	27, 28, 29, 31
deleting work states .....	62	EVGET .....	28, 182
DESTINATION .....	107	EvInp .....	182
DESTINATION_APP .....	108	EVPRO.....	186
Dimension		EVTIM .....	28, 31
property.....	22	Expense .....	27
types .....	38	EXPRESSION .....	137, 141
dimension processing, scheduling .....	21	<b>F</b>	
dimension property, add & delete.....	26	FACTOR.....	137, 141
dimension reserved names.....	19	FIRST.....	136
dimension types .....	19	FIRST_PERIOD .....	108
dimensions, assigning to applications .....	24	FLD.....	137, 141
dimensions, creating .....	19	Flow dimension .....	63
dimensions, processing.....	20	FOR .....	110
disable CALC_EACH_PERIOD .....	209	Forecast .....	28, 38
document subtypes .....	185	Forecasting application .....	38
document types.....	184	FORMAT .....	22, 27, 31, 186
Drill Through		FORMULA.....	27, 32
access .....	182	full process .....	20, 22, 23
creating .....	182	FUNCTION .....	111
Defining .....	182	<b>G</b>	
Drill Through .....	182	GET .....	137, 142
DrillKey .....	182	GO.....	112
drill-through queries.....	183	GO section.....	209
drill-through table .....	184	GO statements .....	206
<b>E</b>		Group dimension .....	63
Editing default messages .....	172	<b>H</b>	
editing work states .....	61	hierarchy .....	22, 27, 28, 29, 31
email addresses.....	42	HLEVEL .....	27, 28, 29, 31
enabling activity audit .....	180	HQ folder.....	15
END .....	102	HTTPS .....	41
ENDADD .....	101	<b>I</b>	
ENDFUNCTION.....	111	ID.....	22, 27, 28, 29, 31

InApp .....	25, 26	MDX queries .....	26
INCLUDE .....	114	MEASURES .....	121, 173
Income		Measures dimension .....	38
INC .....	27	Member ID .....	22
Income .....	27	member sheets .....	22, 27, 28, 29, 186
incremental process .....	20, 23	members .....	22, 27, 29, 31, 38, 173, 182, 186
Intercompany booking rules .....	72	members, adding .....	22
Intercompany dimension required properties .....	30	MEMBERSET .....	121
intercompany eliminations .....	73	Memory variables .....	122
intercompany eliminations detail .....	75	messages .....	22, 37, 172
invalid characters .....	111	metadata .....	22
IP address .....	186	Method codes .....	80
<b>J</b>		modeling .....	29
JOIN .....	114	<b>N</b>	
<b>K</b>		new application .....	11, 36
KPIs, adding .....	163	new application set .....	11
<b>L</b>		new dimension .....	22
LAST_MEMBER .....	115	new users .....	36
legal consolidation appset .....	63	NEXT .....	110, 136
LEVEL .....	31	NO_PARALLEL_QUERY .....	124
LOCAL_CURRENCY .....	116	NOADD .....	137, 140
locks, concurrent .....	37	Not Available .....	172
LOG files .....	206	<b>O</b>	
LOGIC .....	27, 130	OLAP Services .....	173
Logic, best practices .....	196	Open report library .....	15
LOGIC_BY .....	116	Open schedule library .....	15
LOGIC_MODE .....	116	Optimizing .....	22, 37
LOGIC_PROPERTY .....	117	Owner .....	60
LOOKUP .....	117, 142	owner dimension .....	36
<b>M</b>		Owner property .....	26, 29
maintaining dimension properties .....	26	<b>P</b>	
manage application sets .....	11	PARENT .....	27, 29
manage applications .....	36, 37	Parent/child .....	22
manage dimensions .....	22	PARENTH1 .....	29
manage members .....	22	pound (#) sign .....	122
Manager .....	60	PRIOR .....	136
Manager folder .....	15	Process members ....	11, 22, 27, 29, 37, 38, 173
managing work states .....	60	process members from member sheet .....	22
maxstatus .....	68	process options .....	20
MDX .....	27	PROCESS_EACH_MEMBER .....	126
MDX logic .....	196	PROCESS_FAC2 .....	126

processing an application .....	37	member access profile .....	191
processing dimensions .....	20	resolving member access profile conflicts ..	191
processing members .....	22	task profiles .....	191
properties		Security Management	
dimension contains .....	22	general rules .....	191
properties, in dimensions .....	25	security reports .....	52
Proportional .....	80	SELECT .....	132
publishing office documents .....	186	SELECTCASE .....	134
PUT .....	126	Server Manager .....	170
<b>Q</b>		setting concurrent locks .....	37
QUERY_FILTER .....	127	Shell application .....	11, 22, 36, 182
QUERY_SIZE .....	128	SKIP_DIM .....	108, 134
QUERY_TYPE .....	128	SolveOrder .....	32
<b>R</b>		SPCOPYOPENING .....	71
Rate application .....	35	special property - CC .....	32
read/write dimensions .....	24	SPICBOOKING .....	72
REC .....	105, 137, 140	SPICDATA .....	72
Refer		SPRUNCALCACCOUNT .....	68
UNICODE .....	173	SPRUNCONVERSION .....	68
Refresh .....	22	SPRUNELIM .....	73
refresh-after-send .....	207	SPRUNVALID .....	77
RENAME_DIM .....	108, 129	SQL	
reordering work states .....	61	adding members to dimension .....	22
reporting on activity audit .....	180	Query file .....	182
reporting on work states .....	61	SQL Server .....	173, 182
reserved member names .....	21	SQL Server 2000 Analysis Services .....	173
reserved names		SQL logic .....	197
dimensions .....	19	step properties in a BPF .....	57
resetting BPFs .....	54	STORE_ORG .....	134
reviewer custom actions, setting up .....	57	stress test .....	207
RUN_ALLOCATION .....	199	Style property .....	27, 28, 29, 31, 186
RUN_STORED_PROCEDURE .....	129	Style value .....	186
RUNLOGIC .....	130	style-based formatting .....	186
Running the last stored procedure .....	208	SUB .....	135
<b>S</b>		Subaccounts, defining .....	27
scheduling dimension processing .....	21	Subdirectories .....	11
SCOPE_BY .....	132	Subtable dimension .....	63
Script logic documentation updates .....	208	SYSLIB .....	136
secured dimensions .....	22, 24	<b>T</b>	
security .....	11, 22, 28, 29	team definitions, modifying .....	44
Security management .....	191	teams, adding .....	43



templates .....	11, 22, 36	work states	
TEST_WHEN .....	144	adding .....	60
Time		deleting .....	62
TIMEID .....	31	editing .....	61
Time-shift instructions .....	136	editing descriptions .....	61
TOTAL		managing .....	60
TOTAL member .....	31	reordering .....	61
TOTAL .....	31	reporting .....	61
<b>U</b>		work status	
UnaryOperator .....	32	changing application settings .....	36
Update Version .....	22	owner property .....	26
URL .....	186	work status .....	57
US elimination .....	73	work status dimensions .....	36
USE .....	137	Write Back .....	37
users, adding .....	42	WRITE_TO_FAC2 .....	145
<b>V</b>		WRITE_TO_FILE .....	146
Validate		<b>X</b>	
Validate and process members .....	22	XDIM_ADDMEMBERSET .....	146
validating members .....	22	XDIM_DEFAULT .....	146
Validation .....	22, 38	XDIM_FILTER .....	146
Validation rules .....	77	XDIM_GETINPUTSET .....	147
Validation rules detail .....	78	XDIM_GETMEMBERSET .....	148
Viewing security reports .....	52	XDIM_MAXMEMBERS .....	150
<b>W</b>		XDIM_MEMBER .....	150
Web Server .....	186	XDIM_MEMBERSET .....	151
WebFolderDirectory .....	22	XDIM_REQUIRED .....	152
WHEN .....	137	<b>Y</b>	
Wizard folder .....	15	YTD .....	38, 39