

Performance



ADMINISTRATION GUIDE

VERSION 9.4.0 (18)

DATE 12/2022

© Copyright 1998 - 2022 cubus AG - A Serviceware Company. All rights reserved.

Microsoft, Excel, Windows and Windows Server are trademarks or registered trademarks of Microsoft Corporation.

All other names of companies, products and/or services are trademarks of their respective owners.

Hyperion and Essbase are registered trademarks and Hyperion Solutions is a trademark of Oracle.

Subject to change without notice.

All rights reserved. No part of this document may be reproduced, translated or transmitted in any form or by any means, electronic, optical or mechanical, including photocopy, recording, or any information storage system, for any purpose other than the purchaser's personal use, without the prior written consent of cubus AG.

cubus AG assumes no responsibility for typographical errors or inaccuracies in this documentation or for the use, reliability, interoperability, merchantability, or fitness for a particular purpose of the software product and/or programs described herein. This manual is subject to periodic updates. cubus AG reserves the right to modify, augment, or improve this document as well as the products and/or programs described herein.

Contents

Part I Introduction	1
Part II Architecture	2
1 WEB Architecture.....	2
2 OLAP Databases.....	2
3 Relational SQL Database.....	2
4 Performance Client Website.....	2
5 Performance Administration Website.....	3
6 Performance Desktop.....	3
Part III Creating a Performance Application	4
1 Directory Structure.....	4
Performance Installation Directory	4
Web Application Directory	5
Performance Application Directories	6
Global Application Directory	7
2 Creating the Application Directory Structure.....	7
3 Creating the Application.....	7
4 Creating the Performance database(s).....	7
Creating the OLAP Database	8
Checking the Relational Repository	8
5 Creating the OLAP Model.....	8
Unit Dimension	9
6 Application Server Configuration.....	10
Application Server Manager Configuration	10
Manually starting a specific Application Server	11
Application Server Command Line Parameters	12
Connection Information	12
Configuring the Application	13
7 Synchronizing the Performance Databases.....	13
8 Creating Modules.....	13
Creating Workbooks	14
Creating a Performance Analytics / AL Anywhere Module	14
Creating a Group Module	14
Creating a Detail Module	14
9 Configuring the Modules.....	14
10 Setting up Users and Access Permissions.....	15
Part IV Creating a ROLAP Database	16
1 Creating the Relational Model.....	16
Fact Table ABC_FACT_<application_name>	17
Dimension Table DIM_<dimension_name>	19
Table ABC_DIM_<application_name>	20
DIMENSION_TYPE Table	21
Table MEMBER_NAME	22
Table MEMBER_ALIAS	23
Table MEMBER_UDA	23

Table ALIAS_TABLE	24
Table UDA	25
Views	25
View V_DIM_<dimension_name>	25
View V_MEMBER_ALIAS.....	27
View V_MEMBER_UDA	28
2 Administration of Members and Dimensions.....	28
Adding a New Dimension	28
Deleting a Dimension	29
Adding a member	29
Deleting a member	29
3 Mapping Multiple Applications in a Relational Database	29
4 ROLAP Configuration.....	30
Part V Administration Website	32
1 Application List.....	33
2 Global Administration - Application.....	33
Creating an Application	34
Editing an Application	34
Deleting an Application	34
Renaming an Application	34
Exporting Application Data	35
Importing Application Data	35
3 Configuration Parameters.....	35
Global Parameter Dialog	36
Application Parameter Dialog	36
Configuration Parameter Menu Items	37
Copying Configuration Parameters	37
List of Configuration Parameters	37
ABC_WWW_ROOT.....	37
AbcVirtualPath.....	37
ActionHistoryNumberOfTopRecords	38
AdditionalApplicationExportTables	38
ADMIN_WWW_ROOT.....	38
AllowCommentsOnProtectedCells	38
AllowFillRange.....	38
ALProvider	38
ALServer	38
ALServerCommentsHost.....	39
ALServerRestApiPort.....	39
ALServerWebServiceEncryption.....	39
ALTheme	39
AlwaysShowTreeSelection.....	39
AppServerCompression.....	39
AppServerEncryption.....	40
AppServerSchema.....	40
AppServerRestSchema.....	41
AppServerRestPort.....	41
AppServerWebAlias.....	42
AutoInsertNewRow.....	42
AutoSizeColumnHeaders.....	42
AutoSizeRowHeaders.....	42
CheckAddCalc Tag.....	42
CheckSheetBeforeSave	42
CheckUnitAccess	43
CreateUserAccessRightsOnOLAPServerToo.....	43
CreateUserOnOLAPServerToo.....	43
CubusAppServerManagerConnectionString.....	43

CubusPath	44
CustomerBackgroundFileName	44
DistribAccuracy	44
DumpDETCalc	44
DumpGrid	44
DumpMemberSelect	44
EnableExport	44
EnableImport	44
EnableMaintenanceMode	45
EnableWorkbookDesigner	45
EnableWorkflow	45
ExpandedUnitGeneration	45
ExpandModuleOnSelect	45
ExpandRibbonsOnLoad	45
ExportType	45
ForceSave	45
HideRootModule	45
HideToolbarButtons	46
HideTreeIcons	46
IgnoreLockedAddCalcCells	46
IncludeHiddenCellsOnCopy	46
IncludeHiddenCellsOnPaste	46
LogPath	46
MaintenanceModeMessage	46
ModuleMode	46
MSReportServerPath	46
NumberOfSearchRecordsDisplayed	47
Preload	47
PrimaryOLAPDB (and further OLAP databases)	47
<OLAPDB>_SSASCube	48
ProgressIndicatorDelay	49
RefreshOutlineTables	49
ReloadDataAfterCalc	49
ReloadDataAfterSave	49
SelectionBackgroundColor	49
SelectionDisabledTextColor	49
SelectionTextColor	49
SessionTimeout	49
SetModulesExpanded	50
SheetPrintFooter	50
SheetPrintHeader	50
ShowAncestors	50
ShowDimensionName	50
ShowScrollbars	50
ShowSheetHeaders	50
ShowSheetTabs	51
SkipHiddenCellsOnPaste	51
SqlQueryCacheMode	51
SqlQueryTimeout	51
SuppressZeroValue	51
TraceLevel	51
UnitLockType	51
UseMemberNameForTreeExport	52
UseOlapConnectionUser	52
UsePersonalizedOutlines	52
4 Substitution Variables	52
Creating a Substitution Variable	53
5 Concurrent User Report	53
Report Type	53
Configuration	54

6	Log Viewer	54
7	Session	54
	Deleting a Session	55
	Configuring Session Timeouts	55
8	Create DesktopApp package	56
9	Module Administration	57
	Module Administration - General	57
	Module Administration - Specifics	59
	AbcModuleReloadStructure.....	60
	AddCalcViewPosition	60
	AddCalcViewSize.....	60
	ConnectionStringName.....	60
	ContentViewPosition	61
	ContentViewSize.....	61
	CreateEVSession.....	61
	CustomerBackgroundFileName	61
	DetailGridPosition	61
	DetailGridSize.....	62
	DetReport	62
	EvaluationOrder.....	62
	ExpandRibbonsOnLoad	62
	GanttActivityBarClick.....	62
	GanttActivityMilestoneClick.....	63
	HideEvRibbonTabs	63
	HideToolBarButtons	63
	InformationFile	65
	Link	65
	ModuleFile	65
	NavigateWithoutData.....	66
	RefreshOBViews.....	66
	SendPassword.....	66
	ShowBackButton	66
	ShowDimensionName.....	66
	ShowFindControls.....	66
	ShowPageNavigationControls.....	67
	ShowParameterPrompts	67
	ShowPromptAreaButton.....	67
	ShowTabs	67
	URL	67
	UseModuleName.....	67
	UseWorkbookCommentFormatting.....	67
	View	67
	ViewType	68
	Module Specific Compatibility Matrix.....	68
	Module Administration - Execution	70
	Menu Items	72
	Parameterization of Scripts.....	72
	External Calculation Server.....	74
	Input and Output Files of External Programs.....	76
	Module Administration - Unit	77
	Module Administration - Member Select	77
	Creating New Modules	78
	Menu Items	79
	Module Check – Global Section	80
	Moving Modules in Administration Tree	80
	Copying Modules	80
	Detail Module – Menu Items	81
	Create Table	81
	Clear Table	81

Delete Table	81
Workbook Designer	81
Member Select Designer	83
10 User Administration.....	84
User Concept	84
Creating a User	85
Removing a User	86
Renaming a User	86
Menu Items	87
New or Edit User.....	87
Copy User	88
Assigning Roles.....	89
11 Role Administration.....	89
Menu Items	92
Assigning Users	92
Access Permissions of Units to Modules	93
Menu Items	94
Module Access Permissions	94
Menu Items	95
Direct Unit Access Permissions	96
Menu Items	97
12 Unit Administration.....	97
Menu Items	97
13 Unit Lock.....	98
Menu Items	98
14 Submitting/Un-submitting Objects.....	98
Menu Items	99
15 Event Management.....	99
Menu Items	100
16 Client Integration.....	100
Part VI Toolbars	102
1 Hide buttons.....	102
2 Task button.....	102
Part VII Adhoc Analysis	103
1 Requirements.....	103
Part VIII Workflow	104
1 Basic Terms.....	104
Definition	104
Workflow Tasks	104
Workflow Levels	104
Workflow Actions	104
Submit	105
Accept	105
Reject	105
2 Workflow Administration.....	105
Workflow Module Structure	105
Workflow Master Data	106
Workflow Tasks	106
Log Entry	107
Workflow Access Rights	107
Workflow Roles.....	107
User-To-Role Assignment.....	107

Role-To-Unit Assignment.....	107
Role-To-Module Assignment.....	108
Workflow Status Module	108
Workflow Matrix Module	108
Part IX Tools	109
1 Batch Client.....	109
Command Line Parameter	109
Structure of the Batch File	110
SET_HOSTNAME	110
SET_PORTID.....	110
SET_MANAGERPORTID.....	110
START_APP.....	111
STOP_APP	111
LOGIN	111
LOGOUT	111
SET_MODULE.....	111
SET_UNIT	111
SET_OFFGRID.....	112
EXECUTE/CALCULATE.....	112
EXPORT	113
SET_PROPERTY.....	113
SET_PRINTER_TIMEOUT.....	113
SAVE	113
RESET_OUTLINE.....	113
RESET_CACHE.....	113
EXIT	114
Example	114
2 CopyComments.....	114
Part X Appendix A: Wildcards	120
Part XI Appendix B: Customising the Look&Feel	122
1 Change Active Theme.....	122
2 Creating a New Theme Directory.....	122
Part XII Appendix C: Cockpit Configuration	124
1 Cockpit Overview.....	124
Cockpit Content Page	124
2 Output Reports.....	124
Frame Layout Combinations	124
Performance Analytics Frame	126
News Frame	126
Comment Frame	127
MS Reporting Services Frame	127
3 Cockpit Module Configuration.....	128
Adding a Cockpit Module	128
Configuring Cockpit Specifics	128
General Parameters	129
Layout	129
Format	130
Specific	130
Unit	130
Behavior	130
Special	130
Frame Specific Parameters	131
CPFrameX_Type.....	131

CPFrameX_View.....	131
CPFrameX_FrameHeader.....	131
CPFrameX_DependOnList.....	131
CPFrameX_EventArgList.....	132
CPFrameX_FilterList.....	134
CPFrameX_Behaviour.....	135
CPFrameX_Format.....	135

Part XIII Appendix D: Migrating custom HTML page containing Performance Analytics Object to Performance Desktop	138
--	------------

1. Introduction

This guide is intended for the administrator of the Performance tool.

Part 2 illustrates the architecture of Performance and its use in web environments.

Part 3 makes up the main part of this guide. Step-by-step instructions illustrate the procedure for creating an Performance application. This Part also discusses the Administration Website and the structure of the relational database.

Part 4 describes the setup of a ROLAP database.

Part 5 describes the overall administration functions for setting up and configuring an Performance application.

Part 6 describes some generic features of the general toolbar.

Part 7 describes some process management capabilities.

Part 8 describes the administration of the workflow.

Part 9 describes the commands available in the BatchClient tool as well as additional tools.

This guide does not elaborate on the various methodologies available for the creation of a planning model by means of the OLAP tools. This information is provided in the individual OLAP tool manuals.

2. Architecture

WEB Architecture

Performance is designed for use in web based environments. The multi-tier based architecture consists of the database layer, the application layer and the client layer.

The web applications for client and admin access are hosted in the Microsoft Internet Information Server (IIS) as an ASP.NET 4.0 application.

These web applications are communicating with the Application Server Manager and the Application Servers via WCF.

All Performance applications are managed with a single repository that stores all the metadata needed by an application.

OLAP Databases

The OLAP databases serve to map the input data to multidimensional structures. Their comprehensive analysis capabilities support the centralized evaluation and coordination of planning by the Controlling department.

Data may be downloaded from other data sources (e.g. data from the General Ledger) into the OLAP databases to support forecasting or comparisons with previous periods.

By means of the OLAP database, you can save and compare multiple versions.

Performance supports the following technologies for implementing the OLAP databases:

- MOLAP (multidimensional OLAP)
 - Oracle Essbase
- ROLAP (relational OLAP)
 - Relational DB: Microsoft SQL Server
 - Multidimensional DB: MS Analysis Services
- TM1 (multidimensional OLAP)
 - IBM Cognos TM1

The ROLAP variant is implemented via two components: a relational and a multidimensional database. All structural information and data required by Performance is stored in a relational database, i.e. the relational database is mandatory for Performance. For analysis, a multidimensional database can be set up on this basis.

Relational SQL Database

The SQL database stores the meta data for all Performance applications. The meta data includes e.g. users and their access permissions as well as units and their associated modules.

Performance Client Website

All input and computation functions are implemented in the Performance Client Website. The Performance Client Website is the user interface for Performance.

The user needs to be authenticated by Performance or other authentication providers such as

ActiveDirectory, Essbase or LDAP. After the successful authentication the user's specific module structure including the access to the planning or reporting units is displayed.

Performance Administration Website

The Performance Administration Website is the interface for the administrator. The Administration Website is used to configure all application settings.

Performance Desktop

Performance Desktop basically is a desktop application wrapper around the web client allowing you to access Performance without using Internet Explorer. The application can be started by either calling the installation URL (see PerformanceInstallationGuide under installation chapter) or by using the desktop shortcut, which will be created during the installation.

Migration is required in order to use the custom HTML pages containing Performance Analytics Object in the Performance Desktop (see [Appendix D](#)).

3. Creating a Performance Application

Directory Structure

Performance Installation Directory

In the following sections the term

<INSTALL_DIRECTORY>

refers to the installation directory

e.g.: C:\Program Files\cubus\Performance

After the installation of the software, the installation directory contains the following directory structure:

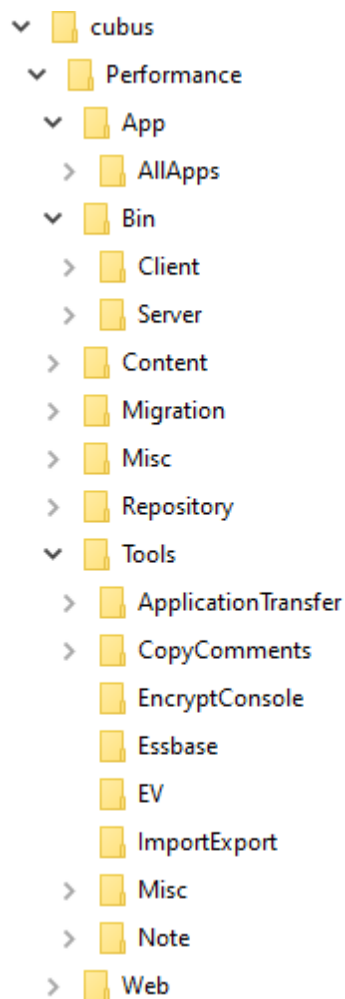
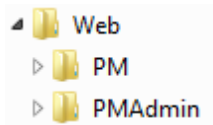


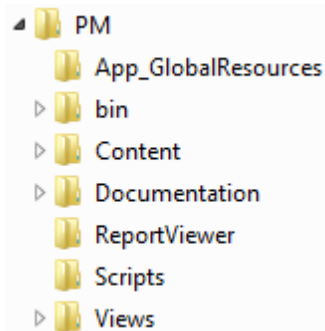
Fig. 1 - Installation Directory

Web Application Directory

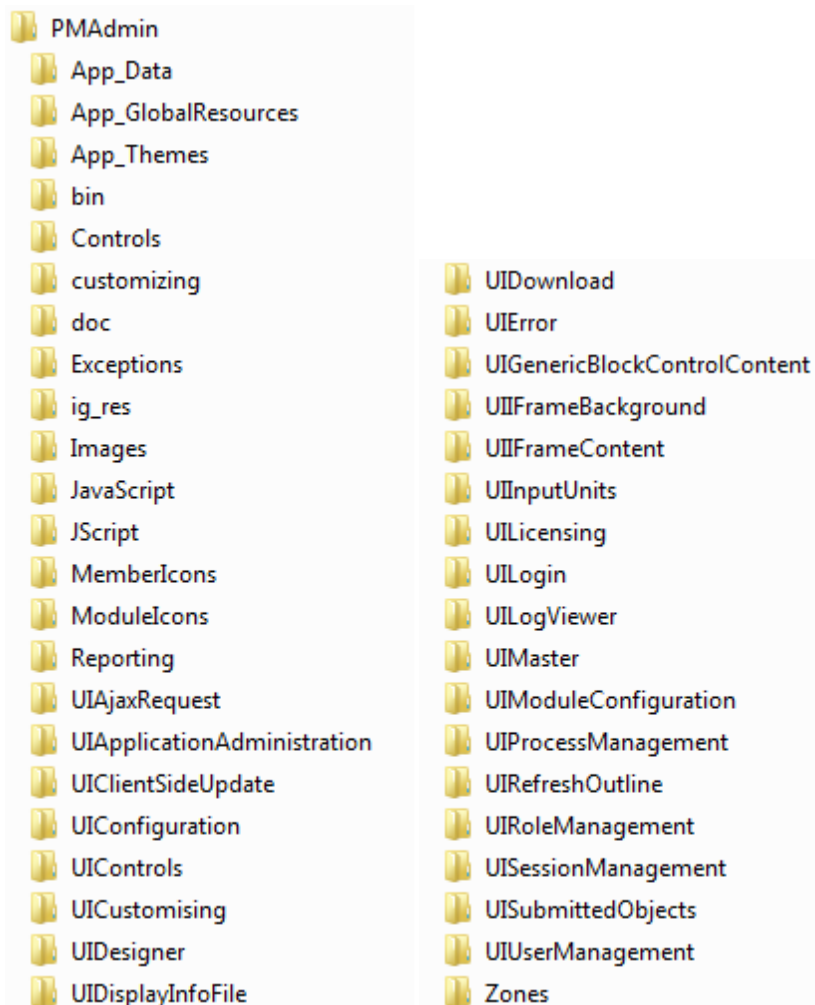
The web applications are stored in the *Web* directory of the Performance directory.



The client application:



The administration application:



This directory structure supports the separation of program data and application data. The idea is that the program data only needs to be installed once on the system.

Performance Application Directories

The application data for the ApplicationServer is contained in the `App` directory. The program data as well as the templates are stored in the `Bin` directory. For detailed information on the structure and contents of the `Bin` directory, refer to the “Installation Manual”.

The `App` directory contains all Performance applications. The next figure illustrates the directory structure of an Performance application (“`AbcSample`”).

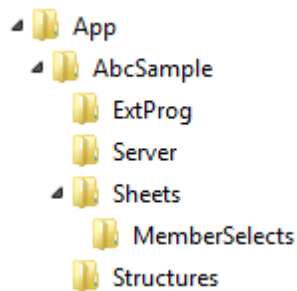


Fig. 2 - AbcSample - Directory Structure

The subdirectory structure of the `AbcSample` directory is explained in the following table.

Directory	Description
<code>ExtProg</code>	External calculation programs
<code>Reports</code>	Optional reports
<code>Server</code>	Log file, batch file for starting the AppServer
<code>Sheets</code>	Performance workbooks, member selection
<code>Structures</code>	The outline structure files (XML)

Following the installation of the software the following steps need to be performed:

1. Create the directory structure for the Performance application
2. Create the Performance application with the administration website
3. Create the OLAP model in the OLAP DB
4. Create the basic configuration of the application
5. Transfer the OLAP model of the planning structure from the OLAP database to the SQL DB (synchronization)
6. Create the modules
7. Configure the modules for use with Performance
8. Set up the users and their associated permissions

A step-by-step instruction is provided in the following chapters.

Global Application Directory

Workbooks can be stored in a global sheets directory. These workbooks can then be used in all applications.

Member select sheets can also be stored in the global MemberSelects directory.

To use this functionality, this directory first has to be created.

Therefore, create a new folder below the `App` directory and name it `AllApps`. Within the directory `AllApps`, create the folder `Sheets`.

If the global member select functionality is also required, create the folder `MemberSelects`.

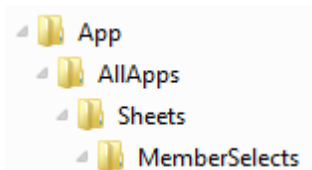


Fig. 3 - AllApps -
Directory Structure

All workbooks located in this directory will be available for all Performance applications and can be selected in the module specifics.

All member selects located in the MemberSelects directory will be available for all applications.

Note: Workbook and member select files located in the application specific sheet or member select directory are loaded and used first – even if the global sheets directory contains a file with the same file name.

Creating the Application Directory Structure

To prepare for creating a new application, simply copy the directory `Content\AbcSample\ABCApp` to `App/<application_name>` (the application directory).

Creating the Application

In the administration website the new Performance application must be created.

Creating the Performance database(s)

The “minimum” configuration of the Performance database consists of:

- at least one **multidimensional database (OLAP DB)** for storing the OLAP data in a multidimensional structure (see also “OLAP Database”)

Additionally, further OLAP databases can be integrated.

The use of multiple OLAP databases (= “multicubes”) in one application provides a number of advantages, e.g.:

- Splitting an application into a number of partial plans with various dimensions (e.g. sales/revenues plan with products and markets; cost planning with cost centers; profit and loss/balance planning with clients).

- Prevention of “irrelevant“ data items in the OLAP database.
- Use of different “ABC_UNIT“ dimensions in one application for separating access to specific data sections.
- Use of different OLAP databases in an Performance application (sales/revenues planning based on MS-AS; profit and loss/balance based on Essbase).

When using multiple OLAP databases in the same Performance application, the following conditions must apply:

- The OLAP databases may use different dimensions.
- Dimensions with the same name must have the same structure. This ensures transparency for the user; i.e. the user is not aware of the different OLAP databases.
- The “Unit“-dimension of OLAP databases are identified by the UDA “ABC_UNIT“ (MOLAP) or by the appropriate dimension type (ROLAP).
- The OLAP databases contain alias tables with identical table names. Identical dimensions use identical alias names for the members.
- A so-called “*PrimaryOLAPDB*“ needs to be defined.

Any further reference to Performance database in this guide, relates to the entire database, consisting of the relational database and the multidimensional database(s).

Creating the OLAP Database

To create an OLAP database using Oracle Essbase™ you first need to create the Essbase application and then the associated Essbase database.

If the ROLAP variant is used, the OLAP database is created by creating the relational database and the associated tables (see "Creating a ROLAP Database" for further details).

Also an Performance administrator needs to be defined for this database. In Essbase, this user needs to have "Application Designer" permissions for this application. If the ROLAP variant is used, this user requires full access to the ROLAP database.

Note: Initially, the sample database supplied with this software includes an administrator and startup user whose user ID is defined during the setup (e.g. admin/password).

Checking the Relational Repository

After the installation of Performance the repository for all Performance applications is available. The default name is “ABCRepository”.

The repository contains a default user defined during the setup (e.g. admin).

Creating the OLAP Model

Performance grants the user maximum flexibility for the creation of a multidimensional database. Initially, the OLAP model is created by means of the OLAP administration tool ("Essbase Administration Services Console " in the case of Oracle Essbase™ or MS SQL Server and MS Analysis Services in the case of ROLAP).

This enables the user to make full use of all functionalities and tools when creating the OLAP database.

Typically, an OLAP module comprises of a minimum of four dimensions. However, Performance is not subject to any restrictions in terms of the number of dimensions in the OLAP model.

To illustrate the functionalities, let us assume that we are dealing with the following model (the OLAP sample database model named “ABCSampl”). This model comprises of seven base dimensions:

- | | |
|-----------------|--|
| 1. Measures | Hierarchy of the planning members, e.g. revenues, cost. |
| 2. FY | Planning period, e.g. year, quarter, month. |
| 3. TS | Time Series containing the years (Y2009, Y2010,..) |
| 4. Scenario | List of scenarios, e.g. actual, budget, variances |
| 5. Organisation | Hierarchy of the planning members (= units), e.g. CC_Prod, CC_Sales, Region1 |
| 6. Product | Product hierarchy |
| 7. Customers | Customer hierarchy |

Unit Dimension

The unit dimension is mandatory in Performance. This dimension is used to control the access permissions. The dimension needs to be tagged so that Performance is able to get the required information:

- MOLAP – database

The dimension of the unit needs to be identified through the UDA (User Defined Attribute) *ABC_UNIT* in the OLAP DB. If more than one unit dimension is needed in the same MOLAP database, then additional UDAs (e.g. *ABC_UNIT_COSTCENTER*, *ABC_UNIT_PRODUCT*) can be defined.

These additional unit dimensions can be used to define further access permissions.

- ROLAP – database

The dimension of the unit needs to be identified through the "ABC_UNIT" dimension type in the *ABC_DIMENSIONS* table. This implies that an associated dimension type with the description "ABC_UNIT" needs to be created in the *DIMENSION_TYPE* table.

The following figure shows an outline of the planning model used in the “AbcSample” sample application.

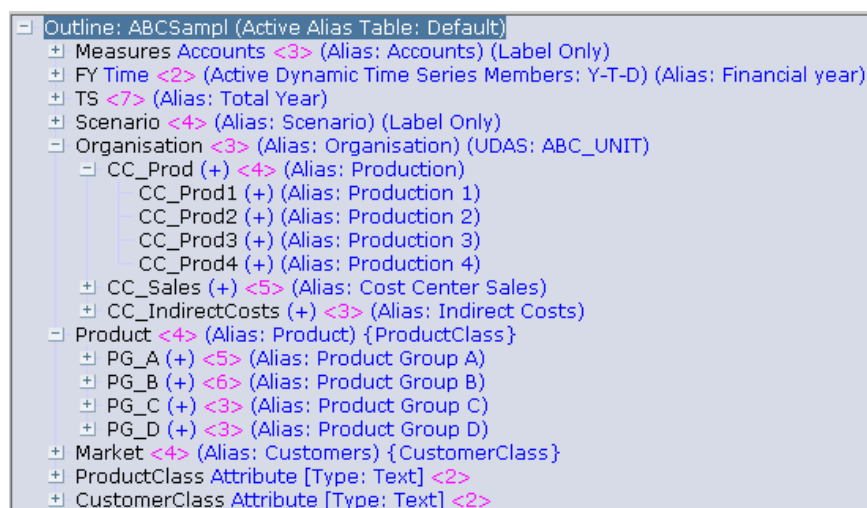


Fig. 4 - “ABCSampl” OLAP model

Application Server Configuration

After the installation of Performance a windows service is installed. This service is named **Serviceware Performance App Server Manager**. This service is responsible for starting and stopping the individual application servers for each application.

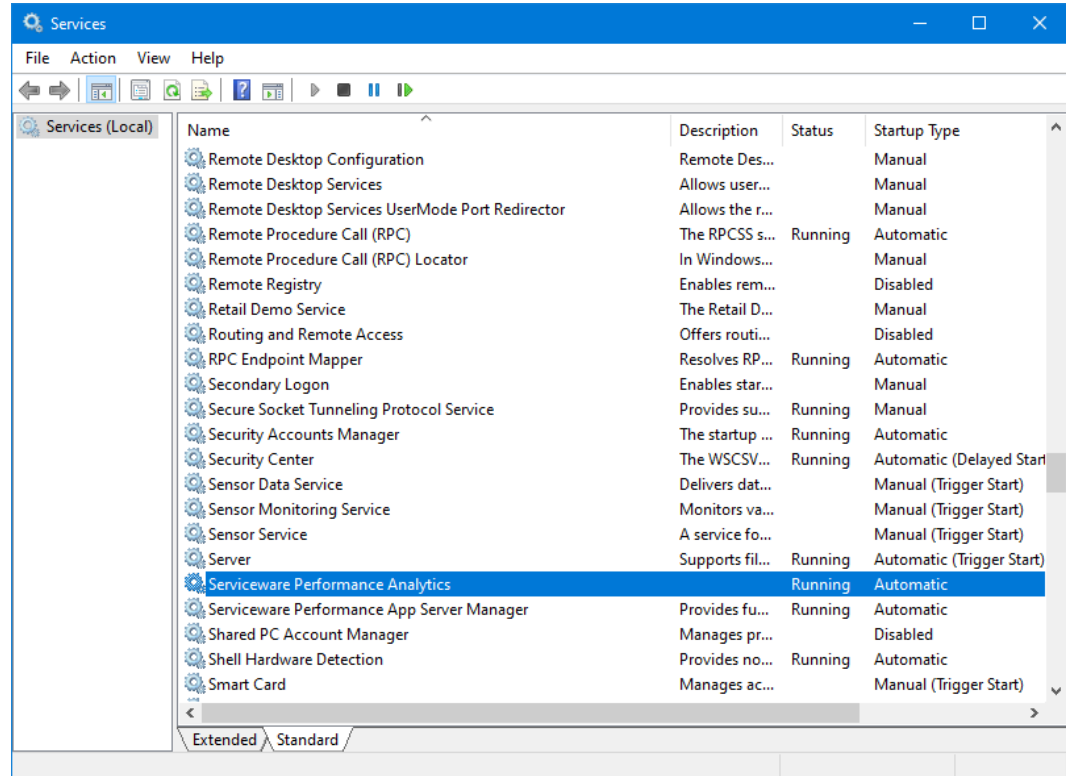


Fig. 5 - Services

Note: The .ini files of the Performance version 6.X and earlier do not exist anymore. The former command line parameters for the application servers have been replaced.

Application Server Manager Configuration

The “Performance App Server Manager” service can be configured by changing parameters in the CubusAppServerManager.exe.config file. Next to the windows service that the setup installs multiple windows services can be set up by copying the “Performance\bin\server” directory and executing CubusAppServerManager.exe -i. Because there cannot exist multiple windows services with the same name the windows services name and description can be configured in the CubusAppServerManager.exe.config. The following configuration lines have to be added to the section “appSettings”:

```
<appSettings>
  <add key="DesktopAppUrl" value="http://cagdemo/FM/FMDesktopApp"></add>
  <add key="ServiceName" value="Performance App Server Manager"></add>
  <add key="ServiceDescription" value="Provides functionality to manage CubusAppServers"></add>
  <add key="ServerRestartTrialsInSeconds" value="0,1,5,10,20,60"></add>
  <add key="ServerManagerStartDelayInSeconds" value="1"></add>
</appSettings>
```

Parameter	Description
-----------	-------------

ServiceName	Name of the windows service when installing it by calling CubusAppServerManager.exe -i
ServiceDescription	Description of the windows service when installing it by calling CubusAppServerManager.exe -i
ServerRestartTrialsInSeconds	When a CubusAppServer that has been started by the CubusAppServerManager gets killed, the CubusAppServerManager tries to restart the CubusAppServer. The ServerRestartTrialsInSeconds parameter defines the time in seconds that the CubusAppServerManager waits before trying to restart the service. Multiple values can be defined separated by comma for the different tries. The last value is used indefinitely for the further tries.
ServerManagerStartDelayInSeconds	The time in seconds that the CubusAppServerManager waits before starting. Can be useful to assure that other services have started before the CubusAppServerManager starts.
DesktopAppUrl	This property defines the URL of the Desktop application. This setting is needed in order to start the DesktopApp from Chrome, Firefox or the Edge browser. If this setting does not exist or is incorrect, you will encounter an error message when you start the Desktop application from one of these browsers. If you are using Internet Explorer, this setting is not needed.

Manually starting a specific Application Server

Usually the Application Server is started and controlled by the Performance administration website. During development of an application it could be useful to start a single Application Server manually.

The directory `<INSTALL_DIRECTORY>/App/AbcSample/Server` contains the server start script `AbcSample_AppServer.bat`. This script contains the following commands:

```

1 @ECHO OFF
2 REM -----
3 REM Sample Application Server Start Script
4 REM -----
5 set APPLICATION_NAME=AbcSample
6 set WINDOW_TITLE="Performance AbcSample"
7
8 REM --- set log file
9 set ABC_TRACE_FILE=AppServer_%APPLICATION_NAME%.log
10
11 REM --- start server
12 ..\..\..\bin\server\CubusAppServer.exe -n %APPLICATION_NAME% -w %WINDOW_TITLE%
13

```

Note: Make sure that the OLAP server is running before you start the Application server.

Application Server Command Line Parameters

To display the parameter list of the Application Server in the console, simply enter:
 CubusAppServer -h

The following parameters are supported:

Parameter	Description	Example	Optional
-h	Returns the list of the supported command line parameters	CubusAppServer -h	✓
-n <application name>	Name of the application.	CubusAppServer -n AbcSample	
-p <cubus path>	Path to the installation directory if not provided by the installation.	CubusAppServer -p C:\NewInstallPath\Performance	✓
-w <title>	Title of the console window of the Application Server.	CubusAppServer -w "Performance Server"	✓

Connection Information

The application server needs to access the configured databases. In order to access these databases the connection strings in the configuration file CubusAppServer.exe.config and CubusAppServerManager.exe.config needs to be configured.

During the installation process of Performance these connection strings are inserted based on the information provided by the installation. If the connection to the configured databases must be changed or new databases must be added, then these strings must be changed.

```
<connectionStrings>
  <add name="ABCRepositoryConnectionString" connectionString="Data Source=cagdemo;Initial Catalog=ABCRepository;
    User ID=&quot;sa&quot;;Password=&quot;#password&quot;;" providerName="System.Data.SqlClient"></add>
  <add name="MOLAP/cagdemo" connectionString="User ID=admin;Password=password"></add>
</connectionStrings>
```

The user configured in the ConnectionString must have database owner access to the relational database.

The user for the MOLAP databases must have the access right to read the MOLAP databases.

If connecting to TM1 databases the connection string needs to be configured in the following way:

```
<add name="TM1/http://<server_name>:<Port>" connectionString="User
ID=<UserId>;Password=<Password>"></add>
<add name="TM1/https://<server_name>:<Port>" connectionString="User
ID=<UserId>;Password=<Password>"></add>
```

Configuring the Application

For the basic setup, specify the OLAP database through the following parameter in the Performance repository:

Parameter	Description	Example
PrimaryOLAPDB	Specifies the first OLAP database. Further OLAP databases may be defined via the Administration user interface.	MOLAP/localhost/ABC/Demo or ROLAP/ABC/Demo

Further configuration changes can be made via the Administration Website. The available configuration parameters and associated usage descriptions are provided in the "Configuration Parameters" section.

Synchronizing the Performance Databases

The OLAP database administration tool (e.g. Essbase Administration Services Console) is used for maintaining the OLAP model.

After changing names and/or aliases in the OLAP model, Performance needs to be synchronized with the OLAP database.

If you want to synchronize the structures without restarting the Application Server, the synchronization can be done via the Administration website.

Caution: The client cache will **not** be updated, i.e. if a client is logged in and already read the OLAPDB structures, these structures will not be updated. The client has to re-login to get the changes.

Note: The OLAP server needs to be started before the Application Server is started.

Make sure that the dimension type required by Performance for unit dimensions is defined in the OLAP database. (See "Create the OLAP model in the OLAP DB".)

Creating Modules

Performance provides the following base module types for the application:

- Workbook
- Detail Module
- Group Module
- Url (Website)
- EV Report
- EV Report with Data Entry
- Cockpit
- Reporting Services

- Workflow

In addition to these base module types also combinations of the base module types e.g. Workbook + Url exist.

Creating Workbooks

Workbooks are created in two steps:

1. Create workbooks with retrieve capability in Excel (e.g. Oracle SmartView).
2. Modify these Excel workbooks to comply with the Performance conventions.

The document “*Workbooks*” provides detailed step-by-step instructions and illustrates the capabilities of Performance workbooks.

Creating a Performance Analytics / AL Anywhere Module

Use the Performance Analytics Client to create the views. Compatible with Internet Explorer and Edge (IE Mode).

Use the AL Anywhere Client to display the views. Compatible with Chrome.

Creating a Group Module

You can provide background images for modules of type “Group”. The following graphic formats are supported: GIF (*.gif), JPEG (.jpg), bitmap (*.bmp). Use any graphics tool of your choice to create these images. Be sure to store these images in the Images folder in the IIS directory.

Creating a Detail Module

Detail modules are defined through Excel workbooks. The document “*Detail Modules*” contains detailed descriptions of the required procedures and of the capabilities provided by detail modules.

Configuring the Modules

The modules specifically created for Performance need to be stored in the application directory in the structure shown below:

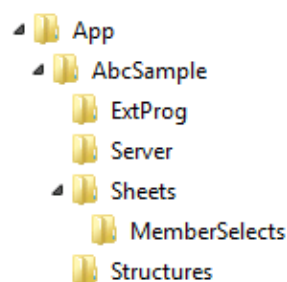


Fig. 6 - Application Directory
(Example)

The Performance administration website is used for module management (adding, removing, structuring, etc.).

Tip: It is a good idea to create a working directory for the workbooks. From this directory, you can then copy workbooks to the appropriate sheet directory for editing.

Setting up Users and Access Permissions

The User Management functionality of the administration website enables you to create users. For each user, you can set access permissions via the roles for units or for modules. Depending on the current settings, the users may automatically be added to the OLAP database.

4. Creating a ROLAP Database

The creation of an application as a ROLAP variant involves the creation of two components: a relational and a multidimensional database.

All outline information and data required are stored relationally in a star scheme.

Based on this star scheme, a multidimensional database can be set up for data analysis.

The following sections describe the relational model of the required star scheme.

Creating the Relational Model

The creation of a relational model requires the initial creation of a relational database.

The following tables need to be created in the database so that the outline information and the data required by Performance can be stored:

Table Name	Description
ABC_FACT_<application_name>	Fact table
ABC_DIM_<application_name>	Description of the dimensions
DIM_<dimension_name>	One dimension table per dimension
MEMBER_NAME	Member name
MEMBER_ALIAS	Alias name
MEMBER_UDA	Attributes (=UDA)
DIMENSION_TYPE	Description of dimension types
ALIAS_TABLE	Description of alias tables
UDA	Description of the member attributes (called "user defined attributes" = "UDA")

Additionally, various views for accessing outlines and data are used. These views are described later on in this document.

The number and designations of dimension tables depend on the defined OLAP model. The following figure shows an example of the database structure of an OLAP model "ABCBC" using the dimensions "Berichtszeilen" (=Accounts), "GJ" (=Fiscal year), "ZR" (=Time series), "Szenario" (=Scenario), "Markt" (=Market), "Produkt" (=Product) and "Organisation" (=Organisation).

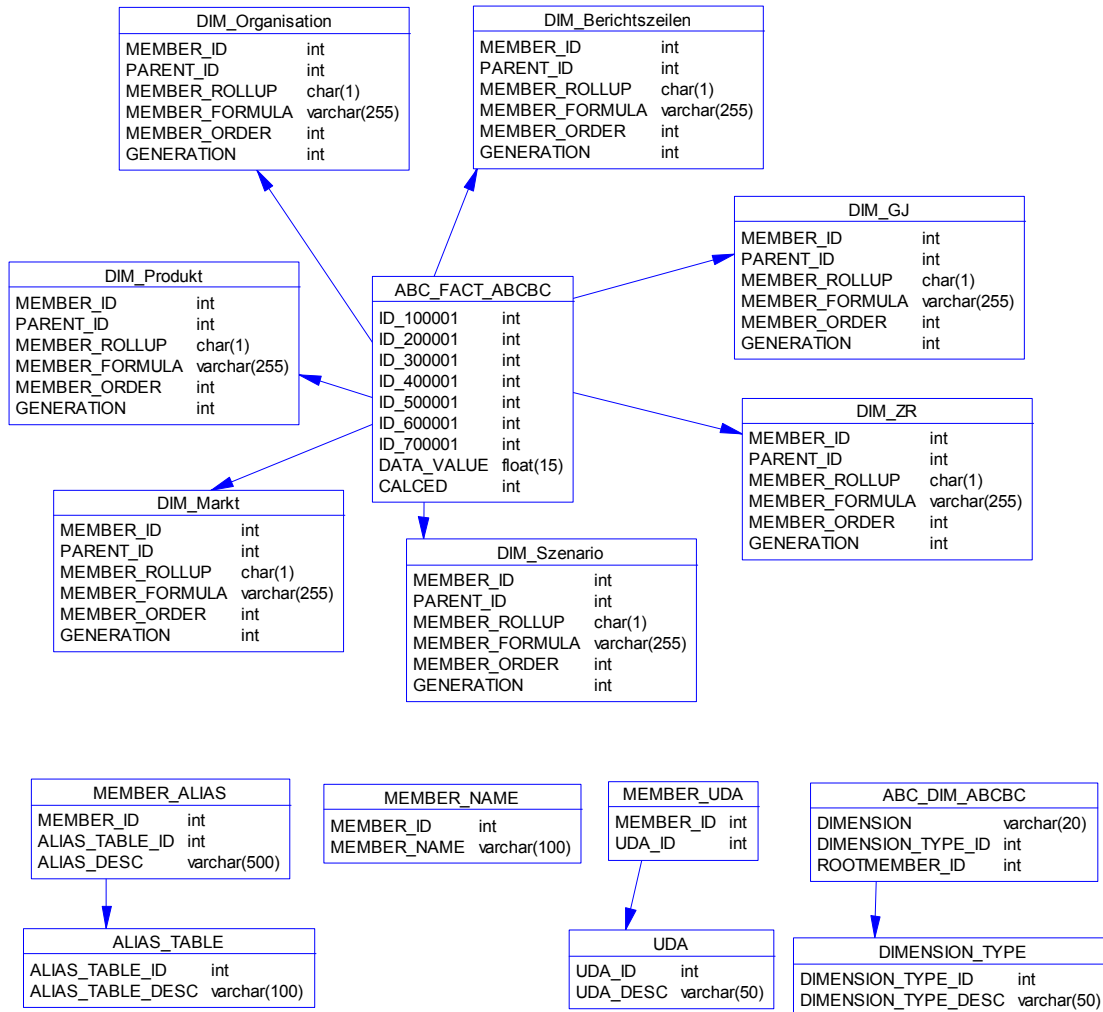


Fig. 7 – ROLAP database structure

The tables and views required for the ROLAP database are discussed in detail in the following sections.

Fact Table ABC_FACT_<application_name>

The fact table is at the core of the star scheme. It contains the data of the ROLAP model.

The name of the fact table contains the name of the application. This ensures that multiple applications can share the same relational database (see also “Mapping Multiple Applications in a Relational Database”).

Structure (Example)

	Column Name	Data Type	Length	Allow Nulls
▶	ID_100001	int	4	
	ID_200001	int	4	
	ID_300001	int	4	
	ID_400001	int	4	
	ID_500001	int	4	
	ID_600001	int	4	
	ID_700001	int	4	
	DATA_VALUE	float	8	✓
	DATA_TYPE	int	4	✓

Content

Column Name	Description
ID_<dimension_ID>	<p>A separate column needs to exist for each dimension. These columns serve for linking the fact table to the dimension tables.</p> <p>These columns have the following name format:</p> <p style="text-align: center;">ID_<dimension_ID></p> <p>where the dimension_ID corresponds to the numerical value of the first element (= root node) of the dimension.</p>
DATA_VALUE	The column contains the numerical value of the records.
DATA_TYPE	<p>Describes the value type. This column can be used e.g. for the further processing of data in MSAS.</p> <p>When storing OLAP data, Performance uses the following values:</p> <ul style="list-style-type: none"> OLAP data cells within a ABC_PRECALC_<postfix> range receive the data type value "-1" (see also the workbook description). In all other cases, the data type is set to "0".

Example

	ID_100001	ID_200001	ID_300001	ID_400001	ID_500001	ID_600001	ID_700001	DATA_VALUE	CALCED
	100369	200001	300006	400028	500027	600006	700002	0	<NULL>
	100369	200004	300006	400028	500027	600006	700002	0	<NULL>
	100369	200005	300006	400028	500027	600006	700002	0	<NULL>
	100369	200006	300006	400028	500027	600006	700002	0	<NULL>
▶	100369	200007	300006	400028	500027	600006	700002	0	<NULL>
	100369	200008	300006	400028	500027	600006	700002	0	<NULL>
	100369	200009	300006	400028	500027	600006	700002	0	<NULL>

Fig. 8 - Example of a Fact Table

Dimension Table DIM_<dimension_name>

For each dimension of the ROLAP model, a DB table

DIM_<dimension_name>

needs to be created which maps the outline information as a parent/child relationship.

Dimension tables at least need to have one child item, additional to the root / parent item. Also all items need to be organized in a hierarchy. No unattached items are allowed.

All dimension tables have the same structure with the following columns:

Structure

	Column Name	Data Type	Length	Allow Nulls
	MEMBER_ID	int	4	✓
	PARENT_ID	int	4	✓
	MEMBER_ROLLUP	char	1	✓
	MEMBER_FORMULA	varchar	1000	✓
	GENERATION	int	4	✓
	MEMBER_ORDER	int	4	✓

Content

Column	Description
MEMBER_ID	Unique numerical identifier for the member from the MEMBER_NAME table
PARENT_ID	Parent member (parent node)
MEMBER_ROLLUP	Consolidation character (+, -, *, /)
MEMBER_FORMULA	Formula for calculating a member (for MSSQL in MDX format)
MEMBER_ORDER	Member sequence among sibling nodes
GENERATION	Generation number of the member

Example

	MEMBER_ID	PARENT_ID	MEMBER_ROLLUP	MEMBER_FORMULA	MEMBER_ORDER	GENERATION
▶	200002	200001	+		200002	2
	200003	200001	+	<NULL>	200003	2
	200004	200002	+	<NULL>	200004	3
	200005	200002	+	<NULL>	200005	3
	200006	200004	+	<NULL>	200006	4
	200007	200004	+	<NULL>	200007	4
	200008	200004	+	<NULL>	200008	4
	200009	200005	+	<NULL>	200009	4

Fig. 9 - Example of a Dimension Table

A unique numerical identifier needs to be defined for each member. This numerical identifier is used internally for accessing the outlines and data. For example, the relationship between the dimension table and the fact table is as follows:

<fact_table>.ID_<dimension_ID> → <dimension_table>.MEMBER_ID

To simplify the assignment of members to dimensions, it is recommended to define a range of numbers for each dimension.

Example:

The following figure illustrates the relationship between the fact table and the dimension table DIM_GJ. Note that the ID of the root node of the dimension GJ is “200001”. Each member of the dimension GJ has a unique numerical ID between “200002” and “299999”.

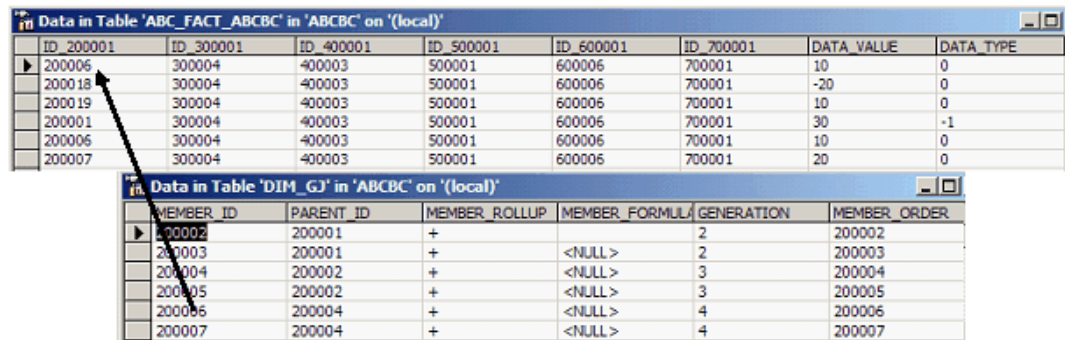


Fig. 10 - Example illustrating the relationship between Fact Table and Dimension Table

Table ABC_DIM_<application_name>

The table ABC_DIM_<application_name> contains an entry for each dimension.

The name of this table also includes the name of the application. This ensures that multiple applications can share the same relational database (see also “Mapping Multiple Applications in a Relational Database”).

This table includes the following columns:

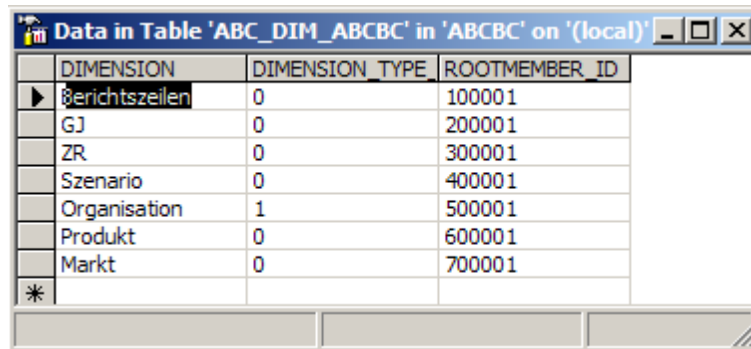
Structure

Column Name	Data Type	Length	Allow Nulls
DIMENSION	varchar	20	
DIMENSION_TYPE_ID	int	4	✓
ROOTMEMBER_ID	int	4	✓

Content

Column	Description
DIMENSION	Name of the dimension
DIMENSION_TYPE_ID	Dimension type as per DIMENSION_TYPE table Note: The dimension of the unit needs to be given the ID of the dimension type "ABC_UNIT".
ROOTMEMBER_ID	Number range and dimension ID of the dimension

Example



DIMENSION	DIMENSION_TYPE	ROOTMEMBER_ID
Berichtszeilen	0	100001
GJ	0	200001
ZR	0	300001
Szenario	0	400001
Organisation	1	500001
Produkt	0	600001
Markt	0	700001
*		

Fig. 11 - Example of ABC_DIM_<application_name>

DIMENSION_TYPE Table

The DIMENSION_TYPE table contains the definitions of the dimension types. This table is structured as follows:

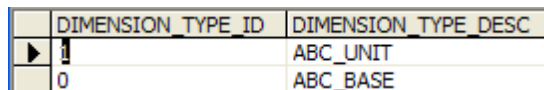
Structure

	Column Name	Data Type	Length	Allow Nulls
	DIMENSION_TYPE_ID	int	4	✓
	DIMENSION_TYPE_DESC	varchar	50	✓

Content

Column	Description
DIMENSION_TYPE_ID	Unique dimension type identifier
DIMENSION_TYPE_DESC	Dimension type description

Example



DIMENSION_TYPE_ID	DIMENSION_TYPE_DESC
1	ABC_UNIT
0	ABC_BASE

Fig. 12 - DIMENSION_TYPE table

Currently, the following dimension types are supported:

- ABC_BASE (=0)
- ABC_UNIT (=1)
- ABC_MONTH (=2)
- ABC_YEAR (=3)

Note: The dimension of the units needs to be given the ID of the dimension type "ABC_UNIT".

Table MEMBER_NAME

In Performance, member names are used to facilitate the maintenance of workbooks and of the configuration (e.g. the definition of access permissions).

However, for internal access to the outline information and data, Performance uses the numerical member IDs; for instance, the fact table contains the member IDs.

The assignment of member IDs and member names to all members is contained in the MEMBER_NAME table.

Note: Both the member IDs and the member names need to be unique.

Structure

	Column Name	Data Type	Length	Allow Nulls
	MEMBER_ID	int	4	
	MEMBER_NAME	varchar	100	✓

Content

Column	Description
MEMBER_ID	Unique numerical member identifier
MEMBER_NAME	Unique member name Note: Spaces at the beginning or end are not allowed.

Example

Member_ID	Member_Name
100001	Berichtszeilen
100002	Absatz/Umsatz
100003	Invest
100004	Personal
100005	Kosten
100006	GuV
100007	Bilanz
100008	CashFlow
100009	Kennzahlen

Fig. 13 - MEMBER_NAME example

Changing Member Names

Be aware of the following when changing member IDs and member names:

- The numerical member IDs serve for the basic identification of outline members. These IDs are used for the internal linkage of the various tables. Consequently, changes to these IDs imply changes to all dependent tables. Therefore, the numerical IDs should be changed only if it is absolutely necessary. Great care must be taken. However, for the workbooks

and the configuration, changes to member IDs are not relevant.

- Changes to member names only need to be performed in the MEMBER_NAME table. Following the change of the member names in the MEMBER_NAME table, this change is automatically applied during the outline update in Performance or after the processing of an OLAP cube in Microsoft Analysis Services (MAS). It may still be necessary to adjust the workbooks and the configuration (definition of access permissions etc.).

Table MEMBER_ALIAS

A member can have one or multiple alias names. These are registered in the MEMBER_ALIAS table.

An alias name is an alternative name for a member. Alias names can be used for purposes such as multi-language naming.

Structure

	Column Name	Data Type	Length	Allow Nulls
	MEMBER_ID	int	4	
	ALIAS_TABLE_ID	int	4	
	ALIAS_DESC	varchar	100	

Content

Column	Description
MEMBER_ID	Unique numerical member identifier
ALIAS_TABLE_ID	Numerical identifier of the alias table
ALIAS_DESC	Alias name

Example

	MEMBER_ID	ALIAS_TABLE_ID	ALIAS_DESC
▶	100376	1	Rechts- und Beratungskosten VT
	100377	1	Gesellschaftsrechtliche Aufwendungen VT
	100378	1	Personalbeschaffungskosten VT
	100379	1	Personalentwicklungskosten VT
	100380	1	Externe Arbeitskräfte VT
	100381	1	Sonstige Verwaltungsaufwendungen VT
	100382	1	Sonstige betriebliche Aufw. VT
	100383	1	Verluste Anlagenabgang VT

Fig. 14 - MEMBER_ALIAS example

Table MEMBER_UDA

A member can have one or multiple attributes (known as "user defined attributes" = UDA). These are registered in the MEMBER_UDA table.

Structure

	Column Name	Data Type	Length	Allow Nulls
	MEMBER_ID	int	4	
	UDA_ID	int	4	✓

Content

Column	Description
MEMBER_ID	Unique numerical member identifier
UDA_ID	Unique numerical attribute identifier

Example

	MEMBER_ID	UDA_ID
▶	100006	10002
	100007	10002
	100008	10002

Fig. 15 - MEMBER_UDA example

Table ALIAS_TABLE

The ALIAS_TABLE table defines all existing alias tables.

An alias is an alternative name for a member. Alias names can be used for purposes such as multi-language naming. The “alias table” is a collection of all alias names in a given language.

Structure

	Column Name	Data Type	Length	Allow Nulls
	ALIAS_TABLE_ID	int	4	
	ALIAS_TABLE_DESC	varchar	100	✓

Content

Column	Description
ALIAS_TABLE_ID	Numerical identifier of the alias table
ALIAS_TABLE_DESC	Name of the alias table

Performance uses the numerical alias table ID for internal access only. However, the alias table name can be specified in the URL that is used for calling the client web page.

Example

ALIAS_TABLE_ID	ALIAS_TABLE_DESC
1	Default
2	English

Fig. 16 - ALIAS_TABLE example

Table UDA

The UDA table contains the descriptions of the attributes (“user defined attributes“ = UDAs).

Structure

Column Name	Data Type	Length	Allow Nulls
UDA_ID	int	4	
UDA_DESC	varchar	50	✓

Content:

Column	Description
UDA_ID	Unique numerical attribute identifier
UDA_DESC	Attribute (=UDA)

Example:

UDA_ID	UDA_DESC
10000	FC
10001	Handelsware
10002	ND
300001	DET_MON

Fig. 17 - UDA example

Views

View V_DIM_<dimension_name>

Since the outline information is distributed across multiple tables (DIM_<dimension_name>, MEMBER_NAME, MEMBER_ALIAS), a view that presents this information collectively needs to be created for each dimension. When reading the dimension information, Performance always accesses these views rather than the dimension table itself.

The views do not contain any information on members of generation 1 (generation 1 corresponds to the dimension). These are read from the table ABC_DIM_<application_name>.

Name

V_DIM_<dimension_name>

Structure (example: V_DIM_GJ)

Column	Alias	Table	Output
MEMBER_ID		DIM_Berichtszeiler	✓
PARENT_ID		DIM_Berichtszeiler	✓
MEMBER_NAME		MEMBER_NAME	✓
MEMBER_NAME	Parent_Member_Name	MEMBER_NAME_1	✓
MEMBER_ROLLL		DIM_Berichtszeiler	✓
MEMBER_FORM		DIM_Berichtszeiler	✓
GENERATION		DIM_Berichtszeiler	✓
MEMBER_ORDE		DIM_Berichtszeiler	✓

Content

Column	Description
MEMBER_ID	Numerical member identifier
PARENT_ID	Numerical identifier of the parent node
MEMBER_NAME	Member name
PARENT_MEMBER_NAME	Member name of the parent node
MEMBER_ROLLUP	Consolidation character (+,-,*,/)
MEMBER_FORMULA	Formula for calculating the member (for MSSQL in MDX format)
GENERATION	Generation number of the member
MEMBER_ORDER	Sequence of members among its sibling nodes

Definition of the view (example: V_DIM_GJ)

```

CREATE VIEW dbo.V_DIM_GJ
AS
SELECT
    dbo.DIM_GJ.MEMBER_ID,
    dbo.DIM_GJ.PARENT_ID,
    dbo.MEMBER_NAME.MEMBER_NAME,
    MEMBER_NAME_1.MEMBER_NAME AS PARENT_MEMBER_NAME,
    dbo.DIM_GJ.MEMBER_ROLLUP,
    dbo.DIM_GJ.MEMBER_FORMULA,
    dbo.DIM_GJ.GENERATION,
    dbo.DIM_GJ.MEMBER_ORDER
FROM    dbo.DIM_GJ
INNER JOIN
    dbo.MEMBER_NAME ON dbo.DIM_GJ.MEMBER_ID =
    dbo.MEMBER_NAME.MEMBER_ID

```

INNER JOIN

```

                                dbo.MEMBER_NAME MEMBER_NAME_1 ON
dbo.DIM_GJ.PARENT_ID = MEMBER_NAME_1.MEMBER_ID

```

Result:

MEMBER_ID	PARENT_ID	MEMBER_NAME	PARENT_MEMBER_NAME	MEMBER_ROLLUP	MEMBER_FORMULA	GENERATION	MEMBER_ORDER
200002	200001	HJ1	GJ	+		2	200002
200003	200001	HJ2	GJ	+	<NULL>	2	200003
200004	200002	Q1	HJ1	+	<NULL>	3	200004
200005	200002	Q2	HJ1	+	<NULL>	3	200005
200006	200004	M01	Q1	+	<NULL>	4	200006
200007	200004	M02	Q1	+	<NULL>	4	200007
200008	200004	M03	Q1	+	<NULL>	4	200008
200009	200005	M04	Q2	+	<NULL>	4	200009

View V_MEMBER_ALIAS

By analogy to the view V_DIM_<dimension_name>, a view needs to be created for the linkage of the member names and alias names.

Structure

Column	Alias	Table	Output
MEMBER_NAME		MEMBER_NAME	✓
ALIAS_TABLE_ID		MEMBER_ALIAS	✓
ALIAS_DESC		MEMBER_ALIAS	✓

Content

Column	Description
MEMBER_NAME	Member name
ALIAS_TABLE_ID	Numerical identifier of the alias table
ALIAS_DESC	Alias name

Example

Definition of the view (MS SQL Server):

```

CREATE VIEW dbo.V_MEMBER_ALIAS
AS
SELECT
                                dbo.MEMBER_NAME.Member_Name,
                                dbo.MEMBER_ALIAS.ALIAS_TABLE_ID,
                                dbo.MEMBER_ALIAS.ALIAS_DESC
FROM
                                dbo.MEMBER_ALIAS
INNER JOIN
                                dbo.MEMBER_NAME ON dbo.MEMBER_ALIAS.MEMBER_ID =
                                dbo.MEMBER_NAME.Member_ID

```

View V_MEMBER_UDA

By analogy to the view V_DIM_<dimension_name>, a view needs to be created for the linkage of the member names and the attributes (UDAs).

Structure

Column	Alias	Table	Output
MEMBER_NAME		MEMBER_NAME	✓
UDA_ID		MEMBER_UDA	✓

Content

Column	Description
MEMBER_NAME	Member name
UDA_ID	Attribute identifier (UDA ID)

Example

Definition of the view (MS SQL Server):

```
CREATE VIEW dbo.V_MEMBER_UDA
AS
SELECT
    dbo.MEMBER_NAME.Member_Name,
    dbo.MEMBER_UDA.UDA_ID
FROM
    dbo.MEMBER_NAME
INNER JOIN
    dbo.MEMBER_UDA ON dbo.MEMBER_NAME.Member_ID =
    dbo.MEMBER_UDA.MEMBER_ID
```

Administration of Members and Dimensions

Adding a New Dimension

The addition of a new dimension involves the following tables:

- ABC_FACT_<application_name>
- ABC_DIM_<application_name>

The required entries need to be added to these tables (refer to the table descriptions earlier in this document).

Next, a dimension table DIM_<dimension_name> needs to be created (described earlier in this document).

Deleting a Dimension

The deletion of a dimension involves the following tables:

- ABC_FACT_< application_name >
- ABC_DIM_< application_name >

The applicable entries need to be removed from these tables (refer to the table descriptions earlier in this document).

If the dimension table is no longer required, it can be deleted. Before doing so, make sure that the dimension table is not used elsewhere.

Adding a member

The addition of a member involves the following tables:

- MEMBER_NAME
- MEMBER_ALIAS
- MEMBER_UDA
- ABC_FACT_<application_name>
- DIM_<dimension_name>

If a new member is to be added, a valid numerical identifier needs to be specified for the new member, and the new member needs to be added to the MEMBER_NAME and MEMBER_ALIAS tables.

The new member needs to be included in the dimension outline via the dimension table. This is done by assigning a parent node to the member.

The new member name is automatically applied during the outline update in Performance or after the processing of a cube in Microsoft Analysis Services (MSAS).

Deleting a member

The deletion of a member involves the following tables:

- MEMBER_NAME
- MEMBER_ALIAS
- MEMBER_UDA
- ABC_FACT_<application_name>
- DIM_<dimension_name>

Before a member can be deleted, all dependent data with the associated member ID needs to be deleted from the database.

Note: When the entries in the fact table are deleted for a member ID, all data for this member is deleted. This is like removing a “slice” from the cube.

Mapping Multiple Applications in a Relational Database

The mapping of multiple applications to the same relational database provides the following advantages:

- Less effort for outline data maintenance
- No redundancies of outline tables

The "OLAP-DB" configuration parameter for the ROLAP DB is used to define the database name and the name of the Performance application

Syntax: ROLAP/<PerformanceAppName>/<ROLAPAppName>

Example: ROLAP/AbcSampleRolap/AbcSampleRolap

The name of the Performance application is a logical name that serves for identifying the associated fact and dimension tables. I.e. in order to be able to load the associated outline data in Performance, the following tables need to be created for each application:

ABC_FACT_<ROLAPAppName> and
 ABC_DIM_<ROLAPAppName>

(also refer to the table descriptions earlier in this document).

ROLAP Configuration

To use Performance with ROLAP databases the CubusAppServerManager.exe.config file first needs to be configured correctly. The first step is to configure the authentication providers. The authentication providers are responsible for authenticating users. An example ROLAP authentication provider can be seen below. The default Essbase authentication provider needs to be replaced with the ROLAP provider authentication provider.

```
<authenticationProviders>
  <provider
type="Cubus.Abc.Services.AuthenticationServices.Provider.SqlProvider,
      Cubus.Abc.Services"
      connectionString="Data Source=DbServer;
                      Initial Catalog=ABCRepository_ROLAP;
                      Persist Security Info=True;" />
</authenticationProviders>
```

It is important to configure the connection string to access the correct server and database. The user that is entered on the login screen will be authenticated using the authentication provider that is configured.

Next the structure provider and connection string must be specified. This also has to be configured in the CubusAppServer.exe.config.

```
<structureProvider>
  <provider id="DefaultROLAP"
      type="Cubus.Abc.Services.StructureServices.FileProvider,
            Cubus.Abc.Services">
    <olapProvider type="Cubus.Abc.Services.RolapProviderService,
                      Cubus.Abc.Services">
      </olapProvider>
    </provider>
  </structureProvider>
```

The structure provider defines which Rolap provider will be used for the OLAP database. It is important that the correct OLAP provider type is defined as shown above. This is a special type for ROLAP databases.

The path where the xml file will be stored follows the convention:

```
<PerformanceInstallPath>\App\<ApplicationId>\Structures  
\<ApplicationId>_<OlapDb>.xml
```

Additionally the ROLAP connection string must be configured where the id follows the convention **ROLAP/<ApplicationId>/<FactTablePostFix>**. For a fact table named ABC_FACT_RolapSample the FactTablePostFix would be RolapSample.

```
<connectionStrings>  
  <add name="ROLAP/SampleApplication/RolapSample"  
        connectionString="Data Source=DbServer;  
                          Initial Catalog=ABCRepository_ROLAP;  
                          Persist Security Info=True;  
                          User ID=DbUser;  
                          Password=DbPassword"  
        providerName="System.Data.SqlClient" />  
</connectionStrings>
```

5. Administration Website

This chapter focuses on the functionality of the Administration Website in order to show the versatility of Performance.

The Administration Website consists of different sections.

The top section contains the global administration tasks for all applications:

- Applications
- Global Parameter (valid for all applications)
- Application Parameter (per application)
- Concurrent Users
- Log Viewer
- Styling
- Session
- Create DesktopApp package

The middle section displays the application list of all configured applications and enables the user to perform the administration tasks for each application:

- Module
- Parameter
- User
- Role
- Event
- Submitted objects
- Unit
- Unit lock

Performance distinguishes between a global administrator and application specific administrators. The global administrator has access to all Performance applications, while application specific administrators only have access to the related applications. (See the user and role access rights for further details.)

The figure below shows the Administration Website entry screen.

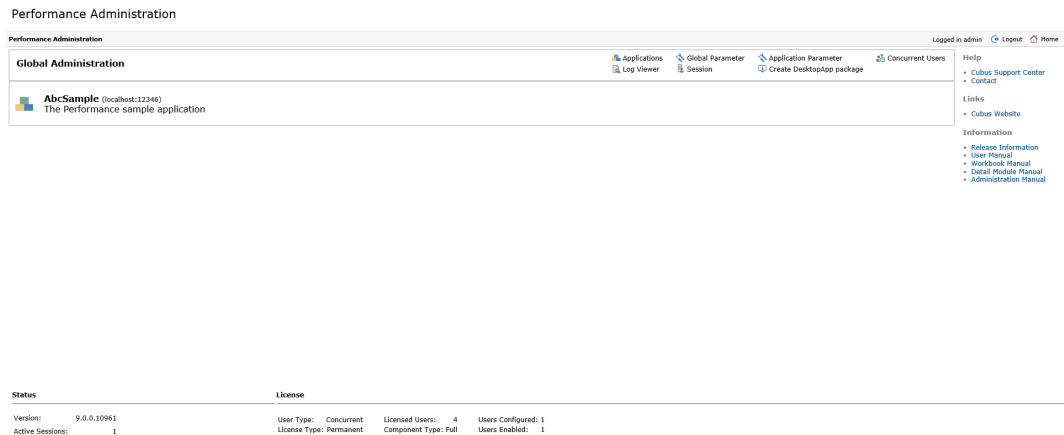


Fig. 18 - Administration Website Main Screen

The following sections provide detailed information on each menu item in the individual dialogs.

Application List

The middle section displays all Performance applications in the system. Every time a new application is added, it appears in this list. To configure the application select the corresponding administration task of the application. You can navigate to the application by simply clicking on the application name.

Global Administration - Application

The top section is responsible for the configuration of all applications.

The figure below shows the application administration dialog.



Fig. 19 - Application dialog

The following list provides an overview of the application properties.

Property	Description
AppID	The unique identification for the applications. This is normally a suitable application name.
ConnectionString	The applications connection string. The string contains the connection information to the CubusAppServer server. The format is <servername>:<port>.
Description	A description of the application.
StartType	The StartType defines if the application server is started automatically with Windows (Automatic) or manually by the user. An additional StartType is "Disabled".
Status	The current status of the application server.

Creating an Application

To create an application select "New" from the menu on the left hand side of the screen. This will then open a new dialog where the application details can be added.

The ApplicationId and ConnectionString are required fields, description is an optional field but it is recommended when there are many different applications. Click "Save" when all fields have been specified. The new application will then appear in the table and also in the application list on the main screen.

Editing an Application

Applications can be edited or updated by selecting an application from the application table and clicking on "Edit" in the menu items. The application description, connection string and start type can be modified.

Note: To rename an application please use the "Rename" option.

Deleting an Application

To delete an application, select the application that you wish to delete from the overview table. When the application is selected click on the "Delete" item in the menu.

A message box will then be displayed asking if you really want to delete the application. Press "OK" to continue and "Cancel" if you no longer wish to delete the application.

Note: If you delete an application all relevant application data stored in the relational database including all modules and roles is deleted. The application directory containing the workbooks, the underlying OLAP or ROLAP database and the Performance Analytics objects are not deleted.

Renaming an Application

To rename an application, select the application that you wish to rename from the application table. Then click on the "Edit" menu item. Enter the new application name in the appropriate text box and click "Save".

Note: By renaming applications no specific application data will be lost, only the application name will be changed.

After the application has been renamed in the repository you have to rename the application directory manually and restart the application server.

Exporting Application Data

Application specific data can easily be exported to an XML file. This is done by selecting which part of the application you wish to export in the “Action Type” dropdown list and then clicking on the “Export” menu item.

Export Type	Description
Application	All relevant application data is exported.
Module	Only Module data is exported.
Unit	Only Unit data is exported.
Role	Only Role data is exported.
User	Only User data is exported.

Importing Application Data

Application data can also be imported using the same principle. Select the type of data file (e.g. MyApp.xml) that needs to be imported. For example, Application or Module. Then click on the “Import” menu item.

Please be aware that importing module data deletes all existing module data first for consistency reasons. This also applies to the Role-to-Model assignment. Make sure to export your Role data as well if you want to use it further.

Configuration Parameters

The functionality of Performance can be customized by means of a number of configuration parameters.

Configuration parameters are divided in four categories.

1. **Global** configuration parameters
2. **Application specific** parameters
3. **Module specific** parameters
4. **Workbook** parameters

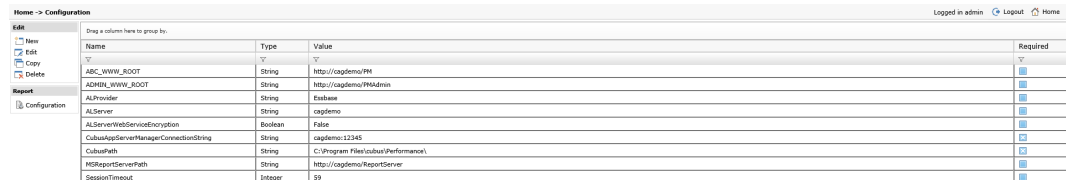
The *global* and *application specific* configuration parameters are used to customize the behaviour of the application (e.g. UsePersonalizedOutlines) or to get installation information (e.g. ABC_WWW_ROOT). *Global* parameters are available and valid for all applications. *Application specific* parameters are only available for the individual application. If a parameter is specified in the global parameter section and in the application section, the application setting overrules the global one.

Module specific parameters are defined in the Performance Admin for a single module using the Specifics tab. The detailed description of module specific parameters can be found in the

chapter [Module Administration - Specifics](#).

Configuration parameters are also used within workbooks or detail modules. Almost every workbook related configuration parameter can also be defined as a global configuration parameter, so that the setting of this parameter is used for all workbooks or detail modules. The detailed description of *workbook configuration* parameters can be found in the Workbook Documentation.

Global Parameter Dialog



Name	Type	Value	Required
ABC_WWW_ROOT	String	http://capdemo/PM	Y
ADMIN_WWW_ROOT	String	http://capdemo/PMAdmin	
ALProvider	String	Enbase	
ALServer	String	capdemo	
ALServerWebServiceEncryption	Boolean	False	
CacheServerManagerConnectionString	String	capdemo:3344	
CubaPath	String	C:\Program Files\Cubus\Performance\	
MSReportServerPath	String	http://capdemo/ReportServer	
SessionTimeout	Integer	59	

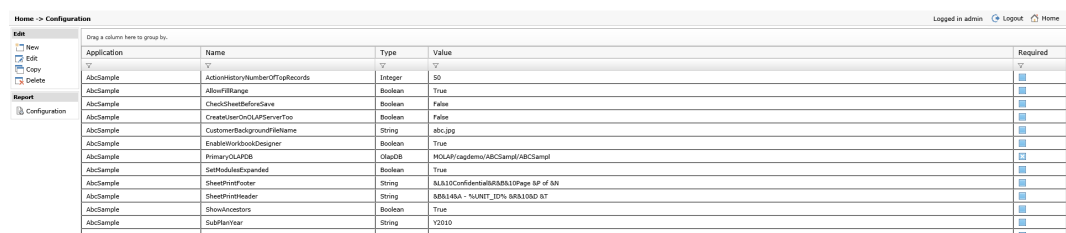
Fig. 20 - Global Configuration Parameter Dialog

The following properties are available in the Configuration Parameters dialog:

Property	Description
Name	This field contains the name of the selected configuration parameter.
Type	The following parameter types are supported: String, Integer, Boolean, Double, OLAPDB. Note: Parameter type double supports up to 4 decimal digits.
Value	This field displays the value of the selected configuration parameter. This value can be changed as required.
Required	Specifies if the parameter is required or not. If the parameter is required, it cannot be deleted.

Application Parameter Dialog

In addition to the global configuration parameter dialog the application parameter dialog contains the list of all parameters for **all** applications.



Application	Name	Type	Value	Required
ABC_Sample	ActionHistoryNumberOfTingRecords	Integer	50	Y
ABC_Sample	ALBofBRange	Boolean	True	
ABC_Sample	CheckSheetBeforeSave	Boolean	False	
ABC_Sample	CreateUserOnOLAPServerToo	Boolean	False	
ABC_Sample	CustomizeBackgroundFilename	String	abc.jpg	
ABC_Sample	EnableModuleDesigner	Boolean	True	
ABC_Sample	PrimaryOLAPDB	OLAPDB	MOLAP/capdemo/ABC_Sample/ABC_Sample	
ABC_Sample	SelfModuleExpanded	Boolean	True	
ABC_Sample	SheetPrintFooter	String	M.A.10Confidential&R&B&L10Page 8P of 8N	
ABC_Sample	SheetPrintHeader	String	88&L4&A - %NUNIT_ID% 88&L10&D 8T	
ABC_Sample	ShowInOptions	Boolean	True	
ABC_Sample	SQLServer	String	Y2010	
ABC_Sample	UserPersonalizedOutlines	Boolean	True	

Fig. 21 - Application Configuration Parameter Dialog (All Applications)

This dialog is identical to the global configuration parameter dialog. It has an additional property for the application name.

Configuration Parameter Menu Items

The following toolbar buttons are available in the Configuration Parameters dialog:

- **New**
Creates a new configuration parameter.
- **Edit**
Updates the selected configuration parameter.
- **Copy**
Copies the selected configuration parameters.
- **Delete**
Deletes the selected configuration parameter.
- **Report**
Display a report of the configuration parameters, which can be printed or exported in PDF format.

Copying Configuration Parameters

Configuration parameters can be copied between applications. To do this, select one or more items from the overview table then click on the “Copy” menu item.

A new dialog will be displayed specifying the configuration parameters that are to be copied.

To copy the selected configuration parameters, select a target application and click the copy button. If one or more configuration parameters already exist in the target application, a message is displayed. In addition to this message the button “Force Copy” is displayed. When pressing this button the selected configuration parameters will overwrite the existing parameters.

List of Configuration Parameters

Performance supports the following global and application specific configuration parameters:

ABC_WWW_ROOT

Type: *String*

This configuration parameter must be set to the URL of the web application (e.g. `http://cubusserver/PM`).

This parameter is needed internally.

AbcVirtualPath

Type: *String*

This configuration parameter contains the web application name. If the ABC_WWW_ROOT is set to `http://cubusserver/cubus`, the AbcVirtualPath is `cubus`.

ActionHistoryNumberOfTopRecords

Type: *Integer*
Default: *50*

This parameter determines the maximum number of entries in the recent modules and detailed history.

AdditionalApplicationExportTables

Type: *String*
Default: *Empty*

This parameter can be used to define additional tables that shall be included in an application export. A default application export will only contain structure tables, no state or detail tables. To include such tables, set this parameter using a comma separated list of the table names without whitespaces.

Example: ABC_SUBMITTED_OBJECTS,DET_WBT_EMPL,Notes

ADMIN_WWW_ROOT

Type: *String*
Default: *http://localhost/PMAdmin*

This configuration parameter must be set to the URL of the administration web application (e.g. <http://cubusserver/PMAdmin>).

This parameter is needed internally.

AllowCommentsOnProtectedCells

See Workbook Documentation for detailed description.

AllowFillRange

See Workbook Documentation for detailed description.

ALProvider

Type: *String*
Default: *Essbase*

This configuration parameter is used to specify the Performance Analytics authentication method. This configuration parameter has the following possible values:

- Essbase
- Windows
- TM1
- OAuth2.0
- OIDC

ALServer

Type: *String*
Default: *localhost*

This configuration parameter is used to specify where the Performance Analytics server is running.

Note: If [ALServerCommentsHost](#) is set this parameter is not used to specify the Performance

Analytics server from which the comments are retrieved.

ALServerCommentsHost

Type: *String*
Default: *same value as ALServer*

This parameter specifies the address of the Performance Analytics server from which comments are to be loaded. If it is not set then the [ALServer](#) parameter is used.

Note: This parameter can be used if Performance and Performance Analytics run behind a reverse proxy and have a different internal and external address.

ALServerRestApiPort

Type: *Integer*
Default: *80*

This parameter defines the REST API Port that is used to connect to the Performance Analytics REST API when using AL Anywhere modules.

ALServerWebServiceEncryption

Type: *Boolean*
Default: *False*

If this parameter is set to *True*, the connection to the ALServer WebService will be encrypted.

Note: If AppServerSchema is set to HTTP *ALServerWebServiceEncryption* has to be set to *False*.

If AppServerSchema is set to HTTPS *ALServerWebServiceEncryption* has to be set to *True*.

ALTheme

Type: *String*
Default: *Ribbon*

This configuration parameter is used to specify the theme for the Performance Analytics reports.

AlwaysShowTreeSelection

Type: *Boolean*
Default: *False*

This configuration parameter is used to define if selection trees for modules, units and off-grid selections should be visible. The default value is *False*. If this value is set to *False*, this means, if there is only one accessible member in the selection tree, then the selection is hidden. If this value is set to *True*, selection trees will always be shown, even if there is only one accessible member.

AppServerCompression

Type: *Boolean*
Default: *True*

This configuration parameter determines whether the data transferred between the CubusAppServer and its clients (WorkbookControl, Web application) is compressed or not. Compression can significantly decrease the amount of data being transferred over the network. This can improve performance especially in environments with low network bandwidth. In environments with high network bandwidth compression could even downgrade performance slightly.

If this parameter is set to *True* the data transferred between the CubusAppServer and its clients

will be compressed.

If this parameter is set to *False* the data transferred between the CubusAppServer and its clients will not be compressed.

Note:

- In order to use compression the .NET Framework 4.5 must be installed.
- Changing this parameter requires the CubusAppServerManager.exe (Windows Service Name = Performance App Server Manager) to be restarted to take effect.

AppServerEncryption

Type: *String*
Default: *Windows*

By default the data transfer between the workbook client and the CubusAppServer is encrypted with TCP over SSL. This is also true for the data transfer between the web application and the CubusAppServer and the CubusAppServerManager.

The global configuration parameter *AppServerEncryption* applies to the CubusAppServerManager and to each CubusAppServer if no application specific configuration parameter for *AppServerEncryption* is configured.

Configure encryption

If you change the global parameter *AppServerEncryption*, the CubusAppServerManager connection strings in the web.config files for the Performance Client and the Performance Admin as well as in the *EVServer.exe.config* need to be adapted. Change the parameter *Encryption* according to the value of the global parameter *AppServerEncryption* (Parameters are separated by semicolon.)

Valid values

- *None* - The connection will not be encrypted.
- *Windows* - The connection will be encrypted using Windows authentication. This is only possible if server and client are in the same Windows domain.
- *Crossdomain* - The connection will be encrypted using a custom encryption mechanism. This parameter can be used if client and server are not in the same windows domain.

Note:

- *AppServerEncryption* is only used if the *AppServerSchema* is set to TCP and ignored for all other *AppServerSchema* parameters. If the *AppServerSchema* is set to HTTP the connection will not be encrypted like when *AppServerEncryption* is set to None. If the *AppServerSchema* is set to HTTPS then SSL will be used for encryption.
- Changing this parameter requires the CubusAppServerManager (the Windows Service) to be restarted to take effect.

AppServerSchema

Type: *String*
Default: *TCP*

This configuration parameter defines which protocol is used to exchange data between the workbook client and the app server. It also applies to the communication between CubusAppServer and CubusAppServerManager.

Configure schema

If you change the global parameter `AppServerSchema`, the `CubusAppServerManager` connection strings in the `web.config` files for the Performance Client and the Performance Admin as well as in the `EVServer.exe.config` need to be adapted. Change the parameter `Schema` according to the value of the global parameter `AppServerSchema` (Parameters are separated by semicolon.)

Valid values

- `TCP`
- `HTTP`
- `HTTPS`

Note:

- Use of `HTTPS` as `AppServerSchema` requires a trusted TLS certificate to be added to each machine running a `CubusAppServer` or `CubusAppServerManager`. This certificate has to be added on the port configured in `CubusAppServerManagerConnectionString`.
- If `HTTP` or `HTTPS` is used Performance can use the same port as existing IIS Web Sites.
- Changing this parameter requires the `CubusAppServerManager` (the Windows Service) to be restarted to take effect.
- If any Batch Files are used while `HTTP` or `HTTPS` is configured then the [SET_MANAGERPORTID](#) parameter has to be set before using the `LOGIN` command.

AppServerRestSchema

Type: *String*
Default: *HTTP*

This configuration parameter defines which protocol is used to exchange data between the Workbook Anywhere and the app server.

Valid values

- `HTTP`
- `HTTPS`

Note:

- Use of `HTTPS` as `AppServerSchema` requires a trusted TLS certificate to be added to each machine running a `CubusAppServer` or `CubusAppServerManager`. This certificate has to be added on the port configured in `CubusAppServerManagerConnectionString`.
- If `HTTP` or `HTTPS` is used Performance can use the same port as existing IIS Web Sites.
- Changing this parameter requires the `CubusAppServerManager` (the Windows Service) to be restarted to take effect.

AppServerRestPort

Type: *Integer*
Default: *80 if [AppServerRestSchema](#) is set to HTTP/443 if [AppServerRestSchema](#) is set to HTTPS*

This configuration parameter defines which port is used to exchange data between the Workbook Anywhere and the app server.

Note:

- Changing this parameter requires the CubusAppServerManager (the Windows Service) to be restarted to take effect.

AppServerWebAlias

Type: *String*
Default: *PMApplications*

This configuration parameter specifies which HTTP/HTTPS path is used to access CubusAppServer and CubusAppServerManager internally.

By default a CubusAppServer can be reached via `http(s)://hostname/PMApplications/AppService_{port}`, while a CubusAppServerManager can be reached via `http(s)://hostname/PMApplications/AppServiceManager`. This can be changed by setting `AppServerWebAlias`.

Configure web alias

If you change the global configuration parameter `AppServerWebAlias`, the CubusAppServerManager connection strings in the `web.config` files for the Performance Client and the Performance Admin as well as in the `EVServer.exe.config` need to be changed as well. Add the parameter `AppServerWebAlias` according to the value of the global parameter `AppServerWebAlias` (Parameters are separated by semicolon.)

Note:

- `AppServerWebAlias` is only used when `AppServerSchema` is set to HTTP or HTTPS. If TCP is used this parameter is ignored.
- Changing this parameter requires the CubusAppServerManager (Windows Service) to be restarted to take effect.
- `AppServerWebAlias` doesn't have to be added to the connection string in the `web.config` files for the Performance Client and Performance Admin if the default value is used.

AutoInsertNewRow

See Workbook Documentation for detailed description.

AutoSizeColumnHeaders

See Workbook Documentation for detailed description.

AutoSizeRowHeaders

See Workbook Documentation for detailed description.

CheckAddCalcTag

See Workbook Documentation for detailed description.

CheckSheetBeforeSave

See Workbook Documentation for detailed description.

CheckUnitAccess

Type: *Boolean*
Default: *True*

This configuration parameter determines whether or not the unit access is checked when opening a module.

If this parameter is set to *False* the unit access is not checked.

If this parameter is set to *True* the unit access is checked. If the unit access is checked, the user will be informed when a unit is already in use by other users. See also the parameter UnitLockType.

CreateUserAccessRightsOnOLAPServerToo

Type: *Boolean*
Default: *False*

This configuration parameter determines whether or not user access rights are automatically added to the OLAP database.

If this parameter is set to *True*, a user gets “CALC” privileges on the OLAP server, if the user has NO privileges on the OLAP database at all. If the user already has privileges, then no change is done.

If this parameter is set to *False* access rights are set up in the relational database only and will NOT be automatically added to the OLAP Server database.

Note: This configuration parameter is available only if a MOLAP database is used. If a ROLAP database is used, this parameter is ignored. When using Essbase frontend tools like Smartview or Essbase Excel Addin the access rights must be configured carefully.

CreateUserOnOLAPServerToo

Type: *Boolean*
Default: *False*

This configuration parameter determines whether or not users created in Performance are automatically added to the OLAP database.

If this parameter is set to *True*, any user management transactions of the Admin module are performed in the relational database as well as in the OLAP database.

If this parameter is set to *False*, users are set up in the relational database only and will NOT be automatically added to the OLAP Server database.

Note: This configuration parameter is available only if a MOLAP database is used. If a ROLAP database is used, this parameter is ignored.

CubusAppServerManagerConnectionString

Type: *String*

This configuration parameter must be set to the hostname and port of the CubusAppServerManager (e.g. localhost:12345).

This parameter is needed internally.

Note:

If AppServerSchema is set to HTTP or HTTPS the port number configured will also be used for all application servers.

If AppServerSchema is set to HTTP or HTTPS Performance can use the same port as existing

IIS Web Sites.

CubusPath

Type: *String*

Contains the installation path of Performance (e.g. C:\Program Files\cubus\Performance). Based on this parameter all other path information can be derived (e.g. SheetPath).

CustomerBackgroundFileName

See chapter Module Administration - Specifics for detailed description.

DistribAccuracy

See Workbook Documentation for detailed description.

DumpDETCalc

Type: *Boolean*

Default: *False*

When this parameter is set to *True* the calculation server for a detail module dumps for every detail record an Excel file containing the calculation result for this record. These files are stored in the directory extprog/<UnitId>.

Note: This parameter should only be used during the development to locate calculation problems in the calculation server. It is not recommended to use it in a production environment.

DumpGrid

Type: *Boolean*

Default: *False*

When this parameter is set to *True* the CubusAppServer dumps all workbook grids to the CubusAppServer log file every time they are being read from / written to the OLAP database.

Note: This parameter should only be used during the development to locate problems that occur while reading or writing data. It is not recommended to use it in a production environment.

DumpMemberSelect

Type: *Boolean*

Default: *False*

When this parameter is set to *True* the member select file is stored after all parameters have been passed to the sheet. The Excel file containing the final membersselect definition is stored in the directory extprog.

Note: This parameter should only be used during the development to locate problems in the membersselect definition. It is not recommended to use it in a production environment.

EnableExport

See Workbook Documentation for detailed description.

EnableImport

See Workbook Documentation for detailed description.

EnableMaintenanceMode

Type: *Boolean*
Default: *False*

This configuration parameter enables the maintenance mode for the entire Performance software or for a specific application.

If the Maintenance Mode is enabled, it is only checked for new and non admin account logons. Administrators will still have access to the application and existing user sessions will not be logged out.

EnableWorkbookDesigner

See Workbook Documentation for detailed description.

EnableWorkflow

Type: *Boolean*
Default: *False*

If this parameter is set to *True*, the task button in the toolbar is visible which allows the user to see the assigned tasks.

ExpandedUnitGeneration

Type: *Integer*
Default: *1*

This configuration parameter can be used to determine the number of expanded levels for the unit tree. For each unit node on the level defined, the unit will be expanded.

Note: The default value for this parameter is 1, i.e. the unit tree shows the starting node and the level directly below the starting node (children). If this parameter is not defined the default value will be used.

ExpandModuleOnSelect

Type: *Boolean*
Default: *False*

If this parameter is set to *True*, selecting a module from the module tree also expands its children automatically.

ExpandRibbonsOnLoad

See chapter Module Administration - Specifics for detailed description.

ExportType

See Workbook Documentation for detailed description.

ForceSave

See Workbook Documentation for detailed description.

HideRootModule

Type: *Boolean*
Default: *False*

If the parameter is set to *True* the root module in the module tree will be hidden.

HideToolBarButtons

See chapter Module Administration - Specifics for detailed description.

HideTreeIcons

Type: *Boolean*

Default: *False*

If the parameter is set to *True*, the icons in the module, unit and member select trees will be hidden.

The text for the nodes will be shown in italic, if the module and unit combination is submitted.

The text for the nodes with no access will be shown with an opacity.

Any symbols regarding the node state will be shown on the right of the text.

IgnoreLockedAddCalcCells

See Workbook Documentation for detailed description.

IncludeHiddenCellsOnCopy

See Workbook Documentation for detailed description.

IncludeHiddenCellsOnPaste

See Workbook Documentation for detailed description.

LogPath

Type: *String*

This configuration parameter is used to determine the path to log files that are to be displayed in the log viewer. It is possible to define one or more paths to log files by separating each path with a semicolon (;) in the value field of the global configuration parameter. See also chapter Log Viewer.

MaintenanceModeMessage

Type: *String*

When maintenance mode is enabled this parameter defines the message displayed to the user after login or selecting an application.

ModuleMode

See Workbook Documentation for detailed description.

Note: This parameter is Workbook only and can't be applied to Detail Modules.

MSReportServerPath

Type: *String*

This configuration parameter can be set to the URL of the MS Reporting services (e.g. <http://localhost/ReportServer>).

NumberOfSearchRecordsDisplayed

Type: *Integer*
Default: *20*

The NumberOfSearchRecordsDisplayed parameter is responsible for specifying how many search results will be shown while searching in the unit and offgrid trees.

Preload

Type: *String*

Preloading allows to shorten the time required for the first application- or module navigation of a user. Therefore, the module- and unit structures will be loaded and cached automatically during the app server start up. The parameter can be defined either in the global- or the application specific configuration and will take effect either for all or for only specific applications respectively.

During the preloading process, the application is fully usable, but might exhibit slight performance losses until the preloading process has completed.

As soon as the module- or unit structure will be changed (e.g. in the module administration), the cache will be cleared and not rebuilt automatically until the next app server restart.

The following parameter values are supported.

- **None**
Same as when it's not defined. No preloading will be performed.
- **Module**
The module structures will be generated and cached for all users automatically.
- **ModuleAndUnit**
The module- and unit structures will be generated and cached for all users and module combinations automatically.

PrimaryOLAPDB (and further OLAP databases)

Type: *OlapDB*

At least one OLAP database needs to be set up in an Performance application. This so-called PrimaryOLAPDB must be specified as a "PrimaryOLAPDB" configuration parameter. Depending on the OLAP database used, the value of this configuration parameter is specified as follows:

- Essbase:
MOLAP/<Server[:port]>/<Application>/<Database>[/<UnitUDA>]

Server is the name or IP-adress of the Essbase server.

Application and Database are the names of the Essbase database.

UnitUDA is an optional parameter which can be used to flag one dimension as the UNIT dimension. The default value is ABC_UNIT.

Example:

MOLAP/localhost/App/Sales

MOLAP/10.10.12.13/App/Sales/ABC_UNIT_COSTCENTER

If Essbase 19 or higher is used the server url [http\(s\)://server\[:port\]](http(s)://server[:port]) or [http\(s\)://server\[:port\]/\[subdir1\]/\[subdir2\]](http(s)://server[:port]/[subdir1]/[subdir2]) is also supported.

Example:

MOLAP/http://localhost:9001/App/Sales

MOLAP/https://localhost:9001/App/Sales/ABC_UNIT_COSTCENTER

MOLAP/https://localhost:9001/essbase/agent/App/Sales

- ROLAP:
 ROLAP/<ApplicationName>/<ROLAP Application>

Example:

ROLAP/ABC_ROLAPDB/Demo

- TM1
 TM1/<Tm1RestApiUrl>/<Tm1ServerName>/<cube>[/<UnitUDA>]

Example:

TM1/http://localhost:9001/sData/SalesCube

TM1/https://localhost:9001/sData/SalesCube

TM1/https://localhost:9001/sData/SalesCube/RegionUnitUDA

All other OLAP databases are defined via configuration parameters as follows:

- Configuration parameter <OLAPDB_ID>, where <OLAPDB_ID> is the internal name of the OLAP database.
- Parameter value like the PrimaryOLAPDB
- Configuration parameter type: "OLAP-DB"

When a configuration parameter of type "OLAP-DB" has been created, deleted or updated, the administrator is asked whether or not the OLAP DB information is to be synchronized.

- Answer "Yes": The server cache is updated; i.e. the structural information of all defined OLAP databases is updated.
- Answer "No": The server cache is not updated. In this case, the structural information can be updated via the menu item "Update outline" or via an restart of the CubusAppServer.

<OLAPDB>_SSASCube

Type: *String*

When using adhoc analysis in workbooks in a ROLAP environment, an additional OLAPDB parameter has to be defined. This parameter contains the connection information to the related Microsoft Analysis Services cube. The name of this parameter is <OLAPDB>_SSASCube.

The value of this parameter is

ROLAP/<SSAS-Server>/<SSAS-Database>/<SSAS-Cube>

SSAS-Server is the name or IP-adress of the Microsoft Analysis Server.

SSAS-Database and SSAS-Cube are the names of the database and cube.

Example:

To use the adhoc analysis functionality within a workbook using the PrimaryOLAPDB, the string parameter with the name PrimaryOLAPDB_SSASCube must be added.

ProgressIndicatorDelay

Type: *Integer*
Default: *0*

This configuration parameter is used to specify the delay time in milliseconds after clicking the save button or starting a calculation until the progress indicator appears.

If this parameter is set to 0, the progress indicator will appear immediately after clicking the save button or starting a calculation.

If this parameter is set to -1, the progress indicator will not appear.

RefreshOutlineTables

Type: *Boolean*
Default: *False*

This configuration parameter is used to specify whether the relational database repository should be filled with the meta data from the multidimensional database.

If this parameter is set to *True*, the database tables OtIMember and OtIAlias will be filled with the members of the multidimensional database in a parent child relationship when it refreshes the multidimensional structures.

This data can then be accessed by SQL queries and used in workbook modules, or member selects.

Note: The parent name of the top most member in a dimension will be the member name itself. The database columns Alias0 - Alias9 in table OtIMember can be used to store the different alias names. Therefore the number of possible alias tables is limited to 10. The maximum length of aliases is 250 characters.

ReloadDataAfterCalc

See Workbook Documentation for detailed description.

ReloadDataAfterSave

See Workbook Documentation for detailed description.

SelectionBackgroundColor

See Workbook Documentation for detailed description.

SelectionDisabledTextColor

See Workbook Documentation for detailed description.

SelectionTextColor

See Workbook Documentation for detailed description.

SessionTimeout

Type: *Integer*
Default: *30*

Specifies the number of minutes a user session stays valid. After this time has expired, the session will be terminated and the user has to log on again.

Note:

- This parameter cannot be set per application. It has to be set as a global configuration parameter.
- Changing this parameter requires the CubusAppServerManager (Windows Service) to be restarted to take effect.
- This parameter sets the following WCF timeout parameters:
 - ReceiveTimeout
 - SendTimeout
 - OpenTimeout
 - CloseTimeout

SetModulesExpanded

Type: *Boolean*
Default: *False*

This configuration parameter is used to specify whether the modules in the module tree should always be set to expanded. If this parameter is set to *True*, all nodes are expanded always.

SheetPrintFooter

See Workbook Documentation for detailed description.

SheetPrintHeader

See Workbook Documentation for detailed description.

ShowAncestors

Type: *Boolean*
Default: *False*

This configuration parameter is used to define whether or not the ancestors of the permissible units will be displayed.

If this parameter is set to *False*, The user will see only the units permitted for the currently selected module and the user.

If this parameter is set to *True*, the user will see the permissible units plus their ancestors. This implies that the permissible units are shown within their hierarchical structure.

ShowDimensionName

See chapter Module Administration - Specifics for detailed description.

ShowScrollbars

See Workbook Documentation for detailed description.

ShowSheetHeaders

See Workbook Documentation for detailed description.

ShowSheetTabs

See Workbook Documentation for detailed description.

SkipHiddenCellsOnPaste

See Workbook Documentation for detailed description.

SqlQueryCacheMode

See Workbook Documentation for detailed description.

SqlQueryTimeout

See Workbook Documentation for detailed description.

SuppressZeroValue

See Workbook Documentation for detailed description.

TraceLevel

Type: *String*

Default: *Information*

This configuration parameter is responsible for setting the trace level in the CubusAppServer log file. This configuration parameter has the following possible values:

- Off
- Error
- Warning
- Information

Note:

Changing this parameter requires the CubusAppServerManager (Windows Service) to be restarted to take effect.

UnitLockType

Type: *String*

Default: *UnitOnly*

This configuration parameter is responsible for determining how the locked units behave. This configuration parameter has three possible values.

- UnitOnly
- Unit&OLAPDB
- Unit&Module

If *UnitOnly* is defined, if the unit is in access anywhere in the application the unit will be locked.

If *Unit&OLAPDB* is defined, if the unit is in access within the same OLAP database and application, the unit will be locked.

If *Unit&Module* is defined, if the unit is in access within the same module and application, the unit will be locked.

UseMemberNameForTreeExport

See Workbook Documentation for detailed description.

UseOlapConnectionUser

Type: *Boolean*
 Default: *False*

This parameter determines whether the user access to the OLAP database is done by the connection user or the logged-in user.

If this parameter is set to *True*, the access will be done by the connection user. The connection user can be configured per OLAP database in the *connectionStrings* section in the CubusAppServer.exe.config.

Example:

```
<connectionStrings>
    <add name="MOLAP/localhost/Sample/Basic" connectionString="User
ID=admin;Password=password"></add>
</connectionStrings>
```

If this parameter is set to *False*, the access will be done by the logged-in user. The user needs the appropriate permissions in the OLAP database to perform the necessary tasks (e.g. Read, Write, Calc).

UsePersonalizedOutlines

Type: *Boolean*
 Default: *False*

This parameter determines whether the user will see all units assigned to the selected module or only the units which the user is able to access.

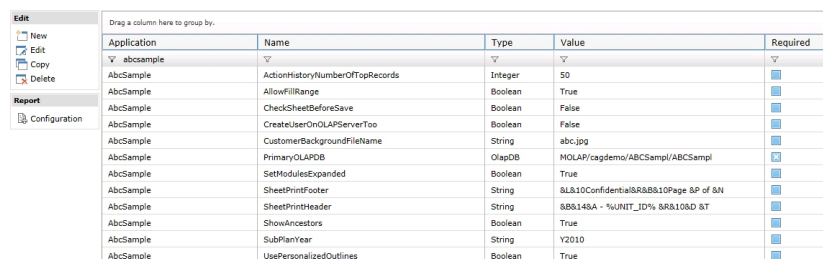
If this parameter is set to *True*, the user will only see the units the user is permitted to access and assigned to the module. Furthermore if the parameter *ShowAncestors* is set to *True*, the ancestors of the units to which the user has access to are also displayed.

If this parameter set to *False*, all units assigned to the selected module are displayed when the user logs in. The units where the user has no access are displayed with a small lock.

Substitution Variables

Substitution variables are custom specific configuration parameters. These substitution variables are defined either in the global configuration or application configuration dialog.

The defined variables can be used in workbooks and detail modules as required.



Application	Name	Type	Value	Required
AbcSample	ActionHistory/LumberOfTopRecords	Integer	50	<input type="checkbox"/>
AbcSample	AllowFillRange	Boolean	True	<input type="checkbox"/>
AbcSample	CheckSheetBeforeSave	Boolean	False	<input type="checkbox"/>
AbcSample	CreateUserOnOLAPServerTop	Boolean	False	<input type="checkbox"/>
AbcSample	CustomerBackgroundFileName	String	abc.jpg	<input type="checkbox"/>
AbcSample	PrimaryOLAPDB	OlapDB	MOLAP/cagdemo/ABCSample/ABCsampl	<input type="checkbox"/>
AbcSample	SetModulesExpanded	Boolean	True	<input type="checkbox"/>
AbcSample	SheetPrintFooter	String	&L&10Confidential&R&B&10Page 8P of 8N	<input type="checkbox"/>
AbcSample	SheetPrintHeader	String	&B&14&A - %UNIT_ID% &R&10&D &T	<input type="checkbox"/>
AbcSample	ShowAncestors	Boolean	True	<input type="checkbox"/>
AbcSample	SubPlanYear	String	Y2010	<input type="checkbox"/>
AbcSample	UsePersonalizedOutlines	Boolean	True	<input type="checkbox"/>

Fig. 22 - Managing Substitution Variables

Creating a Substitution Variable

Press the “New” menu item to define substitution variables.

Specify a unique name for the substitution variable into the “Configuration Parameter” field. Specify the value of the variable in the “Configuration Value” field.

Also, specify the substitution variable type. The following types are supported:

- String
- Integer
- Double (Floating point number)
- Boolean

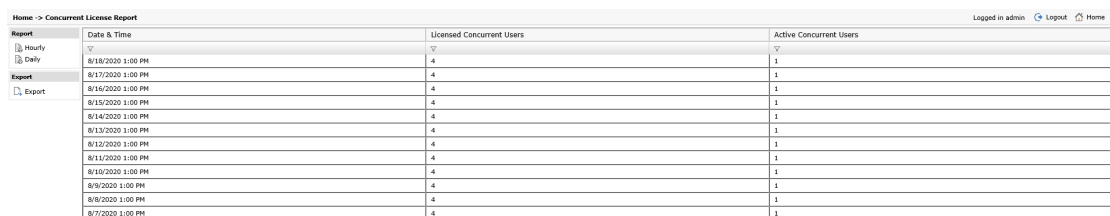
The “Edit” menu item enables you to update the value of the selected variable.

The “Delete” menu item enables you to delete the selected substitution variable.

Concurrent User Report

Performance tracks the number of concurrent users who are logged on. You can view the summary report in this section.

The summary report can be sorted by clicking the header of the column you want to sort by.



Report	Date & Time	Licensed Concurrent Users	Active Concurrent Users
Hourly			
Daily			
Report	8/19/2020 1:00 PM	4	1
	8/17/2020 1:00 PM	4	1
Export	8/15/2020 1:00 PM	4	1
	8/15/2020 1:00 PM	4	1
	8/14/2020 1:00 PM	4	1
	8/13/2020 1:00 PM	4	1
	8/12/2020 1:00 PM	4	1
	8/11/2020 1:00 PM	4	1
	8/10/2020 1:00 PM	4	1
	8/9/2020 1:00 PM	4	1
	8/8/2020 1:00 PM	4	1
	8/7/2020 1:00 PM	4	1

Fig. 23 - Concurrent License Report

Report Type

Two types of report can be selected at the **Report** section located at the top-left of the screen.



Fig. 24 - Report Type

By selecting **Hourly** report, the **Active Concurrent Users** column displays the highest number of active concurrent users during that hour of day.

By selecting **Daily** report, the **Active Concurrent Users** column displays the highest number of active concurrent users during that day.

Configuration

The report is generated by reading the log files at the path specified in the configuration file located at **<PerformanceInstallPath>\Bin\Server\CubusAppServerManager.exe.config**.

In the configuration file, under the **<appSettings>** section you will find two keys, **licenseLogPath** and **licenseLogFrequency**.

```
<appSettings>
  <add key="licenseLogPath" value="C:\Users\Administrator\Desktop\LogFiles\License.log"></add>
  <add key="licenseLogFrequency" value="month"></add>
```

Fig. 25 - License Log Configuration

The **licenseLogPath** specifies the path where the log files are written. You may change the path and the name of the file to be written as you prefer.

The **licenseLogFrequency** determines how often a new log file should be created. For example, specifying **month** for the **licenseLogFrequency** creates a new log file when new month begins.

licenseLogFrequency	Behavior
hour	creates a new log file every hour.
day	creates a new log file every day.
week	creates a new log file every week.
month	creates a new log file every month.
year	creates a new log file every year.

Log Viewer

The log viewer is used to display log files that are generated within the software. It is also possible to download log files using this dialog window.

In the log viewer control it is possible to display the following logs:

- Workbook server trace log
- Server trace log per application
- Database trace log

Note: To display all log files from the server log, the configuration parameter name LogPath needs to be defined. The LogPath value contains the path(s) to the server log file(s).

Session

This is where all user sessions are managed. Every time a user logs in to Performance a session is created for that specific user. The overview screen shows all current user sessions.

If the `SessionCleanupInterval` value is set to zero, automatic session cleanup is disabled. The `SessionCleanupPingInterval` can be disabled by setting the value to zero.

Create DesktopApp package

This chapter describes how to create a deployment package for the Performance Desktop Application.

During the first start of the Performance Application Server Manager after the software installation, the Performance DesktopApp deployment package is created automatically.

It may be required to create additional packages using the Performance Administration in the following scenarios:

- deploy different versions for different environments (e.g. test and production) to the same client machine
- deploy a hotfix

The **source files** for creating the package are located in the folder "`<INSTALL_DIR>\Bin\Server\DesktopApp`". Any changes to the source must be place in this folder.

The deployment package is created in the **output directory** "`<INSTALL_DIR>\Bin\Server\DesktopAppPackage`".

In order to install multiple Performance Desktop Application versions on the same client machine a unique ID is required. This is often the case when a newer software version is installed on the test server and an older software version is installed in the production environment. This may also be the case when using multi-production server deployments. E.g. one server for Europe and a second server for Asia. The **assembly identity name** is used for identifying a unique client. **Product name** is the visible name of the Performance Desktop Application and it appears in the start menu. These values need to be unique in order for different client versions to be deployed in parallel to the same client machine.

Warning: If the assembly identity name is changed, all users need to install a new Performance Desktop application using the URL: http://<yourservername>/<Performance_CLIENT_ALIAS>/PMDesktopApp and uninstall the old one (if they don't need it). Otherwise there will be two (or more) versions of the Performance Desktop application in the system.

Version number is the number for the next package. It will be automatically increased every time you create a new package.

We recommend to sign the deployment package. Therefore you need a code signing certificate. The certificate file has to be copied in the directory `<INSTALL_DIR>\Bin\Server\`. In the section **appSettings** in the file `CubusAppServerManager.exe.config` add the key `CodeSigningCertFileName` with certificate file name as value and `CodeSigningCertFilePassword` with the password for the certificate as value.

Note: The `appSettings` section can be **encrypted** using the `aspnet_regiis.exe` tool that is used to encrypt connection strings. During the package creation the server checks for an Internet connection. If there is no certificate specified or no internet connection exists, the package will be created, but **not** signed with the certificate. In this case the package will get a **publicKeyToken** equal to 0. During installing or updating the Performance Desktop Application on the client machine the "Unknown Publisher" security warning appears. The client will receive the same warning when there is no internet connection when installing the package.

If you do not provide a code signing certificate but you want to install multiple Desktop Applications on the same machine from different locations, every package must have a different

version number. Otherwise it will not be possible to install a second application with the same version and the same publicKeyToken.

Module Administration

The modules are managed via the Module dialog. This dialog can also be used to define execution steps and to grant direct access permissions for individual modules.

When you click on the "Module" menu item in the navigation area, the following dialog is shown.

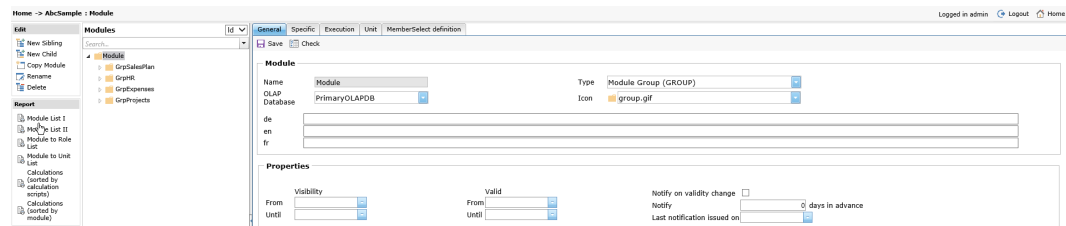


Fig. 27 - Module dialog

The left-hand panel lists the options for creating, renaming and deleting modules.

The module tree displays all current modules and their hierarchy.

The panel on the right displays the properties of the selected module. The properties are grouped into several panes.

- General
- Specific
- Execution
- Unit
- MemberSelect definition

Module Administration - General

The following module properties can be configured here.

Name

This is where the name of the module is defined. The name of each module needs to be unique within each application.

Note: The module names must not contain special characters.

Type

Use this field to specify the module type. Each module type is described below.

- AL_ANYWHERE
The AL_ANYWHERE type displays AL Anywhere reports. For pure visualization of data in report format, any Performance Analytics view/folder can be integrated in Performance.
- ABCMODULE_EV
This module is for workbook content and Performance Analytics content displayed within

the same module.

- **COCKPIT**
This module is used to display a cockpit consisting of different contents like Performance Analytics or Reporting Services reports. Cockpit modules are not supported in the Desktop App.
- **DET**
This module is used to display detail content. Any input templates can be created in Excel (refer to the user document entitled "Detail Modules" for further information).
- **DET_URL**
This module divides the page in two parts. The upper part is used to display detail content. The lower part can display a web page (URL).
- **EV_DATA_ENTRY**
The EV_DATA_ENTRY type displays Performance Analytics reports with data entry enabled and the possibility to submit. Any Performance Analytics view/folder can be integrated in Performance.
- **EV_MODULE**
The EV_MODULE type displays Performance Analytics reports. For pure visualization of data in report format, any Performance Analytics view/folder can be integrated in Performance. Note that these views do not permit any values to be written back to the OLAP DB.
- **GROUP**
Used to specify a group of modules, normally used as a parent module.
- **MODULE**
This module is used to display workbook content. Any input templates can be created in Excel (refer to the user document entitled "Workbooks" for further information).
- **MODULE_URL**
This module is used to display workbook content in combination with any web page. Any input templates can be created in Excel (refer to the user document entitled "Workbooks" for further information). This module can also display a web page on the same content page.
- **REPORTING_SERVICES**
This module is used to display Microsoft Reporting Service reports.
- **URL**
The URL modules are used to display external content, for example, a web page.
- **WORKFLOW**
This module is used as a group module for a workflow where users can submit, accept and reject on the workflow.
- **WORKBOOK_ANYWHERE**
This module is used to display workbook content with the new Workbook Anywhere client. Any input templates can be created in Excel (refer to the user document entitled "Workbooks" for further information).

Note: Please see the module-specifics section for more information on different module types.

OLAP Database

An OLAP database needs to be assigned to each module. This database serves as the "master" database for the module. The access permissions are defined via the dimension of the units of this OLAP DB. The default is the *PrimaryOLAPDB* of the application.

Icon

An icon for the module can be specified. This icon is the image that is displayed in the module tree. The icons that are standard are:

- folder.gif
This is the default image that is displayed if no icon is specified.
- bsc.gif
This icon can be used for scorecarding modules.
- det.gif
This icon is for all detail modules.
- ev.gif
This icon is for the Performance Analytics modules.
- group.gif
This image is for all groups.
- module.gif
This icon represents workbook modules.
- url.gif
This image is for URL modules.

Visibility

A module can be set to visible or invisible by the administrator depending on a given date.

- The "From" field defines the date from which the module is visible. If this field is empty, the module is visible immediately.
- The "Until" field defines the date until which the module is visible. If this field is empty, the module is visible without restrictions.

Valid

A module can be set to be valid (i.e. enabled) for the user depending on a given date.

- The "From" field defines the date from which the module is valid. If this field is empty, the module is valid immediately.
- The "To" field defines the date until which the module is valid. If this field is empty, the module is valid without restrictions.

Description

Use this field to enter descriptions to be displayed to the user. These descriptions need to be specified in the applicable language.

When entering the description, you need to select the applicable language from the associated combo box.

Module Administration - Specifics

The module-specifics define the properties of a module. The next sections describes all available module specific parameters in detail.

AbcModuleReloadStructure

Type: *String*

This parameter controls the behavior after performing a *Save* or an *Execution Step* on a workbook.

Supported values are:

- **<Empty>**
The behavior does not change.
- **ClientOnly**
After pressing save or running an execution, the unit and offgrid trees in the client web application will be reloaded.
- **ServerOnly**
After pressing save or running an execution, a complete outline structure reset will be performed on the server. The unit and offgrid trees in the client web application will **NOT** be reloaded.
- **ClientAndServer**
After pressing save or running an execution, a complete outline structure reset will be performed on the server. Afterwards, the unit and offgrid trees in the client web application will be reloaded.

AddCalcViewPosition

Type: *String*
Default: *Bottom*

Specifies where the additional calculations pane is to be displayed.

Possible values are

- Top
- Bottom
- Left
- Right

AddCalcViewSize

Type: *Integer*
Default: 300

Defines the size of the additional calculations pane in pixel.

ConnectionStringName

Type: *String*

This parameter allows a separate database for saving and loading all data related to a detail module. The value for this parameter is also the value for the name attribute in the connectionString section of the Cubus.AppServer.exe.config file.

Microsoft SQL Server

```
<add name="DETMSSQLRepositoryConnectionString"
      connectionString="Data Source=(local);
                      Initial Catalog=DETREPOSITORY;
                      Persist Security Info=True;
                      User ID=user; Password=password"
      providerName="System.Data.SqlClient" />
```

Oracle

```
<add name="ConnectionStringName"
      connectionString="Data Source=SID;
                      User ID=youruser/schema;
                      Password=yourpassword;"
      providerName="Oracle.DataAccess.Client" />
```

If this value is empty, the default `AbcRepositoryConnectionString` name will be used to access the database to save and load detail module data. Otherwise the value will be used and must be defined in the configuration file `Cubus.AppServer.Exe.Config`.

ContentViewPosition

Type: *String*
Default: *Top*

Specifies where the main content of combined modules is to be displayed.

Possible values are

- Top
- Bottom
- Left
- Right

Note: The main content of combined modules always refers to the workbook or detail module content.

ContentViewSize

Type: *Integer*
Default: *half of the available screen area*

Defines the size of the main content window in pixel.

CreateEVSession

Type: *Boolean*
Default: *False*

Used for URL modules. Creates an Performance Analytics session using the credentials provided to Performance during the login and passes the Performance Analytics session ID to the URL as an additional request parameter *EvSessionId*.

The session id of Performance Analytics can be used in custom specific URL modules using the Performance Analytics object.

CustomerBackgroundFileName

Type: *String*

Specifies an image file (e. g. "My_File.png") that can be displayed as a background. The image file must be located in the directory `ABC_WWW_ROOT/Content/Themes/<CurrentTheme>/`.

DetailGridPosition

Type: *String*
Default: *Left*

Specifies where the Detail Grid for a Detail Form module is to be displayed.

Possible values are

- Top
- Bottom
- Left
- Right
- Hidden

DetailGridSize

Type: *Integer*
Default: 500

Defines the size of the Detail Grid for a Detail Form module in pixel.

DetReport

Type: *String*

Specifies the name of a Crystal Report (*.rpt) which is used to display the detail records for printing.

After the rpt file has been created it has to be copied to [CUBUS_WEB_APPLICATION] \Reporting\Reports (for SQL Server databases) or to [CUBUS_WEB_APPLICATION]\Reporting \ReportsOracle (for Oracle databases).

EvaluationOrder

Type: *String*
Default: *ByStep*

The evaluation order determines how the workbook is parsed.
The following options are available:

- **ByStep**
This is the default value. ByStep indicates that the parsing steps are following in sequence for all sheets defined. For example, the offgrid tags will be parsed for all sheets, then the ongrid tags for all sheets, then the data will be read for all sheets, etc.
- **BySheet**
This parameter indicates that the sheets will be parsed in the order in which they are defined. For example, sheet 1 will be parsed, then the data will be read. After the parsing of sheet 1 has taken place, sheet 2 will then be evaluated.

ExpandRibbonsOnLoad

Type: *Boolean*
Default: *True*

If this parameter is set to true or false, navigating to a module automatically expands or collapses the ribbon area depending on the specified value.

GanttActivityBarClick

Type: *String*

Used for Performance Analytics modules. This parameter determines what happens when the user clicks on a bar of a Gantt chart in the Performance Analytics view.

A URL can be defined to call another Performance module, using the POV of the Gantt bar clicked. The URL needs to be of the following structure:

`cubus://nav/?mod=<ModuleName>&unit=${<Region>}&<ProductId>=${<Product>}`

<ModuleName>: Name of the module you want to navigate to

<Region>: Name of the dimension (in the Performance Analytics view) that is used as unit in the new Performance module

<ProductId>: Name of an additional offgrid dimension in the new Performance module

<Product>: Name of the additional offgrid dimension in the Performance Analytics View

For the parameters that are put into `$(...)` the value of the currently selected POV will be handed into the URL.

GanttActivityMilestoneClick

Type: *String*

Used for Performance Analytics modules. This parameter determines what happens when the user clicks on a milestone of a Gantt chart in the Performance Analytics view.

A URL can be defined to call another Performance module, using the POV of the Gantt milestone clicked. The URL needs to be of the same structure as for `GanttActivityMilestoneClick`.

HideEvRibbonTabs

Type: *String*

This parameter can be used to hide tabs on the button toolbar for Performance Analytics Data Entry modules. It is possible to hide multiple tabs by separating them with a comma, e.g.: "Layout,Format,..."

Possible values are

- Navigate
- Member
- Layout
- Format
- Source
- Canvas
- Comments

HideToolbarButtons

Type: *String*

This parameter can be used to hide toolbar buttons in the toolbar at the top left corner of the screen, in the ribbon and backstage and in the context menu. It is possible to hide multiple items by separating them with a comma, e.g.: "Comment,Submit,Save..."

The following table gives an overview of the possible values and the areas they affect:

Value	Area
AddCalc	Home ribbon, context menu
AdHocAnalysis	Home ribbon, context menu
ChangeAlias	context menu (workbook only)
Comment	Home ribbon, Layout ribbon, Comments tab
Copy	Home ribbon, context menu
Cut	Home ribbon, context menu
Delete	Home ribbon, context menu (detail module only)
Distribution	Home ribbon, context menu
DrillThrough	Home ribbon, context menu
Email	toolbar
Execute	Home ribbon, context menu
Export	toolbar, backstage, context menu
ExportOlap	context menu (workbook only)
ExportPowerPoint	context menu (workbook only)
ExportRecords	context menu (detail module only)
GoalSeek	Home ribbon, context menu
ImportRecords	context menu (detail module only)
Info	toolbar
Layout	Layout tab
MemberSelect	Home ribbon, context menu
New	Home ribbon (detail module only)
Paste	Home ribbon, context menu
Print	toolbar, backstage, context menu
PrintAddCalc	backstage
PrintRecord	backstage (detail module only)
Redo	toolbar, context menu
Reload	toolbar, context menu
ReloadModule	toolbar
Save	toolbar, backstage, context menu
Submit	Home ribbon
Undo	toolbar, context menu

InformationFile

Type: *String*

The information file for the module.

Optionally, the administrator can provide module-related information (such as a help file for the module) for the user. The associated filename needs to be specified in this field.

When the user clicks the "Info" button in a module on the mini ribbon toolbar, the contents of these files will be displayed in the ribbon backstage.

The ABC_WWW_ROOT configuration parameter needs to be set accordingly (see "Configuration Parameters"). The HTML document needs to be stored in a language-specific below ABC_WWW_ROOT.

Path: ABC_WWW_ROOT\Content\Customizing\<LANGUAGE>\<HTML-Filename>

Optionally, an anchor tag can be provided in order to navigate to a specific section in the information file.

Example: InformationFile.html#Info1

In this case the click on the info file button will navigate to the element with the id *Info1* inside the *InformationFile.html*.

Note: The button is visible only if the the specified file exists in the file system.

Link

Type: *String*

For workbook module:

A URL called when the context menu item "Drill through..." is clicked. The point of view for the selected numeric OLAP cell is passed to the URL. The value of the cell is passed in the request parameter "Value".

For detail module:

A URL called when the context menu item "Drill through..." is clicked. The point of view for the selected detail data is passed to the URL. The following request parameters are passed in the URL:

- Value
The value of the selected cell
- FieldName
The name of the selected cell
- TableName
The name of the database table
- DetId
The DetId of the selected detail data

ModuleFile

Type: *String*

The workbook to be displayed.

NavigateWithoutData

Type: *String*
Default: *Never*

This parameter determines how data should be read.
The following parameters are available:

- **Never**
The data is always loaded automatically.
- **Always**
The data always has to be loaded manually by clicking on the retrieve data button.
- **OnChange**
The data is initially loaded when the module is opened, however after selecting a new member, the user has to click on the retrieve data button.

RefreshOBViews

Type: *Boolean*
Default: *False*

Used when embedded in Oracle Business Intelligence to refresh the dashboard content after saving data. The default value is *False*.

SendPassword

Type: *String*

If and how to pass the user password to the URL-module.

- **No**
User password is not passed to the URL-module.
- **Yes**
User password is passed to the URL-module.
- **Encrypted**
User password is encrypted and passed to the URL-module.

Note: This could be a security issue.

ShowBackButton

Type: *Boolean*
Default: *False*

Shows or hides the back button in the report.

ShowDimensionName

Type: *Boolean*
Default: *True*

Specifies if the dimension names are to be displayed in the navigation panels.

ShowFindControls

Type: *Boolean*
Default: *False*

Shows or hides the find controls in the report.

ShowPageNavigationControls

Type: *Boolean*
Default: *False*

Shows or hides the page navigation controls in the report.

ShowParameterPrompts

Type: *Boolean*
Default: *False*

Shows or hides the parameter prompt in the report.

ShowPromptAreaButton

Type: *Boolean*
Default: *False*

Shows or hides the prompt area button in the report.

ShowTabs

Type: *Boolean*
Default: *True*

This parameter determines whether the Performance Analytics tabs should be displayed in the reports.

URL

Type: *String*

The URL that you wish to display in the content window.

UseModuleName

Type: *Boolean*
Default: *False*

This parameter determines whether the module description should be used to set the main title in the Performance Analytics view. The title will only be displayed in the Performance Analytics view, if it has been activated in the stored view. If multiple views are opened in the module, the title will be set to each of them.

UseWorkbookCommentFormatting

Type: *Boolean*
Default: *False*

The parameter is available only for Performance Analytics reports. It specifies, whether the comments shall be styled using the current Performance Analytics view (*false*), or the workbook template defined for this application (*true*).

View

Type: *String*

For Performance Analytics and AL Anywhere modules:

The path of the Performance Analytics Report that you wish to display.
E.g. /AbcSample/Sales/SalesReport

For reporting services modules:

The reporting services report that is to be displayed.

ViewType

Type: *String*

For Detail Modules:

Defines the type of the detail module:

- Form
- List

For Performance Analytics and AL Anywhere Modules:

The view type offers different functionalities of Performance Analytics Reports. The following parameters are available:

- **Static**
Static Report, no user interaction
- **OffgridOnly**
Report, allowing the user to select off-grid items
- **UiActive**
Report, allowing the user to make modifications that are possible without the menu. Only for AL Anywhere modules.
- **Adhoc**
The entire Performance Analytics functionality can be used for the creation, development, and modification of reports.
- **AdhocSave**
The entire Performance Analytics functionality can be used for the creation, development, and modification of reports. In addition, views can be saved with a new name.

Module Specific Compatibility Matrix

	A B C M O D U L E - E V	D E T	D E T - U R L	E V - D A T A - E N T R Y	E V - M O D U L E	G R O U P	M O D U L E	M O D U L E - U R L	R E P O R T I N G - S E R V I C E S	U R L	W O R K F L O W
Module Specific											
AbcModuleReloadStructure		X	X				X	X			

AddCalcViewPosition	X						X	X			
AddCalcViewSize	X						X	X			
ConnectionStringName		X	X								
ContentViewPosition	X		X				X	X			
ContentViewSize	X		X				X	X			
CreateEVSession										X	
CustomerBackgroundFileName						X					
DetailGridPosition		X	X								
DetailGridSize		X	X								
DetReport		X	X								
EvaluationOrder	X	X	X				X	X			
ExpandRibbonsOnLoad	X	X	X	X	X	X	X	X	X	X	X
GanttActivityBarClick	X			X	X						
GanttActivityMilestoneClick	X			X	X						
HideEvRibbonTabs				X							
HideToolBarButtons	X	X	X				X	X			
InformationFile	X	X	X				X	X		X	X
Link	X	X	X				X	X			
ModuleFile	X	X	X				X	X			
NavigateWithoutData	X	X	X				X	X			
RefreshOBViews		X	X				X	X			
SendPassword	X						X	X		X	
ShowBackButton									X		
ShowDimensionName	X	X	X	X	X	X	X	X	X	X	
ShowFindControls									X		
ShowPageNavigationControls									X		
ShowParameterPrompts									X		
ShowPromptAreaButton									X		
ShowTabs				X	X						
URL			X					X		X	
UseModuleName	X			X	X						
UseWorkbookCommentFormatting	X			X	X						
View	X			X	X				X		
ViewType	X	X		X	X						

Module Administration - Execution

For modules containing workbooks or detail modules, multiple execution scripts can be defined. Execution steps can be one of the following items:

- OLAP calculation script
- External program (e.g. batch script)
- Calculation server for detail modules

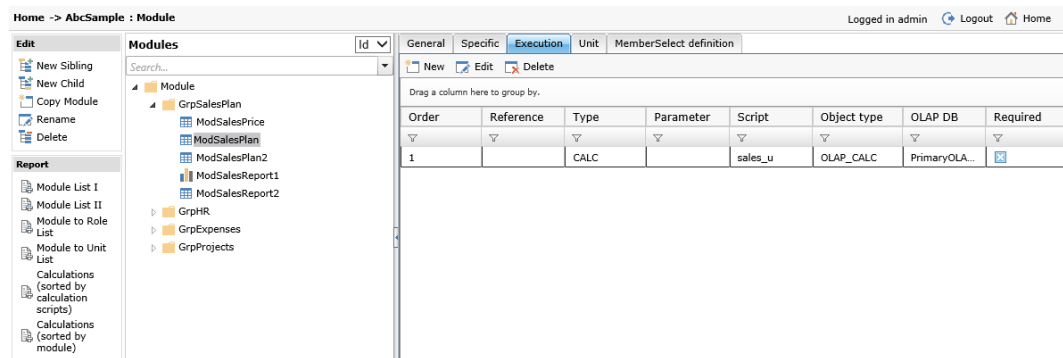


Fig. 28 - Module Execution Dialog

The defined scripts are listed in the order of execution.

Note: The order column indicates the sequence of execution steps. The order can be configured by editing a script. However two scripts cannot be assigned to the same order value. For example, script A and script B cannot both have the order 1.

Note: Only modules containing workbooks or detail modules support starting module executions.

The following can be specified for each script:

Reference

The reference is used to provide a unique identifier for execution steps. This field can be used in workbooks to start execution steps.

This field is optional.

Type

The type of the execution step indicates when the step is to be executed. Via a combo box, you can select from the following types:

Type	Description
Calc	The execution step is explicitly started by the user via a execution menu item in the workbook.

Pre-Open	The execution step is automatically executed before the workbook is read in.
Pre-Save	The execution step is automatically executed before the workbook is saved.
De-Select	The execution step is automatically executed when an offgrid member is selected for the previously selected member
Pre-Select	The execution step is automatically executed before the change of an "off-grid" unit.
Pre-Close	The execution step is automatically executed before the workbook is closed.
Post-Save	The execution step is automatically executed after the workbook has been saved.
Post-Submit	The execution step is automatically executed after the workbook has been locked. Note: In addition this type is triggered by all actions of a workflow module (submit, accept, reject) when used in conjunction with the object type EXT_PROG.

Execution

The start of an execution step of type "CALC" can be defined as:

- "required": the execution step will be executed in any case; or as
- "optional": the execution step is available to the user for selection

Note: Execution steps other than type "CALC" are always required and cannot be selected by the users when executing calculation scripts in the client.

Set the checkbox to indicate a required step.

Object Type

In this field, the object type can be selected from a combo box. This type specifies if the calculation will execute within the OLAP DB or by means of external programs. The following object types are available for selection:

- OLAP-Calculation
OLAP calculations are only possible for OLAP MOLAP databases
- External Programs
- Detail Module Calculation

OLAP Database

An OLAP database needs to be assigned to each calculation. By default, this is the "master" database of the associated module. Depending on the object type of the calculation, the specified OLAP database is applied as follows:

- OLAP calculation
The calculation is performed on the specified OLAP database.
- External program

The OLAP database can be passed as a parameter to the calculation (parameter &OLAPDB).

If the external program generates an output file (parameter &OUTPUTFILE), these data will be stored in the specified OLAP database.

- Detail module calculation

IF no OLAP database is specified on the output sheet of the calculation workbook, the calculated data will be stored in the OLAP database of the calculation.

Script

To enable the initiation of execution scripts, specify the name of the associated OLAP calculation script (without file extension), the name of the detail module calculation workbook or the name of the external calculation program in the "Calculation Script" field.

If an OLAP calculation script is to be used, make sure that this script is stored in the database directory of the specified OLAP database.

In the case of a detail module calculation workbook or an external program, this needs to be stored in the appropriate directory on the server side. Detail module calculation workbooks need to be stored in the <Sheets> directory, while external programs need to be stored in the <ExtProg> directory of the application.

Program Options

This input field is only used for external programs.

For external programs, you can specify any required call parameters here (see "External Calculation Server").

Duration

The administrator can specify the estimated duration (as plain text) of each calculation step. The user will see this information when selecting calculation steps.

Menu Items

The following toolbar buttons are available in the execution dialog:

- **New**
Adds a new execution script.
- **Edit**
Edits the selected execution script.
- **Delete**
Deletes the selected execution script.
- **Save**
Saves the changes made to the execution script.
- **Cancel**
Cancels the changes made to the execution script and returns to the list of scripts.

Parameterization of Scripts

OLAP calculation scripts can be parameterized in order to selectively calculate parts of the OLAP database. Depending on the nature of this parameterization, the calculation is executed

for registered units only or for members of the off-grid dimensions. External programs can also be parametrized.

Off-grid dimensions are defined by a "MemberSelect" instruction in the workbook (refer to the "Workbooks" document).

When you create calculation scripts, Performance automatically replaces the variables used for the members of the off-grid dimensions by corresponding members.

The definition of such variables start with the "&" character followed by the name of the dimension containing the member to be calculated.

The variables &UNIT_ID and &UNIT_NAME for the unit dimension are exceptions. Both of these variables are replaced by the currently selected member (&UNIT_ID) or by its alias name (&UNIT_NAME) from the unit dimension.

In addition to the offgrid dimensions, other calculation parameters and substitution variables can be parameterized. These calculation parameters need to be defined in the workbook. In the calculation script, the "&" character followed by the parameter name needs to be specified.

List of available parameters:

- &UNIT_ID (member name of the current unit)
- &UNIT_NAME (alias name of the current unit)
- &<MemberSelect-ID> where <MemberSelect-ID> is the unique identifier of a MemberSelect function of the module
- &<CalcParam> where <CalcParam> is a defined calculation parameter of the module
- &<SubstVar> where <SubstVar> is a Performance substitution variable used in the module
- &USER_ID (user id of the current user)
- &USER_NAME (user name of the current user)
- &MODULE_ID
- &OLAPDB
- &LEVEL (only available for Workflow modules with object type EXT_PROG; the level of the task that has been submitted, accepted or rejected)
- &ACTION_TYPE (only available for Workflow modules with object type EXT_PROG; the action that has been performed on the current task, e.g. Submitted, Accepted or Rejected)
- &COMMENT (only available for Workflow modules with object type EXT_PROG; the comment that has been entered for the action that has been performed on the current task)
- &TEMPLATEFILE=<filename>
 The template file parameter can be used to define a template file for sending mails in a workflow process. The parameter defines a file in the ExtProg folder of the application. If this file contains the marker &COMMENT then this marker is replaced with the content of the &COMMENT parameter. This enables the handling of workflow comments with special characters.

Example:

Input Parameter:

&TEMPLATEFILE=MailTemplate.txt

Output Parameter:

ExtProg\MailTemplate_635295354931269167.txt

The file MailTemplate_635295354931269167.txt will contain the workflow comment.

- &RUNASYNC

This parameter is used to activate the asynchronous mode for external programs, which is particularly useful for long-running scripts. The parameter ensures that the system does not wait for the end of the script and thus prevents the program from freezing. The parameter is passed through without any changes, so it is possible to identify later whether the script was started in asynchronous mode.

Example 1:

All other dimensions are calculated for the registered unit.

```
FIX("&UNIT_ID")
    CALC DIM (Dim1,Dim2,...);
ENDFIX
```

Example 2:

All other dimensions are calculated for the registered unit and for the selected member of the time dimension.

```
FIX("&UNIT_ID", "&Time")
    CALC DIM (Dim1,Dim2,...);
ENDFIX
```

Note: The calculation types "Pre-Open" and "Pre-Select" are exceptions. For this calculation, the off-grid members cannot be parameterized because these may not yet be known when the calculation is executed.

Note: The OLAP calculation scripts can only be used for MOLAP databases.

Note: Offgrid parameters, calculation parameters and substitution variables that start with the same string like reserved parameters cannot be used for parameterization. When e.g. using an offgrid parameter called USER_NAME_LONG only the reserved part USER_NAME would be replaced. For a user with the name James the parameter output would be James_LONG and the expected offgrid parameter USER_NAME_LONG would not be replaced.

External Calculation Server

An external data calculation program can be specified in Performance. An external calculation server handles special computation tasks that the OLAP database and/or the Excel workbooks can only handle with difficulty or not at all.

This calculation server is an autonomous program which is capable of reading data from Performance via a defined interface, which processes this data and which then returns the results via the interface to Performance.

The call interface for the calculation program is specified as follows:

```
<program_name> <program_options>
```

Example of an external program:

If e.g. a shell script is to be started, this cannot be done directly but via a batch file (*.bat) or via a preceding sh.exe:

```
sh.exe <absolute path>/<prog.sh>
```

e.g.:

```
sh.exe C:/Program Files/cubus/Performance/App/AbcSample/  
ExtProg/prog.sh
```

This requires the sh.exe to be available in the directory <ExtProg>.

Program specific parameters for the external calculation program can be specified via the "Program Options" field in the Module Execution dialog. Before the external program is called, these variables are replaced by their values and the command line is set up accordingly.

The parameterization of the calculation server follows the same principles as the parameterization of the calculation scripts.

Supported parameters are:

- &UNIT_ID (member name of the current unit)
- &UNIT_NAME (alias name of the current unit)
- &<MemberSelect-ID> where <MemberSelect-ID> is the unique identifier of a MemberSelect function of the module
- &<CalcParam> where <CalcParam> is defined calculation parameter of the module
- &<SubstVar> where <SubstVar> is a defined substitution variable of the module
- &USER_ID (ID of the current user)
- &USER_NAME (user name of the current user)
- &OLAPDB (name of the assigned OLAP database of the calculation program)

Additionally the following parameters are supported:

- **&INPUTFILE**

When this parameter is specified, an input file for external programs is created. This file can be created for modules of type "workbook" as well as for modules of type "detail module".

It is stored in

```
<App. directory>/ExtProg/<Unit>/abcsheetin.txt
```

For a more detailed description of the structure of this file, refer to "Input and Output Files of External Programs".

- **&OUTPUTFILE**

If the external program generates an output file for further processing by Performance, this parameter needs to be specified. The data of the output file will be stored in the OLAP database specified for the external program. Performance can read in this output file only if it has the same format as the input file &INPUTFILE. Exceptions are the lines with the "HeaderRows" and "HeaderCols" statements which are not needed in this output file.

The output file is stored in

```
<App. directory>/ExtProg/<Unit>/abcsheetout.txt
```

For a more detailed description of the structure of this file, refer to "Input and Output Files of External Programs".

Note: Performance can process an &OUTPUTFILE only for modules of type "workbook".

- **&ERRORFILE**

If the "&ERRORFILE" parameter is set, an error file is generated that contains all error messages of the external program. In the event of an error, these messages are directly passed on the user. This file does not require any specific formatting.

- **&DEBUGFILE**

If this parameter is set, a debug file is generated which contains program-specific information for detecting computation errors. This file does not require any specific formatting.

Input and Output Files of External Programs

If the parameter &INPUTFILE is specified for modules of type "workbook", all cells at the intersection of a column and row marked in the Excel workbook by the On-Grid tag "Export" (see "Workbook") are copied to the export file.

The following is an example of an input file.

[Sheet1]

"Sheetname=Interest"

"HeaderRows=6"

"HeaderCols=1"

	"Plan 2000"	"Plan 2000"	"Plan 2000"
	"New business"	"New business"	"New business"
	"Product"	"Product"	"Product"
	"Division"	"Division"	"Division"
	"Object"	"Object"	"Object"
	"Interest3M"	"Interest6M"	"Interest1Y"
"P1"	2.88	3.13	3.35
"P2"	2.88	3.13	3.35
"P3"	2.88	3.13	3.35
"P4"	2.95	3.23.44	
"P5"	2.95	3.23.44	
"P6"	2.95	3.23.44	
"P7"	3.03	3.28	3.54
"P8"	3.03	3.28	3.54
"P9"	3.03	3.28	3.54
"P10"	3.1	3.35	3.64
"P11"	3.1	3.35	3.64
"P12"	3.1	3.35	3.64

The input file (&INPUTFILE) and the output file (&OUTPUTFILE) for workbooks use the following structure:

1. A separate section is generated for each table within the input workbook.
2. This section is incrementally numbered (starting with 1).
3. The 1st line in this section contains the "Sheetname" parameter.
4. The 2nd line in this section contains the "HeaderRows" parameter. This parameter specifies the number of header rows in the table.
5. The 3rd line in this section contains the "HeaderCols" parameter. This parameter specifies the number of header columns in the table.
6. This is followed by any number of lines made up of cells (by analogy to an Excel table) separated by tabs.

If the parameter &INPUTFILE is specified for modules of type "detail module", the data are exported to a tab separated text file. The first line of this export file contains the column names of the database table in the sequence defined in the detail module (worksheet "Fields"). All other lines contain the records, with the data cells formatted like the associated table fields. This implies that text fields are formatted as text, date fields are formatted as dates, etc.

If the detail module contains sensitive data fields and if the user cannot access these fields, the values in these columns are masked ("*****").

Example:

DET_ID	UNIT_ID	LASTNAME	START_DATE	END_DATE	EMP_TYPE
DET_EMPLOYEE_000151	ANGESTELLT	Region1	Schwarz	2002-01-01	3000-01-01
	SALARY				
					80,0
DET_EMPLOYEE_000152	ANGESTELLT	Region1	Schmid	2002-07-01	3000-01-01
					40,0

Module Administration - Unit

To define access permissions, you can either select the applicable units via the tree view in the "Unit" tab (right-hand side of the window) or you can enter the applicable unit name into the input field. When entering the unit name manually, you may use regular expressions.

There are special characters for the regular expression that have to be escaped if you want to use it in a regular context. Those are the followings; [] { } () - + * ? . , \ ^ | \$. You can escape a special character by putting a backslash ahead of them. If you want to use brackets like in a regular context for example, "Region1 (South)"; you have to escape them like this, "Region1 \ (South)".

Additionally, you can specify whether a given permission applies to the specified unit only (context: Member (M)) or for the unit and all of its subunits (context: Member&Subtree (M&S)).

Module Administration - Member Select

Member select assignments can be defined under the member select tab.

Member selects can be defined on three different levels.

- Global member select
- Module member select
- Specific member select

If the *global* member select option is selected then a list is displayed showing all the possible global member select definitions. Member selects can be assigned and unassigned using the left and right arrows. The order in which the member selects appear in the client can be changed using the up and down arrows.

To use global member selects, a member select file with the name “_global_member_select.xls” needs to be defined in the following folder.

```
<Install_Dir>/App/<Application_Name>/Sheets/MemberSelects/  
_global_member_select.xls
```

If a *module* specific member select sheet exists, then all member select definitions will automatically be assigned to this module when this option is selected. To use module specific member selects, a member select file with the following prefix needs to be defined in the following folder.

```
<Install_Dir>/App/<Application_Name>/Sheets/MemberSelects/  
<Module_Id>_member_select.xls
```

If a *specific* member select sheet is assigned, then the list of available member selects in this file are displayed.

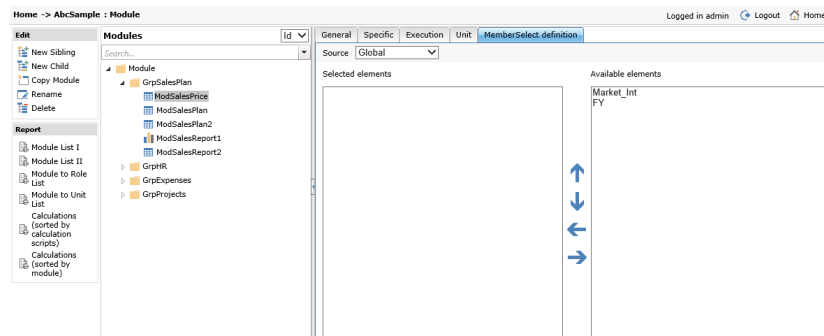


Fig. 29 - Member Select Dialog

Creating New Modules

To create new module, click the "Add Sibling" or "Add Child" menu.

Initially, the "Modules" root node is created. The name and properties of the "Modules" root node may be modified. However, note that the root node cannot be deleted, and no sibling nodes can be added to the root node.

For each module, the following needs to be specified in the dialog box:

Input Field	Description
Name	Mandatory
Type	Mandatory
OLAP Database	Mandatory
Icon	Optional
Visibility	Optional
Valid	Optional
Description	Optional
Specifics	Optional
Execution	Optional
Units	Optional
Member Select	Optional

Menu Items

The following menu buttons are available in the Module Administration:

- **New Sibling**
Adds a new sibling module.
- **New Child**
Adds a new child module.
- **Copy Module**
Copies the current specified module to a new location.
- **Rename**
Renames the selected module.
- **Delete**
Deletes the selected module.
- **Save**
Saves the data that is currently entered in to the dialog boxes.
- **Cancel**
Cancels the changes.

Note that not all menu items are always available. This depends on the selected tab in the configuration dialog.

Other menu items can include:

- **New**
Adds a new item to the chosen category.
- **Edit**
Edits the selected item.
- **Delete**
Deletes the selected item.

Module Check – Global Section

After a module has been created a module check can be carried out. To execute the module check click on the “Check” button in general module configuration dialog. The check functionality is responsible for checking the following:

- A unit has been defined.
- An icon has been defined.
- The member select returns data.
- The member select query syntax is correct.
- A ModuleFile, View, URL has been defined for the module types MODULE, EV_MODULE and URL respectively.
- A view type has been defined for the EV_MODULE type.

After clicking on the “Check” button the user will be informed if everything is correctly configured, or if the module configuration is not correct.

Moving Modules in Administration Tree

Modules can easily be moved in the module administration dialog. Modules can either be moved as a child of an existing module or as a sibling.

To move a module as a child module, drag and drop it on to the module that you wish to be the parent. After releasing the mouse button the module will then be moved as a child.

To move a module as a sibling module, drag and drop it on to the module that you wish to be the sibling while ensuring that the control (CTRL) key is pressed. The key must be pressed when releasing the mouse button to ensure that the module is moved as a sibling.

Copying Modules

Defined modules can easily be copied to a new location. Copying a module will make an exact copy of the selected module (including specifics/calculation/units/member selects).

To copy a module select the module that you wish to copy and select the “Copy Module” item from the menu. A new dialog will be displayed when the new module name can be entered. Select the target application and the location of the module to be copied and select from the option box if the module is to be copied as a child or a sibling.

It is possible to copy the module subtree by checking the copy subtree option. The switch to target application after copy option will select the module that is to be copied after the copy takes place. The use PrimaryOLAPDB as default option will assign the PrimaryOLAPDB to the module, if the configured OLAP database is not defined in the target application.

To copy the module, click “Save” from the menu.

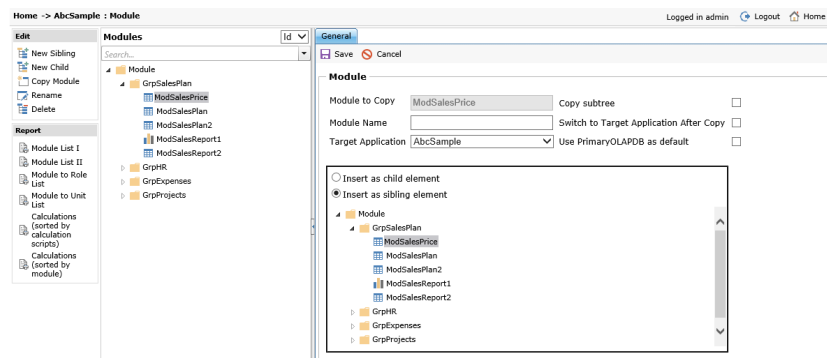


Fig. 30 - Copy Module

Detail Module – Menu Items

Create Table

If the configured module has a module type “DET” the corresponding database table can be managed.



For modules with the module type “DET” the *Create Table* option will appear in the admin toolbar. The *Create Table* option provides the functionality for database tables to be created automatically from the definition in the detail module.

It is important that the detail module workbook file has been defined in the module specifics section. If no detail module workbook or table definition is found for the specified module no table can be created in the database.

If the table already exists and contains data then the table cannot be overwritten. However if the table exists and there is no data then the existing table will be overwritten by the new definition.

Clear Table

It is possible to delete the detail module table contents. This can be done by clicking on the “Clear Table” button. By clicking this button, all database records will be deleted for the defined detail module database table.

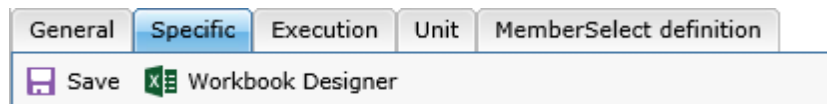
Delete Table

To delete a detail module database table, click on the “Delete Table” button. By clicking the delete table button the database table for the configured detail module will be removed.

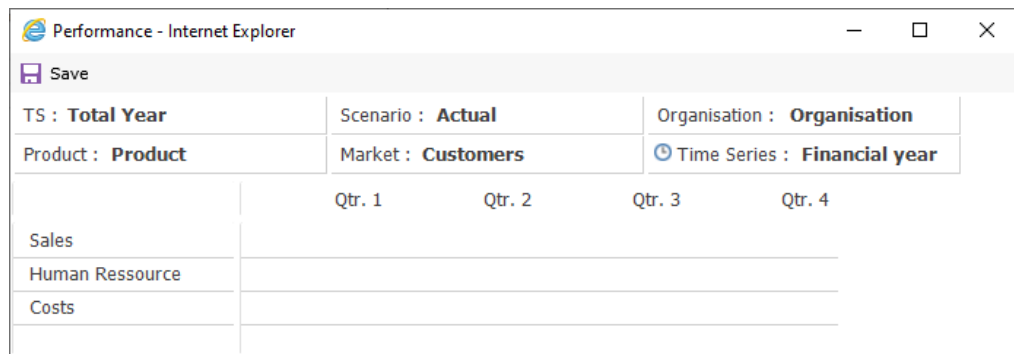
Workbook Designer

The workbook designer can be used to create workbooks for data entry. The result of the workbook designer can be used directly for data entry.

If the module is of type workbook module, the Workbook Designer button will be displayed in the module specifics menu, in the module administration.



Clicking the button “Workbook Designer” will open the workbook designer in a new window. The Workbook Designer uses the Performance Analytics to define the layout.



The workbook layout can be defined, by dragging and dropping rows, columns, offgrids and also selecting new members.

	Jan	Feb	Mar	Apr	May	Jun
Quantity	2.954.770,00	3.005.150,00	2.807.784,00	2.966.918,00	2.826.468,00	2.725.694,00
Price	7,86	8,02	8,45	8,07	8,04	8,09
Rebate	1.613.004,57	1.496.508,82	1.944.021,89	1.720.881,82	1.480.587,92	1.596.599,74
Rebate in pct	6,95	6,21	8,19	7,18	6,51	7,24
Cash Discount	348.244,26	361.449,30	356.066,97	359.301,06	341.000,82	330.656,34
Cash Discount %	1,50	1,50	1,50	1,50	1,50	1,50
Other Discounts						
Salaries & wages						
Gesetzl. Sozialauf...						
No. of employees						
FTE						
Depreciation						
Personalaufwand						
Occupancy costs						
Automation costs						
Machinery						
Communication	3.446,00	3.446,00	3.446,00	3.446,00	3.446,00	3.446,00
Office supplies	647,00	647,00	647,00	647,00	647,00	647,00
Postage, freight c...	5.806,00	5.806,00	5.806,00	5.806,00	5.806,00	5.806,00
Fleet expenses	9.993,00	9.993,00	9.993,00	9.993,00	9.993,00	9.993,00
Travel expenses	2.021,00	2.021,00	2.021,00	2.021,00	2.021,00	2.021,00
Advertising / mar...	9.297,00	9.297,00	82.047,00	9.297,00	9.297,00	9.297,00
Insurance, contrib...	7.591,00	2.741,00	2.741,00	2.741,00	2.741,00	2.741,00
Human resources...		1.940,00	5.578,00	2.910,00		
Other administrati...	7.332,00	7.332,00	7.332,00	7.332,00	7.332,00	7.332,00

After the workbook layout has been defined, the result can be saved by clicking the “Save” button.

A dialog will appear and the user is prompted to enter a file name and extension (xls or xlsx), and specify the directory where the workbook should be saved.

On saving the workbook, if the option “Set path in module specifics” is checked, the file path will be automatically set for the parameter *ModuleFile*.

After the file has been saved, it can be opened in Excel for advanced editing or opened directly in the client.

The decimal places configured in the designer are also applied for each data cell in the workbook.

Member Select Designer

The member select designer can be used to create simple member selects without any technical knowledge.

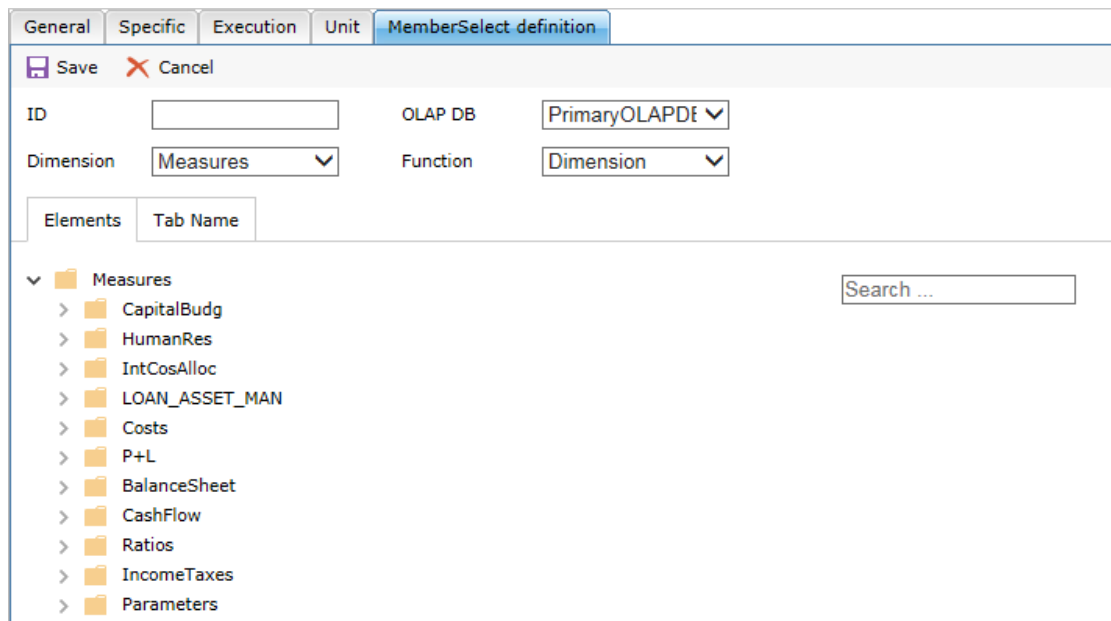


Fig. 31 - Member Select Designer

To create a new member select, select the *Designer* source in the *MemberSelect definition* window and click on the *New* button.



Fig. 32 - Member Select Toolbar

A member select ID has to be specified, where no duplicate IDs within the same Performance application are allowed. Next to the ID the OLAP database can be selected. In the second row the desired dimension and selection function can be chosen.

The following functions are available:

- **Dimension**
The entire dimension is being used. No elements can be selected within the tree.
- **Member**
Only the selected members are included in the member select.
- **IDescendants**
The selected members and all of their descendants are included in the member select.
- **Descendants**
The descendants of the selected members are included in the member select.
- **IChildren**
The selected members and all of their children are included in the member select.
- **Children**

The children of the selected members are included in the member select.

- **UDA**
A list of all available UDAs in the selected dimension are displayed. Only members which have one of the selected UDAs are included in the member select
- **Attribute**
All available attributes in the selected dimension are displayed. Only members which have one of the selected attributes are included in the member select.
- **Level 0**
All lowest members of the selected dimension are included in the member select. No elements can be selected within the tree.

To search for a member within the member tree (not available for *Attribute* and *UDA*) the name of the member can be typed into the search text box. The search results will appear below the text box.

To navigate through the result list, the up, down and return keys can be used on the keyboard or the desired member can be clicked with the mouse.

The *Tab Name* section allows the title of the member select to be defined for each available language.

User Administration

The following sections introduce the user concept. This is followed by a description of the dialog for user administration.

Note: The available functionalities may differ, depending on whether a MOLAP or a ROLAP database is used.

User Concept

Performance users need to be set up in both the Performance repository and the *PrimaryOLAPDB*. Also, the users need to be set up in all OLAP databases whose data they are to access (OLAP databases of the relevant modules).

Note that, when user data is managed via Performance, user data is updated automatically in the OLAP databases only if the configuration parameters "CreateUserOnOLAPServerToo" and/or "CreateUserAccessRightsOnOLAPServerToo" are set to "True". Otherwise, user data need to be updated manually in the OLAP databases.

Note: The configuration parameters "CreateUserOnOLAPServerToo" and "CreateUserAccessRightsOnOLAPServerToo" are available only if MOLAP databases are used. For ROLAP databases these parameters are ignored.

Access Permissions

In Performance, access permissions are defined by assigning roles. To be able to use the full functionality (read, write, calculate data), the user needs appropriate permissions in the OLAP database (calculate permission). Any user set up in the OLAP database via Performance automatically receives these permissions.

The former EncryptIdentity functionality is replaced by the connection user. When the

configuration parameter UseOlapConnectionUser is set to true, the whole access to the underlying OLAP database is handled by the connection user configured in the CubusAppServer.exe.config file.

The “User Administration” dialog can be accessed by clicking on the user item in the main menu. The following screen will then be displayed where the Performance users can be configured.

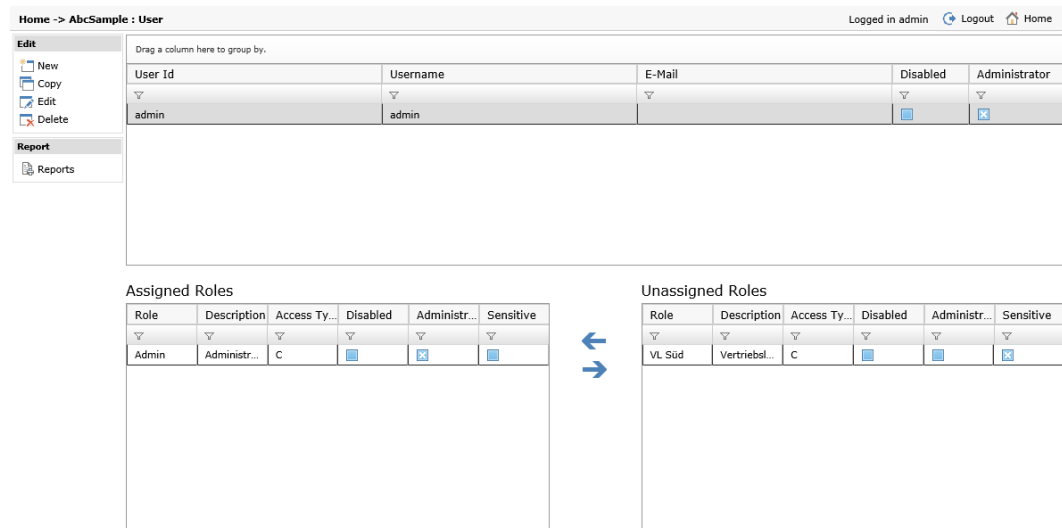


Fig. 33 - User/Role Configuration

Users' permissions to access modules and units are defined via roles. First, you need to define roles that imply specific access permissions. In addition, you can set up users with specific roles.

These roles can then be assigned to one or multiple users. By the same token, each user can be assigned one or multiple roles. At least one role needs to be assigned to each user. Optionally, any user can have multiple roles.

Creating a User

Use the Administration Website to create new users.

Repository

The user data is stored without password in the Performance repository.

OLAP Database

If the "CreateUserOnOLAPServerToo" configuration parameter is not specified or if it is set to FALSE, no action is performed in the OLAP databases.

If the "CreateUserOnOLAPServerToo" configuration parameter is set to TRUE, the user is created or updated in each configured OLAP database:

- If the user name and password are specified, the user is created (unless the user already exists). The user's database access permissions are then adjusted.
- If no password is specified, the system checks whether or not the user already exists. If

the user exists, the database access permissions are adjusted. If the user does not exist, an error message is issued.

- Access permissions of the users:
Admin: Create User and Application Designer.
Client: Calculate

Removing a User

Use the Administration Website to remove users.

Repository

The user data are removed from the Performance repository.

OLAP Database

If the "CreateUserOnOLAPServerToo" configuration parameter is not specified or if it is set to FALSE, no action is performed in the OLAP databases.

If the "CreateUserOnOLAPServerToo" configuration parameter is set to TRUE, the user's access permissions to the configured databases are removed.

Renaming a User

Use the Administration Website to rename a user.

Repository

The user data are updated in the Performance repository.

OLAP Database

If the "CreateUserOnOLAPServerToo" configuration parameter is not specified or if it is set to FALSE, no action is performed in the OLAP databases.

If the "CreateUserOnOLAPServerToo" configuration parameter is set to TRUE, the user is created or updated in each configured OLAP database. Be sure to distinguish between the following cases:

- The old user does not exist in the OLAP database and the new user does not exist in the OLAP database:
The new user is created in the OLAP database (see Creating a User).
- The old user does not exist in the OLAP database but the new user already exists in the OLAP database:
The existing (new) user is updated, i.e. the access permissions to the OLAP database are adjusted. If a password is specified, the user's password is also updated (see Creating a User).
- The old user exists in the OLAP database but the new user does not exist in the OLAP database:
The old user is renamed and the access permissions to the OLAP database are adjusted (see Creating a User).

- The old user exists in the OLAP database and the new user exists in the OLAP database: The action is aborted and an associated error message is returned.

Menu Items

The following toolbar buttons are available in the User dialog:

- **New**
Creates a new user.
- **Copy**
Creates a copy of an existing user with/without role assignments
- **Edit**
Changes user information.
- **Delete**
Deletes the selected user.

New or Edit User

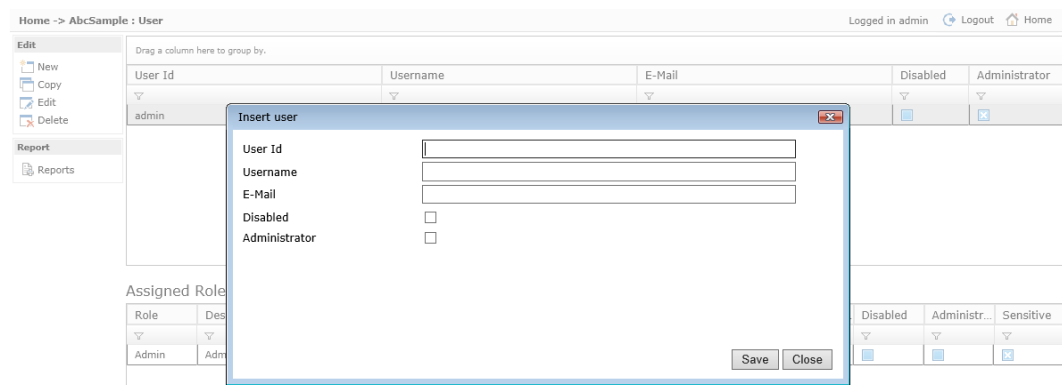


Fig. 34 - User administration dialog

The user dialog contains the following fields:

UserId

The UserId is a unique identifier that the user will use to log into Performance.

Username

This field is used to specify the actual users' name.

E-Mail

Optional field to enter the E-mail address of the user. Can be used to send E-mail based on a workflow.

Password / Confirm

Use the "Password" input field to specify the password the user needs to enter to identify him/

her when logging on to the Performance client. The password is required when the user is to be created in the OLAP database or when the user's password is changed in the OLAP database ("CreateUserOnOLAPServerToo" configuration parameter set to "True"). Note that the password is subject to the restrictions of the OLAP database. If the user already exists in the OLAP database or if the "CreateUserOnOLAPServerToo" configuration parameter is set to "False", the password does not need to be specified and will be ignored.

Administrator

Use this field to grant administrator permissions (e.g. access to the Admin Client, etc.) to a user.

Note: Administrator permissions can be granted to users as well as to roles (see "Role"). The difference between a user that has the administrator flag set and users that have a role with administrator permissions is that a user with administrator flag can see all applications, add users and change global configuration parameters, whereas users with a administrator role can only see the application where the user has a administrator role. Users with an administrator role also can't change any global configuration parameters or create users within their application.

Disabled

Use this checkbox to deactivate/activate the selected user. For example, you can deactivate or activate users who only require temporary access to the Performance Client.

Note: In addition to deactivating users, you can also explicitly deactivate roles (see "Role").

Copy User

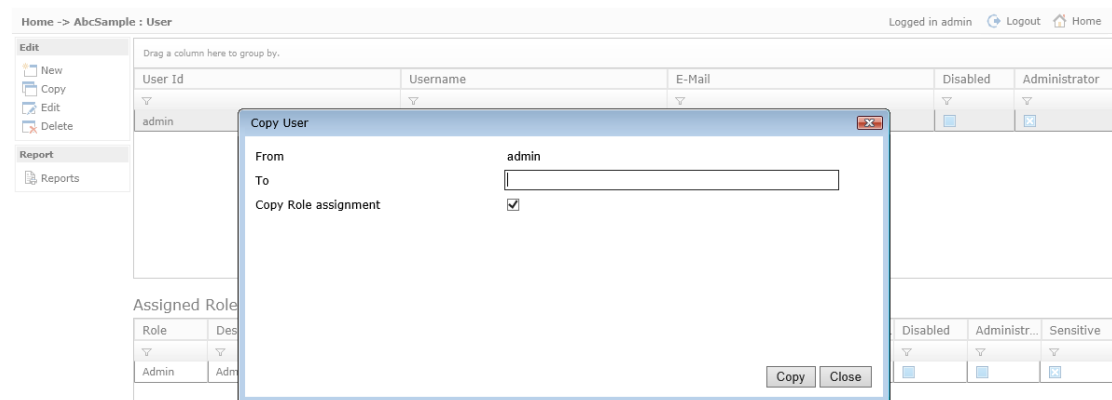


Fig. 35 - User administration dialog for copy user

The user dialog contains the following fields

From

This field contains the UserID of the selected user. This is the source user which is going to be copied.

To

This field is used to specify the UserID of the new (copied) user.

Password / Confirm

The password is required when the user is to be created in the OLAP database or when the user's password is changed in the OLAP database ("CreateUserOnOLAPServerToo" configuration parameter set to "True"). For further information see "Creating a User".

Copy role assignment

Select this option, if the new user should be assigned to the same roles as the source user.

Assigning Roles

Roles can be assigned in the dialog box "Assigned Roles". Simply select a role and then use the arrow buttons to assign and un-assign roles to users. The right-hand list box lists all roles available in Performance, while the left-hand list box lists the ones that are currently assigned.

Use this dialog to assign one or multiple roles to the selected user. You may add or remove multiple roles.

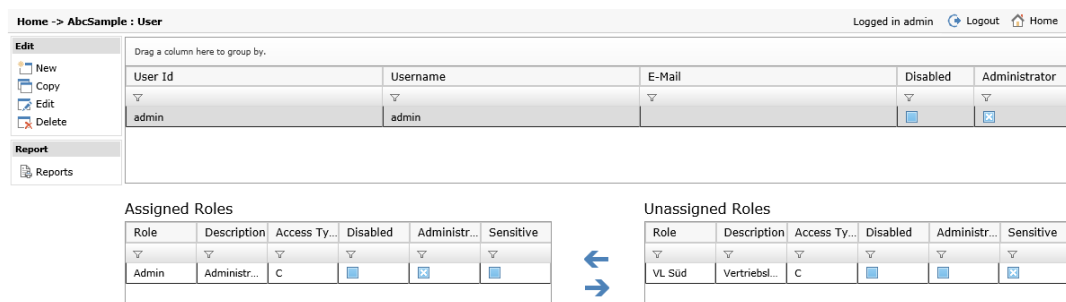


Fig. 36 - User dialog with separate pane for role assignment

If the assignment of roles leads to any clashes in permissions (e.g., the first role grants the user a permission while the second role denies the same permission), the granted permission overrides the denied permission.

Once a user has been granted a permission, that permission remains valid.

Role Administration

The user roles can be defined in the role administration dialog. The role is at the heart of the permission concept in Performance. A role must be assigned to one or more modules, and a module must be assigned to one or more units.

For a detailed description of the assignment of permissions, refer to "Access Permissions".

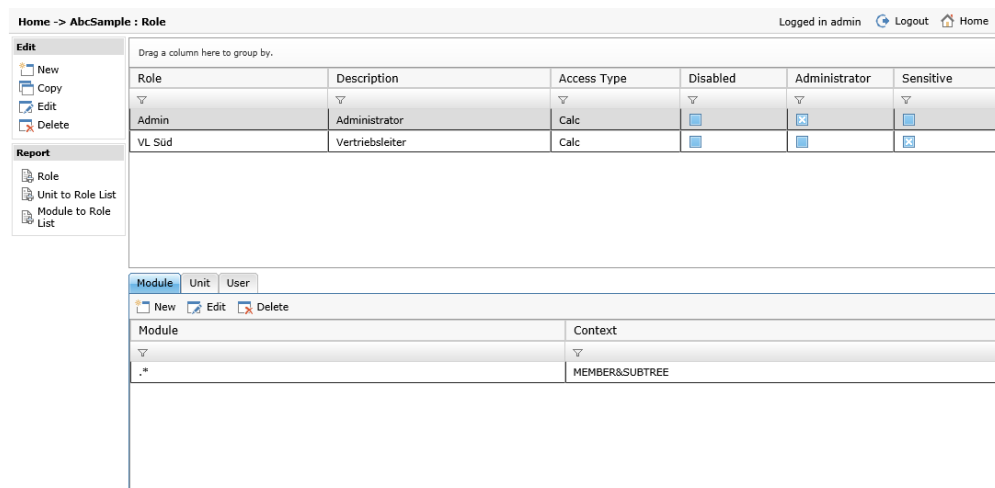


Fig. 37 - Role administration dialog

Use the “New/Edit” Roles dialog to specify the following:

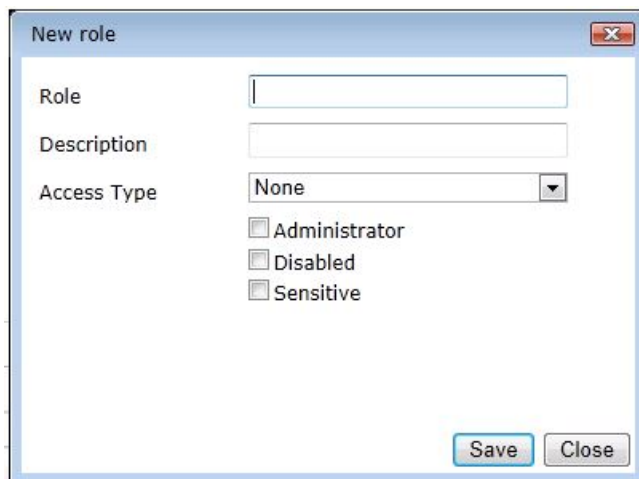


Fig. 38 - Role Dialog for New/Edit Role

RoleId

Use this field to specify a useful name for the role. The role name is only shown to the administrator when managing roles via the Administration Website. The RoleId is also unique.

Description

Use this input field to enter a description or note. This description is only shown to the administrator when managing roles via the Administration Website.

AccessType

This field defines the role permissions. The following permissions can be assigned:

- None
The role only has no access to the modules. The module tree is visible but the module content is not accessible.
- Read permission
The role only has read access to the modules.

- Write permission
The role has both read and write access to the modules.
- Calculation permission
The role has write permission and can also initiate execution steps (of type "CALC").

Administrator

Use this field to define whether or not the defined role includes administrator permissions (e.g. access to the Administration Website, etc.).

Note: In addition to granting role-specific administrator permissions, you can also grant user-specific permissions (see "User"). The difference between a user that has the administrator flag set and users that have a role with administrator permissions is that a user with administrator flag can see all applications, add users and change global configuration parameters, whereas users with a administrator role can only see the application where the user has a administrator role. Users with an administrator role also can't change any global configuration parameters or create users within their application.

Disabled

Use this checkbox to deactivate/activate the selected role. For example, you can deactivate or activate roles for users (or user groups) that only require temporary access to the Performance Client.

Note: In addition to deactivating roles, you can also explicitly deactivate users (see "User").

Sensitive

The "Sensitive" option can be used to specify whether or not the role has sensitive access permissions. For sensitive roles, data fields marked as sensitive in detail modules are masked. (For example: If the "salary" field in a detail module is marked as a sensitive field, users of a sensitive role will see the masked value "*****" instead of the actual value. For further information, refer to the "Detail Modules" document).

Use the "Copy" Roles dialog to specify the following:

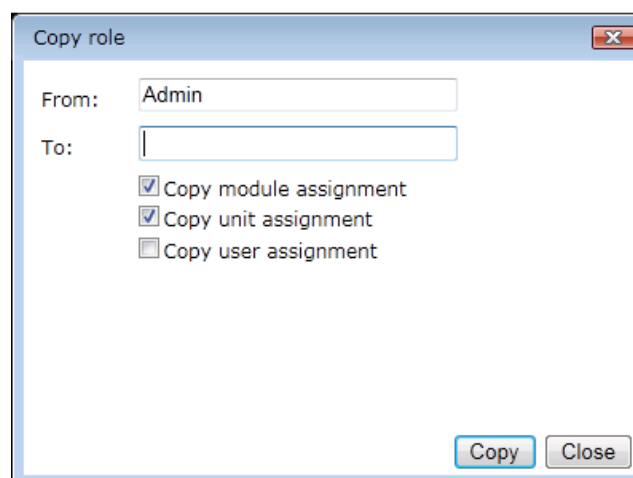


Fig. 39 - Role Dialog for Copy Role

From

This field shows the role id of the source role (read only).

To

Use this field to specify the name for the new role. The Roleid is must be unique.

Copy module assignment (checked by default)

Select this option, if the new role should receive the same module assignments as the source role. For further information, refer to “Module Access Permissions“.

Copy unit assignment (checked by default)

Select this option, if the new role should receive the same module assignments as the source role. For further information, refer to “Access Permissions of Units to Modules”

Copy user assignment

Select this option, if the new role should be assigned to the same users as the source role.

Menu Items

The following toolbar buttons are available in the Role dialog:

- **New**
Creates a new role.
- **Copy**
Copies an existing role with assignments (if selected)
- **Edit**
Saves changes to the role.
- **Delete**
Deletes the selected role.

Assigning Users

Roles can be assigned to users in two places.

- The user tab in the roles dialog
- The assigned roles section in the user administration dialog.

There are two table views listing all the users and all of the roles. There are also two arrows which assign the roles or users between the two dialog boxes. To assign a user to the a role, click on the user and then move the user to the corresponding role dialog box. You may assign multiple roles and users at the same time. Access Permissions

The user log on to a module via the Performance Client Services. The Module Administration or User Administration dialogs are used to define which roles/users can access which units or modules.

Defining the access permissions requires the following procedure:

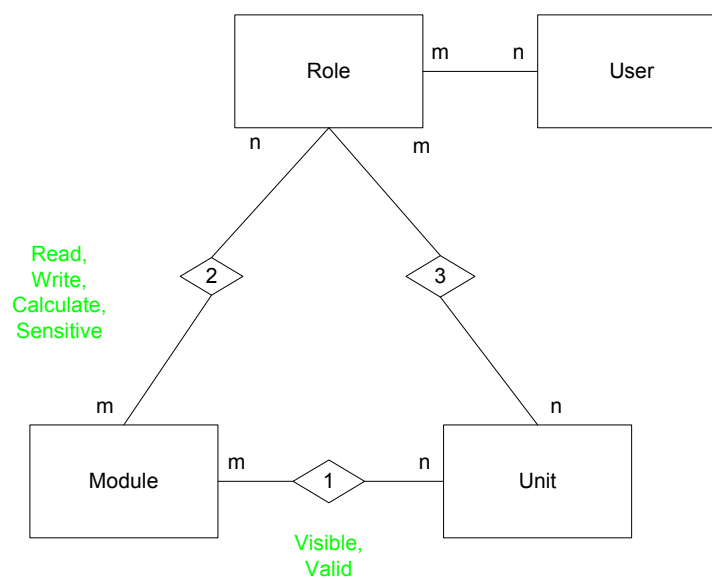
1. Access to units is defined via the modules

For a module, access to one or more units must be defined (see "Access Permissions of Units to Modules").

2. Access to modules is defined via the role
 The permission to access modules is assigned directly to a role (see "Module Access Permissions").
3. Assigning units to roles
 If units are explicitly assigned to a role, the users assigned to this role can access the units that are set up for the selected module as well as for the role itself (intersection set).

A user can be assigned to one or multiple roles and, by analogy, one or multiple users can be assigned to the same role.

The following diagram illustrates the associated relationships:



Access Permissions of Units to Modules

For each module, you can define which units are permitted to access it. Access can only be defined for units from the so-called "master" database of the module.

Press the "Assign Units" toolbar button of the module administration dialog to display an additional window in which you can define the unit access permissions for each module.

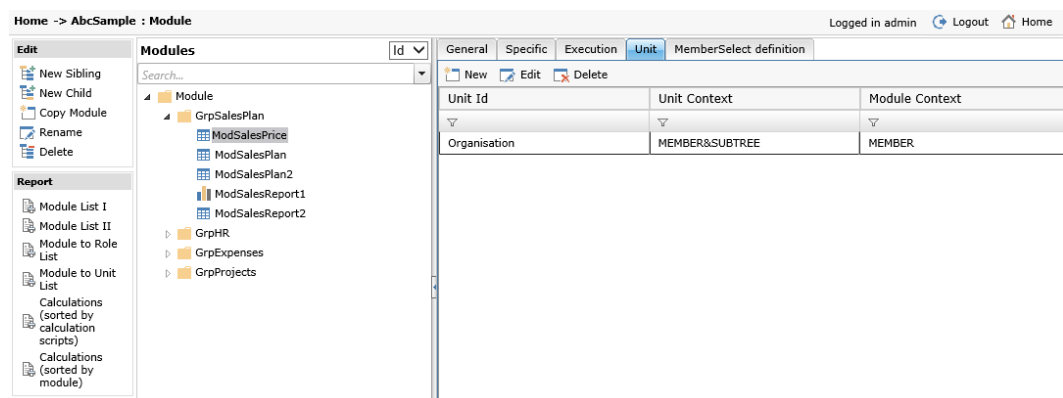


Fig. 40 - Overview tab with unit access permissions

Overview tab

The Units tab lists the access permissions defined for the selected module. To add entries, press the “New” toolbar button. To update existing entries, click the “Edit” toolbar button. To delete entries, click the “Delete” toolbar button.

The right-hand pane contains the following fields:

Unit

To define access permissions, you can either select the applicable units via the tree view in the “Unit” tab (right-hand side of the window) or you can enter the applicable unit name into the input field. When entering the unit name manually, you may use regular expressions.

There are special characters for the regular expression that have to be escaped if you want to use it in a regular context. Those are the followings; [] { } () - + * ? . , \ ^ | \$. You can escape a special character by putting a backslash ahead of them. If you want to use brackets like in a regular context for example, "Region1 (South)"; you have to escape them like this, "Region1 \ (South)".

Additionally, you can specify whether a given permission applies to the specified unit only (context: Member (M)) or for the unit and all of its subunits (context: Member&Subtree (M&S)).

Menu Items

The following toolbar buttons are available in this dialog:

- **New**
Creates a new record.
- **Edit**
Updates the currently selected entry in the Overview table.
- **Delete**
Deletes the selected entry from the overview table.

Module Access Permissions

Modules can be assigned to each role. The “Module Access Permissions” dialog can be started in the Role administration dialog.

Press the “Assign Modules” toolbar button to display an additional window in which you can define the module access permissions for each role.

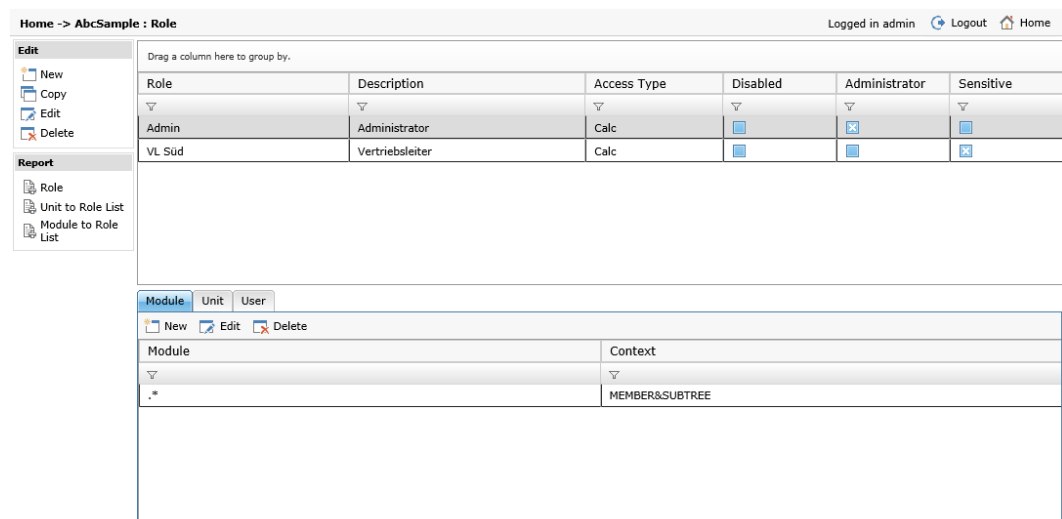


Fig. 41 - Module access permissions

Overview Tab

The Overview tab lists the access permissions defined for the selected role. To add entries, press the “New” toolbar button. To update existing entries, click the “Update” toolbar button. To delete entries, click the “Delete” toolbar button.

The right-hand section of the lower pane contains the following fields:

Module

To define access permissions, you can either select the applicable modules via the tree view in the “Module” tab (left-hand side of the window) or you can enter the applicable module name into the input field. When entering the module name manually, you may use regular expressions.

There are special characters for the regular expression that have to be escaped if you want to use it in a regular context. Those are the followings; [] { } () - + * ? . , \ ^ | \$. You can escape a special character by putting a backslash ahead of them. If you want to use brackets like in a regular context for example, "Module (Test)"; you have to escape them like this, "Module \(Test\)".

Access permissions can be specified either for the selected module (context: Member (M)) or for the module and all of its submodules (context: Member&Subtree (M&S)).

Menu Items

The following toolbar buttons are available in this dialog:

- **New**
Creates and initializes a new record.
- **Update**
Updates the selected entry in the overview table.
- **Delete**
Deletes the selected record. Note that this menu item is available for marked entries in the overview table only.
- **Info**

Displays information on this dialog.

- **Close**
Closes the Module Access dialog.

Direct Unit Access Permissions

In addition to assigning modules to roles and assigning units to modules, you can restrict the unit access permissions for a role. Via the Role Administration screen, you can invoke the dialog for defining direct access permissions via the “Unit Access” toolbar button.

If units are explicitly assigned to a role, the users assigned to this role can access the units that are set up for the selected module as well as for the role itself (intersection set).

Note: The access permissions of roles to units are defined without specifying an OLAP database. This implies that the assignment is “OLAP DB unspecific” – i.e. the permission applies to any OLAP database that includes an corresponding unit.

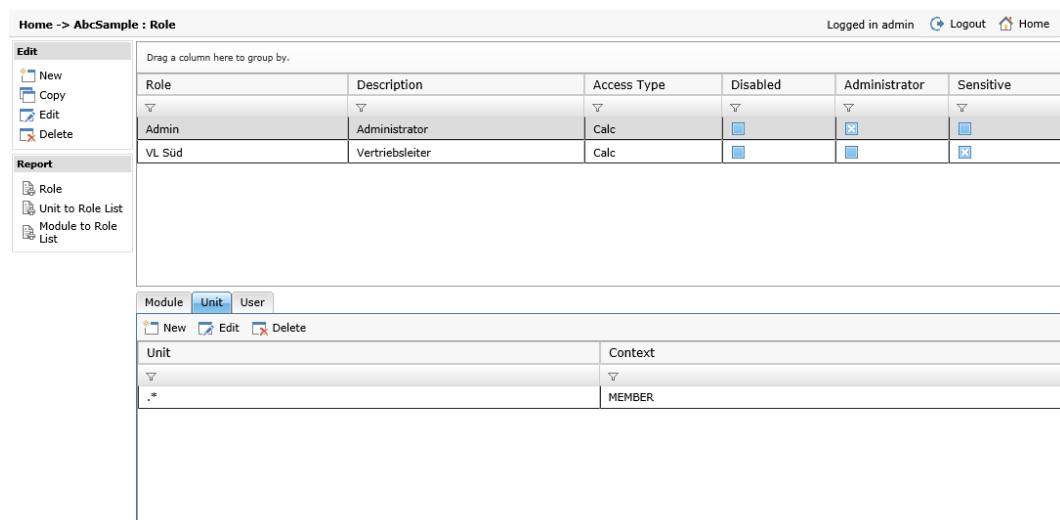


Fig. 42 - Direct Unit Access Permissions

Overview Tab

The Overview tab lists the defined access permissions. To add entries, press the “New” toolbar button. To update existing entries, click the “Update” toolbar button. To delete entries, click the “Delete” toolbar button.

The right-hand section of the lower pane contains the following fields:

Unit

To define access permissions, you can either select the applicable units via the tree view in the “Unit” tab (left-hand side of the window) or you can enter the applicable unit name into the input field. When entering the unit name manually, you may use regular expressions.

Additionally, you can specify whether a given permission applies to the specified unit only (context: Member (M)) or for the unit and all of its subunits (context: Member&Subtree (M&S)).

Menu Items

The following toolbar buttons are available in this dialog:

- **New**
Creates and initializes a new record.
- **Update**
Updates the selected entry in the overview table.
- **Delete**
Deletes the selected record. Note that this menu item is available for marked entries in the overview table only.
- **Info**
Displays information on the dialog.
- **Close**
Closes the Direct Unit Access Permissions dialog.

Unit Administration

The synchronization of the workbook server with the configured OLAP databases takes place automatically when the application server starts. The structures (member names), alias tables, alias names, etc., are read into memory (cache) during this process.

If you want to synchronize the structures without restarting the application server (e.g. following changes of alias names or the configuration of a new OLAP database), the synchronization can be started via this menu item for each OLAP-Database separately. It is also possible to refresh all database outlines by selecting the “Refresh All” option. The last refresh date shows, when the database structure was last updated.

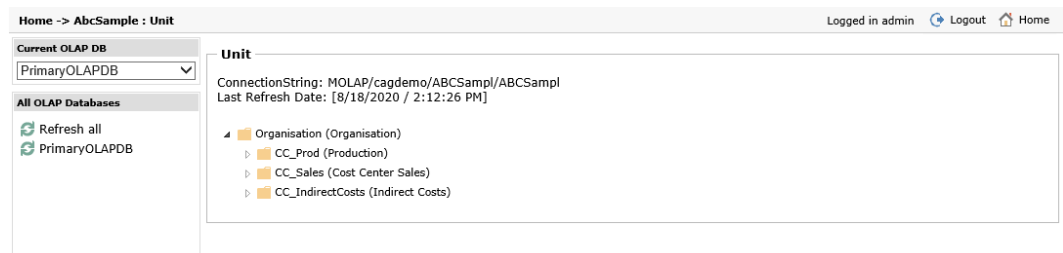


Fig. 43 - Unit Dialog

Note: In this dialog, you can only view the units defined in the OLAP model. Structural changes (e.g. adding or removing units) are only allowed in the OLAP DB.

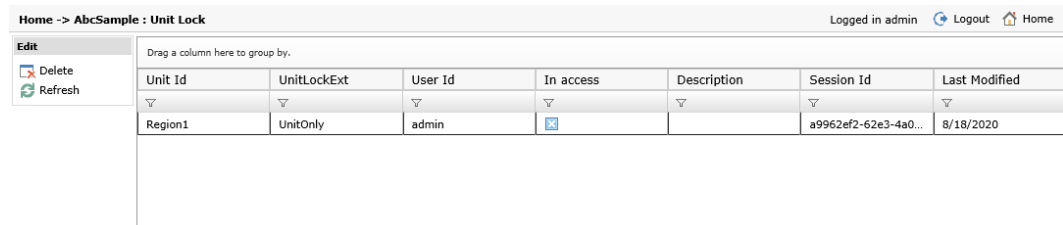
Menu Items

The following toolbar buttons are available in the Units dialog:

- **Refresh All**
Gets the latest structure for all configured OLAP database.
- **PrimaryOLAPDB (and further OLAP databases)**
Refresh only the specific OLAP database structure.

Unit Lock

The Unit Lock dialog is responsible for unlocking units that are currently in access by users. A locked unit should be automatically cleaned up in the database when the user's session ends. However if for some reason the unit needs to be unlocked manually, it can be done using the delete menu item.



Unit Id	UnitLockExt	User Id	In access	Description	Session Id	Last Modified
Region1	UnitOnly	admin	[lock icon]		a9962ef2-62e3-4a0...	8/18/2020

Fig. 44 – Unit Lock Dialog

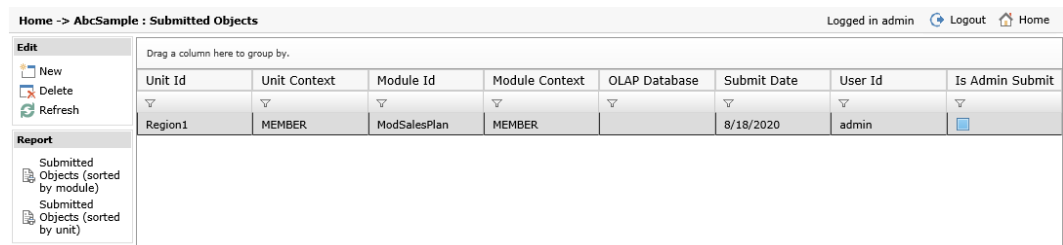
Menu Items

The following toolbar buttons are available in the Units dialog:

- **Delete**
Deletes a locked unit. Therefore unlocks the unit that is currently in access by a particular user.
- **Report**
Displays a report of the locked units.

Submitting/Un-submitting Objects

The administrator can submit units and modules. Objects can be submitted separately (e.g. you can submit a unit or a single module for all units) or combined (e.g. specific modules for a number of units). The associated locking is "OLAP DB unspecific". For example, if the administrator defines a lock for "Reg.*" units for all modules, each module will be locked for the units in the associated "master" database that match the "Reg.*" pattern.



Unit Id	Unit Context	Module Id	Module Context	OLAP Database	Submit Date	User Id	Is Admin Submit
Region1	MEMBER	ModSalesPlan	MEMBER		8/18/2020	admin	[lock icon]

Fig. 45 - Submit Objects Dialog

Overview

The overview displays all submitted units and modules. In this table, you can add or delete entries.

To add entries, press the "New" toolbar button. To delete entries, click the "Delete" toolbar button.

Unit

For the lock, the appropriate objects can be selected in the “Units” and “Modules” tree views or they can be specified manually in the associated input field. Regular expressions can be used in the input field. An empty input field corresponds to the regular expression “.*”. Additionally, you can specify whether the lock is to apply to the specified member only or to the entire subtree of the member.

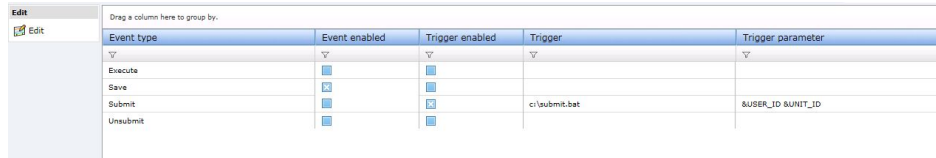
Menu Items

The following toolbar buttons are available in this dialog:

- **New**
Creates a new Submitted Object.
- **Delete**
Deletes a selected Submitted Object.

Event Management

This dialog is used for configuring events and their associated triggers.



Event type	Event enabled	Trigger enabled	Trigger	Trigger parameter
Execute	<input type="checkbox"/>	<input type="checkbox"/>		
Save	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Submit	<input type="checkbox"/>	<input type="checkbox"/>	c:\submit.bat	\$USER_ID \$UNIT_ID
Unsubmit	<input type="checkbox"/>	<input type="checkbox"/>		

Fig. 46 - Event Management Setup

The Event Management dialog includes the following fields:

EventType

The following events can be configured:

- Execute
- Save
- Submit
- Unsubmit

No further events can be added.

Event enabled

Defines whether an event is enabled or not. If an event is enabled a log entry will be written to the EventLog table when the event is fired.

Trigger enabled

Defines whether a trigger is enabled or not.

Trigger

External program launched by the event, provided that the associated trigger is enabled (active).

Be sure to specify the absolute pathname.

For example, you can specify a program such as an external shell script that sends mails to specific users.

Note: The program will be executed asynchronously.

Trigger Parameters

Additional, freely selectable parameters to be passed to the external program. When you specify any of the following parameters, Performance will replace these by the actual value:

- &APPLICATION_ID
- &USER_ID
- &MODULE_ID
- &UNIT_ID

Menu Items

The Process Management dialog includes the following toolbar buttons:

- **Edit**
You can update the configuration parameters by pressing the “Edit” toolbar button.

Client Integration

The following administration modules can be integrated into the client using the corresponding URLs in an URL Module:

Administration URL

Module	URL
Module	\$ADMIN_WWW_ROOT\$/UIModuleConfiguration/AbcModulConfiguration.aspx?MasterPage=false
User	\$ADMIN_WWW_ROOT\$/UIUserManagement/AbcUserManagement.aspx?MasterPage=false
Role	\$ADMIN_WWW_ROOT\$/UIRoleManagement/AbcRoleManagement.aspx?MasterPage=false
Unit	\$ADMIN_WWW_ROOT\$/UIInputUnits/AbcUnit.aspx?MasterPage=false
Unit Lock	\$ADMIN_WWW_ROOT\$/UIInputUnits/AbcUnitLock.aspx?MasterPage=false
Submitted Objects	\$ADMIN_WWW_ROOT\$/UISubmittedObjects/AbcSubmittedObjects.aspx?MasterPage=false
Event Management	\$ADMIN_WWW_ROOT\$/UIProcessManagement/AbcProcessManagement.aspx?MasterPage=false

Note: When integrating administration modules into the client, the following customHeader has to be set in the web.config of the Performance Client web application:

```
<system.webServer>
  <httpProtocol>
    <customHeaders>
      <add name="X-UA-Compatible" value="IE=9"></add>
```

```
        </customHeaders>  
    </httpProtocol>  
</system.webServer>
```

6. Toolbars

It is possible to configure some aspects of the toolbars in Performance. The chapter below describes what can be configured in the toolbars and how to do this.

The general toolbar can be found on the top right and is always shown.



Logged in cubusadmin  Logout  Home

Hide buttons

Single buttons can be hidden by using the configuration parameter *HideToolbarButtons*.

Task button

The task button is only visible when the configuration parameter *EnableWorkflow* is true.

7. Adhoc Analysis

The Adhoc Analysis allows the user to open a new Performance Analytics report on the fly, which automatically has the same point of view as the current workbook. Therefore, a new temporary view will be created within the user's private folder.

Requirements

To use the Adhoc Analysis, your installation needs to meet the following requirements.

- Performance Analytics Client has to be installed
- Performance Analytics has to be configured in the Performance Administration
- Private Folders have to be enabled in Performance Analytics
- A database, which matches the workbook's OLAP database, has to be set up in Performance Analytics

In addition, the user account needs the following system role permissions:

- Manage Folders
- Manage Views/Canvases
- Has Private Folder

8. Workflow

This chapter describes the workflow functionality of Performance and how to configure it.

Basic Terms

Definition

A workflow in Performance consists of a group of tasks that users have to complete in a certain order on certain levels to accomplish a higher task (e.g. to complete the sales plan for an organization). The workflow is completed when the user on the highest level has finished their task by submitting it.

Example:

An example for a workflow could be a sales planning cycle.

First the *Regional Managers* have to input their plan data and to submit their regions (units). After a submission the region is locked for input.

Then the *Sales Manager* has to review the different inputs and either accept or reject the individual regions. Rejecting would open the region for input again and the *Regional Manager* can modify their data.

After the Sales Manager has accepted all regions the *CFO* can review the sales plan. The *CFO* has the task to accept or reject the sales plan. After accepting it the sales workflow is finished.

Workflow Tasks

A task within a workflow is assigned to a certain user and a certain unit. The task either includes entering data through several modules in Performance or to validate data that has been entered by accepting or rejecting it.




Workflow Levels

A workflow can consist of several levels that depend on each other. Tasks on one level have to be completed before tasks on the next level can be executed. E.g. the regional sales managers have to finish the sales plan for their regions before the sales manager can approve the plan on the next level.

Data entry in Performance modules is only possible on the first level. After submitting data on the first level the data cannot be modified by the following tasks on higher levels. Tasks on higher levels only consist of accepting or rejecting data that has been entered on the first level.

Workflow Actions

Different actions are allowed for tasks on different levels. A user can execute actions on tasks that have been assigned to them as well as on tasks one level below their level in the workflow.

Button	Description
 Submit	Submit the current task
 Accept	Accept the current task
 Reject	Reject the current task

Submit

By submitting a task the user confirms that they have finished it. All descendant modules of the workflow will be locked for input (for the context module and unit) after the user submits on the first level.

When a user submits their task on a level above the first workflow level this means implicitly that the user accepts all tasks one level below their level in the workflow. All tasks that have not been submitted at this time will be submitted automatically.

Accept

A user can only accept tasks of users below their workflow level. This indicates that the lower level task is finished and is approved without submitting.

Reject

A user can reject a task of user below their level in the workflow. When a task on the first level is rejected, the unit is automatically reopened for input.

When the user clicks the reject button on their own task (which is only possible on a level above the first workflow level) they do not reject their own task but all tasks below their level in the workflow. This automatically reopens the units again.

Workflow Administration

This chapter describes how to set up a workflow in Performance. All necessary Excel sheets can be found in the sheets folder of the ABCSample application.

Workflow Module Structure

The first step when setting up a workflow is to define a workflow module. This module has the type WORKFLOW and will be the place where actions for the tasks (submit, accept, reject) will be executed.

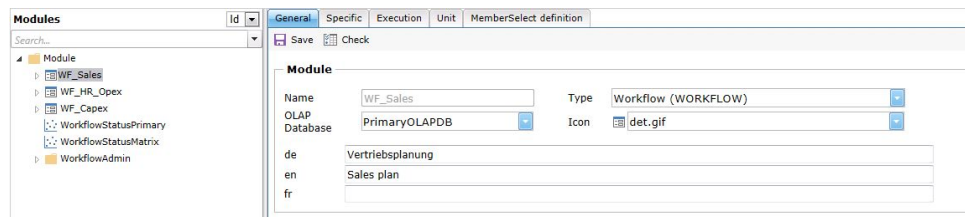


Fig. 47 -Workflow Module

All modules that are defined as descendants of the workflow module belong to this workflow. All modules that need to be assigned to the workflow must be defined as descendants of the workflow module. This can consist of any types of modules e.g. workbook modules, report modules.

Workflow Master Data

The workflow master module (WorkflowMaster.xls) allows to enter the master data for a workflow.



Fig. 48 - Workflow Master

The following fields can be filled:

Field	Description
Workflow name	The name of the workflow
Module	The workflow module to which the workflow is assigned. Only modules of the type WORKFLOW can be selected.
Level1 ... Level4	The names of the different levels that the workflow comprises.
Description	The description for the workflow.

Workflow Tasks

The workflow detail module (WorkflowDetail.xls) allows the assignment of users to the different units and levels of the workflow. Each assignment is considered as a task.

Workflow - Sales

Unit	Level	Owner
Organisation	CFO	CFO
Sales	Sales	Ben
South	Plan	Peter
North	Plan	Peter
East	Plan	John
West	Plan	John
Internet.	Plan	John

Fig. 49 - Workflow Detail

Log Entry

All activities in a workflow are stored in the repository. The first log entry is created when the task is created.

Workflow Access Rights

The workflow access rights are based on the existing access rights of Performance (roles and users). When the tasks are saved the appropriate roles and Role-To-User access rights are created automatically.

Workflow Roles

For each task a new role is created. The role id has the following format:

<ModuleId>\${UnitId}\${Level}.

Example

ModuleId: WF_Sales
 UnitId: Region1
 Level: Level1
 RoleId: <WF_Sales>\${Region1}\${Level1}

User-To-Role Assignment

The new role will be assigned to the selected user.

Role-To-Unit Assignment

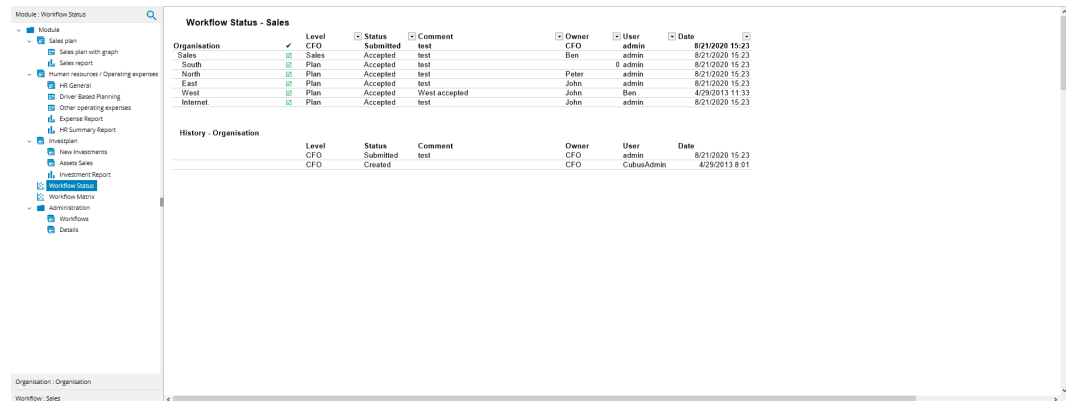
The role will get access to the selected unit and all of its descendants.

Role-To-Module Assignment

The role will get access to the workflow module and all of its descendants.

Workflow Status Module

The workflow status module (WorkflowOverview.xlsx) can be set up to show the status of a single workflow.



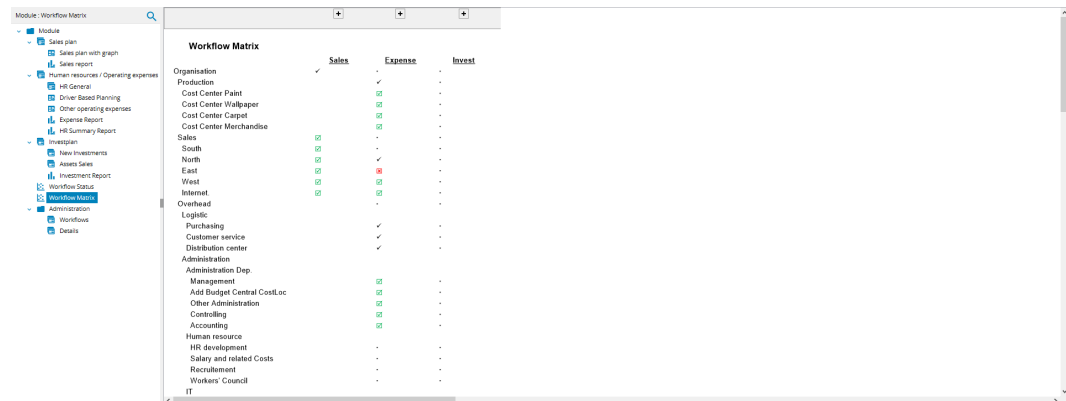
Organisation	Level	Status	Comment	Owner	User	Date
CFO	CFO	Submitted	test	CFO	admin	8/21/2020 15:23
Sales	Sales	Accepted	test	Ben	admin	8/21/2020 15:23
South	Plan	Accepted	test	Ben	admin	8/21/2020 15:23
North	Plan	Accepted	test	Peter	admin	8/21/2020 15:23
East	Plan	Accepted	test	John	admin	8/21/2020 15:23
West	Plan	Accepted	West accepted	John	Ben	4/29/2013 11:33
Internet	Plan	Accepted	test	John	admin	8/21/2020 15:23

Organisation	Level	Status	Comment	Owner	User	Date
CFO	CFO	Submitted	test	CFO	admin	8/21/2020 15:23
CFO	CFO	Created	test	CFO	CubusAdmin	4/29/2013 8:01

Fig. 50 - Workflow Status

Workflow Matrix Module

The workflow matrix module (WorkflowMatrix.xlsx) can be set up to show the status of all workflows.



	Sales	Expense	Invest
Production	✓	✓	-
Cost Center Paint	☐	☐	-
Cost Center Wallpaper	☐	-	-
Cost Center Carpet	☐	-	-
Cost Center Merchandise	☐	-	-
Sales	☐	-	-
South	☐	-	-
North	☐	✓	-
East	☐	☐	-
West	☐	☐	-
Internet	☐	☐	-
Overhead	-	-	-
Logistic	-	-	-
Purchasing	✓	-	-
Customer service	✓	-	-
Distribution center	✓	-	-
Administration	-	-	-
Administration Dep.	-	-	-
Management	☐	-	-
Add Budget Central Cost Loc	☐	-	-
Other Administration	☐	-	-
Controlling	☐	-	-
Accounting	☐	-	-
Human resource	-	-	-
HR development	-	-	-
Salary and related Costs	-	-	-
Recruitment	-	-	-
Workers' Council	-	-	-
IT	-	-	-

Fig. 51 - Workflow Matrix

9. Tools

Batch Client

When formulas in the OLAP-Model, OLAP calculation scripts, CalcServer workbooks or external calculations have been changed, it may be necessary to perform a recalculation for some or all units.

This recalculation can be initiated by means of the *BatchClient* tool.

This method can be used for purposes such as the periodic nightly initiation of calculations via a scheduler.

Command Line Parameter

The program is called via the command line (shell) as follows:

```
BatchClient -b <BatchFile> -a <ApplicationId>  
            -l <LogFile> [-c] [-x] [-p <AppServerEncryption>] [-r] [-  
d <Delay>]  
            [-s <AppServerSchema>] [-n <Alias>]
```

The "BatchFile" contains the data for login to the Application Server, the modules and units to be selected, and the tasks to be performed. The structure of this file and the associated commands are described in the following sections.

The "ApplicationId" is the name of the corresponding application (e.g. ABCSample).

The "LogFile" contains the log of all commands executed.

The "-c" runs the BatchClient in console mode. I.e. no user interface is displayed.

The "Delay" is a value in milliseconds and defines the delay between the execution of two commands. The default value is 500.

The "-x" aborts the Batchclient when an error occurs. If this option is not set, the BatchClient continues with the processing.

The "-p" configures the data encryption between the BatchClient and the application server and must match the application configuration parameter *AppServerEncryption* (Valid values: *None*, *Windows* (default), *Crossdomain*).

The "-r" disables the data compression between the BatchClient and the application server. This option has to be set if the configuration parameter *AppServerCompression* is set to *true* for the application.

The "-s" configures the transport protocol used between the BatchClient and the application server. It needs to match the application configuration parameter *AppServerSchema* (Valid values: *TCP* (default), *HTTP*, *HTTPS*).

The "-n" configures the HTTP/HTTPS path used to access the application server. It is only used if -s is *HTTP* or *HTTPS* and if the application configuration parameter *AppServerWebAlias* has been configured.

During installation, the batch program is stored in:

```
<INSTALL_DIRECTORY>\Bin\Client
```

Note: This batch program can be run only if the Workbook Client is installed.

Structure of the Batch File

Note the following when composing the batch file:

- The file may contain blank lines and comment lines at any arbitrary position.
- Comment lines always begin with # or //.
- The command SET_HOSTNAME followed by the command SET_PORTID or SET_MANAGERPORTID must **always** be called **before** any other commands can be used.
- Any blanks in parameters need to be masked by quotation marks (e.g. "abc demo/ password" as login info.)

The commands available for use with BatchClient are discussed in the following section.

SET_HOSTNAME

The parameter "HostName" contains the name of the IP address of the computer running the Application Server.

This must **always** be the **first** command in a batch file (apart from blank lines or comment lines).

Syntax: **SET_HOSTNAME** <HostName>

Example: **SET_HOSTNAME localhost**

SET_PORTID

The parameter "PortID" contains the associated port of the Application Server.

This must **always** be the **second** command in a batch file (apart from blank lines or comment lines).

If the BatchClient is only used to start or stop applications the parameter "ManagerPortID" has to be used instead.

Syntax: **SET_PORTID** <PortID>

Example: **SET_PORTID 12346**

SET_MANAGERPORTID

The parameter "ManagerPortID" contains the associated port of the Application Server Manager.

If the BatchClient is used to start or stop applications this parameter has to be set.

Syntax: **SET_MANAGERPORTID** <PortID>

Example: **SET_MANAGERPORTID 12345**

Note: If [AppServerSchema](#) has been set to HTTP or HTTPS this parameter has to be set before

using the **LOGIN** command.

START_APP

The parameter "StartApp" contains the ID of the application that needs to be started.

This command will start the CubusAppServer for the specified application.

Syntax: **START_APP <AppID>**

Example: **START_APP AbcSample**

STOP_APP

The parameter "StopApp" contains the ID of the application that needs to be stopped.

This command will stop the CubusAppServer for the specified application.

Syntax: **STOP_APP <AppID>**

Example: **STOP_APP AbcSample**

LOGIN

This command is used for logging in on the Application Server by means of the specified user ID. Any blanks in the name need to be masked by quotation marks.

Multiple logins are not allowed.

Syntax: **LOGIN <UserID>/<Password>**

Example: **LOGIN "abc demo/abc123"**

LOGOUT

This command is used for logging out the logged-in user from the Application Server.

Syntax: **LOGOUT**

SET_MODULE

This command is used for selecting the module. The parameter "ModuleId" is the module id (not the language dependent name). A LOGIN must have taken place before the module can be selected. The batch file can switch between any arbitrary number of modules.

Syntax: **SET_MODULE <ModuleId>**

Example: **SET_MODULE Vertriebsplanung.xls**

SET_UNIT

This command is used for selecting the unit. A LOGIN must have taken place before a SET_UNIT. The batch file can switch between any arbitrary number of units.

Syntax: **SET_UNIT <unit>**

Example: **SET_UNIT Region1**

Note: A Unit must be set explicitly in the batch client before taking other actions. Unlike in the user interface the batch client does not set a default unit set after choosing a module.

SET_OFFGRID

This command is used for selecting an offgrid member. A SET_MODULE for the corresponding module must have taken place before a SET_OFFGRID.

Valid Offgrid IDs are the MemberSelectIds from a *MemberSelect* definition.

Syntax: **SET_OFFGRID <Offgrid-ID> <Member-ID>**

Example: **SET_OFFGRID Produkt Prod_1**

Note: If an offgrid is defined in a module, it must be set explicitly in the batch client before taking other actions. Unlike in the user interface the batch client does not set a default offgrid after choosing a module.

EXECUTE/CALCULATE

This command is used for starting an execution step:

- If no execution script is specified, all required executionsteps provided for the selected module are initiated.
- If an execution script is specified, the specified optional execution script is processed in addition to the required once.

Syntax: **CALCULATE**

Or: **EXECUTE**

Or: **CALCULATE < ExecutionScriptId >**

Or: **EXECUTE <ExecutionScriptId>**

Example: **EXECUTE**

EXECUTE p_detmod_calcserver.xls

Note: A execution script needs to be specified for the optional scripts of type CALC. Required execution scripts are initiated automatically during calculation.

Only a single optional execution script can be specified. If multiple optional scripts are to be initiated, the call

EXECUTE <ExecutionScriptId >

needs to be specified for each of these additional optional scripts.

EXPORT

This command is used for exporting a workbook as a PDF or Excel file.

Syntax: **EXPORT** <export filename> <export type>

Example: **EXPORT** "C:\ExportedPDFDocuments\my.pdf" pdf
 EXPORT "C:\ExportedExcelSheets\my.xlsx" excel

Note: For PDF export the PDF995 printer software must be installed on the machine where the batch client is running.

SET_PROPERTY

This command can be used to set specific properties of the application. Valid **properties** are:

- LANGUAGE <LanguageID>
- ALIASTABLE <AliasName>

The language and the alias table can be switched any number of times. If the properties are not set, the ABC default settings will be used (Language = *de*, Aliastable = *Default*).

Syntax: **SET_PROPERTY** <Property> [<PropertyValue>]

Example: **SET_PROPERTY** LANGUAGE de
 SET_PROPERTY ALIASTABLE Default

SET_PRINTER_TIMEOUT

This command is used only in combination with the Export command.

Syntax: **SET_PRINTER_TIMEOUT** <time in milliseconds>

Example: **SET_PRINTER_TIMEOUT** 5000

SAVE

This command is used for saving the workbook data. This command is optional.

Syntax: **SAVE**

RESET_OUTLINE

This command resets the specified OLAP database on the server.

Syntax: **RESET_OUTLINE** <olapDbName>

Example: **RESET_OUTLINE** PrimaryOLAPDB

RESET_CACHE

This command resets the module, role and unit cache on the server.

Syntax: **RESET_CACHE**

EXIT

This command is used for closing the BatchClient window following processing. This command is optional.

Syntax: **EXIT**

Example

```
# BatchClient example
SET_HOSTNAME localhost
SET_PORTID 12345
LOGIN "abc demo"/abc123

# calculating human resource module
SET_MODULE p_detmod_input.xls
SET_UNIT Kst_Prod1
EXECUTE p_detmod_calcserver.xls

# calculating sales module
SET_MODULE au_VertriebsplanungA.xls
SET_UNIT Region1
SET_OFFGRID Produkt Prod_1
EXECUTE

# export the current module
SET_PRINTER_TIMEOUT 3000
EXPORT "C:\test\test.pdf" pdf
EXPORT "C:\test\test.xlsx" excel

LOGOUT
EXIT
```

CopyComments

The CopyComments tool is used to create copies from existing comments or additional calculations. The console can also be used to find and replace certain values within the NOTE_ID column and the saved additional calculation sheet.

A comment could be text comment (cell, off-grid, module, detail), an attachment (document) or an additional calculation. The tool is a console application and can be used in batch processing or as an execution step integrated in a module.

This tool allows to duplicate an existing comment entry or additional calculation by changing a part of the CommentKey (comment) / NOTE_ID (additional calculation) field. These fields are a

unique key for each comment / additional calculation.

Note: The CommentKey field is located in the Performance Analytics database repository (COMMENTINFO table) and the NOTE_ID can be found in the Performance database repository (Notes table).

The CopyComments console provides the following command line options:

CopyComments

```
[-h|-?]
-a <ApplicationID>
-t <CellComments|ModuleComments|OffgridComments|DetailComments|AllComments|AddCalc>
-c <current value> -n <new value>
-b <AbortOnDuplicate|SkipDuplicate|OverwriteDuplicate>
-l <logfile|console> -d <dimension> -s "<where-clause>"
-u <Performance Analytics user> -p <Performance Analytics user password> -o <OlapDB> -
[-m] --UpdateAddCalc
```

Note:

- Depending on which object type (-t) is used, some parameters are not needed or optional.
- The UpdateAddCalc option is only valid for AddCalc. If this option is used, the behaviour parameter (-b) has no effect

Parameter	Description
-h	Display help message.
-?	Displays usage.
-a	The application id (needed when copying additional calculations).
-t	The object type determines which comment data records are included by the copy comment process. Possible values are: CellComments all cell comments of type attachment or text ModuleComments all module comments of type attachment or text OffgridComments all offgrid comments of type attachment or text DetailComments all detail comments of type attachment or text AddCalc all additional calculation AllComments all comments of type cell, module, offgrid
-c	The current value which will be replaced by the value from the -n parameter.
-n	The new value which will be used to replace the current value.
-b	The behaviour used when an error occurs due to duplicate entries: AbortOnDuplicate: The whole copy comments process is aborted. SkipDuplicate: The current data record is skipped and the copy comments process continues with the next data record. OverwriteDuplicate: The current data record overwrites the duplicate data record.

	<p>MergeCommentEntries: The source comment entries are merged with the destination comment entries. The date and time that the comment was created determines the merged comment order.</p> <p>The default behaviour is "AbortOnDuplicate"</p>
-l	<p>If an error occurs and any of the following two values are set the faulty data records are written either into a logfile or the console window.</p> <p>Console: All data records containing errors are written to the console window.</p> <p>Logfile: All data records containing errors are written into the log file.</p> <p>The default value is Logfile.</p>
-d	<p>The key of the value, which has to be replaced. This parameter is required when copying cell or detail comments.</p> <p>For cell comments this is the dimension name and for detail comments this is the detail module fieldname defined in the worksheet.</p> <p>Note: This parameter is also required when the parameter -t is set to AllComments.</p>
-s	<p>Optional sql where conditions (without WHERE command), which are applied to the COMMENTINFO table.</p> <p>Example:</p> <p>CommentKey = 'MyValue' OR CommentKey LIKE '%MyOtherValue%'</p> <p>It is also possible to include a sub select into the optional where condition.</p> <p>Example:</p> <p>Id in (SELECT CommentInfoID FROM CommentEntry WHERE Author = 'admin')</p> <p>Note: The value of this parameter is passed to the where condition without any syntax checks. Ensure that the correct syntax for the database is used.</p>
-u	Username for Performance Analytics (needed for copy comments)
-p	Password for Performance Analytics user (needed for copy comments)
-r	<p>The specific Relation-DB (needed when copying detail comments).</p> <p>e. g. DET:myHostName\ABCRepository</p> <p>DET = fixed Prefix</p> <p>myHostName = machine name where Performance Analytics Server is installed</p> <p>ABCRepository = database name of the ABC repository</p>
-o	<p>The specific Olap-DB (needed when copying cell comments).</p> <p>e. g. Essbase:MyServer\MyApp\MyDB</p> <p>Essbase = Data source type</p>

	<p>MyServer = OLAP server</p> <p>MyApp\MyDB = Database name</p>
-m	<p>If the console option migration mode (-m) is used, all Performance comments will be migrated to the comments tables (COMMENTINFO, COMMENTENTRY, COMMENTENTRYATTACHMENT) in the Performance Analytics repository. In this case all other parameters (except -l logfile) will be ignored by the console. You also have to ensure that the mentioned comment tables are empty.</p>
-z	<p>Determines the existing comment behavior. This is an optional parameter and can be used to rename comment keys (move) or to delete comments.</p> <p>None = The comments are copied to the new location.</p> <p>Rename = The comments are not duplicated, the comment key is renamed, this moves the comment to a new location. For example Qtr1 to Qtr2.</p> <p>DeleteOnly = No comments are copied and the found comments are deleted.</p> <p>Default is none, the comments will be copied by default.</p>
-- UpdateAddCalc	<p>The --UpdateAddCalc option is only valid in combination with the object type (-t) AddCalc. When this option is used, all occurrences of the current value (-c) in the NOTE_ID and AddCalc sheet will be replaced with the new value (-n). That means no copy will be created. The replace option will also happen even if the value to replace was not found in the NOTE_ID.</p>

Sample 1:

All cell comments (text and attachment) from the year 2011 (comments with a CommentKey containing the text *Y2011* in the dimension Year) belonging to the application BudgetApp should be duplicated by replacing the text *Y2011* with *Y2012*.

When a duplicate entry is detected this entry will be overwritten.

```
CopyComments      -t CellComments
                  -o Essbase:MyServer\MyApp\MyDB
                  -d Year -c Y2011 -n Y2012
                  -b OverwriteDuplicate -u MyUser -p MyPassword
                  -l MyErrorFile.log
```

Sample 2:

All comment entries (cell comments, module comments, offgrid comments) of the first forecast *Forecast1* are copied to the second forecast *Forecast2*.

Duplicate entries are skipped and the error messages appear in the console window.

```
CopyComments      -t AllComments
                  -o Essbase:MyServer\MyApp\MyDB
                  -d Scenario -c Forecast1 -n Forecast2
                  -b SkipDuplicate -u MyUser -p MyPassword
                  -l Console
```

Sample 3:

All comment entries (cell comments, module comments, offgrid comments) of the first forecast *Forecast1* are copied to the second forecast *Forecast2*, when the CommentKey contains the word 'Sales'.

Duplicate entries are skipped and the error messages appear in the console window.

```
CopyComments      -t AllComments
                  -o Essbase:MyServer\MyApp\MyDB
                  -d Scenario -c Forecast1 -n Forecast2
                  -b SkipDuplicate -u MyUser -p MyPassword -l Console
                  -s "CommentKey LIKE '%Sales%'"
```

Sample 4:

All comments on detail entries from the database *ABCRepository* on host *MyHostname* where the detail module fieldname column *Scenario* has the value *Forecast1* are copied to the second forecast *Forecast2*.

Duplicate entries are skipped and the error messages appear in the console window.

```
CopyComments      -t DetailComments
                  -r DET:MyHostname\ABCRepository
                  -d Scenario -c Forecast1 -n Forecast2
                  -b SkipDuplicate -u MyUser -p MyPassword -l
Console
```

Sample 5:

All comment entries (cell comments, module comments, offgrid comments) are deleted from the repository for member *Forecast1*.

```
CopyComments      -t AllComments
                  -o Essbase:MyServer\MyApp\MyDB
                  -d Scenario -c Forecast1
                  -b SkipDuplicate -u MyUser -p MyPassword -z
DeleteOnly
                  -l Console
```

Sample 6:

All comment entries (cell comments, module comments, offgrid comments) of the first forecast *Forecast1* are moved to the second forecast *Forecast2*.

If comments already exist, the comment histories are merged.

```
CopyComments      -t AllComments
                  -o Essbase:MyServer\MyApp\MyDB
                  -d Scenario -c Forecast1 -n Forecast2
                  -b MergeCommentEntries -u MyUser -p MyPassword -z
Rename
                  -l Console
```

Sample 7:

All additional calculations in the application *MyApplication* will be copied, when the NOTE_ID contains the value *Forecast1*. In the copy the old value will be replaced with the new value *Forecast2*. This replacement operations take place in the NOTE_ID and the references within

the additional calculation sheet.

When a duplicate entry is detected this entry will be overwritten.

CopyComments

```
-a MyApplication -t AddCalc  
-c Forecast1 -n Forecast2  
-b OverwriteDuplicate -l Console
```

Sample 8:

All additional calculations in the application *MyApplication* will be updated with the new value *Forecast2*, where the NOTE_ID or a reference within the additional calculation sheet contains the value *Forecast1*.

CopyComments

```
-a MyApplication -t AddCalc  
-c Forecast1 -n Forecast2  
--UpdateAddCalc -l Console
```


10. Appendix A: Wildcards

By using “wildcards” in various dialogs, you can selectively address parts of unit, modules, etc., that “match” the specified pattern.

Wildcards serve to define pattern which are compared with actual names. Such wildcards are also known as “regular expressions”.

Pattern matching (e.g. VT.*) to find an existing name (such as VT39) is subject to the following rules:

1. Any character other than a "wildcard character" (see below) stands for itself.
2. The backslash (\) character preceding a wildcard overrides the wildcard and stands for the character itself.
3. The following wildcards can be used:

Wildcard	Description
.	Placeholder for any character
[A-Z]	Placeholder for a capital (upper case) letter
[a-z]	Placeholder for a lower case letter
[0-9]	Placeholder for a number
[A-Za-z]	Placeholder for upper and lower case letters
[A-Za-z0-9]	Placeholder for upper case and lower case letters and numbers
*	Repeat character This character is always evaluated in conjunction with the preceding character or wildcard. Stands for any repetition of the preceding character. Null repetitions are also permitted.
+	Repeat character This character is always evaluated in conjunction with the preceding character or wildcard. Stands for any repetition of the preceding character. Null repetitions are not permitted.

Examples:

Pattern	Description
VT.*	This pattern matches any name that begins with "VT"; e.g.: VT12 VTABCDEF VT1 and also: VT
VT..	This pattern matches any name that begins with "VT" and that is exactly 4 characters long; e.g.: VT12 VTAB
VT[0-9].*	This pattern matches any name that begins with "VT" and whose third character is a digit; any other characters may follow. E.g.: VT1ABCDE VT9
VT.*-001	This pattern matches any name that begins with "VT" and that ends with "-001". Any other characters may be included between these two arguments. For example: VT-001 VT100-001
.*-XYZ	This pattern matches any name that end with "-XYZ"; any characters may precede this string. For example: VT-XYZ VT100-XYZ

11. Appendix B: Customising the Look&Feel

The look and feel of Performance is defined within themes (the default is SilverTheme). To create a new theme it is recommended to create copy an existing theme into a new theme folder and the modify its content (see *Creating a new theme directory*).

As there are two different web applications for client and administration, different themes can be created and configured for each application.

Change Active Theme

Switching between themes permanently is usually done by changing the corresponding values in the web.config files. As there are two web applications for administration and client, the theme can be selected per application.

Change the theme for the administration web application:

- Open the web.config of the PMAAdmin web application
- Find all occurrences of the old theme name (e.g. SilverTheme)
- Replace all occurrences by the new theme name (e.g. MyNewTheme)

Change the theme for the client web application:

- Open the web.config of the *PM* web application
- Look for the *appSettings* section
- Look for the *add*-element with the key *Theme*
- Change the value of the element to the name of the new theme

If it's not required to switch the active theme permanently, it can be switched only for the current session by using an URL parameter. This can be required e.g. for testing purposes or special applications. Simply append the URL parameter *Theme* and pass the name of the requested theme as its value.

Client example: <http://localhost/PM/?Theme=MyNewTheme>

Admin example: <http://localhost/PMAAdmin/Login.aspx?Theme=MyNewTheme>

Creating a New Theme Directory

Create a new theme directory for the administration web application:

- Navigate to the PMAAdmin web application directory (e.g. C:\Program Files\cubus\Performance\Web\PMAdmin)
- Open the App_Themes folder
- Create a copy of the folder of an existing theme (e.g. SilverTheme)
- Rename the new folder and set it to the name of your new theme

Create a new theme directory for the client web application:

- Navigate to the PM web application directory (e.g. C:\Program Files\cubus\Performance\Web\PM)
- Open the Content and then the Themes folder
- Create a copy of an existing theme by
 - copying the CSS-file of the theme (e.g. SilverTheme.css)
 - copying the data-folder of the theme (e.g. the SilverTheme folder)
- Rename both, the new CSS file and the new data-folder to the name of your new theme

12. Appendix C: Cockpit Configuration

One of the module types in Performance is the cockpit module. Cockpit allows to combine different components on one screen. Cockpit also allows more interaction between reports and the possibility to display more than one type of report. Please note that Cockpit module is not supported in the Performance Desktop.

This appendix explains the cockpit module in further detail.

Cockpit Overview

Cockpit Content Page

Cockpit modules are displayed using the Cockpit Content page (CockpitContent.aspx). The content page has been designed for the purpose of Cockpit and to give extra functionality to the user.

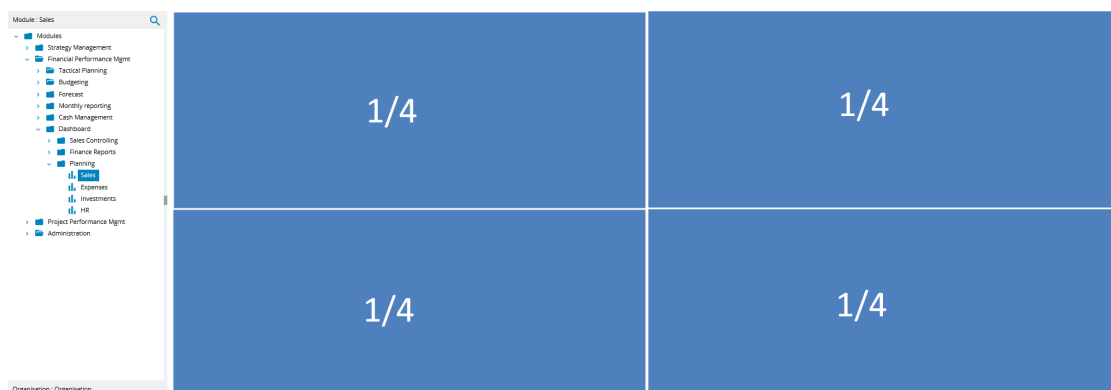
The content page has been split in to four different frames which act as placeholders for the content of Cockpit. Each Cockpit module and each frame within the Cockpit module can be configured to contain different content.

Output Reports

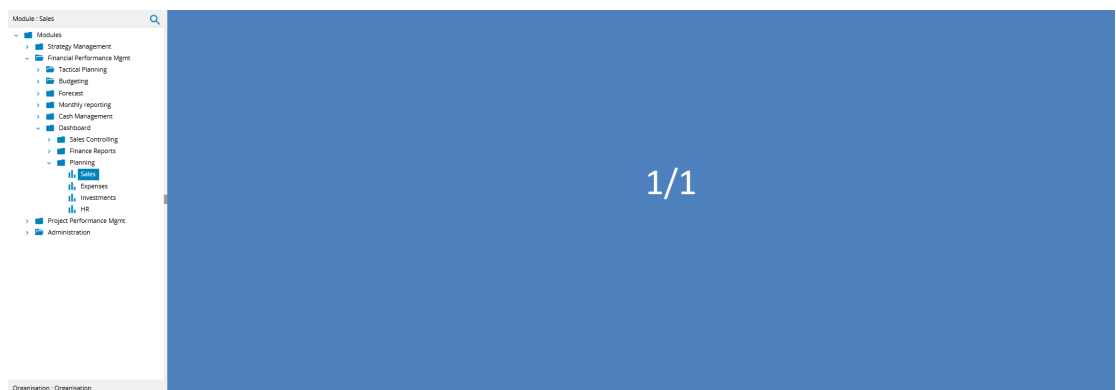
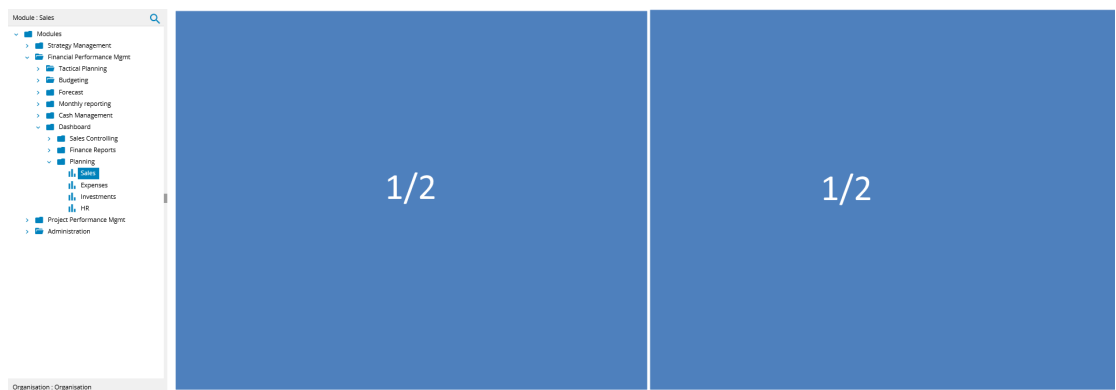
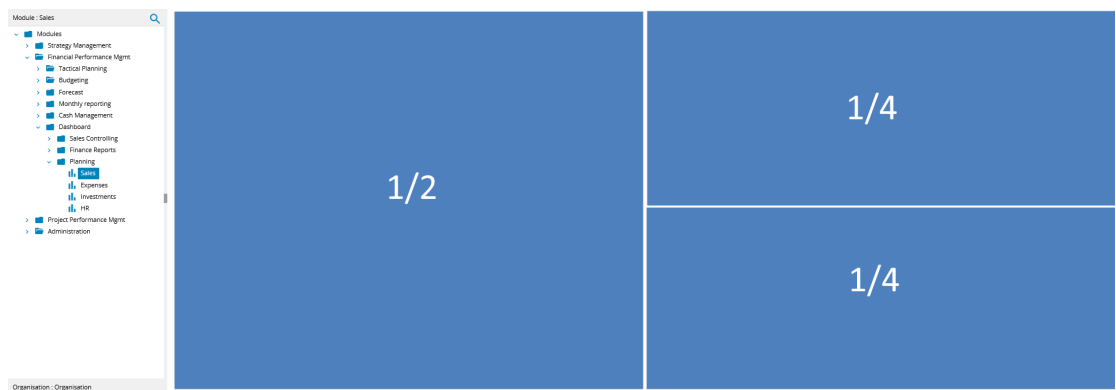
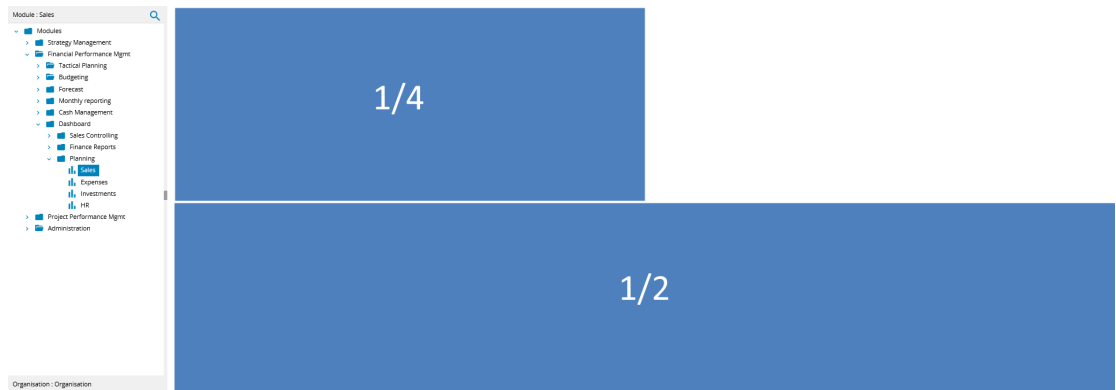
Frame Layout Combinations

Each frame can be configured to be displayed in different positions within the content page. The content page uses a 2x2 matrix to configure the different frame positions.

Below is a screenshot showing how the content frames are defined.



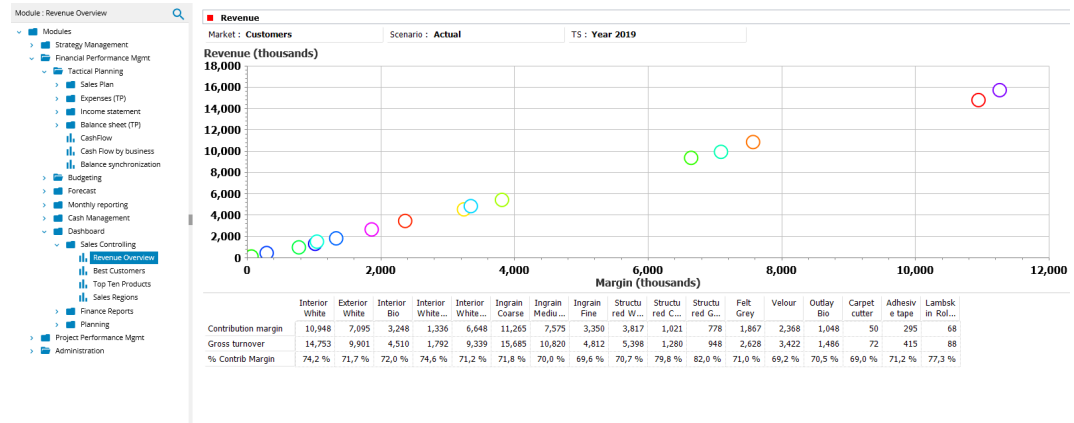
The chosen implementation allows for different layout combinations of different types of reports. The different layout combinations can be seen in the images below.



Please see the module configuration section for further information on configuring content page layouts.

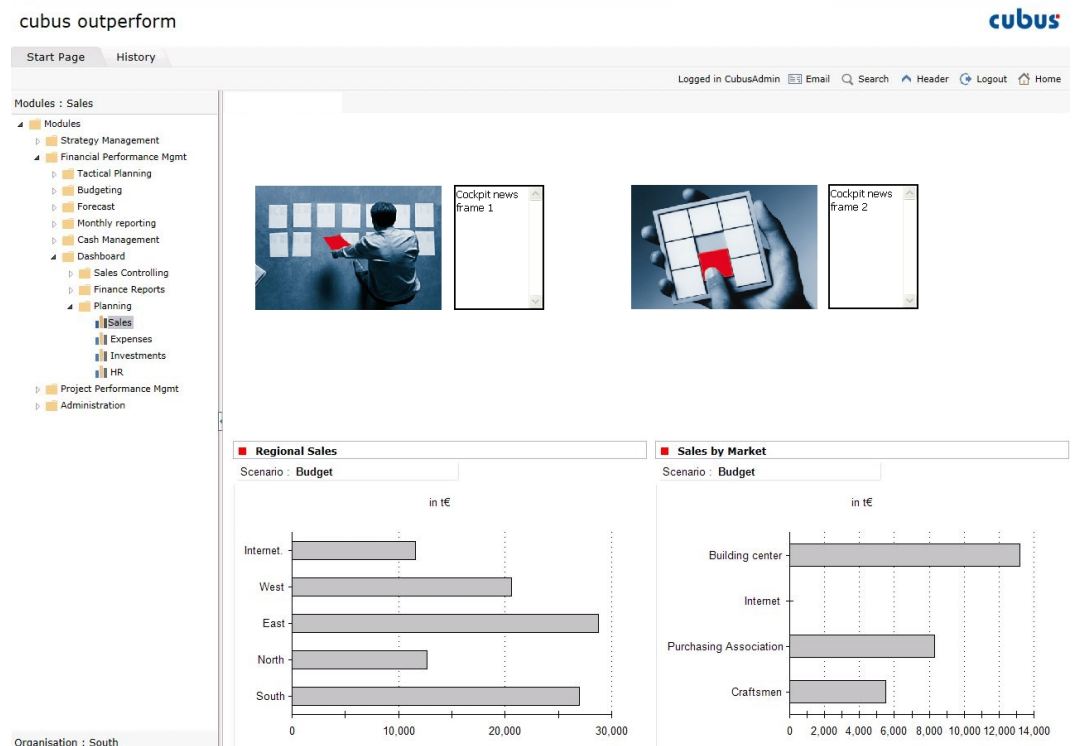
Performance Analytics Frame

The EV frame is used to display different reports from Performance Analytics. Up to four reports can be displayed within an Cockpit module.



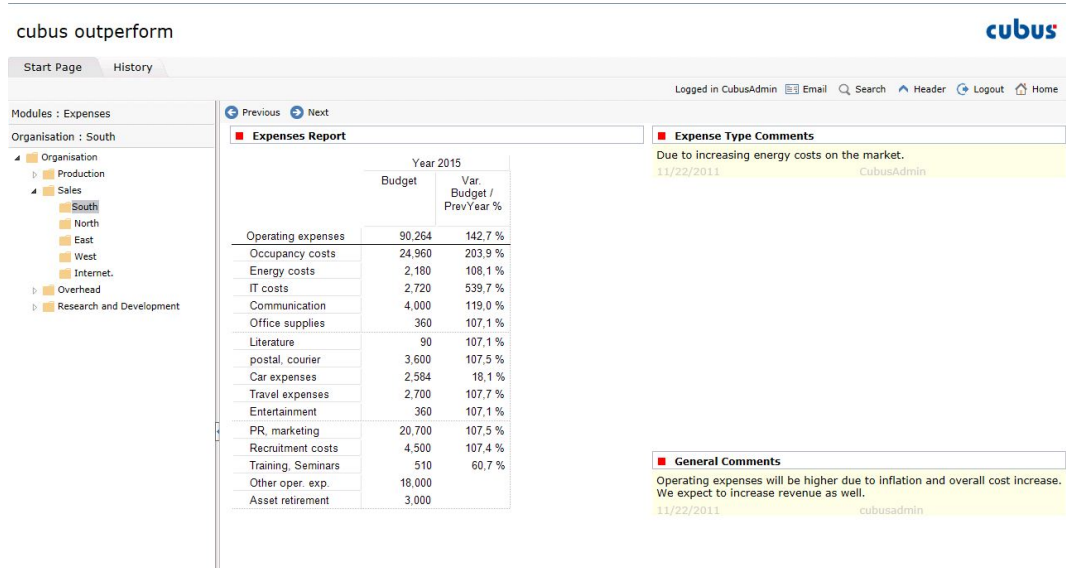
News Frame

The cockpit news frames are shown below. The news frame allows for users to specify news text, an image and a hyperlink to a web page where more information can be displayed.



Comment Frame

Cockpit comment frames are used to display different comments that have been entered in to the system. Different comments can be displayed, such as module comments and cell comments.



The screenshot shows the 'cubus outperform' application interface. On the left is a navigation tree for 'Expenses' under 'Organisation : South'. The main content area displays an 'Expenses Report' table for 'Year 2015' with columns for 'Budget' and 'Var. Budget / PrevYear %'. To the right of the table are two comment frames: 'Expense Type Comments' and 'General Comments', both containing text about increasing energy costs and inflation.

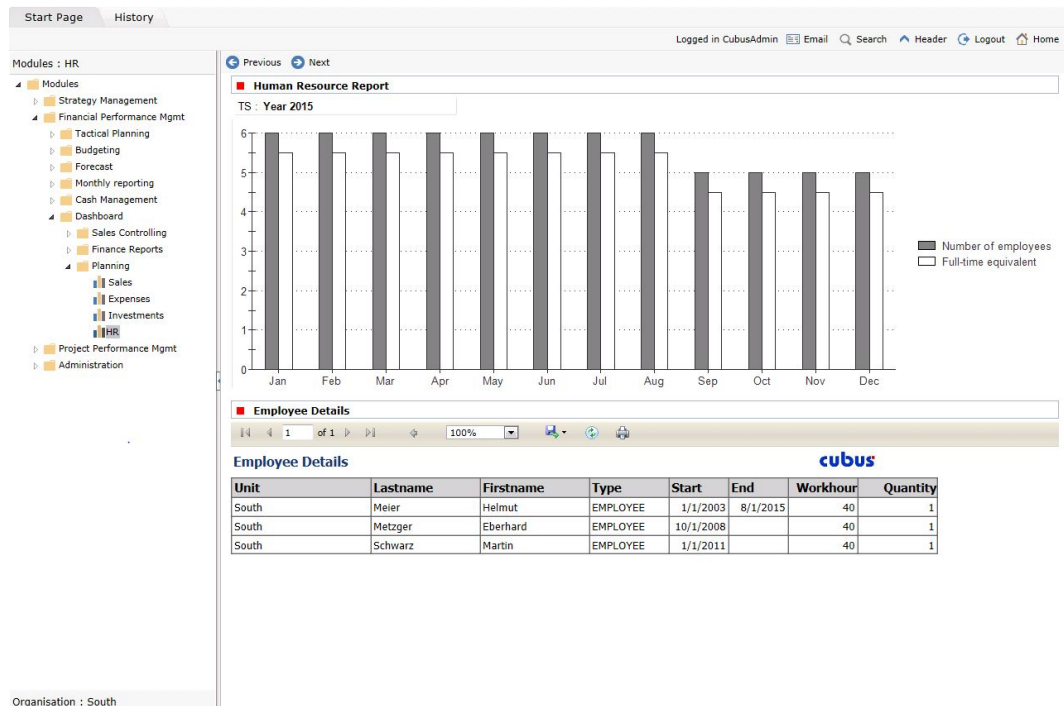
	Budget	Var. Budget / PrevYear %
Operating expenses	90,264	142.7 %
Occupancy costs	24,960	203.9 %
Energy costs	2,180	108.1 %
IT costs	2,720	539.7 %
Communication	4,000	119.0 %
Office supplies	360	107.1 %
Literature	90	107.1 %
postal, courier	3,600	107.5 %
Car expenses	2,584	18.1 %
Travel expenses	2,700	107.7 %
Entertainment	360	107.1 %
PR, marketing	20,700	107.5 %
Recruitment costs	4,500	107.4 %
Training, Seminars	510	60.7 %
Other oper. exp.	18,000	
Asset retirement	3,000	

Expense Type Comments
Due to increasing energy costs on the market.
11/22/2011 CubusAdmin

General Comments
Operating expenses will be higher due to inflation and overall cost increase. We expect to increase revenue as well.
11/22/2011 cubusadmin

MS Reporting Services Frame

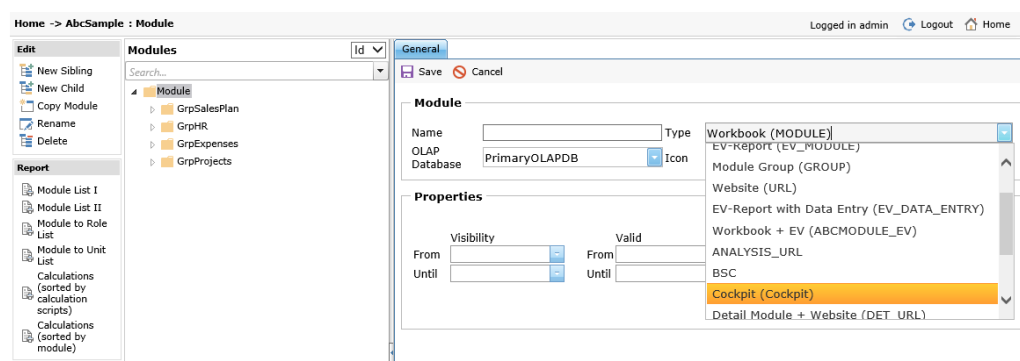
The MS Reporting Services Frame is used to display MS Reports from SQL Server. The reports are interactive and can display different outputs depending on the Performance Analytics report member selection. The reports can also display different information depending on unit selections.



Cockpit Module Configuration

Adding a Cockpit Module

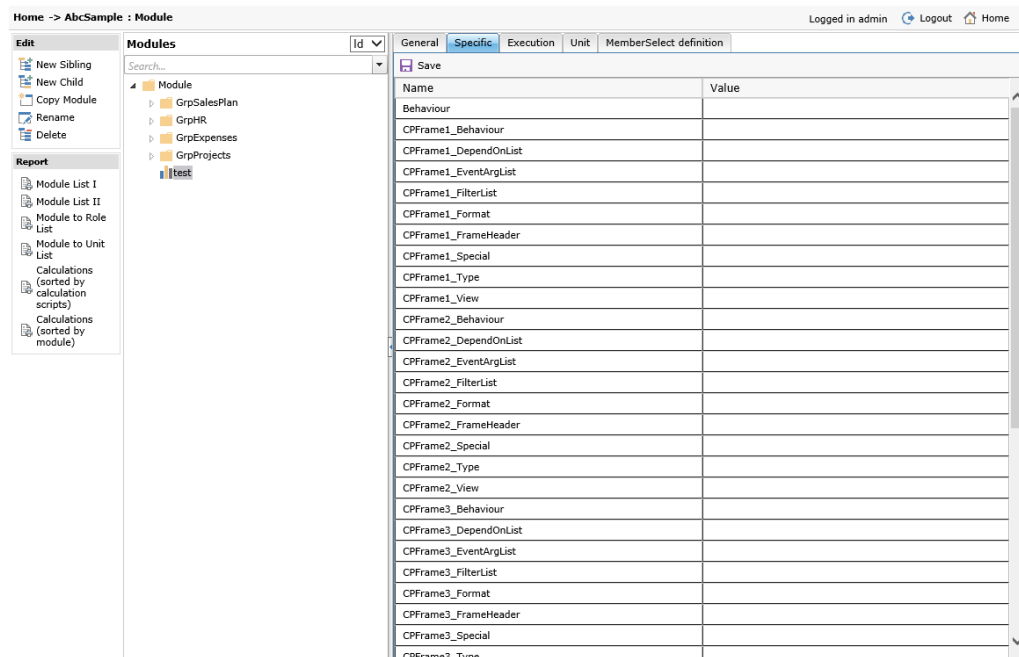
A new cockpit module is added as any other module. In the "Module Type" dropdown list, it is possible to choose the module type "Cockpit".



After the new module has been saved more configuration options are available. Use **CockpitContent.aspx** for the module content page.

Configuring Cockpit Specifics

The content and the behavior of the Cockpit modules can be configured after the creation of the module. This can be done in the specifics tab. The functionality of some parameters differs depending on the content type that needs to be displayed in the cockpit frame.



General Parameters

Layout

The layout parameter defines the number, the size and the position of each single frame.

Q1	Q2
Q3	Q4

CockpitContent.aspx uses these names to define the layout of the module.

In order to use all four frames, the layout parameter value must look like the following: Q1, Q2, Q3, Q4.

Should one frame not be displayed, its name just has to be left out (e.g. Q1, Q2, Q3).

The standard size of the frame is $\frac{1}{4}$ of the displayable content page.

The frame (quarters) have to be combined to modify the size of an frame. This is done by using the ampersand (and sign).

To display two horizontally aligned rectangles, Q1 needs to be combined with Q2 as well as Q3 with Q4. So the value for the layout parameter would look like the following: Q1&Q2, Q3&Q4. Therefore it is possible to configure many different user outputs.

Note that the number of the identification has nothing to do with the CPFrame number from the "module specifics". This is explained later in this documentation.

The table below shows example configurations.

Layout	Configuration				
<table border="1" style="width: 100%; height: 100%;"> <tr> <td style="width: 50%; text-align: center;">Q1</td> <td style="width: 50%; text-align: center;">Q2</td> </tr> <tr> <td style="text-align: center;">Q3</td> <td style="text-align: center;">Q4</td> </tr> </table>	Q1	Q2	Q3	Q4	<p>All four frames are rendered.</p> <p>Q1,Q2,Q3,Q4</p>
Q1	Q2				
Q3	Q4				
<table border="1" style="width: 100%; height: 100%;"> <tr> <td style="width: 50%; text-align: center;">Q1</td> <td style="width: 50%; text-align: center;">Q2</td> </tr> <tr> <td style="text-align: center;">Q3</td> <td></td> </tr> </table>	Q1	Q2	Q3		<p>Only three frames are displayed. Q4 is not rendered.</p> <p>Q1,Q2,Q3</p>
Q1	Q2				
Q3					
<table border="1" style="width: 100%; height: 100%;"> <tr> <td style="width: 100%; text-align: center;">Q1</td> </tr> <tr> <td style="text-align: center;">Q3</td> </tr> </table>	Q1	Q3	<p>Two frames are displayed.</p> <p>Q1&Q2,Q3&Q4</p>		
Q1					
Q3					
<table border="1" style="width: 100%; height: 100%;"> <tr> <td style="width: 100%; text-align: center;">Q1</td> </tr> </table>	Q1	<p>One frame is displayed</p> <p>Q1&Q2&Q3&Q4</p>			
Q1					

Format

Not yet used.

Specific

Not yet used.

Unit

The name of the dimension from the Essbase outline that is flagged as the "UNIT" dimension (e.g. contains the UDA ABC_UNIT).

Behavior

Not yet used.

Special

Not yet used.

Frame Specific Parameters

A description for each frame specific is not required as the specifics for frame 1 are the same for frame 2, 3 and 4.

CPFrameX_Type

The web page that is to be loaded in the frame X is declared in the “Type“ parameter. This web page must be in the “CPFrame” directory, in the Zones folder of the web directory.

CPFrameX_View

The view parameter is used in the frame depending on the content to be displayed.

Type	Value
EVFrame.aspx	The value of the view parameter defines the Performance Analytics report that needs to be loaded together with the directory path. e.g.: /ABCDemo/Cockpit/P&L table
CommentFrame.aspx	Not used
MSReportingServicesFrame.aspx	The value of the view parameter defines the Reporting Services report that needs to be loaded together with the directory path. e.g.: /ABCDemo/Cockpit/Report

CPFrameX_FrameHeader

The FrameHeader includes the title of the frame X. It’s possible to define titles for different languages. It’s necessary to include the title in brackets to specify the language version.

```
(de)Überschrift(/de)(en)Header(/en)
```

If the parameter is empty, no title is displayed.

CPFrameX_DependOnList

This parameter is the same for each type and determines from which frame information can be received.

Value	Description
Content	The unique name of the content page. This value gets information from the ABC navigation.
CPFrame1	Unique name of the first frame.
CPFrame2	Unique name of the second frame.
CPFrame3	Unique name of the third frame.
CPFrame4	Unique name of the forth frame.

CPFrameX_EventArgList

The EventArgList parameter is valid for all frames and defines a list of events which can trigger this frame.

The single event names must appear between quote marks and separated by commas.

The available events are:

Name	Description
LoadAbc	Triggered when the module is loaded. ABCPOVState: Contains all ABC navigation information for this module.
All	The frame reacts on all defined events. ABCPOVState: Depends on the event that was triggered.
ActionOnLoad	Triggered when an frame is loaded. Used to synchronize frames. ABCPOVState: Passes all information about the dimensions and their members if the Performance Analytics triggers this event.
ActionOnCellClick	Performance Analytics event that is triggered when the user clicks in a cell in the report. ABCPOVState: Passes the dimension names and respectively the member names which identify the cell.

ActionMemberClick	<p>Performance Analytics event that is triggered when the user clicks on a member. Whether the member is in the Inspread, Offspread or Chart is irrelevant.</p> <p>ABCPOVState: Passes the member name which was clicked, respectively the dimension name.</p>
OffspreadLayoutChanged	<p>Performance Analytics event triggered when an Offspread dimension changes its "location" property. Changes may occur if the user pulls the dimension into the Inspread or drags it from the Inspread into the Offspread.</p> <p>ABCPOVState: Passes all information about the dimensions and respectively their members.</p>
RowLayoutChanged	<p>Performance Analytics event triggered when an inspread dimension changes its "location" property in a row. This occurs when the dimension is moved into another column within the same row or into the Offspread. Also, moving a dimension from a column or from the Offspread triggers this event.</p> <p>ABCPOVState: Passes all information about the dimensions and respectively their members.</p>
ColumnLayoutChanged	<p>Performance Analytics event triggered when an Inspread dimension changes its "location" property in a column. This occurs when the dimension is moved into another row within the same column or into the Offspread. Also, moving a dimension from a row or from the Offspread triggers this event.</p> <p>ABCPOVState: Passes all information about the dimensions and respectively their members.</p>
ActionShowSelect-MembersDialog	<p>Performance Analytics event triggered when a member is clicked in the report and the member selection dialog is shown. Whether the member is in the Offspread, Inspread or Chart is irrelevant. This event is used to disable the member selection dialog.</p> <p>ABCPOVState: Passes no information.</p>

<p>InspreadSelectionChanged</p>	<p>Performance Analytics event triggered when the member selection of an Inspread dimension is changed. This is also the case when an Offspread dimension changes its "location" property.</p> <p>ABCPOVState:</p> <p>Passes all information about the dimensions and respectively their members.</p>
<p>OffspreadSelectionChanged</p>	<p>Performance Analytics event triggered when the member selection of an Offspread dimension is changed. This is also the case when an Inspread dimension changes its "location" property to Offspread.</p> <p>ABCPOVState:</p> <p>Passes all information about the dimensions and respectively their members.</p>
<p>PostEvLoad</p>	<p>This event is only compatible when using the EVFrame, however it must not be specified in the frame which contains the EVFrame. This event is triggered after the Performance Analytics content has been loaded.</p> <p>ABCPOVState:</p> <p>Passes all information about the dimensions and respectively their members.</p>

CPFrameX_FilterList

The "FilterList" contains all the dimensions that can upon a valid event be updated. The dimension names must appear between quote marks and be separated by commas.

CPFrameX_Behaviour

The “Behavior” parameter is used depending on the content to be displayed in the frame.

Type	Description
EVFrame.aspx	The value of the “Behaviour” parameter contains the dimension names for which no member selection dialog should be shown. The dimension names are enclosed with quote marks and separated by commas.
CommentFrame.aspx NewsFrame.aspx MSReportingServicesFrame.aspx	Not used.

CPFrameX_Format

The “Format” parameter is used depending on the content to be displayed in the frame.

Performance Analytics:

Determines which UI elements to display in the Performance Analytics report. These parameters conform to the Performance Analytics documentation. If an element is hidden, it cannot be redisplayed later by using a menu option.

Option	Meaning
UIAuthorisationAll	All elements are displayed.
UIAuthorisationOffspreadBar	Offspread bar
UIAuthorisationTable	Table
UIAuthorisationRowHeaders	Row headers
UIAuthorisationColumnHeader	Column headers
UIAuthorisationTableData	Table data
UIAuthorisationTableSplitter	Table splitter
UIAuthorisationDataEntryOptions	Data entry Option
UIAuthorisationChart	Chart
UIAuthorisationChartBar	Chart bar
UIAuthorisationChartSplitter	Chart splitter
UIAuthorisationChartData	Chart data

UIAuthorisationDrillTrough	Drill Trough
UIAuthorisationDrillThroughColumns	Drill Trough columns
UIAuthorisationDrillThroughScripts	Drill Trough scripts
UIAuthorisationDrillThroughBar	Drill Trough bar
UIAuthorisationComments	Comments
UIAuthorisationTooltips	Tooltips
UIAuthorisationTooltipUDAs	Tooltips UDA's
UIAuthorisationTooltipProperties	Tooltips properties
UIAuthorisationTooltipAttributes	Tooltips attributes
UIAuthorisationTooltipFormulas	Tooltips formulas
UIAuthorisationTooltipTrafficLights	Tooltips traffic lights
UIAuthorisationTooltipMemberName	Tooltips member name
UIAuthorisationToolbar	Toolbar
UIAuthorisationToolbarText	Toolbar text
UIAuthorisationHelpButton	"Help" button
UIAuthorisationViewsButton	"Views" button
UIAuthorisationPreviewBar	Print preview bar
UIAuthorisationPreviewBarText	Print preview bar text
UIAuthorisationTabBar	Tab bar
UIAuthorisationWarnings	Warnings
UIAuthorisationGetData	"Get Data" button
UIAuthorisationBusyIndicator	Activity status
UIAuthorisationLocalViews	Local views

The elements are structured in a hierarchy and the display options of parent elements are passed to its children. Therefore, the order in which elements are hidden is relevant.

For instance, if the table is deactivated, all the elements beneath are hidden. If one of these elements needs to be shown for the user, then it must be explicitly set to active.

The syntax of this parameter comprises the name, a Boolean value which activates or deactivates the element. The name and the value are separated by a colon. The table below

shows how this can be done.

UIAuthorisationOffspreadBar:False	Deactivate the Offspread bar
UIAuthorisationTable:False	Deactivate the entire table
UIAuthorisationColumnHeader:True	Reactivate the column headers of a deactivated table.

The commands must be between quote marks and separated by commas. The implementation of the elements occurs according to the order they were entered.

13. Appendix D: Migrating custom HTML page containing Performance Analytics Object to Performance Desktop

Migration is required in order to use the custom HTML pages containing EV Object in the Performance Desktop. There are two things that need to be made for the migration.

1. The object tag `<object name="EVOBJECT">` cannot be interpreted in the Performance Desktop and needs to be replaced by `<div id="<OBJECT_ID>">`. The new placeholder element is a regular HTML element and has all the properties of an HTML element.
2. The following lines of javascript have to be added at the end of the page's `<body>` tag.

```
<script>
    var EVOBJECT = parent.PMDesktop.InitializeEVById("<OBJECT_ID>");
    EVOBJECT.Server = "<YOUR_AL_SERVER_NAME>"
    EVOBJECT.HttpProtocol = "<http/https>"
</script>
```

Where `<OBJECT_ID>` is the id of our placeholder element. `<YOUR_AL_SERVER_LOCATION>` is the name of your AL Server, for example, "localhost". And the usual `HttpProtocol` is "http".

Code Sample of the new custom HTML page:

```
<html>
  <head>
    <title>EV Custom Implementation</title>
  </head>
  <script>
    function OpenView() {
      EVOBJECT.Views.Load( "/EVView" );
    }

    function ServerCredentials() {
      return new Array("admin","password");
    }

    function DataSourceCredentials() {
      return new Array("admin","password");
    }
  </script>
  <body>
    <div style="margin: 50px;">
      <input type="button" value="Open View" onclick="OpenView();" />
    </div>
    <div id="EV" style="width: 800px; height:600px;"></div>
    <script>
      var EVOBJECT = parent.PMDesktop.InitializeEVById("EV");
      EVOBJECT.Server = "localhost";
      EVOBJECT.HttpProtocol = "http";

      EVOBJECT.attachEvent("NeedDataSourceCredentials", DataSourceCredentials);
      EVOBJECT.attachEvent("NeedServerCredentials", ServerCredentials);
    </script>
  </body>
</html>
```