myObjectiveOLAP Version 2.9.8

Table of contents

Introduction

Introduction

This document is your primary help and technical reference for use of the myObjectiveOLAP for Microsoft Excel application.

This Preface contains these topics:

Audience
Documentation Accessibility Related Documents Passwords in Code Examples Conventions

Audience

This help file and the document "myObjectiveOLAP Provider for Microsoft Excel Technical Reference Guide" are intended for programmers who are developing applications to access an Oracle OLAP database using the myObjectiveOLAP provider. This documentation is also valuable to systems analysts, project managers, and others interested in the development of database applications.

To use this document, you must be familiar with Microsoft Visual Basic for Applications (VBA) or a comparable object orientated language.
Users should also be familiar with the use of the Oracle OLAP Data Manipulation Language (DML) to access information in an OLAP database.

Documentation Accessibility

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line. However, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that SDMC does not own or control. SDMC neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Related Documents

For more information, see these Oracle resources:

Passwords in Code Examples

For simplicity in demonstrating this product, code examples do not perform the password management techniques that a deployed system normally uses in a production environment.

Introducing myObjectiveOLAP

myObjectiveOLAP Client Overview

myObjectiveOLAP is a data provider for the Oracle OLAP database, using and inheriting interfaces from the Oracle ODP .Net framework.

The myObjectiveOLAP framework allows native providers to expose Oracle OLAP specific features and data types. The myObjectiveOLAP framework provides an automation layer, with high performance and robust data type control, between the Microsoft Windows client application and the Oracle OLAP database.

The myObjectiveOLAP provider for MS Excel uses Oracle's native APIs to offer fast and reliable access to the Oracle OLAP engine within the Oracle database. It exposes many of the Oracle OLAP data manipulation language commands and functions to the client.

The myObjectiveOLAP framework offers additional APIs and graphical interfaces for working with both data and structures of the Escendo (OFA / OSA replacement) product.

.

Conventions

Conventions

The following text conventions are used in this document:

Convention	Meaning
Boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
mono-space	Mono-space type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.
[VALUE]	Square bracket encapsulation equates to a user supplied value.

myObjectiveOLAPXL.dll Exposed functions

myObjectiveOLAP Files and Components

myObjectiveOLAPXL.dll

myObjectiveOLAPXL.dll is the core library that is used by any of the myObjectiveOLAP applications. myObjectiveOLAPXL.dll is not optional and must be installed in order to connect Microsoft Excel to the Oracle OLAP database using the myObjectiveOLAP data provider. myObjectiveOLAPXL.dll contains pre-defined functions which are exposed to Microsoft Excel.

myObjectiveOLAPXL.dll Exposed functions

These functions are grouped into four sections:

Common Functions

Common functions enable the end user to interact either with the myObjectiveOLAP library itself or to execute commands or retrieve output from the Oracle OLAP server application.

The functions include a number of low level API's that do minimal checking before attempting to execute within the server side environment. It is best practice to only use these APIs if myObjectiveOLAP does not offer a function to do this for you. By using the myObjectiveOLAP functions in your code, additional pre-execution checks are performed and enhanced error trapping is available to you.

GUI Functions

A number of functions offered by the myObjectiveOLAP framework can provide the end-user with a graphical interface into the Oracle OLAP option.

Common Options

Common options enable the end-user to control Oracle OLAP server side options.

Reporting Functions

These are Excel worksheet cell based functions which can be used by end users to develop rich reporting solutions

Accessing Exposed Functions

Accessing Exposed Functions

Getting an object reference to myObjectiveOLAP from VBA

In all of the examples shown, you will see a preceding "o." in front of the myObjectiveOLAP function, this is the object reference to the myObjectiveOLAP library.

You must also generate an object reference either by using this example or creating your own.

To instantiate the myObjectiveOLAP from Microsoft Visual Basic for Applications or Microsoft Visual Basic you must bind myObjectiveOLAP to an object that you can then reference.

In all of the examples shown we do this check by calling the regQ function which is shown below.

The regQ function binds the myObjectiveOLAPXL.AddinModule to the Global object "o". Once "o" has been bound you can use it to reference the myObjectiveOLAP functions i.e. o.connect o.mooAttached etc.

```
Global o As Object
Global oregistered As Boolean
Public Function regQ() As Boolean

If oregistered = False Then
Set o = Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object
oregistered = True
```

```
Else
   regQ = True
End If
End Function
```

Accessing functions within the myObjectiveOLAP library directly

Instead of creating an object reference as above you can access functions directly by fully qualifying the function as below:

```
Dim ret_ as string

ret_ =
Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object.mooExecuteO
nly("LMT TIME to NULL")
```

This method can be safer, especially in Excel versions lower than 2007 where the object reference can be lost.

Legal

myObjectiveOLAP Release 2.9.8.32

PrimaryAuthor: Robert Taylor

If you do not agree to the terms of this EULA, do not install or use the SOFTWARE PRODUCT.

What's new in V 2.9.8

myObjectiveOLAP (2.9.8)   Release Note (2014-01-10)

2.9.8 Adds new features, improves performance and resolves some issues.

This release is a significant release which requires an upgrade of your myObjectiveOLAP Server installation if implemented.

Notable BUGS & Requests included in this release:

Some of the features listed below were included in 2.9.7, but as that release was not externally released to the customer base they are being summarised within the 2.9.8 schedule.

2.9.8 - Major   - Re-implementation of the Process Manager to enable parallel processing and Read-only, Read-Write and multi Sub AW processing.

2.9.8 - Major   - Implementation of a capstone/sub AW model to allow data and or processes to be partitioned within a single myObjectiveOLAP Server installation.

2.9.8 - Major   - Significant changes to the Relational Explorer graphical SQL query builder with Excel integration.

2.9.8 - of note   - Improvements to the Data Explorer, multi-dimensional graphical reporting suite for OLAP analysis on Oracle OLAP Analytic Workspaces.

2.9.8 - of note   - Implementation of Scheduled Reports enabling end-users to define a report through Data Explorer for latest execution and deliver-by-email distribution.

2.9.8 - of note   - Improvements to Session Manager and enhanced kill session engine support RAC installations,

2.9.8 - of note   - Brand new code Editor for editing Oracle OLAP DML,

193 Individual bugs or enhancement requests have been closed in 2.9.8 For specific bug inquiry status please raise a ticket or review the bug status within the support.myobjectiveolap.com support system.

Clients running myObjectiveOLAP Server should also apply the 2.9.8 server patch.

2.9.8   - Server patch is fully backwards compatible with earlier releases and is designed to take advantage of the improvements in the 2.9.8 client.

myObjectiveOLAP (2.9.7) Release Note

2.9.7 Was a closed release cycle.

myObjectiveOLAP (2.9.6) Release Note (2013-07-14)

2.9.6 Adds new features, improves performance and resolves some issues.

This release is recommended to be applied to all existing clients. This release is fully backward compatible with all existing releases.

Notable BUGS & Requests included in this release:

2.9.6 - of note - New myObjectiveOLAP Server Data Explorer graphical tool added, please see the Data Explorer topic.

2.9.6 - of note - CST-REQ#2013020910000097 - If a mooCellQDR formula is refreshed in Excel when no connection to the Oracle OLAP database has been made then the value "999999" is returned instead of "0". mooCellQDRT (text retrieve) now reports: "You are not connected to Oracle"

2.9.6 - of note - CST-REQ#2013041210000100. Changes to the session manager to provide more information on the users activities, including the current Oracle OLAP DML execution. There is also a significant change in the way in which Oracle sessions are killed.

2.9.6 - of note - New mooFRM API which replaces the mooFR API. Improvements:

- performance improvements in data retrieval and the network transfer protocol.
- dimension value and measure descriptions can be retrieved in a single database pass.
- multi dimension down and across supported (up to 5 dimensions on each axis).
- better error reporting to the client calling.

2.9.6 - minor - BUG#1107 MetaData and OLAP DML code backup, now includes backup of the magazines recording backups and OLAP checkout, commit information

2.9.6 - minor - CST-REQ#2013053110000108 Add a new process management web viewer.

2.9.6 - minor - BUG#1100 Process Manager no longer displays "toolstriplabel1" if a non-admin profile enters the Process Manager viewer and does not press Refresh.

2.9.6 - minor - Other minor fixes.

Clients running myObjectiveOLAP Server should also apply the 2.9.6 server patch.

2.9.6 - Server patch is fully backwards compatible with earlier releases and is designed to take advantage of the improvements in the 2.9.6 client.
In addition there is a new mooWebServices daemon which adds a web based Process Management viewer.
Failure to apply the 2.9.6 server patch will not cause client issues with earlier clients, however, improvements in performance will not be available if the client utilises the mooFRM API.

Components of patch:

- 2013-07-14-myobjectiveolapserver-296-Build20130701-moocode.eif
- 2013-07-14-myobjectiveolapserver-296-Build20130701-pack.sql

The 2.9.6 Server patch is available from the support portal.

myObjectiveOLAP (2.9.5) Release Note (2013-05-13)

2.9.5.1 was a technical release to add functionality to aide a specific enterprise customer.

2.9.5.1 - Explanation, 2.9.5.1 was released to add supporting functionality to a aide an infrastructure requirements of an enterprise client. 2.9.5.1 included ported functionality from the 2.9.6 branch notably a V 1.0 release of Data Explorer.

2.9.5.1 Is not available from the support portal, all functionality is included within the 2.9.6 release.

myObjectiveOLAP (2.9.4.1) Release Note (2013-02-10)

2.9.4.1 is a technical update to 2.9.4

This release is recommended to be applied to all existing clients. This release is fully backward compatible with all existing releases.

2.9.4.1 - of note - REQ#10102 Change mooCellQDR & mooCellQDRT to use bind variables instead of submitting the query as a text string. Reduces the stress on the shared_pool which in implementations with large numbers of clients can cause potential latch locks. This version increases client QDR performance by 4 - 6 times in environments which can benefit from the update.

myObjectiveOLAP (2.9.4) Release Note (2013-01-27)

2.9.4 Adds new features and resolves some issues.

This release is recommended to be applied to all existing clients. This release is fully backward compatible with all existing releases.

2.9.4 - of note - CST-REQ#013011810000011-BZ:101 Add ability to have a single mooApplicationSettings.xml file loaded from the install directory.
2.9.4 - DocOnly - CST-REQ#2013012710000011 Document how to load the myObjectiveOLAPXL.dll in a user account which did not install the product on the same machine as myObjectiveOLAP was originally installed.

myObjectiveOLAP (2.9.3.8) Release Note (2013-01-13)

2.9.3.8 is a minor update to 2.9.3

This release primarily is a non-technical update and adds requested certification to certain Windows 8 and Office combinations.

Please read the 2.9.3 updates for a list of all changes in 2.9.3

2.9.3.8 - of note - CST-REQ#100033 Certify Windows 8 (32 & 64 bit), Office 2013 (32-bit).

2.9.3.8 - minor - Fix to Export to CSV in Relational Explorer, whilst data was extracted correctly a controlled error was generated on completion.

2.9.3.8 - minor - REQ#100048 Add a control to Session Manager to limit the details displayed. If no detail selected then only a single record per session is displayed. Useful for those customers with many concurrent sessions.

myObjectiveOLAP (2.9.3) Release Note (2012-12-31)

2.9.3 This is a large update, please read the Release Notes carefully.

This release is strongly recommended to be applied to all existing clients. This release is fully backward compatible with all existing releases.

Notable changes in this version

2.9.3 - minor - Numerous improvements and changes

2.9.3 - minor - Workflow updates

2.9.3 - minor - Console warning you should clear your history if gt 20k lines performance impact

2.9.3 - minor - Fixed a bug in how OLAP handles ASCII 32 --> 127 chars.

2.9.2 - minor - Workflow updates

2.9.2 - minor - AWM Compatibility improvements

2.9.2 - minor - Multi-threading model changes and fixes

2.9.2 - minor - REQ#100052 Change the way Session Manager disposes of sessions
REQ#100053 Hide kill button in Session Manager if logged in as MOOUSER but allow them to open the Session Manager
Small improvements to MOO.EXTERNAL.CALL API
UI Improvements and changes
Backward compatibility changes to mooApplicationSettings.xml

2.9.1 - minor - Relational Explorer added

2.9.1 - minor - Updates to Workflow

2.9.1 - minor - Security restrictions hardened on MOOUSER

Notable BUGS & Requests included in previous updates

2.8.1 - minor - Workflow added, number of small improvements and fixes.

2.7.1 - minor - Rollup Patch, UI Improvements, Dimensional & Cube Explorer exit BETA, updates to mooFR

2.6.4 - minor - BUG#1042 -- 1056 Updates to Dimensional & Cube Explorer

2.6.3 - minor - Added R1 Cube Explorer (BETA)
Updates to Dimensional Explorer

2.6.2 - minor - Updates to Dimensional Explorer

2.6.1 - minor - Numerous including but not limited to:
Updated Ribbon graphics
Updates to Process Builder
Dimensional maintenance (BETA)
Added MOO FONT_SIZE
Small fixes to Editor windows and Recall form

2.5.3 - minor - REQ#1037 Copy Process window added.

2.5.2 - minor - REQ#1036 mooGetDimList ([Dimension Name], True/False. The mooGetDimList function returns a one dimensional array containing dimension values from a dimension within Oracle OLAP

2.5.1 - major - BUG#1009 Fixes to multi-database connection threading.

2.4.2 - major - BUG#1010 Fixes to frmRecall to stop it crashing if it has not got anything to recall

2.4.1 - minor - REQ#1007 Added showSaveScript interface

2.4.1 - minor - BUG#1009 Enable resize of edit wind

2.3.4 - minor - REQ#1006 Update to mooServerLogin

2.3.4 - minor - BUG#1008 Enable resize of edit window

2.3.2 - minor - BUG#1007: Changed the description placed in the dba_scheduler for the moo pm

2.3.2 - minor - BUG#1006: Refresh Data

2.3.1 - minor - Process Manager standard reports. and updates to the help menus

2.3.1 - minor - REQ#004: update to new help system

2.2.1 - major - Underlying ADX shim loader has been migrated to ADX2010. Please ensure you delete your previous myObjectiveOLAP install completely.

2.2.1 - minor - REQ#37: Allow the running of the Process Manager for "Just the next task"

2.2.1 - minor - REQ#35: New menu item under MooServer --> "Backup mooServer Code AWs" which backs up all meta-data AWs to a valid CDA

2.2.1 - minor - REQ#30: When saving a connection file, the user is now asked if they wish to save a default file 'serverDefault.xml' if they say Yes, then the contents of this file are loaded every time the Connection window is loaded.

2.2.1 - minor - REQ#45: A 64bit ADX Loader shim will be distributed with future releases. This will enable customers using Office 2010 64bit to use the myObjectiveOLAP add-in. This is still classed as **BETA** functionality and the support matrix remains unchanged (below) .

2.2.1 - minor - BUG#1: RESOLVED PR.CFG(PR.COL 'DESC') can not be NA

2.2.1 - minor - BUG#6: New menu item under MooServer --> Moo Health Check which checks the status of mooServer meta-data by calling moo.meta.check

2.2.1 - minor - BUG#7: Fix to Process Submission window to ensure that databases are attached correctly

2.2.1- minor - BUG#12: OLAP DML can start with MOO but not MOO{SPACE} now

2.2.1- minor - BUG#16: Add a Change Password window to the mooServer login window.

2.2.1 - minor - BUG#19: Program editor changed to not leave mooprgtexttemp text variables on the server.

2.2.1 - minor - BUG#20: Allow the ability to create Admin user accounts through the User Manager.

2.2.1 - minor - BUG#25: Allow the ability to remove the standard connection editor from the myObjectiveOLAP menu, set: ALLOW_DB_CONNECT FALSE in mooApplicationSettings.xml

This is documented in 148-mooApplicationSettingsINTERNAL.rtfd

2.2.1 - minor - BUG#26: Program editor no longer leaves blank lines at the bottom of an edited program

2.2.1- minor - BUG#28: Un-used toolstripMenu removed from prSubmitFRM

2.2.1 - minor - BUG#29: Do not accidentally reset the users password when changing other details.

2.2.1 - minor - BUG#41: Update process screen after deleting queued process

2.2.1 - minor - BUG#43: Icon change

2.2.1 - minor - BUG#44: Add the ability to run ALTER SYSTEM DISCONNECT instead of ALTER SYSTEM KILL SESSION in session manager

1.4.7 - major - A number of core components are now multi-threaded enabling faster execution for a number of actions.

1.4.7 - major - myObjectiveOLAP is now supported in multiple Excel sessions (processes) on the same client PC running Excel 2010 without conflict or locks between sessions.

1.4.7 - note worthy - Read and execute an OLAP script file.

A new menu item has been added to the Advanced Menu Group.

Create a text file in your favorite text editor containing one or more OLAP DML statements.

Save your file with a .moo extension.

Select your file from the dialog box enabled through the Read OLAP Script file menu item.

myObjectiveOLAP will execute your OLAP DML and any output from the Oracle OLAP engine will be printed (off) to a file of the same name and client directory location as the original script file but with a .out extension

1.4.7 - note worthy - Addition of XML mooApplicationsSettings.xml file. This enables an administrator to disable advanced menu items distributed to the client PC estate.

Please see: http://myobjectiveolap.com/documents/mooApplicationSettings.rtfd for further information

Please see: http://myobjectiveolap.com/documents/mooApplicationSettings.xml for an example XML file which enables all functionality

Please note the above XML file enables all settings with the exception of Escendo functionality. To enable Escendo set the relevant key to true.

1.4.7 - minor - Improvements in the internal mechanism which binds the ribbon menu

1.4.7 - minor - Updates to look and feel of all GUI screens

1.4.7 - minor - Adjustments to the OLAP command editor to make full screen mode more comfortable

1.4.7 - minor - Updates to the on-line help.

1.4.7 - minor - Minor bug fixes and improvements.

1.4.7 - minor - Updates to the Supported versions list to exclude 64bit versions of Microsoft Office

1.4.6 - Internal version not publicly distributed

1.4.5 - Minor bug fixes and improvements

1.4.5 - Addition of mooCellQDR Excel function, which can be used as a drop in replacement for the Express XPCellQDR function format identical.

1.4.5 - Addition of mooDimDesc Excel function, which can be used as a drop in replacement for the Express XPDimDesc function. Format similar, see Technical Reference Guide 1.4.5.

1.4.5 - Addition of OLAP Session Manager, you must have ALTER_SYSTEM in order to kill OLAP sessions.

1.4.5 - Addition of Escendo Connection Editor. Enables the end user to define a connection to an Escendo enabled application. Ensure connection security compatibility with Escendo Corps suite of applications .

1.4.4 Internal version not publicly distributed

1.4.3 - Fixed a reported bug which meant that the password was not stored correctly in the connection xml file.
1.4.3 - Description of entry box on connection screen changed from "SID" to "Service Name"

1.4.2 - Minor bug fixes
1.4.1 - Cell limit per individual array retrieve processed by the mooFr function increased from 600,500 to 15,300,000

Installation Instructions

- Close all running instances of Excel De-install the current installation of myObjectiveOLAP either:

Recommended:

Through the Windows Control Panel Add/Remove Programs pane.or by running your original installation setup.exe file and choose Remove.

Alternatively

note: You must use the original setup.exe not setup.exe included within your installed release:

. 1.4.1 - Run the setup.exe included in 2010-06-11-moo-4.1.zip

. 1.4.2 - Run the setup.exe included in 2010-08-18-moo-4.1.zip

. 1.4.3 – Run the setup.exe included in 2010-11-12-moo-1.4.3.zip

. 1.4.4 – Run the setup.exe included in 2011-04-16-moo-1.4.4.zip

. 1.4.5 – Run the setup.exe included in 2011-09-12-moo-1.4.5.zip

. 1.4.6 – Run the setup.exe included in 2011-10-18-moo-1.4.6.zip

. 1.4.7 – Run the setup.exe included in 2011-11-11-moo-1.47.zip

Supported Server Configuration:

Oracle OLAP 11g R1 & R2
Oracle OLAP 10g R1 & R2

Supported Microsoft Windows client operating system:

Microsoft Windows XP Service Pack 2	(32 bit)
Microsoft Windows XP Service Pack 3	(32 bit)
Microsoft Windows 7 (All)	(32 bit)
Microsoft Windows 7 (All)	(64 bit)
Microsoft Windows 8 (All)	(32 bit)
Microsoft Windows 8 (All)	(64 bit)

Supported Microsoft Office:

Microsoft Excel 2003	(32 bit)
Microsoft Excel 2007	(32 bit)
Microsoft Excel 2010	(32 bit)
Microsoft Excel 2013	(32 bit)

64bit Microsoft Office

64 bit releases of Microsoft Office are NOT supported and will NOT work due to a change Microsoft have made to

the mechanism by which extensions communicate with Excel. This does not stop you running 32bit Office on 64bit Windows.

A 64 bit compatible release of myObjectiveOLAP is available in beta, please contact us if you wish to participate in the testing process.

Unsupported:

The following platforms are not supported, although we will help if we can. The information pertaining to myObjectiveOLAPs ability to install and work correctly with older server and client configurations is provided based on customer feedback and has not been independently verified by myObjectiveOLAP development.

Oracle OLAP 9i R1 + this configuration has not been tested at all.
Oracle OLAP 9i R2 + this configuration has been reported as working by a number of customers.
Microsoft Office 2000 + this configuration has been reported as working by a number of customers and is in production use.
Microsoft VISTA ALL variants + this configuration has been reported as working by a number of users and is in production use.
Microsoft Office ALL 64 bit variants + Please see statement above
Terminal or thin-client + This includes but is not limited to: VDI, Microsoft RDC, Citrix. This configuration has been reported as working by a number of customers (Citrix, Windows Server 2008 & 2012).

Legal Notices

Getting Started

Getting Started

In this chapter you will learn and understand the following

Chapter	Summary
System Requirements	Understand the minimum PC requirements. Understand any pre-requisite software Understand database minimum requirements
Files	Understand what files are installed when you install myObjectiveOLAP Client and their purpose Understand the ODAC and .NET pre-requisite software Installing ODAC
Getting Help	Find help and support
Installing	Installing myObjectiveOLAP Client Uninstalling myObjectiveOLAP Client Upgrading myObjectiveOLAP

System requirements

System Requirements

Oracle Data Access Components

You must install the ODAC Driver before using myObjectiveOLAP.

You can download a copy of this from the Oracle website:

ODAC 11.2 Release 3 (11.2.0.2.1) with Xcopy Deployment

http://www.oracle.com/technetwork/database/windows/downloads/utilsoft-087491.html
(URL Correct at time of writing)

myObjectiveOLAP 2010 (V2.2) only supports Version: 2.112.2.0 of Oracle.DataAccess.dll

You must ensure you have the correct version installed on your computer, otherwise you will get an "`Oracle.DataAccess TYPE Error`" when trying to use myObjectiveOLAP functions.

Supported Server Configuration:

Oracle Database with OLAP 11g R1 & R2
Oracle Database with OLAP 10g R1 & R2

Oracle Database with OLAP 11.2.0.3 recommended

Supported Microsoft Windows client operating system:

Microsoft Windows XP Service Pack 2	(32 bit)
Microsoft Windows XP Service Pack 3	(32 bit)
Microsoft Windows 7	(32 bit)
Microsoft Windows 7	(64 bit)
Microsoft Windows 8	(32 bit)
Microsoft Windows 8	(64 bit)

Supported Microsoft Office:

Microsoft Excel 2003	(32 bit)
Microsoft Excel 2007	(32 bit)
Microsoft Excel 2010	(32 bit)

64 bit releases of Microsoft Office are NOT supported and will NOT work due to a change Microsoft have made to the mechanism by which extensions communicate with Excel.
A 64 bit compatible release of myObjectiveOLAP is available in beta, please contact us if you wish to participate in the testing process.

Unsupported Platforms:

The following platforms are not supported, although we will help if we can.

The information pertaining to myObjectiveOLAPs ability to install and work correctly with older server and client configurations
is provided based on customer feedback and has not been independently verified by myObjectiveOLAP development.

Oracle OLAP 9i R1	+ this configuration has not been tested at all
Oracle OLAP 9i R2	+ this configuration has been reported as working by a number of users.
Microsoft Office 2000	+ this configuration has been reported as working by a number of users and is in production use.
Terminal or thin-client	+ This includes but is not limited to: VDI, Microsoft RDC, Citrix.

Microsoft Office ALL 64 bit variants

Files

Files created or used by myObjectiveOLAP

Oracle ODAC Driver

C:\Program Files\oracleODAC2\odp.net\bin\2.x\Oracle.DataAccess.dll

Oracle.DataAccess.dll

myObjectiveOLAP

C:\Documents and Settings\{username}\Application Data\myObjectiveOLAP\myObjectiveOLAP

Microsoft extensibility and Interop Libraries

Interop.VBIDE.dll
Extensibility.dll
Interop.Office.dll
Interop.Excel.dll

Com-Shim Loader and registration Libraries, Executable and Manifest

adxloader.dll.manifest
AddinExpress.MSO.2005.dll
adxregistrator.exe
AddinExpress.XL.2005.dll

myObjectiveOLAP Core Libraries

myObjectiveOLAPXL.dll

adxloader.myObjectiveOLAPXL.dll
adxloader64.myObjectiveOLAPXL.dll

myObjectiveOLAP Configuration Files

C:\Documents and Settings\{username}\Local Settings\

mooApplicationSettings.xml
{Connection_File_name(s)}.xml

System Dependecies

System Dependencies

Before installing or using myObjectiveOLAP you must ensure the PC you are installing on has the following library and runtime environment installed:

ODAC 11.2 Release 3 (11.2.0.2.1) with Xcopy Deployment

myObjectiveOLAP links to the Oracle Data Provider for .NET 2.0 libraries available from Oracle.
The Oracle Data Provider for .NET 2.0 offers high performance and efficient access to Oracle data sources

.NET Runtime 2.0

myObjectiveOLAP makes use of the .NET framework, we have intentionally continued to use the 2.0 version of the framework to maximize the possibility that this will not be a new dependency on your corporate PC estate.

Future note
myObjectiveOLAP development plan is to migrate to the .NET 4 release by myObjectiveOLAP 3.0. This has a current but not binding scheduled release for January 2013. .NET Framework 4.0 was release in 2010.

Installing the Oracle Data Access Provider

ODAC 11.2 Release 3 (11.2.0.2.1) with Xcopy Deployment

myObjectiveOLAP 2010 (V2.2) only supports Version: 2.112.2.0 of Oracle.DataAccess.dll
You must ensure you have the correct version installed on your computer, otherwise you will get an "`Oracle.DataAccess TYPE Error`" when trying to use myObjectiveOLAP functions.

Getting Oracle Data Provider for .NET 2.0
To get the latest version of the Oracle Data Provider for .NET 2.0 you can Google:

"oracle odac 11g xcopy download"

Alternatively, version 11.2.0.2.1 can be downloaded from this URL:

http://www.oracle.com/technetwork/database/windows/downloads/utilsoft-087491.html
(URL Correct at time of writing)

Preparing to Install

Unzip ODAC112021Xcopy.zip into a temporary directory on the computer you wish to install ODAC and myObjectiveOLAP on, for example:

```
C:\ODAC112021Xcopy
```

Install the Oracle Data Provider for .NET 2.0 to a directory on your PC. In this example we will install ODAC to c:\oracleOdac

We will install Oracle Data Provider for .NET 2.0 so that we can see any information returned from the install.bat program

Windows start [Menu Button] → Run → cmd.exe Press [OK]

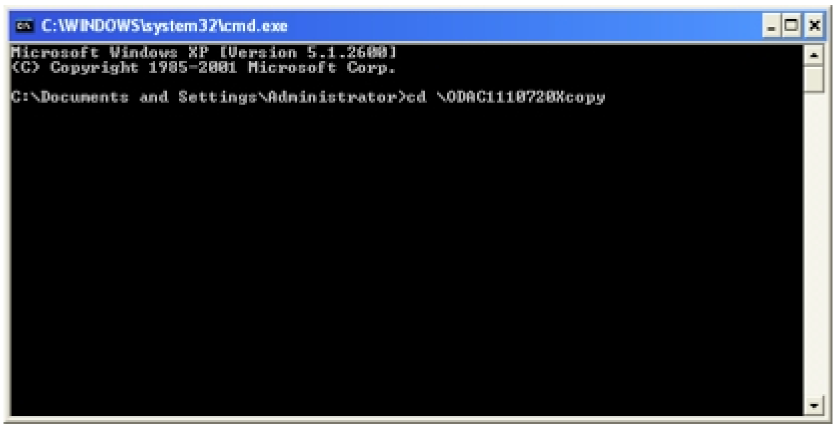
Hint

On Windows 7 right click on cmd.exe and press Run as Administrator. Enter your Admin Credentials.

Change the directory to your temporary directory

```
cd \ODAC112021Xcopy
```

[where ODAC112021Xcopy is your temporary staging directory] Press [Enter key]



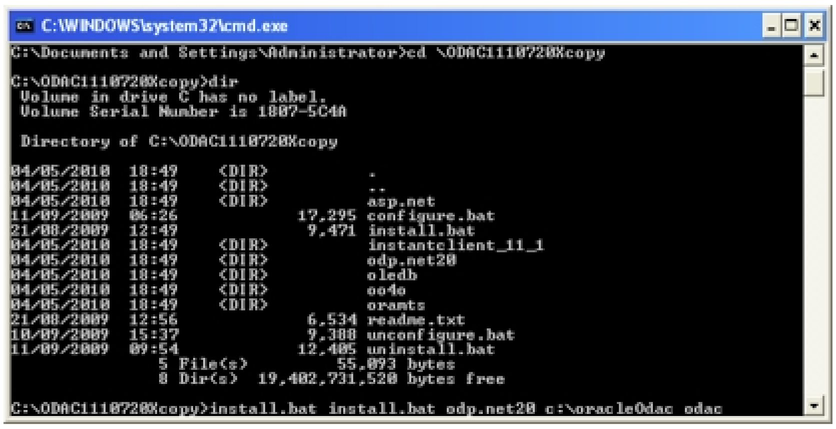
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd \ODAC1110720Xcopy
```

Enter the following command, changing the location of your temporary directory and where you wish to install Oracle Data Provider for .NET 2.0 as appropriate.

install.bat odp.net20 c:\oracleOdac odac

Command	**Explanation**
Install.bat	This is the install executable made available by Oracle
odp.net20	This is the ODAC component we want to install
C:\oracleOdac	This is the directory we want to install ODAC into.
Odac	This is the Oracle_Home name we are giving our install

Press [Enter]



```
C:\Documents and Settings\Administrator>cd \ODAC1110720Xcopy

C:\ODAC1110720Xcopy>dir
 Volume in drive C has no label.
 Volume Serial Number is 1807-5C4A

 Directory of C:\ODAC1110720Xcopy

04/05/2010  18:49    <DIR>          .
04/05/2010  18:49    <DIR>          ..
04/05/2010  18:49    <DIR>          asp.net
11/09/2009  06:26            17,295 configure.bat
21/08/2009  12:49             9,471 install.bat
04/05/2010  18:49    <DIR>          instantclient_11_1
04/05/2010  18:49    <DIR>          odp.net20
04/05/2010  18:49    <DIR>          oledb
04/05/2010  18:49    <DIR>          oo4o
04/05/2010  18:49    <DIR>          oramts
21/08/2009  12:56             6,534 readme.txt
10/09/2009  15:37             9,388 unconfigure.bat
11/09/2009  09:54            12,405 uninstall.bat
               5 File(s)         55,093 bytes
               8 Dir(s)  19,402,731,520 bytes free

C:\ODAC1110720Xcopy>install.bat install.bat odp.net20 c:\oracleOdac odac
```

When the command line returns Oracle Data Provider for .NET 2.0 will have been installed.

Getting help

Official Support

Support for myObjectiveOLAP can be obtained in a number of ways:

Licensed use and annual support direct from myObjectiveOLAP

Customers of myObjectiveOLAP with a valid license and annual support plan can create support requests through the myObjectiveOLAP.com support portal.

http://support.myobjectiveolap.com/otrs/index.pl

The support portal also enables download of all the myObjectiveOLAP.com software including version classified as BETA.

The support portal also enables you to search for FAQ and technical documents on the use of myObjectiveOLAP together with example myObjectiveOLAP Excel macro workbooks.

An annual support agreement with SDMC Consulting Limited

Customers of SDMC Consulting Limited who have an annual support agreement (with a myObjectiveOLAP clause) in place to support their in-house application are licensed for use of myObjectiveOLAP.

They should contact their assigned SDMC support representative if they have questions or problems with myObjectiveOLAP. If necessary they will log a Support Request on the customers behalf.

Licensed use and annual support of Escendo

Customers of Escendo Corporation's Escendo suite of products with a valid license and annual support plan from Escendo Corporation should contact Escendo Support if they have questions or problems with myObjectiveOLAP. If necessary Escendo Support will escalate a Support Request to myObjectiveOLAP on the customers behalf.

Installing

Installing myObjectiveOLAP

The following chapters will take you through installing the myObjectiveOLAP addin.

Installing myObjectiveOLAP

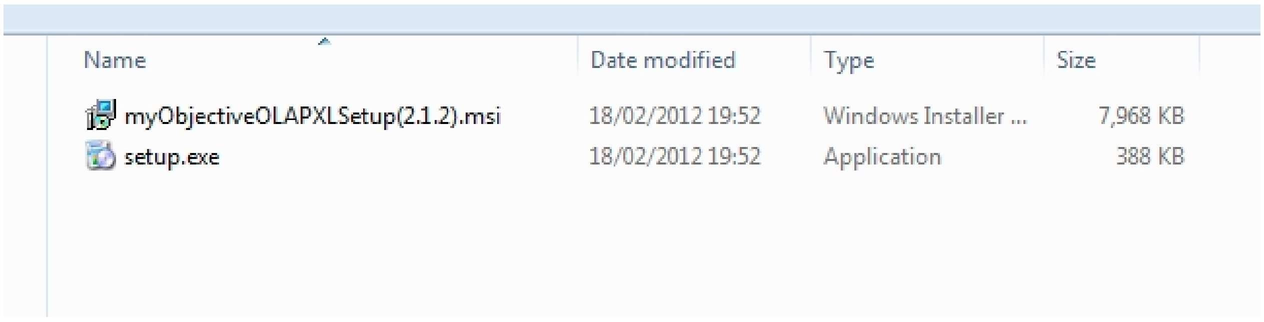
Installing myObjectiveOLAP

Prerequisites

Ensure you have met the prerequisite requirements.

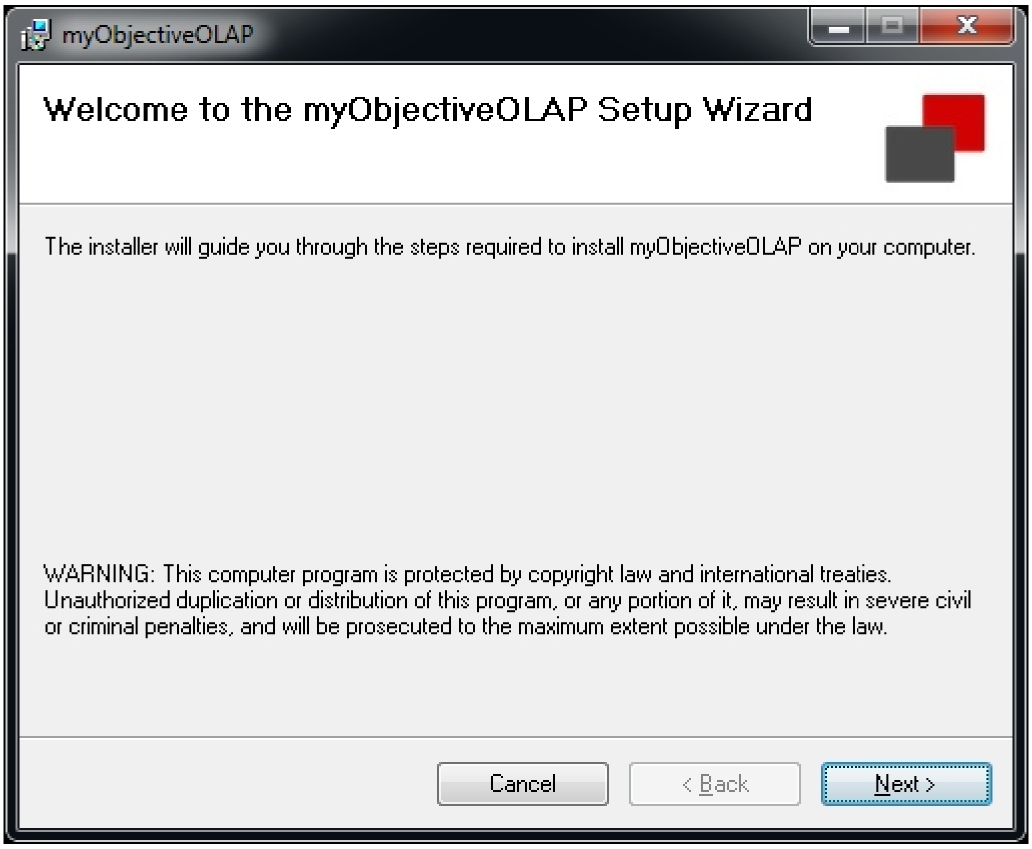
- Oracle Data Access Provider

- .Net Runtime environment



Name	Date modified	Type	Size
myObjectiveOLAPXLSetup(2.1.2).msi	18/02/2012 19:52	Windows Installer ...	7,968 KB
setup.exe	18/02/2012 19:52	Application	388 KB

Double-click the Setup executable.

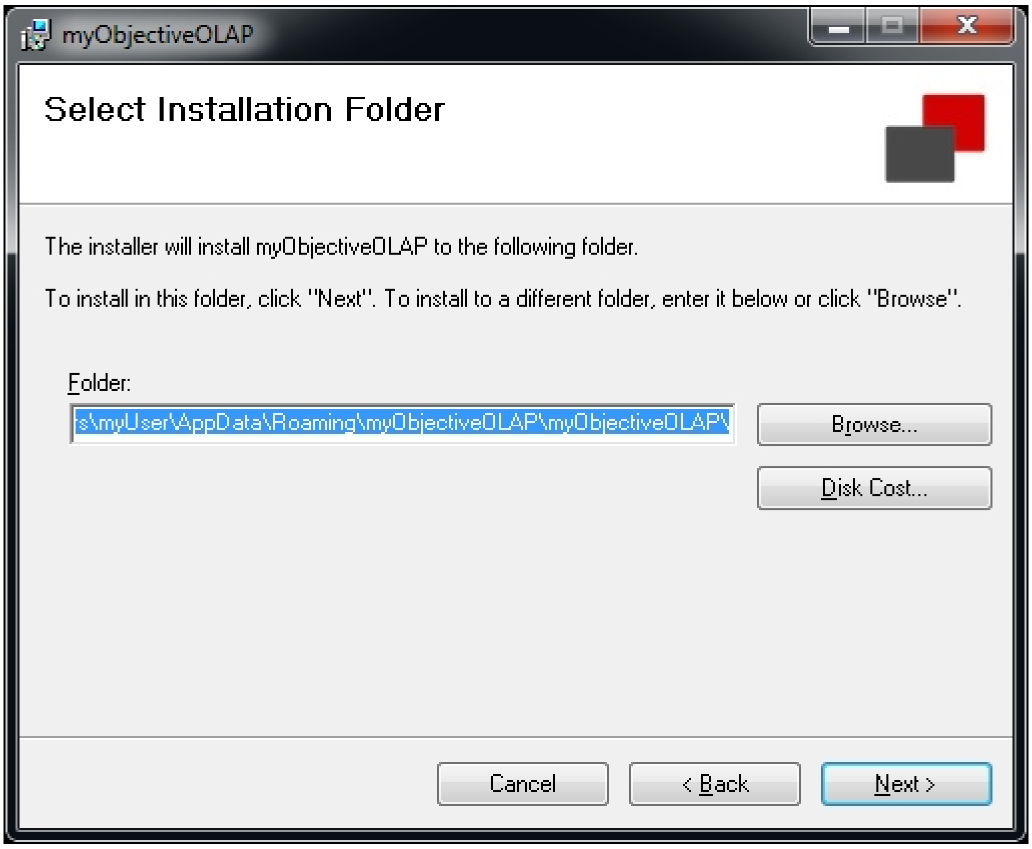


Welcome to the myObjectiveOLAP Setup Wizard

The installer will guide you through the steps required to install myObjectiveOLAP on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

Press [Next]



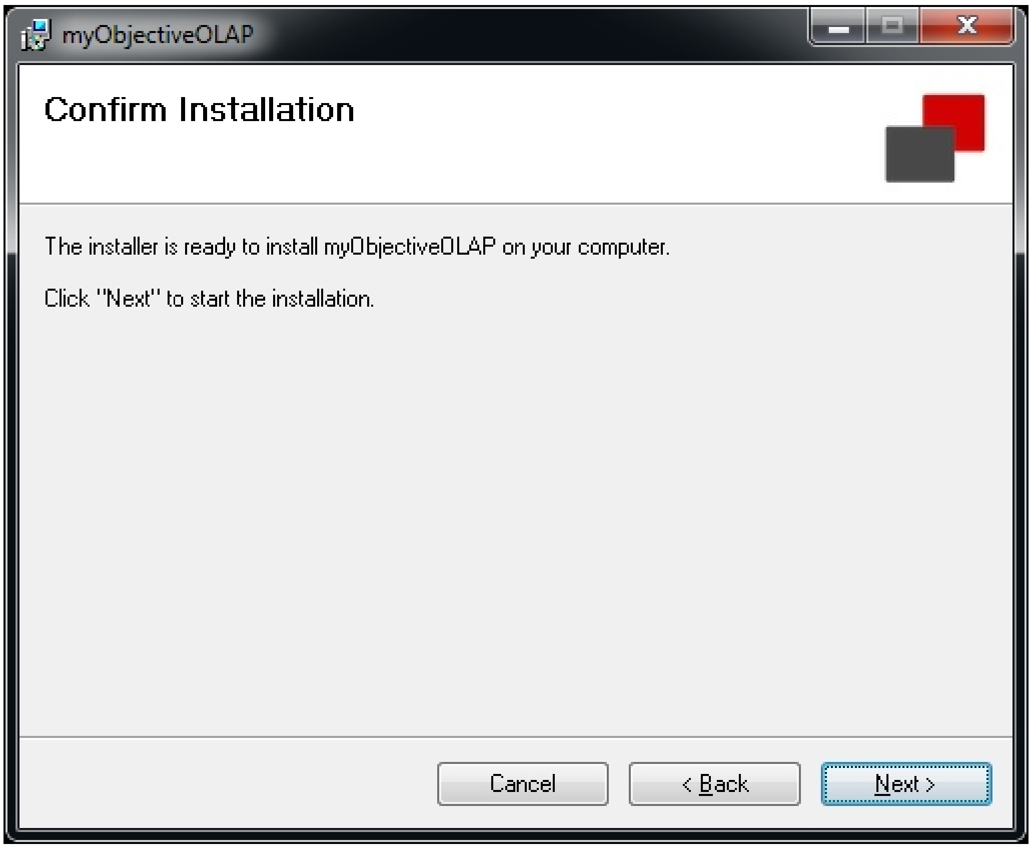
Select Installation Folder

The installer will install myObjectiveOLAP to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder:
s\myUser\AppData\Roaming\myObjectiveOLAP\myObjectiveOLAP\

Choose or select the default directory location you want to install myObjectiveOLAP into.



Confirm Installation

The installer is ready to install myObjectiveOLAP on your computer.

Click "Next" to start the installation.

Confirm Installation

If you are happy with your choices press [Next] to confirm that you want to proceed with the installation of myObjectiveOLAP.

myObjectiveOLAP will now be installed.

Press Close

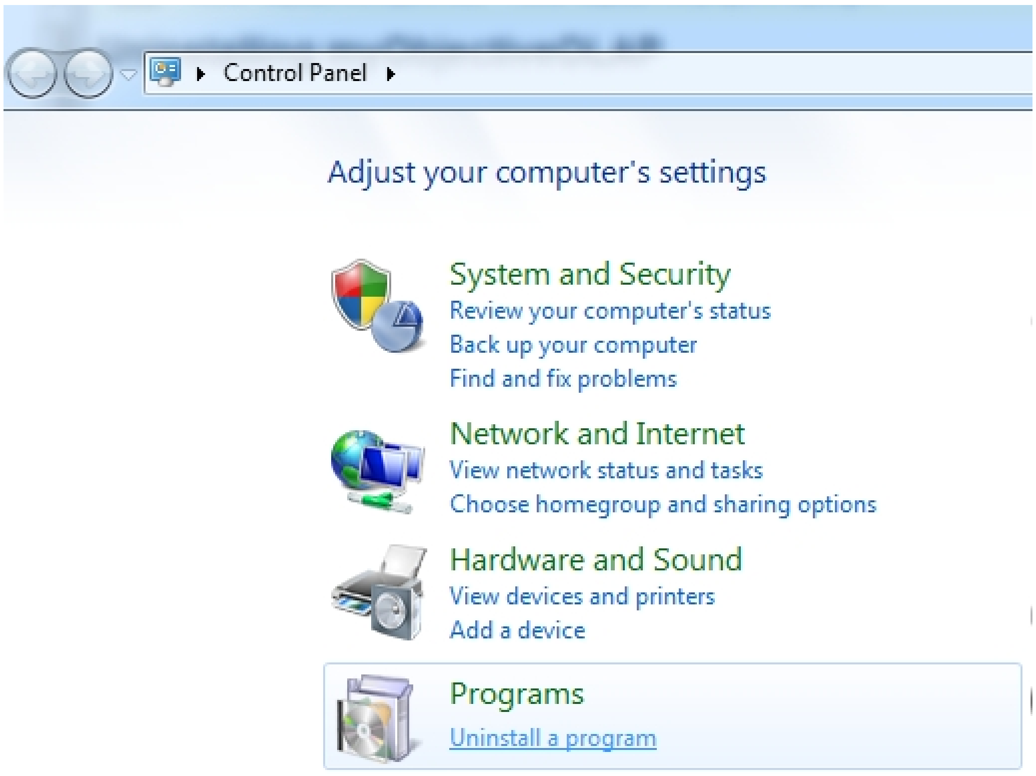
Uninstalling myObjectiveOLAP

Uninstalling myObjectiveOLAP

To uninstall myObjectiveOLAP:

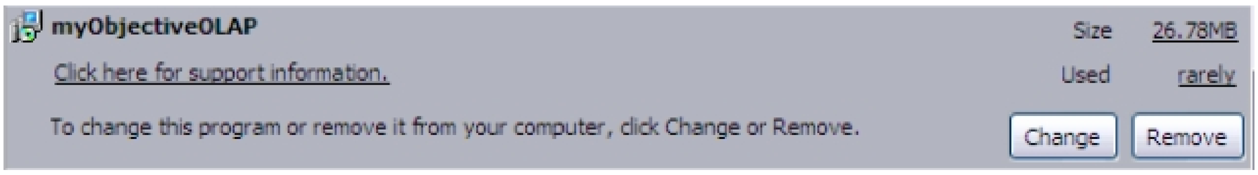
Open the Windows Control Panel.

Select, Programs, Uninstall a Program.



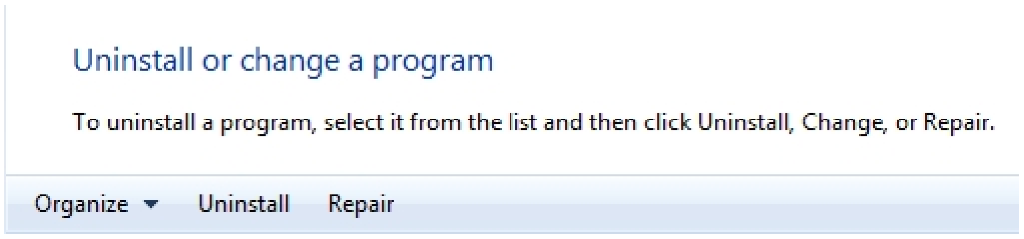
Highlight the myObjectiveOLAP entry in the list of installed programs.

Choose Remove and follow the instructions.



To change this program or remove it from your computer, click Change or Remove.

or choose Uninstall.


Uninstall or change a program

To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

Cleaning up any setting or connection files.

If you are completely uninstalling, and not just preparing to upgrade you may wish to remove the mooApplicationSettings.xml file, and any connection files. These are by default stored at the following location

C:\Documents and Settings\{username}\Local Settings\

If you are upgrading then you probably want to keep these files. If an upgrade of your files is required myObjectiveOLAP will prompt you to upgrade your xml file.

Upgrading from a previous version

Upgrading myObjectiveOLAP

When upgrading the myObjectiveOLAP client you should uninstall your current copy of the software and install the newer version supplied to you.

- Uninstalling myObjectiveOLAP Client

Please follow this link for instructions on uninstalling the myObjectiveOLAP client software. Uninstalling myObjectiveOLAP will not remove any local client configuration files such as your connection files, or mooApplicationSettings.xml file.

- Installing myObjectiveOLAP Client

Please follow this link for instructions on installing the myObjectiveOLAP client software. Installing myObjectiveOLAP will not remove any local client configuration files such as your connection files, or mooApplicationSettings.xml file already installed from a previous installation.

- Move or copy your client configuration files.

You may wish to reuse client configuration or connection files from a previous installation. To do this you should copy the contents of your application data directory to your new application data directory.

Locate your local application data directory,

This is an example on Excel 2003

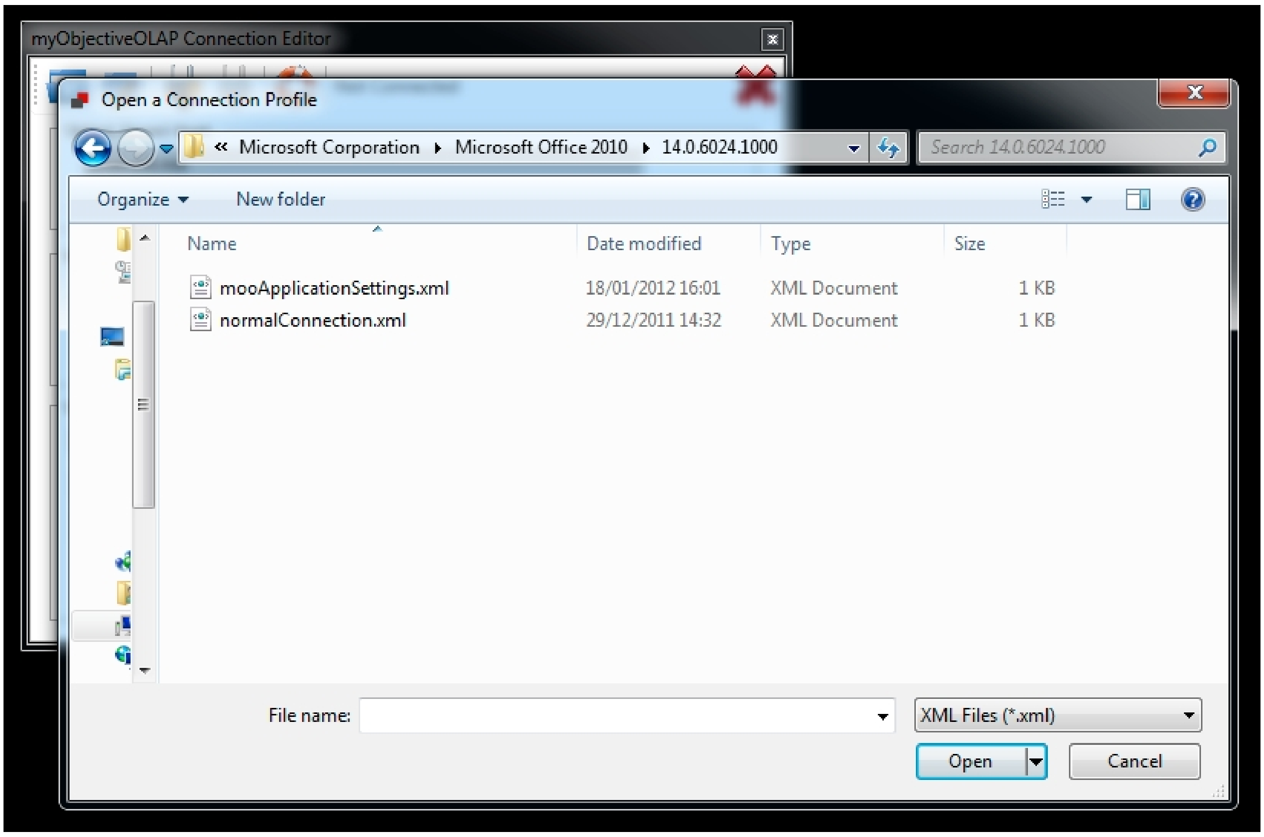
C:\Documents and Settings\{username}\Local Settings\Application Data\

This is an example on Excel 2010

C:\Users\{username\AppData\Local\

Hint

The easy way to identify your current local directory is to start the myObjectiveOLAP Connection Editor and then press "Open" this will open a file browser wizard and display the current directory location in the location bar as shown below:


myObjectiveOLAP Connection Editor
Open a Connection Profile
« Microsoft Corporation ▸ Microsoft Office 2010 ▸ 14.0.6024.1000
Search 14.0.6024.1000
Organize
New folder
File name:
XML Files (*.xml)
Open
Cancel

Name	Date modified	Type	Size
mooApplicationSettings.xml	18/01/2012 16:01	XML Document	1 KB
normalConnection.xml	29/12/2011 14:32	XML Document	1 KB

Application Configuration Files

myObjectiveOLAP Configuration Files

myObjectiveOLAP makes use of a number of configuration files. These are split into two groups:

Application Settings

myObjectiveOLAP gives you the ability to customize the menu items that are displayed to your end-users. This configuration is stored in the mooApplicationSettings.xml file stored within the application user data directory:

The file can either be deployed via a distributed software installation mechanism or configured locally on each user PC.

Connection Files

myObjectiveOLAP makes use of a number of configuration files. These are split into two groups:

mooApplicationSettings.xml

mooApplicationSettings.xml File.

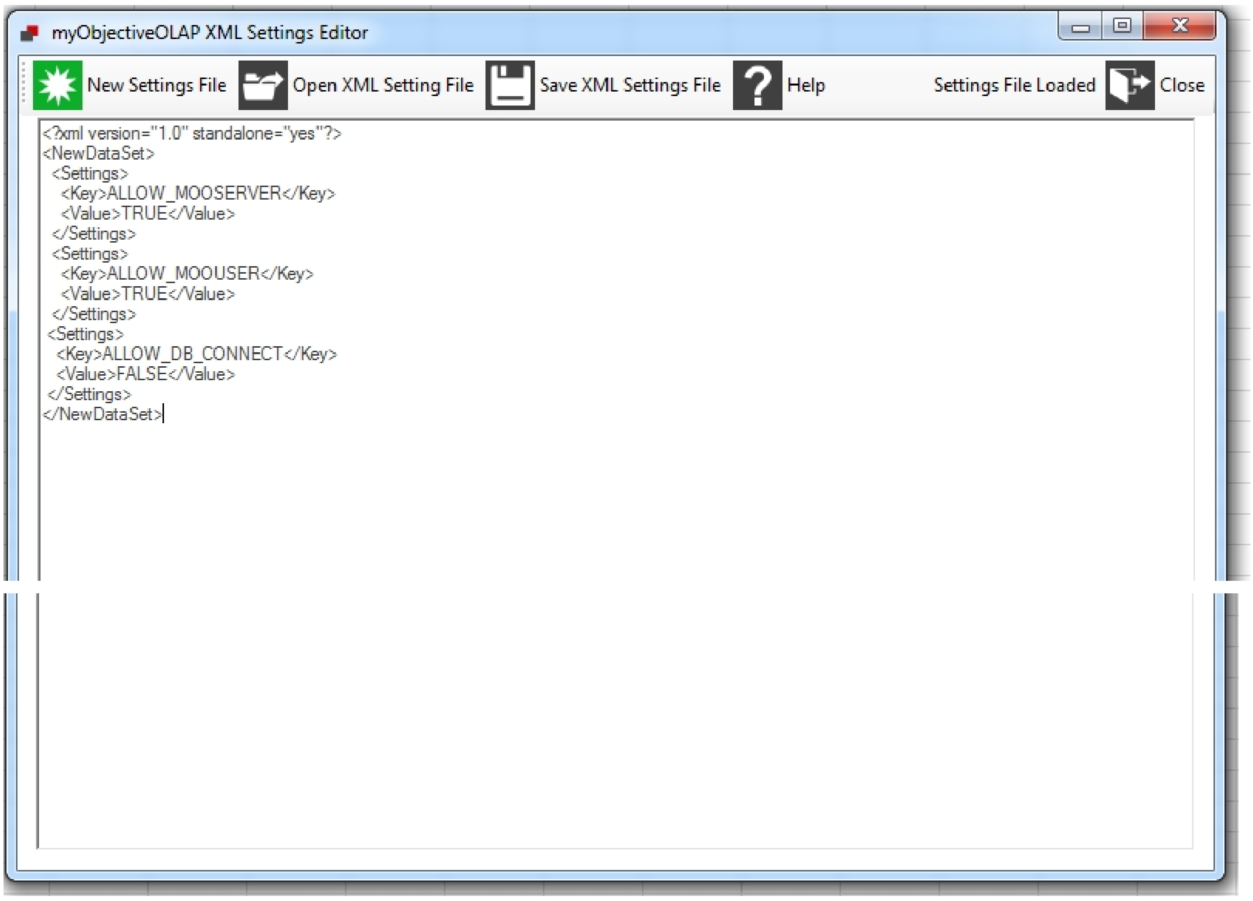
myObjectiveOLAP gives you the ability to customize the menu items that are displayed to your end-users. This configuration is stored in the mooApplicationSettings.xml file stored within the application user data directory:

The file can either be deployed via a distributed software installation mechanism or configured locally on each user PC.

mooApplicationSettings Editor.

When you first install myObjectiveOLAP a menu item "Application Settings Editor" will be created within the myObjectiveOLAP menu group in Excel:

You can use the mooApplicationSettings Editor to create a new mooapplicationSettings.xml file or amend an existing one.



```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>ALLOW_MOOSERVER</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_MOOUSER</Key>
    <Value>TRUE</Value>
  </Settings>
 <Settings>
   <Key>ALLOW_DB_CONNECT</Key>
   <Value>FALSE</Value>
 </Settings>
</NewDataSet>
```

Restricting access to the mooApplicationSettings Editor

User access to edit the mooApplicationSettings editor can be disabled, please see Restricting access to users

Directory to save the file in:

The completed file should be placed in the directory which is represented by `the LocalApplicationData system variable.`

Typically this is either one of the following directories:

```
C:\Users\{username}\AppData\Local
C:\Documents and Settings\{username}\Local Settings\
```

Shared Settings File
Versions 2.9.4 and later only

It is supported to place the mooApplicationSettings.xml file in the directory where myObjectiveOLAP is installed.

You must ensure that the user has permission to read this directory.
If two settings file are placed in LocalApplicationData & the install directory, the users personal version will take precedence and the file saved in the install location will be ignored.
This is primarily useful for customers using myObjectiveOLAP in a terminal or thin-client environment.

```
Please note, myObjectiveOLAP is NOT supported in a terminal or thin-client
environment, however Support will offer advice to customers on a best
endeavors basis.
```

Full ApplicationSettings File

Full mooApplicationSettings.xml file

The following file would enable all menu items of myObjectiveOLAP within Microsoft Excel.

Notes

See: Restricting access to users. to further lock down the menu items.

XML

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>ALLOW_MOOSERVER</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_MOOUSER</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_ESCENDO</Key>
    <Value>TRUE</Value>
  </Settings>
</NewDataSet>
```

OLAP Only

OLAP only mooApplicationSettings.xml file

The following file would only enable the standard database connect screen and would hide access to mooServer and Escendo only menu items.
This would be a valid configuration for normal Oracle OLAP installations.

Notes

See: Restricting access to users. to further lock down the menu items.

XML

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
```

```
  <Settings>
    <Key>ALLOW_MOOSERVER</Key>
    <Value>FALSE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_MOOUSER</Key>
    <Value>FALSE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_ESCENDO</Key>
    <Value>FALSE</Value>
  </Settings>
</NewDataSet>
```

mooServer - User Profile

mooServer User Profile mooApplicationSettings.xml file

The following file would enable the necessary menu items for a mooServer user profile.
This would be a valid configuration for normal myObjectieOLAP Server installations.

Notes

We do not have to restrict Escendo menu items, they are not displayed unless `ALLOW_ESCENDO` is specifically set to TRUE.

We set `ALLOW_DB_CONNECT` to FALSE so that the OLAP only standard connection menu is removed to avoid confusion.

See: Restricting access to users. to further lock down the menu items.

XML

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>ALLOW_MOOSERVER</Key>
    <Value>FALSE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_MOOUSER</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_DB_CONNECT</Key>
    <Value>FALSE</Value>
  </Settings>
</NewDataSet>
```

mooServer - DBA Profile

mooServer DBA Profile mooApplicationSettings.xml file

The following file would enable the necessary menu items for a mooServer DBA profile.
This would be a valid configuration for normal myObjectieOLAP Server installations.

Notes

We do not have to restrict Escendo menu items, they are not displayed unless `ALLOW_ESCENDO` is specifically set to TRUE.

We set `ALLOW_DB_CONNECT` to FALSE so that the OLAP only standard connection menu is removed to avoid confusion.

See: Restricting access to users. to further lock down the menu items.

XML

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>ALLOW_MOOSERVER</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_MOOUSER</Key>
    <Value>TRUE</Value>
  </Settings>
</NewDataSet>
```

Escendo enabled profile

mooServer DBA Profile mooApplicationSettings.xml file

The following file would enable the necessary menu items for a mooServer DBA profile.
This would be a valid configuration for normal myObjectieOLAP Server installations.

Notes

We do not have to restrict mooServer menu items, they are not displayed unless `ALLOW_MOOSERVER` or `ALLOW_MOOUSER` is specifically set to TRUE.

We set `ALLOW_DB_CONNECT` to FALSE so that the OLAP only standard connection menu is removed to avoid confusion.

See: Restricting access to users. to further lock down the menu items.

XML

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>ALLOW_ESCENDO</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_DB_CONNECT</Key>
    <Value>FALSE</Value>
  </Settings>
</NewDataSet>
```

Restricting access to users

Restricting Access

Purpose

A feature request from multiple sources asked for the ability to limit the Advanced functionality from normal end-user installs of myObjectiveOLAP.

In addition: From myObjectiveOLAP Version 1.4.7 if you wish to enable Escendo related functionality you MUST create a mooApplicationSettings.xml file with at least the following content:

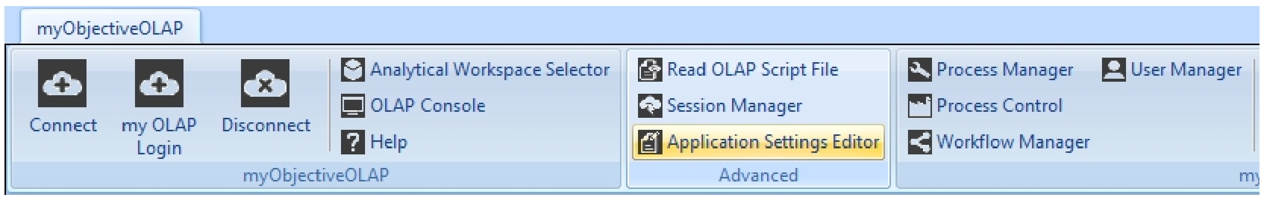
```
<?xml version="1.0" standalone="yes"?>
```

```
<NewDataSet>
  <Settings>
    <Key>ALLOW_ESCENDO</Key>
    <Value>TRUE</Value>
  </Settings>
 </NewDataSet>
```

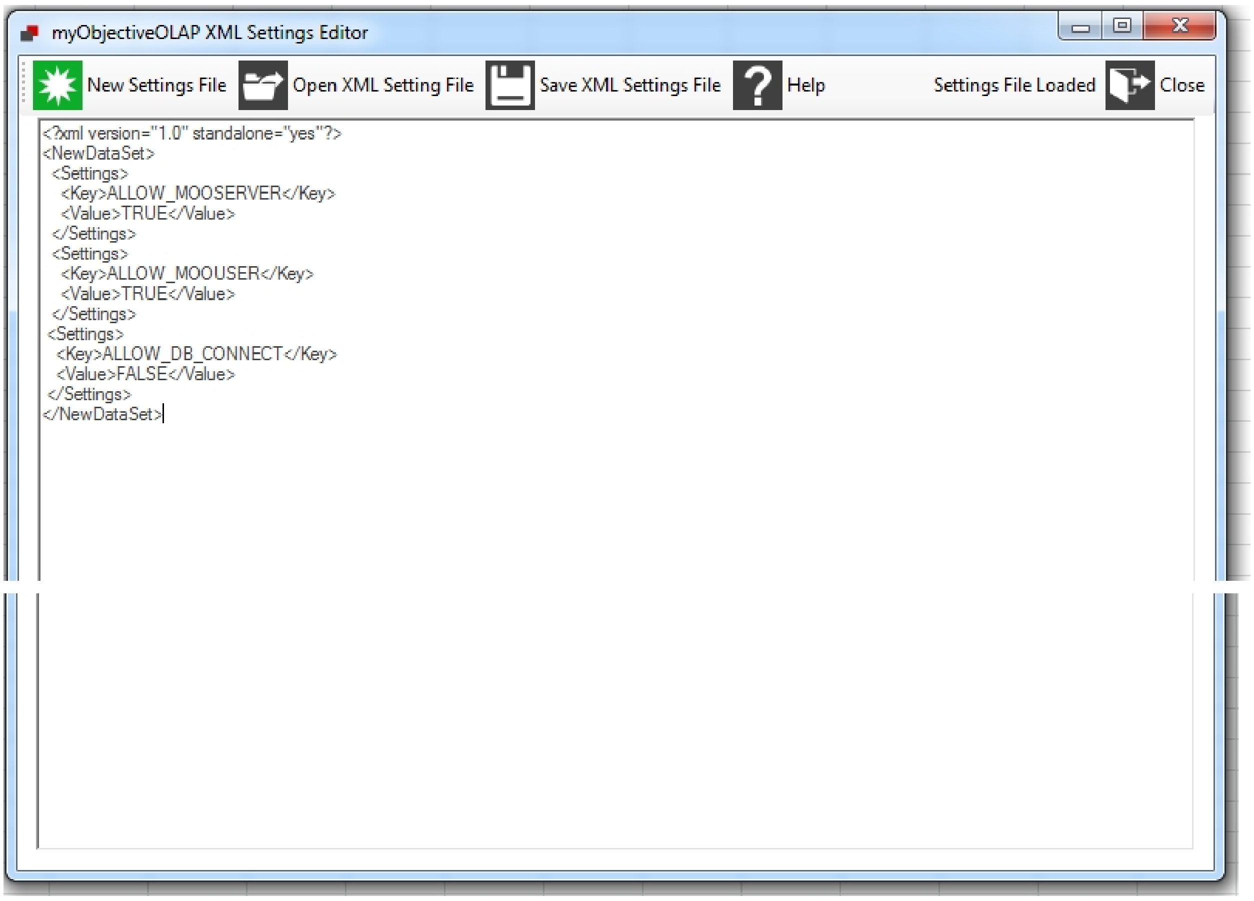
Administrators wishing to enable or disable functionality in myObjectiveOLAP should read this technical note.

Use:

When starting Excel for the first time after installation of myObjectiveOLAP an administrator should use the Application Settings Editor to create an applications setting file.



To create a new settings file add a valid XML schema to the editor window:



```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
 <Settings>
  <Key>ALLOW_MOOSERVER</Key>
  <Value>TRUE</Value>
 </Settings>
 <Settings>
  <Key>ALLOW_MOOUSER</Key>
  <Value>TRUE</Value>
 </Settings>
 <Settings>
  <Key>ALLOW_DB_CONNECT</Key>
  <Value>FALSE</Value>
 </Settings>
</NewDataSet>
```

An example of an XML schema which enables all functionality can be found here.

To edit an existing settings file, use the Open XML Settings File menu and follow the dialog.
To save an existing settings file, use the Save XML Settings File menu and follow the dialog.

Example:

If an administrator wanted to limit the ability for a user to see the Session Manager screen they could create a mooApplicationSettings.xml file with the following contents:

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>ALLOW_OLAP_SESSION_MANAGER</Key>
    <Value>FALSE</Value>
  </Settings>
 </NewDataSet>
```

Keys:

A full list of customizable Key values is shown below:

Standard

ALLOW_OLAP_CONSOLE

Enables or disables the ability for an end-user to open the OLAP command and OLAP program editor windows.

DEFAULT: Enabled

ALLOW_OLAP_SESSION_MANAGER

Enables or disables the ability for an end-user to open the OLAP Session Manager window.

DEFAULT: Enabled

ALLOW_XML_SETTINGS_EDITOR

Enables or disables the ability for an end-user to open the Application Settings Manager.

DEFAULT: Enabled

If you wish to limit functionality you should set this to FALSE.

ALLOW_OLAP_COMMAND_BAR

Enables or disables the ability for an end-user to open the OLAP Command Bar, which would enable the end user to work-around restrictions DEFAULT: Enabled

ALLOW_OLAP_SCRIPT

Enables or disables the ability for an end-user to open the Read OLAP Script File window, which would enable them to execute DML directly   DEFAULT: Enabled

ALLOW_DB_CONNECT

Enables or disables the standard login window, this is only useful if ALLOW_ESCENDO or ALLOW_MOOUSER is enabled.

DEFAULT: Enabled

Escendo

ALLOW_ESCENDO

Enables or disables the ability for an end-user to open the Escendo Connection window, or use Escendo formula in Excel.

DEFAULT: Disabled

mooServer

ALLOW_MOOSERVER

Enables or disables the ability for an end-user to access mooServer control screens

DEFAULT: Disabled

ALLOW_MOOUSER

Enables or disables the mooServer User login

DEFAULT: Disabled

ALLOW_OLAP_SCHEDULER

Enables or disables the minimalist OLAP Scheduler window

DEFAULT: Disabled

ALLOW_PROCESS_CONTROL

Enables or disables the complete OLAP Process Management window

DEFAULT: Disabled

ALLOW_USER_MANAGEMENT

Enables or disables the mooServer User Management window

DEFAULT: Disabled

If ALLOW_MOOSERVER is not set to TRUE, setting more granular controls to TRUE will have no affect. Setting ALLOW_MOOSERVER to TRUE and then removing functionality by setting individual components to FALSE will remove options.

An example mooApplicationSettings.xml file, which enables all myObjectiveOLAP functionality is shown below:

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>ALLOW_ESCENDO</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_MOOSERVER</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_MOOUSER</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_DB_CONNECT</Key>
    <Value>TRUE</Value>
  </Settings>
</NewDataSet>
```

An example mooApplicationSettings.xml file, for myObjectiveOLAP Server users is shown below:

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>ALLOW_MOOSERVER</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_MOOUSER</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_DB_CONNECT</Key>
    <Value>FALSE</Value>
  </Settings>
</NewDataSet>
```

If all advanced functionality is disabled the parent Menu Group will be disabled as well.

Restart Excel

You must restart Excel for the changes to update the myObjectiveOLAP menu.

Files created

Save XML Settings will allow you to create a copy of your XML file with any name, but for it to be valid it must be named mooApplicationSettings.xml
The file must be located in the "Local User Application Data Path" directory, together with the users Connection files.

An example of this path is shown below:

C:\Documents and Settings\myUser\Local Settings\Application Data\Microsoft Corporation\Microsoft Office 2003 \11.0.8341

Connection Files

Connecting to Oracle OLAP.

myObjectiveOLAP supports three connection types:

Standard Oracle OLAP Connection

A standard Oracle OLAP connection should be used when connecting to an OLAP environment that is managed through Oracle's standard AWM tools and conforms to the Oracle Standard Form model.

mooServer Connection

A myObjectiveOLAP mooServer Connection supports additional server side work flow, data submission and reporting tools.
This type of connection should only be used with a mooServer enabled environment.

Escendo Connection

Escendo Connection supports connecting to Escendo Corporations, Escendo Suite of OLAP enabled Reporting, Budgeting and Planning Applications.

Connecting to Oracle OLAP is covered in more detail here.

Example path

By default myObjectiveOLAP will look in the a location similar to below for pre-saved Connection Files.

```
C:\Documents and Settings\{username}\Local Settings\
```

Connecting

Connecting to Oracle OLAP.

myObjectiveOLAP supports three connection types:

Standard Oracle OLAP Connection

A standard Oracle OLAP connection should be used when connecting to an OLAP environment that is managed through Oracle's standard AWM tools and conforms to the Oracle Standard Form model.

mooServer Connection

A myObjectiveOLAP mooServer Connection supports additional server side work flow, data submission and reporting tools.
This type of connection should only be used with a mooServer enabled environment.

Escendo Connection

An Escendo Connection supports connecting to Escendo Corporations, Escendo Suite of OLAP enabled Reporting, Budgeting and Planning Applications.

Example path

By default myObjectiveOLAP will look in the a location similar to below for pre-saved Connection Files.

```
C:\Documents and Settings\{username}\Local Settings\
```

Locate your application data directory,

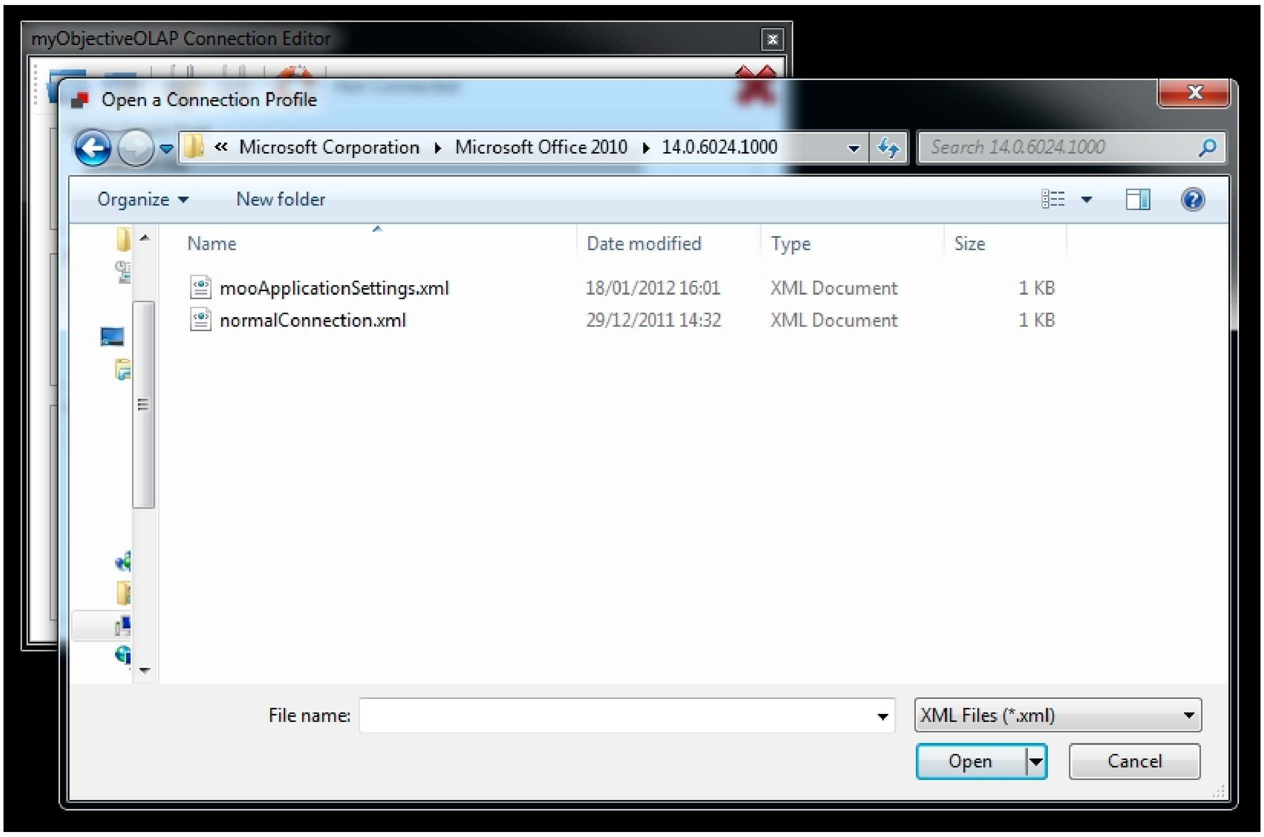
This is an example on Excel 2003

C:\Documents and Settings\{username}\Local Settings\

This is an example on Excel 2010

C:\Users\{username\AppData\Local\

Hint
The easy way to identify your current local directory is to start the myObjectiveOLAP Connection Editor and then press "Open" this will open a file browser wizard and display the current directory location in the location bar as shown below:


myObjectiveOLAP Connection Editor
Open a Connection Profile
« Microsoft Corporation ▸ Microsoft Office 2010 ▸ 14.0.6024.1000
Search 14.0.6024.1000
Organize
New folder
File name:
XML Files (*.xml)
Open
Cancel

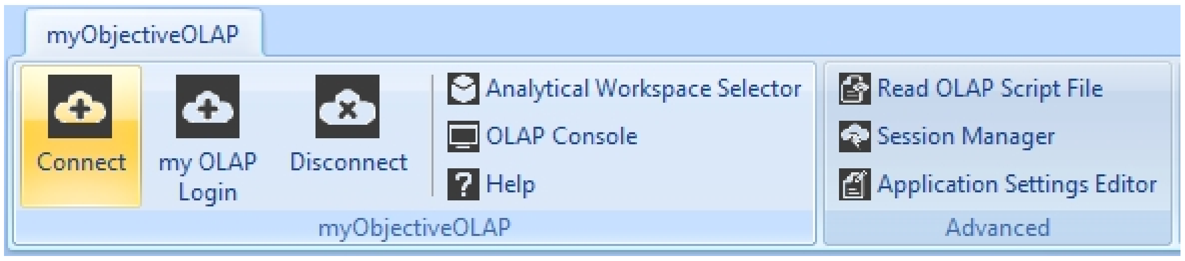
Name	Date modified	Type	Size
mooApplicationSettings.xml	18/01/2012 16:01	XML Document	1 KB
normalConnection.xml	29/12/2011 14:32	XML Document	1 KB

Standard OLAP Connection

Standard Oracle OLAP Connection

A standard Oracle OLAP connection should be used when connecting to an OLAP environment that is managed through Oracle's standard AWM tool and conforms to the Oracle Standard Form model.

The Connection Editor is found within the main myObjectiveOLAP menu group.

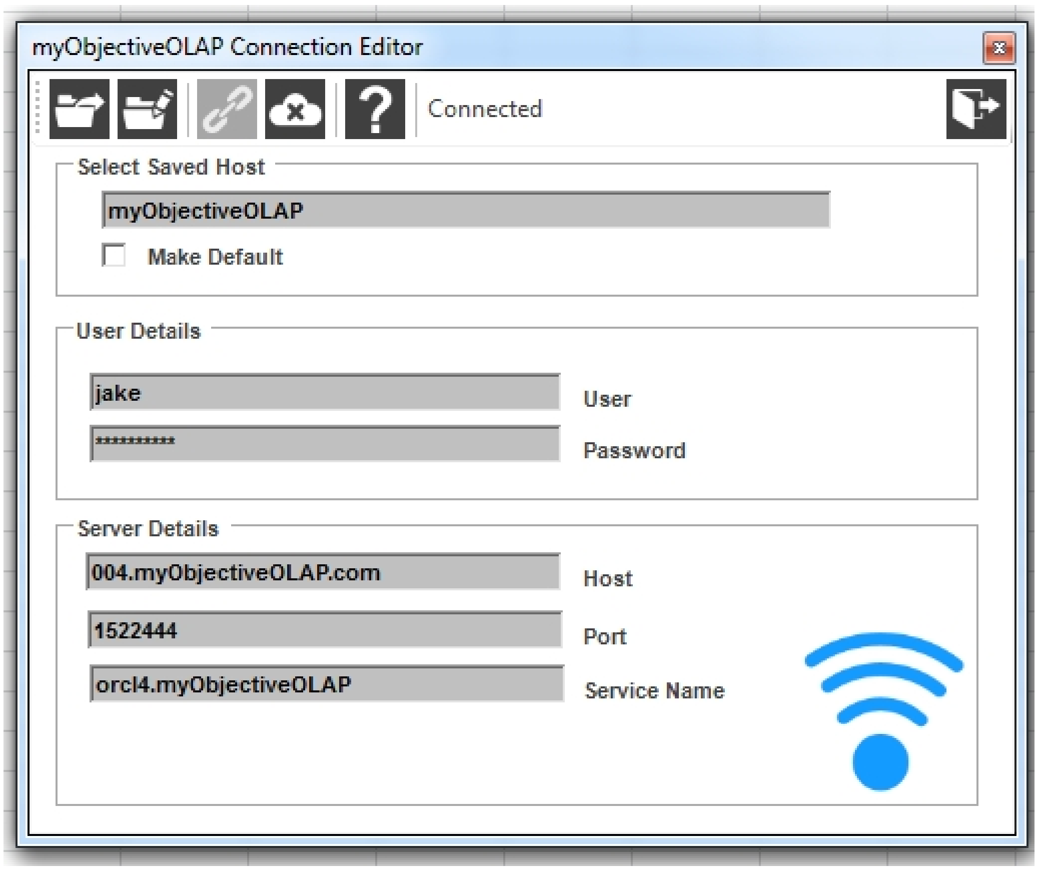


Standard Connection Editor Window

The standard Connection editor window enables the user to enter connection details associated with an Oracle OLAP enabled database instance.

The User can perform the following actions:

Open	–	Open an existing connection file.
Save	–	Save a new Standard Oracle OLAP connection file.
Connect	–	Initiate a connection to an Oracle OLAP enabled database instance based on the entered connection information.
Disconnect	–	Disconnect from an existing connection.
Help	–	Open this Help Topic.
Close	–	Closes the Standard Connection Editor Window



myObjectiveOLAP
jake
****
004.myObjectiveOLAP.com
1522444
orcl4.myObjectiveOLAP

Standard Connection File

By default myObjectiveOLAP will look in the a location similar to below for pre-saved connection Files.

`C:\Documents and Settings\{username}\Local Settings\`

The following shows an example Standard Oracle OLAP Connection xml file:

```xml
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>SAVE</Key>
    <Value>My Saved Bookmark</Value>
  </Settings>
  <Settings>
    <Key>SERVER</Key>
    <Value>myhost.com</Value>
  </Settings>
  <Settings>
    <Key>PORT</Key>
    <Value>1521</Value>
  </Settings>
  <Settings>
    <Key>SID</Key>
    <Value>orcl</Value>
  </Settings>
  <Settings>
    <Key>USER</Key>
    <Value>olapsys</Value>
  </Settings>
  <Settings>
    <Key>PASS</Key>
    <Value>dlasidoapsOIOPdhaoshiad==</Value>
  </Settings>
</NewDataSet>
```

The following keys are stored:

Key		**Description**
Save	–	User friendly description of the connection
Server	–	The hostname or IP address of the server you wish to connect to.
Port	–	The port of the Oracle database instance you wish to connect to.
User	–	The username of the Oracle user you wish to connect with.
PASS	–	An encrypted hash of the Oracle password.

You can save a connection file with any filename supported by the Microsoft Windows file system.

If you check the Make Default option when saving a connection file a second file called Settings.xml is automatically created.
Any time the Standard Connection Editor window is opened the contents of Settings.xml will be loaded if it exists.

Displaying the Standard Connection Window from VBA

You can trigger the connection window to be displayed from within Excel VBA by calling the `mooShowConnFrm` function, as described here.

Connecting From the Console

If you have created a Default Settings.xml as described above you can type MOO CONOLAP in the OLAP Console Command Entry Window and myObjectiveOLAP will try to automatically connect.

Connecting From VBA

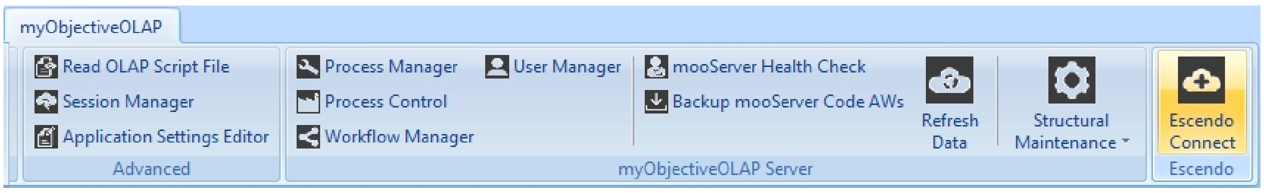
If you have created a Default Settings.xml as described above you can use the `connect()` function from within Excel as described here.

Escendo Connection

Escendo Application Connection

An Escendo Connection supports connecting to Escendo Corporations, Escendo Suite of OLAP enabled reporting, budgeting and planning applications.

The Escendo Connect, connection editor is found within the Escendo myObjectiveOLAP menu group.

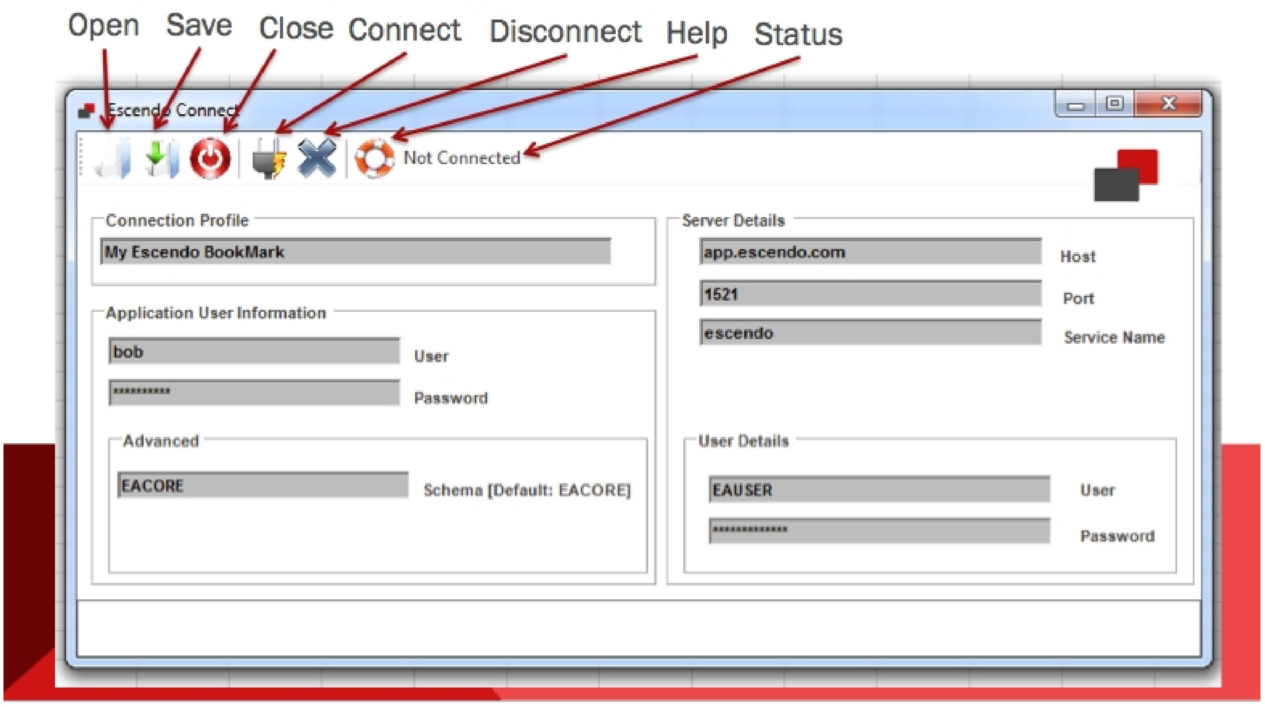


Escendo Connect, Connection Editor Window

The standard Connection editor window enables the user to enter connection details associated with an Oracle OLAP enabled database instance.

The User can perform the following actions:

Open	–	Open an existing connection file.
Save	–	Save a new Standard Oracle OLAP connection file.
Connect	–	Initiate a connection to an Oracle OLAP enabled database instance based on the entered connection information.
Disconnect	–	Disconnect from an existing connection.
Close	–	Closes the Escendo Connect connection window
Help	–	Open this Help Topic.



Not Connected

Connection Profile
My Escendo BookMark

Application User Information
bob User
********** Password

Advanced
EACORE Schema [Default: EACORE]

Server Details
app.escendo.com Host
1521 Port
escendo Service Name

User Details
EAUSER User
************** Password

Escendo Connection File

By default myObjectiveOLAP will look in the a location similar to below for pre-saved connection Files.

```
C:\Documents and Settings\{username}\Local Settings\
```

The following shows an example Escendo connection xml file:

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>CONNProfile</Key>
    <Value>My Escendo BookMark</Value>
  </Settings>
  <Settings>
    <Key>APP_UNAME</Key>
    <Value>bob</Value>
  </Settings>
  <Settings>
    <Key>APP_PASS</Key>
    <Value>HeJNgRKh/dzU2/bWT3z/sw==</Value>
  </Settings>
  <Settings>
    <Key>HOST</Key>
    <Value>app.escendo.com</Value>
  </Settings>
  <Settings>
    <Key>PORT</Key>
    <Value>1521</Value>
  </Settings>
  <Settings>
    <Key>SID</Key>
    <Value>escendo</Value>
  </Settings>
  <Settings>
    <Key>DB_UNAME</Key>
    <Value>EAUSER</Value>
  </Settings>
  <Settings>
    <Key>DB_PASS</Key>
    <Value>oGVXlXUp0SXgUweR7fDVNg==</Value>
  </Settings>
  <Settings>
    <Key>SCHEMA</Key>
    <Value>EACORE</Value>
  </Settings>
</NewDataSet>
```

The following keys are stored:

Key		**Description**
CONNProfile	--	User friendly description of the Escendo connection profile
APP_UNAME	--	The username of the Escendo application user
APP_PASS	--	The encrypted password hash of the Escendo application user
HOST	--	The hostname or IP address of the Escendo enabled Oracle OLAP server you wish to connect to.
Port	--	The port of the Oracle database instance you wish to connect to.
SID	--	The SID of the Oracle database instance you wish to connect to.
DB_UNAME	--	The username of the database connection, this is normally EAUSER
DB_PASS	--	The encrypted password hash of the database connection, this is normally set to never expire, as the user security has been delegated to the Escendo application. If it is changed new connection xml files must be distributed or update on change.
SCHEMA	--	This is the Oracle schema in which the Escendo application resides. In a default installation this is normally EACORE.

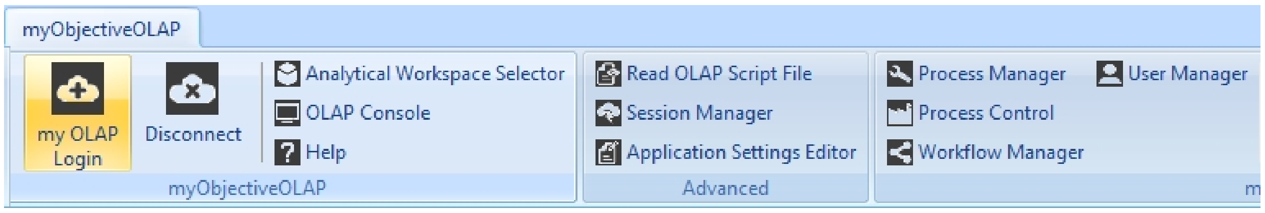
You can save a connection file with any filename supported by the Microsoft Windows file system.

mooServer Connection

mooServer Connection

A myObjectiveOLAP mooServer Connection supports additional server side work flow, data submission and reporting tools.
This type of connection should only be used with a mooServer enabled environment.

The mooServer "my OLAP Login" connection editor is found within the main myObjectiveOLAP menu group.

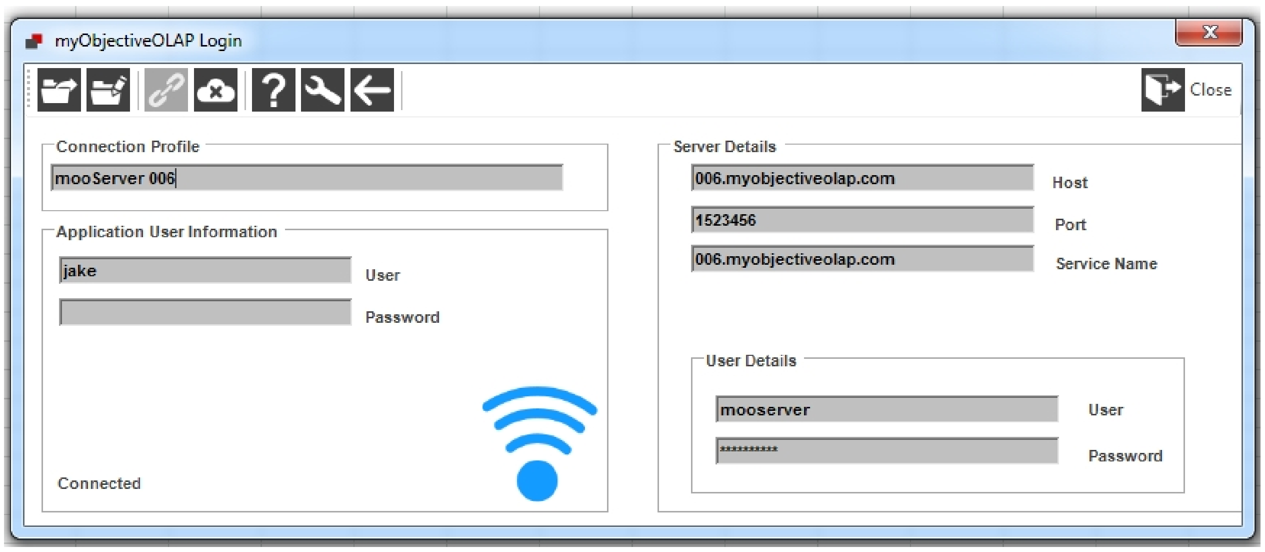


mooServer Connection Editor Window

The "my OLAP Login" connection editor window enables the user to enter connection details associated with an mooServer enabled Oracle OLAP enabled database instance.

The User can perform the following actions:

Open	–	Open an existing connection file.
Save	–	Save a new Standard Oracle OLAP connection file.
Connect	–	Initiate a connection to an Oracle OLAP enabled database instance based on the entered connection information.
Disconnect	–	Disconnect from an existing connection.
Help	–	Open this Help Topic.
Close	–	Closes the Standard Connection Editor Window



Connection Profile

mooServer 006

Application User Information

jake User

Password

Connected

Server Details

006.myobjectiveolap.com Host

1523456 Port

006.myobjectiveolap.com Service Name

User Details

mooserver User

********** Password

mooServer Connection File

By default myObjectiveOLAP will look in the a location similar to below for pre-saved connection Files.

```
C:\Documents and Settings\{username}\Local Settings\
```

The following shows an example mooServer connection xml file:

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>CONNProfile</Key>
    <Value>mooServer</Value>
  </Settings>
  <Settings>
    <Key>APP_UNAME</Key>
    <Value>bob</Value>
  </Settings>
  <Settings>
    <Key>HOST</Key>
    <Value>mooserver.myobjectiveolap.com</Value>
  </Settings>
  <Settings>
    <Key>PORT</Key>
    <Value>1521</Value>
  </Settings>
  <Settings>
```

```
    <Key>SID</Key>
    <Value>moo</Value>
  </Settings>
  <Settings>
    <Key>DB_UNAME</Key>
    <Value>mooserver</Value>
  </Settings>
  <Settings>
    <Key>DB_PASS</Key>
    <Value>jdsakldjLJKLJDASJDSKALJEEjkk</Value>
  </Settings>
</NewDataSet>
```

The following keys are stored:

Key		**Description**
CONNProfile	--	User friendly description of the mooServer connection profile
APP_UNAME	--	The username of the mooServer application user
HOST	--	The hostname or IP address of the mooServer enabled Oracle OLAP server you wish to connect to.
Port	--	The port of the Oracle database instance you wish to connect to.
SID	--	The SID of the Oracle database instance you wish to connect to.
DB_UNAME	--	The username of the database connection, this is normally mooserver
DB_PASS	--	The encrypted password hash of the database connection, this is normally set to never expire, as the user security has been delegated to the mooServer application. If it is changed new connection xml files must be distributed or update on change.

Note: mooServer does not allow the storing of the application password.

You can save a connection file with any filename supported by the Microsoft Windows file system.

Graphical Tools

Graphical Tools

myObjectiveOLAP contains a number of graphical tools intended to automate and help the end-user or developer.

These tools are split into two containers:

Standard Tools

User Tools

[Analytic Workspace Selector]
[Application Settings Editor]

Developer or DBA Tools

[OLAP Console]
[Read OLAP Script File]
[Session Manager]

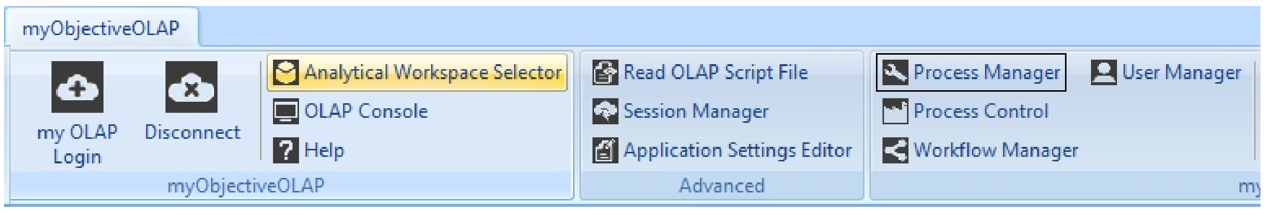
myObjectiveOLAP Server Tools

Please see: myObjectiveOLAP Server Tools

Analytic Workspace Selector

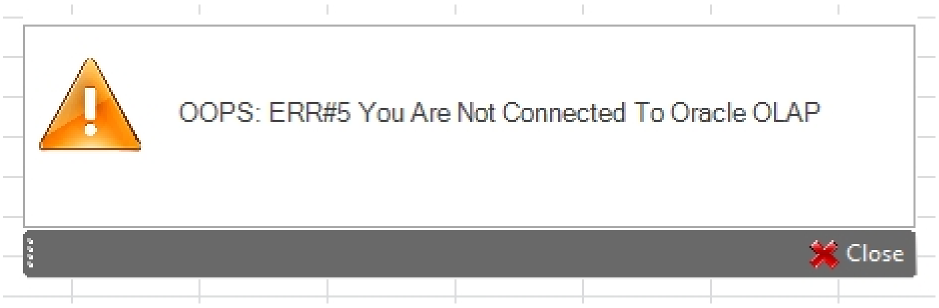
Analytic Workspace Selector

The Analytic Workspace Selector is found within the main myObjectiveOLAP menu group.

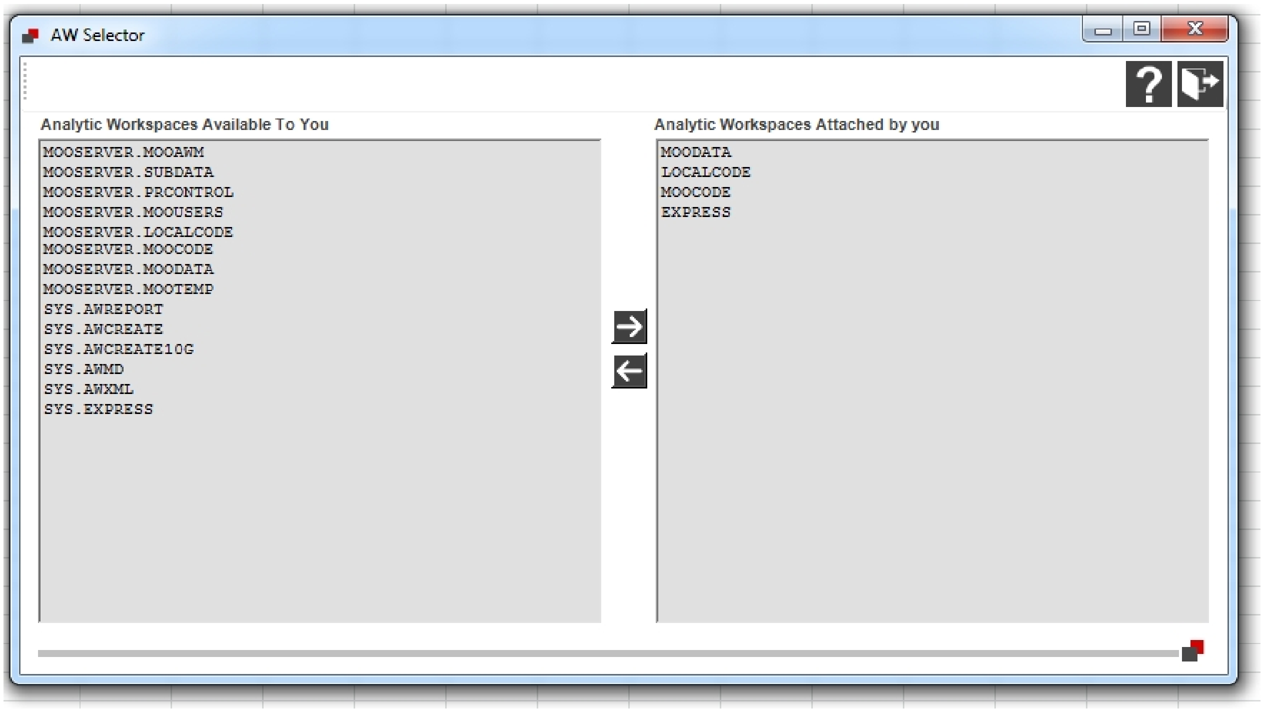


Notes

You can only open the Analytic Workspace Selector if you are connected to an Oracle OLAP database. If you attempt to open it before being connected you will see the following message:


OOPS: ERR#5 You Are Not Connected To Oracle OLAP

On opening the Analytic Workspace Selector you will be presented with a list of Analytic Workspaces available to you (left List Box) and Analytic Workspaces attached already by you (right List Box).


AW Selector
Analytic Workspaces Available To You
Analytic Workspaces Attached by you

```
MOOSERVER.MOOAWM
MOOSERVER.SUBDATA
MOOSERVER.PRCONTROL
MOOSERVER.MOOUSERS
MOOSERVER.LOCALCODE
MOOSERVER.MOOCODE
MOOSERVER.MOODATA
MOOSERVER.MOOTEMP
SYS.AWREPORT
SYS.AWCREATE
SYS.AWCREATE10G
SYS.AWMD
SYS.AWXML
SYS.EXPRESS
```

```
MOODATA
LOCALCODE
MOOCODE
EXPRESS
```

Attaching an Analytic Workspace

"Double Clicking" on the Analytic Workspace name in the "Analytic Workspaces Available To You" list will attach the Analytic Workspace in Read Only mode and update the "Attached By You" list.
Alternatively, highlighting the Analytic Workspace in the "Available To You" list and pressing the Right arrow will attach the highlighted Analytic Workspace.

Detaching an Analytic Workspace

"Double Clicking" on the Analytic Workspace name in the "Analytic Workspaces Attached By You" list will detach the Analytic Workspace.
Alternatively, highlighting the Analytic Workspace in the "Analytic Workspaces Attached By You" list and pressing the Left arrow will detach the highlighted Analytic Workspace.

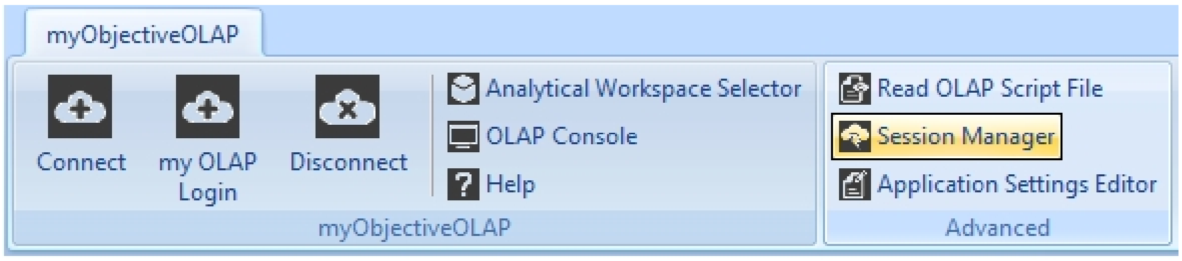
Changing the Analytic Workspace Attach Order

If you already have the Analytic Workspace attached and wish to make a specific Analytic Workspace first in the attached order you can "Double Click" on an Analytic Workspace name in the "Analytic Workspaces Available To You" list. This will attach the Analytic Workspace in Read Only mode to the first position and update the "Attached By You" list to represent this.

Session Manager

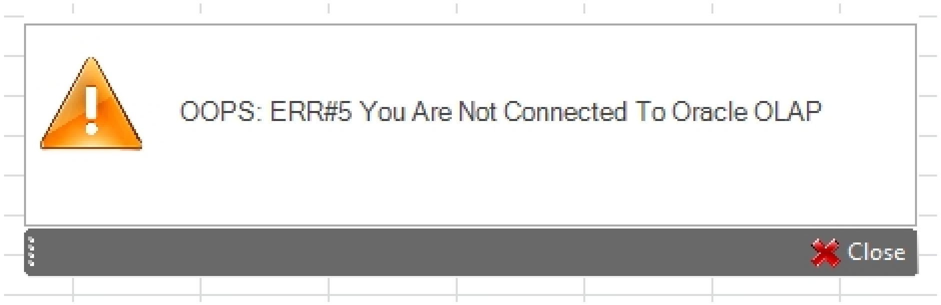
Session Manager

The Analytic Workspace Selector is found within the Advanced myObjectiveOLAP menu group.



Notes

You can only open the Analytic Workspace Selector if you are connected to an Oracle OLAP database. If you attempt to open it before being connected you will see the following message:



OOPS: ERR#5 You Are Not Connected To Oracle OLAP

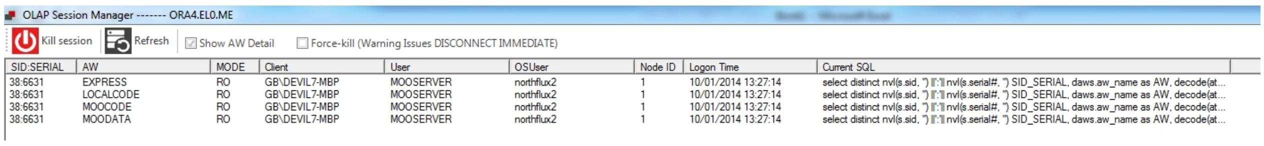
In order to use the Session Manager tool your Oracle user must have been assigned the `'ALTER SYSTEM'` privilege, otherwise you will see the following message in the Session Manager ribbon menu:

"You Do Not Have The Correct Oracle Roles to Kill Sessions"

On opening the Session Manager with the correct privilege you will be presented with a list of OLAP Sessions.

The OLAP Session Manager lists the following information:

```
SID:SERIAL
AW
MODE
Client PC Name
Database User
OS User Client
Database Node the client is connected to
Logon time
Last executed SQL
```



OLAP Session Manager ------- ORA4.EL0.ME

[✓ Show AW Detail] Force-kill (Warning Issues DISCONNECT IMMEDIATE)

SID:SERIAL	AW	MODE	Client	User	OSUser	Node ID	Logon Time	Current SQL
38:6631	EXPRESS	RO	GB\DEVIL7-MBP	MOOSERVER	northflux2	1	10/01/2014 13:27:14	select distinct nvl(s.sid, '') ||':'|| nvl(s.serial#, '') SID_SERIAL, daws.aw_name as AW, decode(at...
38:6631	LOCALCODE	RO	GB\DEVIL7-MBP	MOOSERVER	northflux2	1	10/01/2014 13:27:14	select distinct nvl(s.sid, '') ||':'|| nvl(s.serial#, '') SID_SERIAL, daws.aw_name as AW, decode(at...
38:6631	MOOCODE	RO	GB\DEVIL7-MBP	MOOSERVER	northflux2	1	10/01/2014 13:27:14	select distinct nvl(s.sid, '') ||':'|| nvl(s.serial#, '') SID_SERIAL, daws.aw_name as AW, decode(at...
38:6631	MOODATA	RO	GB\DEVIL7-MBP	MOOSERVER	northflux2	1	10/01/2014 13:27:14	select distinct nvl(s.sid, '') ||':'|| nvl(s.serial#, '') SID_SERIAL, daws.aw_name as AW, decode(at...

Killing a Session

Highlighting a SSESID and pressing Kill Session will attempt to kill the highlighted session by executing the following SQL Statement:

```
ALTER SYSTEM KILL SESSION [SSESID]
```

If you check the Force Kill option before pressing Kill Session will execute the following SQL Statement:

```
ALTER SYSTEM DISCONNECT SESSION [SSESID]
```

After killing the session the Session List will be refreshed.

You can not kill your own session.

OLAP Session

In order for an Oracle session to be classed as an OLAP Session a single OLAP DML statement must have been executed, this includes attaching an Analytic Workspace.

Application Settings Editor

mooApplicationSettings.xml File.

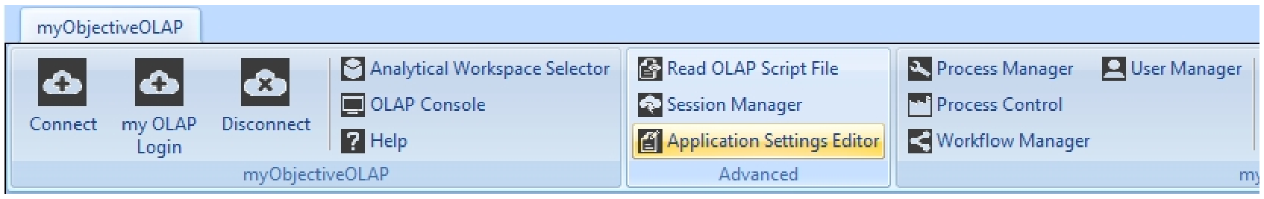
myObjectiveOLAP gives you the ability to customize the menu items that are displayed to your end-users. This configuration is stored in the mooApplicationSettings.xml file stored within the application user data directory:

The file can either be deployed via a distributed software installation mechanism or configured locally on each user PC.

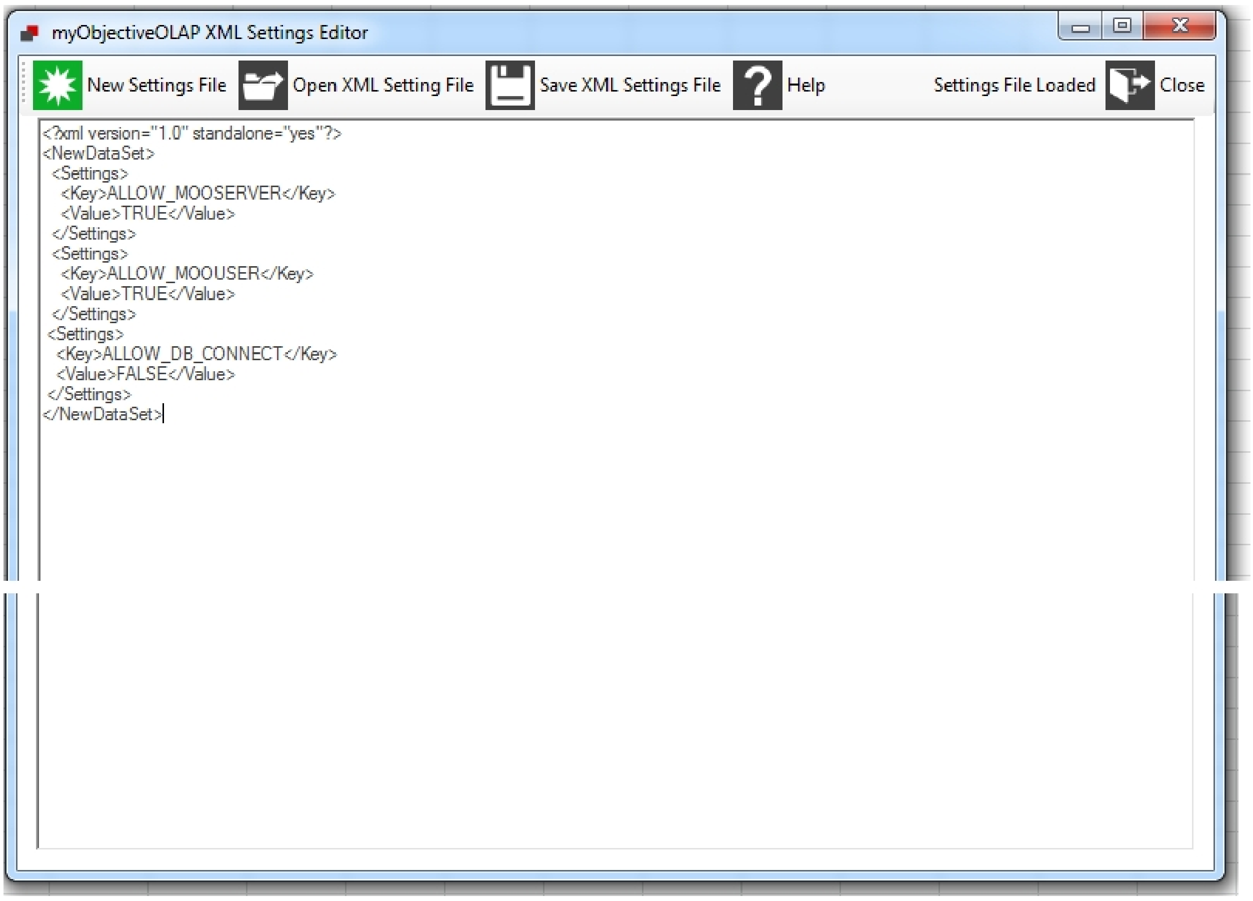
Please see mooApplicationSettings.xml topic for sample configuration files.

mooApplicationSettings Editor.

When you first install myObjectiveOLAP a menu item "Application Settings Editor" will be created within the myObjectiveOLAP menu group in Excel:



You can use the mooApplicationSettings Editor to create a new mooapplicationSettings.xml file or amend an existing one.



```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>ALLOW_MOOSERVER</Key>
    <Value>TRUE</Value>
  </Settings>
  <Settings>
    <Key>ALLOW_MOOUSER</Key>
    <Value>TRUE</Value>
  </Settings>
 <Settings>
   <Key>ALLOW_DB_CONNECT</Key>
   <Value>FALSE</Value>
 </Settings>
</NewDataSet>
```

Restricting access to the mooApplicationSettings Editor

User access to edit the mooApplicationSettings editor can be disabled, please see Restricting access to users

Example path

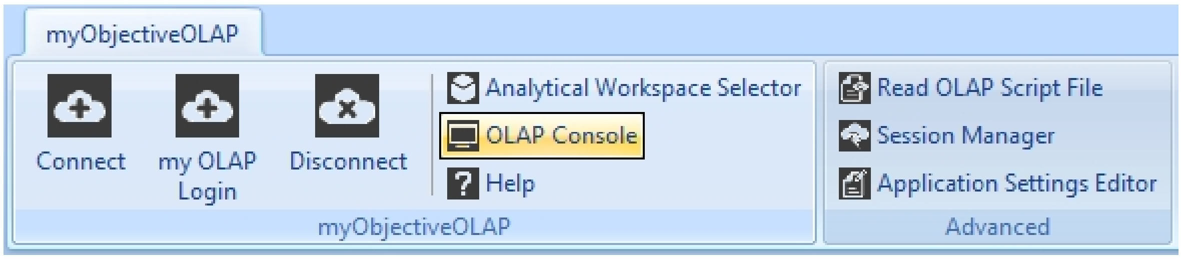
```
C:\Users\{username}\AppData\Local
```

OLAP Console

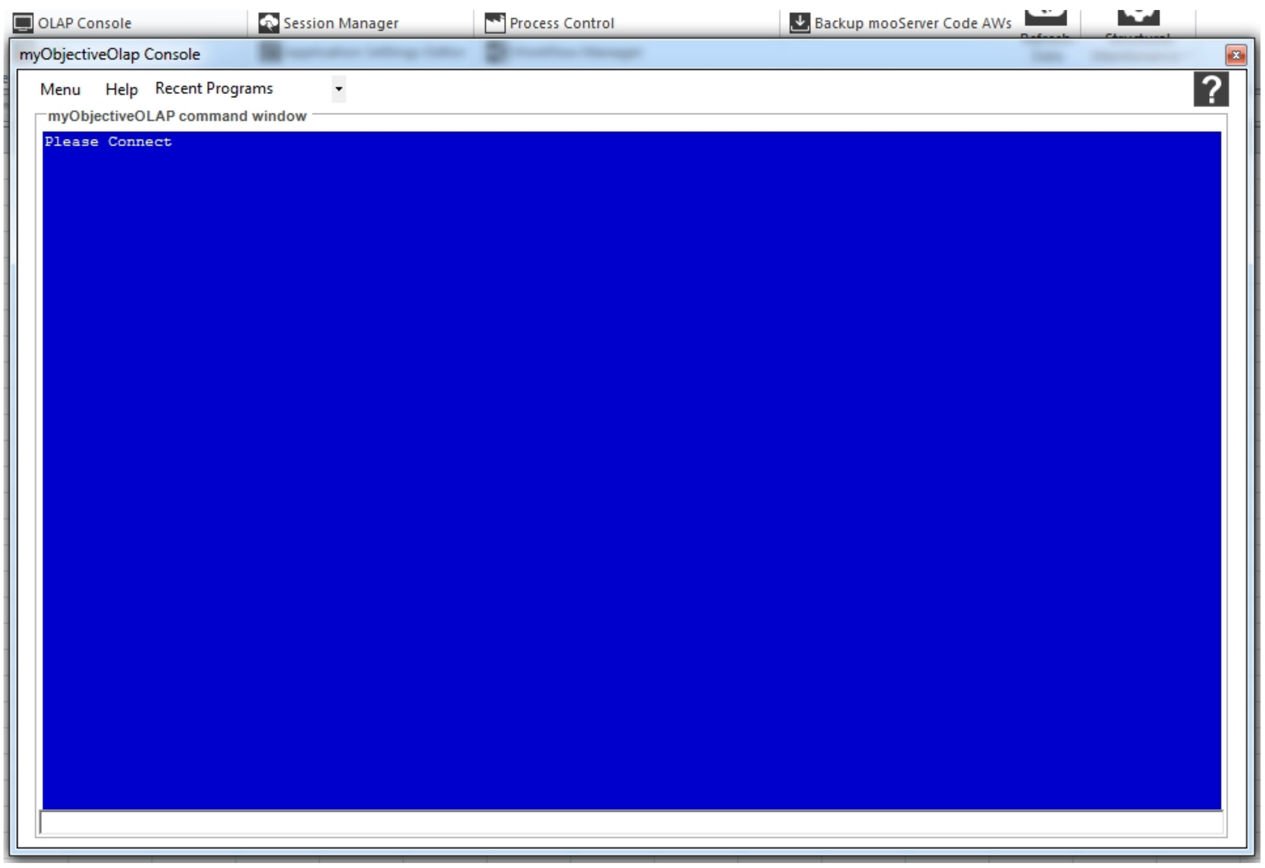
OLAP Console

The OLAP Console is found within the main myObjectiveOLAP menu group.

The OLAP Console enables a user or developer to execute Oracle OLAP DML directly within the database. It also enables the editing of OLAP programs.



Use of the OLAP Console.

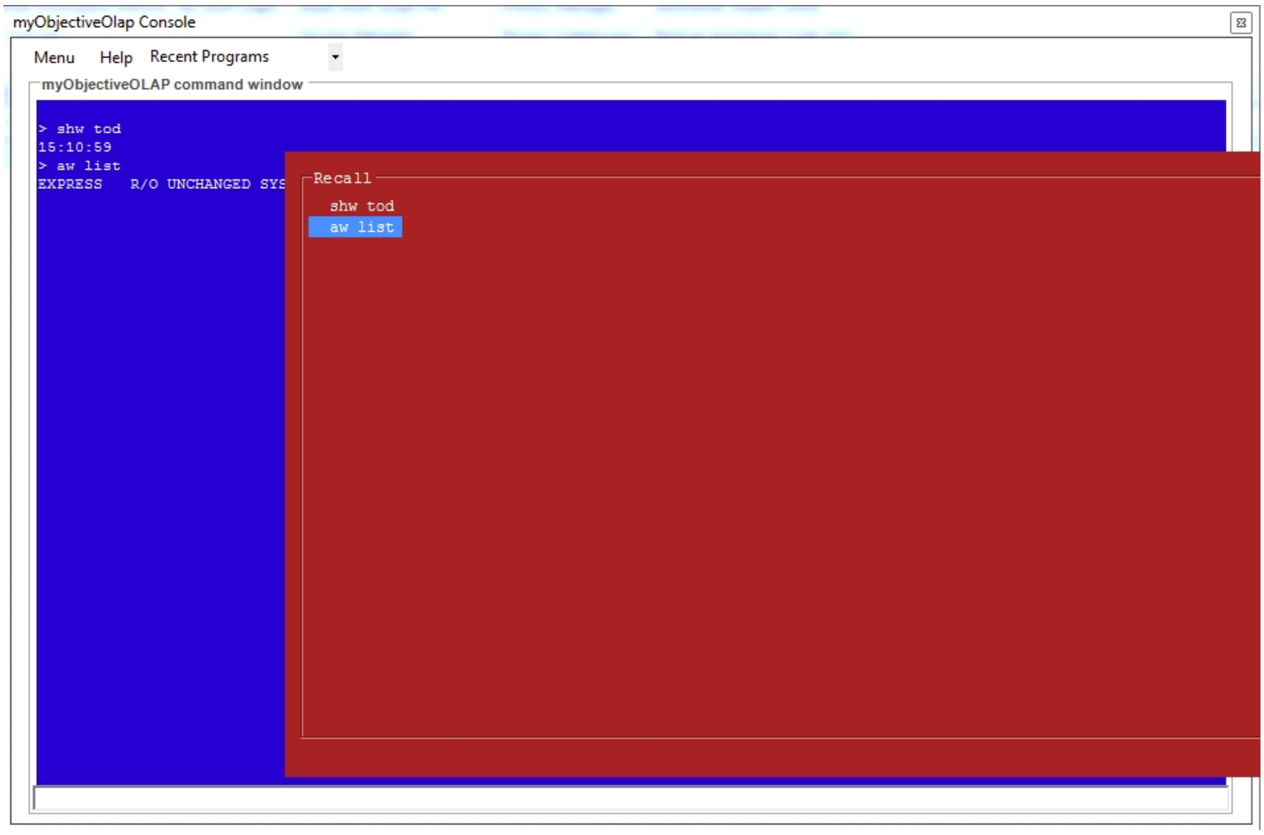


```
Please Connect
```

Commands are entered in the Command Entry Window.

Use of the OLAP Console.

Commands entered into the command entry window are stored for the current session in the command recall window:



```
> shw tod
15:10:59
> aw list
EXPRESS   R/O UNCHANGED SYS
```

```
Recall
  shw tod
  aw list
```

The following keys are used with the Recall window:

```
      F2                        Opens the Recall window.
      Cursor keys (up/down)     Navigate up and down the list of recalled
commands.
      ENTER                     Flag a specific command as to executed again.
Un-flag commands previously selected.
      F10                       Close the recall window and execute commands
flagged.
                                Commands are executed in order of recall NOT
selection.
      ESCAPE                    Close the recall window do not execute any
commands flagged.
```

Commands are split into three types:

Oracle DML Statements

Oracle DML statements can be entered and output from the Oracle OLAP engine viewed.

Warning - Unlike legacy XCA or SNAPI protocols, Oracle will attempt to display potentially very large volumes of data.
Ensure you are sure of the status of any variable before executing a report (rpr) DML statement.

MOO Script

Entering "MOO HELP↵" in the command window lists a series of MOO commands.

```
MooScript HELP, Warning this is not supported
----------------------------------------------
disconOlap (0)      --  Disconnects Oracle
conOlap             --  Reads Saved XML and Connects to Oracle
CLS                 --  Clears the screen
Debug               --  Toggles debugging on and off
Scroll              --  Scrolls Down
ALL_AWS             --  Diagnostics on AWs Available
```

To execute any of the commands above prefix them with MOO:

Example:

```
MOO CONOLAP
```

Would read a pre-saved settings.xml file and connect to Oracle OLAP.

Local Commands

LOTF

Issuing an LOTF [LOCAL_PATH\File] command will direct all future output to the designated file. Local outfile can be disabled by issuing a LOTF EOF (End Of File) statement.

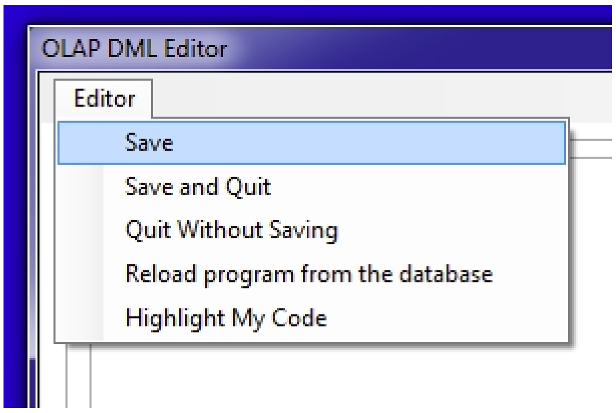
Example:

```
LOTF c:\myObjectiveOLAP.txt
shw tod
LOTF EOF
```

Edit (edt)

Issuing "edit [program_name]" in the command entry window will open the OLAP DML Editor window populated with the code for the program passed as an argument.

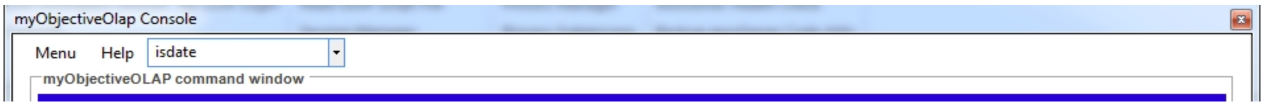
A number of menu items are available to you in the OLAP DML Editor window



```
Save                                --  Saves any changes made to the current
program back to the Analytic Workspace
Save and Quit                       --  Saves any changes made to the current
program back to the Analytic Workspace
                                        and closes the OLAP DML Editor window.
Quit without Saving                 --  Close the current OLAP DML Editor window,
changes made to the current program

                                        are not saved back to the Analytic

Workspace.
Reload program from the database --  Reloads the currently edited program from
the Analytic Workspace

                                        , changes are discarded.
Highlight My Code                   --  Color highlights:  Comments, Commands,
Functions.
```

**Warning* - Save and Save and Quit, does not change the attached mode of your Analytic Workspace, if you are Read Only your program will be saved back to the AW but will not be Read Write saved.*
**Warning* - Save and Save and Quit, does not execute an "upd; commit" you are responsible with permanently updating your Analytic Workspace.*

Previously edited program names are stored in the menu drop down box. Selecting a program from the drop down menu causes the program editor to open the program.



Notes

Commands executed through the OLAP Console can be seen by the Oracle OLAP recap command.

Restricting access to the OLAP Console.

Access to the OLAP Console can be restricted by creating an ALLOW_OLAP_CONSOLE settings key within the mooApplicationSettings.xml file and flagging the key as FALSE.

The following mooApplicationSettings.xml example would disable the OLAP Console. By default the OLAP Console is enabled.

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Settings>
    <Key>ALLOW_OLAP_CONSOLE</Key>
    <Value>FALSE</Value>
  </Settings>
 </NewDataSet>
```

Oracle OLAP DML Editor

Oracle OLAP DML Editor

From within the myObjectiveOLAP OLAP Console you can edit Oracle OLAP objects such as:

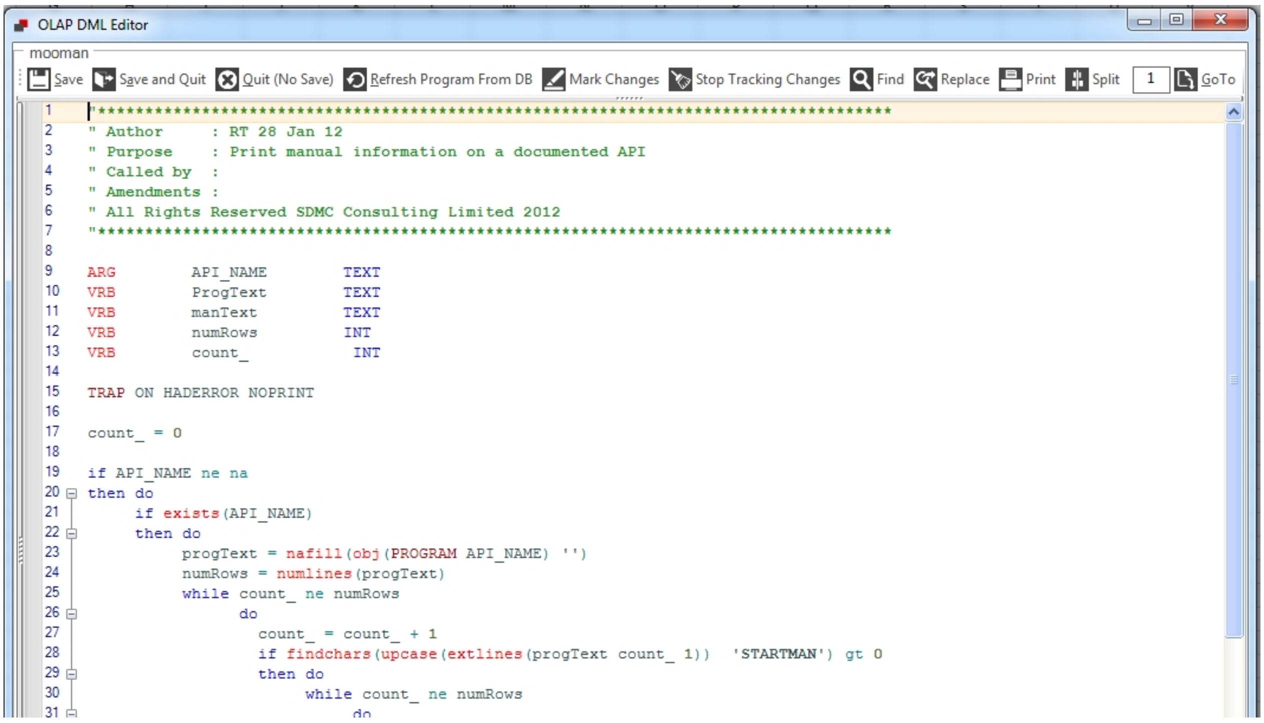
- `Programs`
- `Aggmaps`
- `Models`
- `Single cell variables`

Two commands exist in order to initialise the Editor window:

OLAP Console Command	Edited Object
EDIT [EDT]	● Programs ● Aggmaps ● Models
EDITV [EDTV]	Single Cell Variables

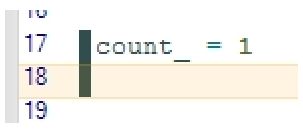
Editor

myObjectiveOLAP includes a modern, fast and feature rich editor for manipulating and creating the Oracle OLAP object types above. It is able to open and syntax highlight Oracle OLAP DML programs, thousands of lines long in less than a second.



```
"*************************************************************************************
" Author      : RT 28 Jan 12
" Purpose     : Print manual information on a documented API
" Called by   :
" Amendments  :
" All Rights Reserved SDMC Consulting Limited 2012
"*************************************************************************************

ARG        API_NAME        TEXT
VRB        ProgText        TEXT
VRB        manText         TEXT
VRB        numRows         INT
VRB        count_           INT

TRAP ON HADERROR NOPRINT

count_ = 0

if API_NAME ne na
then do
     if exists(API_NAME)
     then do
          progText = nafill(obj(PROGRAM API_NAME) '')
          numRows = numlines(progText)
          while count_ ne numRows
                do
                  count_ = count_ + 1
                  if findchars(upcase(extlines(progText count_ 1))  'STARTMAN') gt 0
                  then do
                       while count_ ne numRows
                            do
```

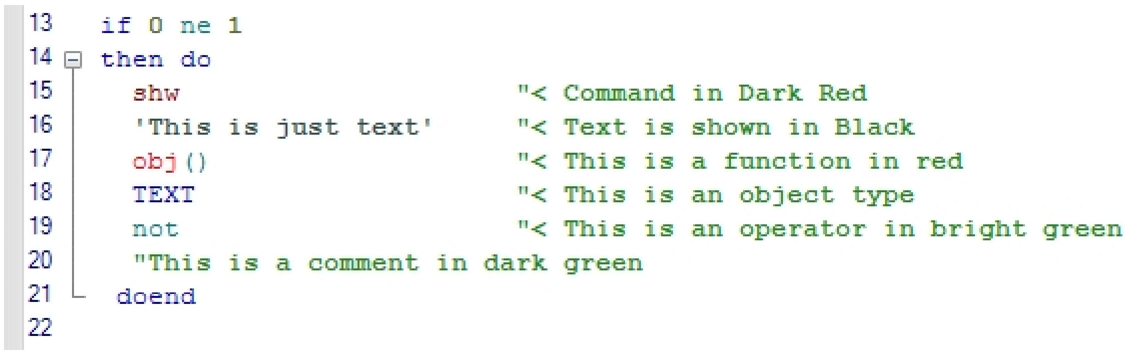
Functionality Overview

Ribbon Menu Option	Purpose
mooman	Name of the object being edited
Save	Saves the contents of the edited object back to the Oracle OLAP Analytic Workspace
Save and Quit	Saves the contents of the edited object back to the Oracle OLAP Analytic Workspace, and then exits the editor
Quit (No Save)	Exits the editor without saving any changes back to the Oracle OLAP Analytic Workspace
Refresh Program From DB	Refreshes the last saved version of the object back into the editor
Mark Changes	Marks all rows that are changed from that point forward:  17 count_ = 1 18 19
Stop Tracking Changes	Stops tracking changes to the object
Find	Starts the Find tool. See below for more information.

Replace	Starts the Replace tool. See below for more information
Print	Starts the code printing engine
Split	Enables or disables the code window splitter
1 GoTo	Goto a specific row number within your code

Code Formatting

As you can see below the myObjectiveOLAP code Editor automatically formats as you type based on the Oracle OLAP lexicon.

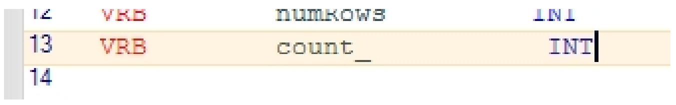


```
if 0 ne 1
then do
   shw                          "< Command in Dark Red
   'This is just text'          "< Text is shown in Black
   obj()                        "< This is a function in red
   TEXT                         "< This is an object type
   not                          "< This is an operator in bright green
   "This is a comment in dark green
  doend
```

Row Numbers, Current and Highlighted Rows

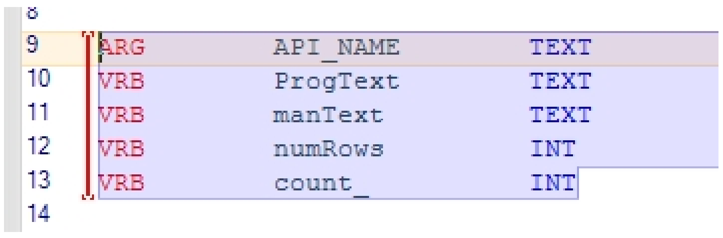
myObjectiveOLAP adds row numbers to the editor window to aide navigation.

The row you are currently editing is highlighted as below:



```
VRB         numRows          INT
VRB         count_            INT
```

If you select multiple rows this is highlighted thus;



```
ARG         API_NAME         TEXT
VRB         ProgText         TEXT
VRB         manText          TEXT
VRB         numRows          INT
VRB         count_           INT
```

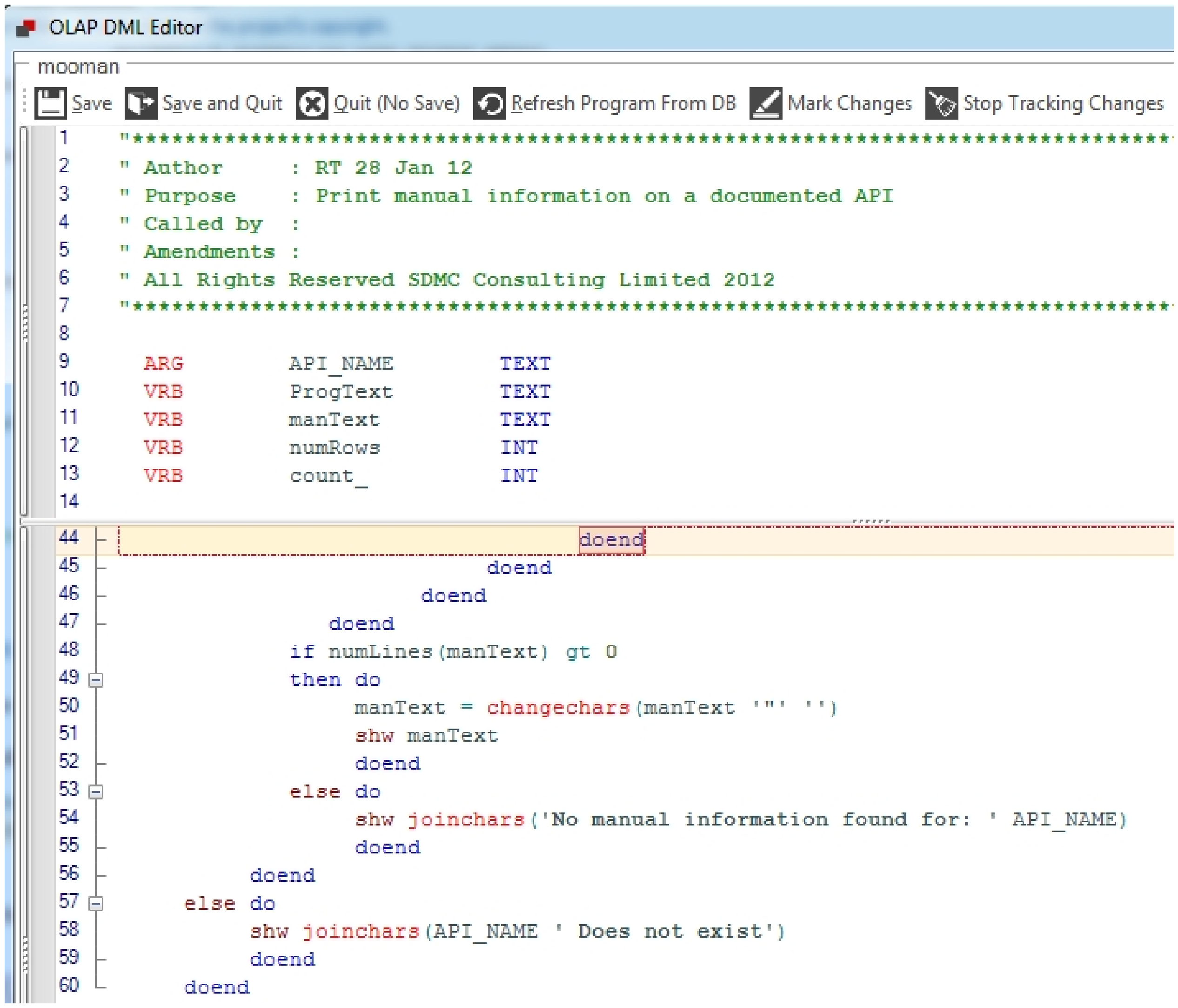
With the code highlighted and a red vertical bar on the left hand-side.

Code Splitter Tool

It can often be useful when editing large programs to be able to split your code window, the myObjectiveOLAP DML editor supports this functionality:

At the top of the Editor window you will see 5 small dots:

Grab the dots with your mouse and pull the pane down:



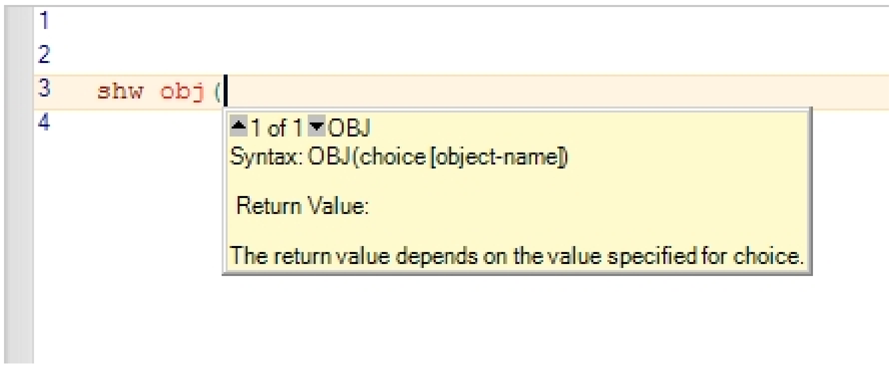
```
1   "**********************************************************************
2   " Author      : RT 28 Jan 12
3   " Purpose     : Print manual information on a documented API
4   " Called by   :
5   " Amendments  :
6   " All Rights Reserved SDMC Consulting Limited 2012
7   "**********************************************************************
8
9     ARG         API_NAME        TEXT
10    VRB         ProgText        TEXT
11    VRB         manText         TEXT
12    VRB         numRows         INT
13    VRB         count_          INT
14
44                                                  doend
45                                             doend
46                                        doend
47                                 doend
48                              if numLines(manText) gt 0
49                              then do
50                                   manText = changechars(manText '"' '')
51                                   shw manText
52                                   doend
53                              else do
54                                   shw joinchars('No manual information found for: ' API_NAME)
55                                   doend
56                        doend
57                   else do
58                        shw joinchars(API_NAME ' Does not exist')
59                        doend
60                   doend
```

As you can see from above we are able to see our variable declarations (rows 9 -- 14) but then in the lower pane we have been able to scroll further down in the code.

Quick Info

Oracle OLAP DML is one of the richest programming languages available.   The myObjectiveOLAP editor helps you remember the subtleties of the language by prompting you with syntax when it recognizes a key DML phrase. This is designed to enable you to utilize Oracle OLAP quicker, by enabling faster coding with fewer mistakes.

Examples of this feature is shown below.

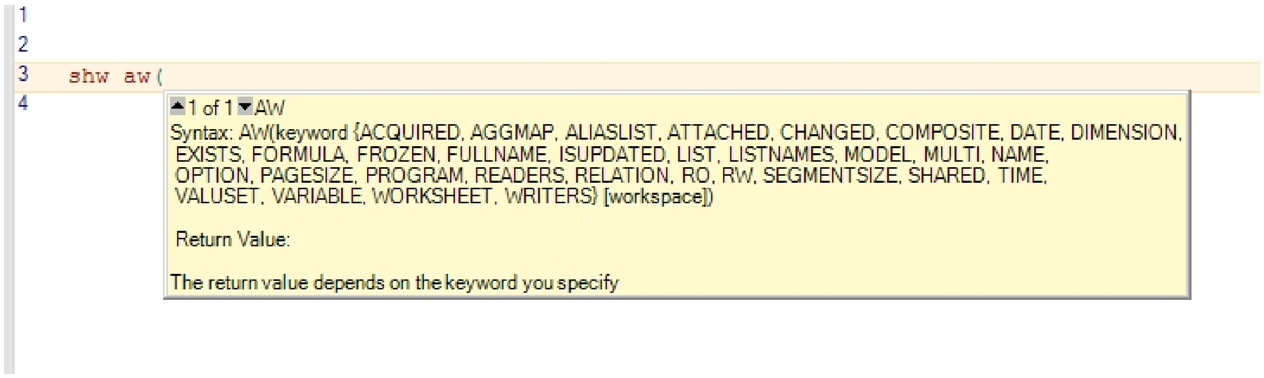


```
shw obj (
```

▲1 of 1▼OBJ
Syntax: OBJ(choice [object-name])

Return Value:

The return value depends on the value specified for choice.



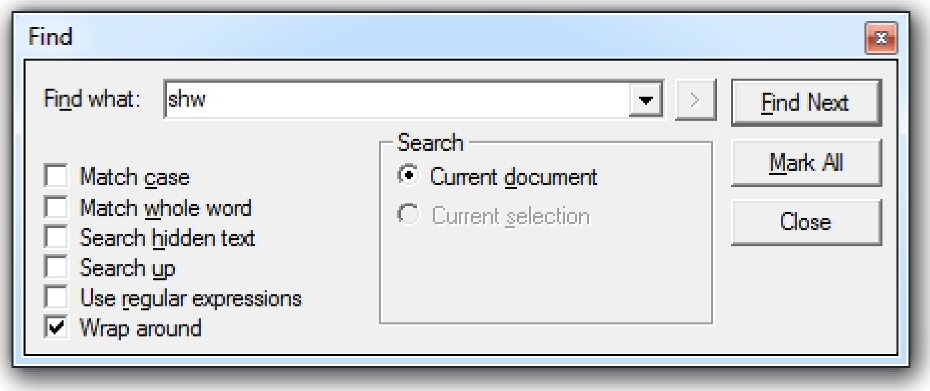
```
shw aw (
```

▲1 of 1▼AW
Syntax: AW(keyword {ACQUIRED, AGGMAP, ALIASLIST, ATTACHED, CHANGED, COMPOSITE, DATE, DIMENSION, EXISTS, FORMULA, FROZEN, FULLNAME, ISUPDATED, LIST, LISTNAMES, MODEL, MULTI, NAME, OPTION, PAGESIZE, PROGRAM, READERS, RELATION, RO, RW, SEGMENTSIZE, SHARED, TIME, VALUSET, VARIABLE, WORKSHEET, WRITERS} [workspace])

Return Value:

The return value depends on the keyword you specify

Find and Replace Tools



Find

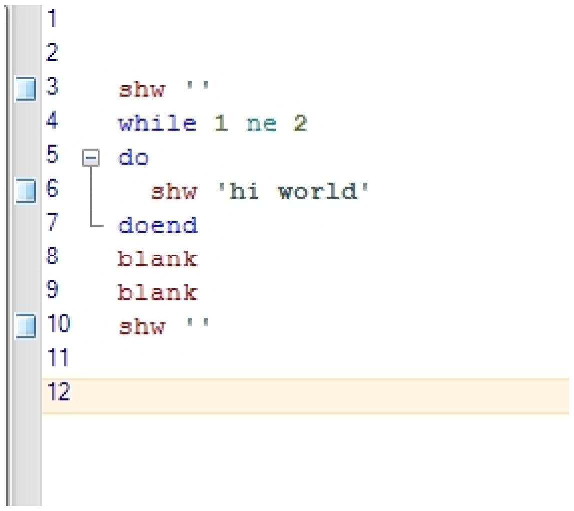
Find what: shw

[ ] Match case
[ ] Match whole word
[ ] Search hidden text
[ ] Search up
[ ] Use regular expressions
[✓] Wrap around

Search
[● Current document]
○ Current selection

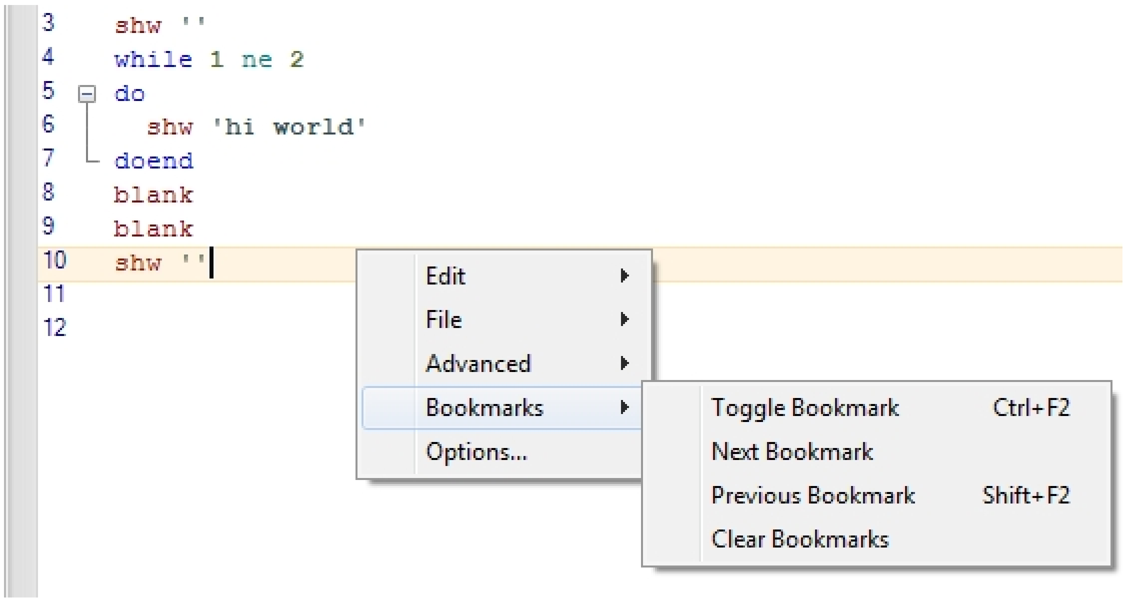
The Find tool enables fast searching for strings within an Editor window, as well as finding it also can Mark all occurrences of a specific search criteria.

In the following example we have searched for "shw" and the Editor window has now displayed a blue cube against the row where the criteria has been found:



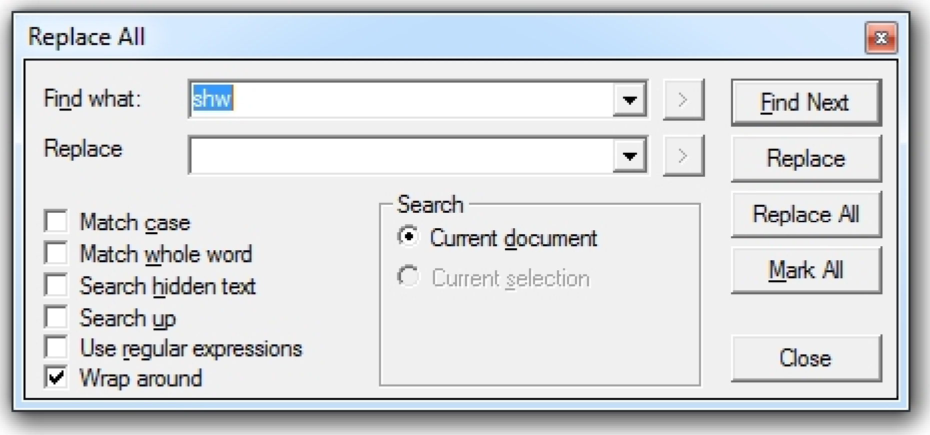
```


shw ''
while 1 ne 2
do
  shw 'hi world'
doend
blank
blank
shw ''


```

To clear our bookmarks we can right-click, and clear then through the Bookmarks --> Clear Bookmarks menu:



```
shw ''
while 1 ne 2
do
  shw 'hi world'
doend
blank
blank
shw ''


```

As you would expect the Editor also contains a fully featured Find and Replace tool which is able to find and replace based on Case, Whole Words or Regular Expressions



Replace All

Find what: shw

Replace:

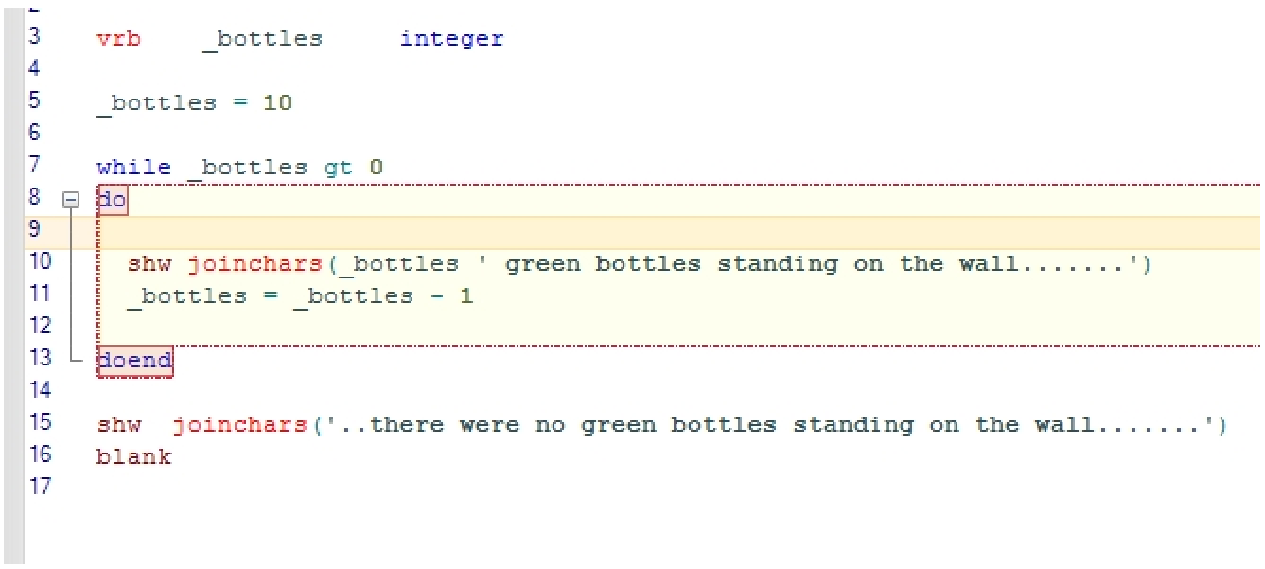
- [ ] Match case
- [ ] Match whole word
- [ ] Search hidden text
- [ ] Search up
- [ ] Use regular expressions
- [x] Wrap around

Search: [✓ Current document] Current selection

Auto outlining

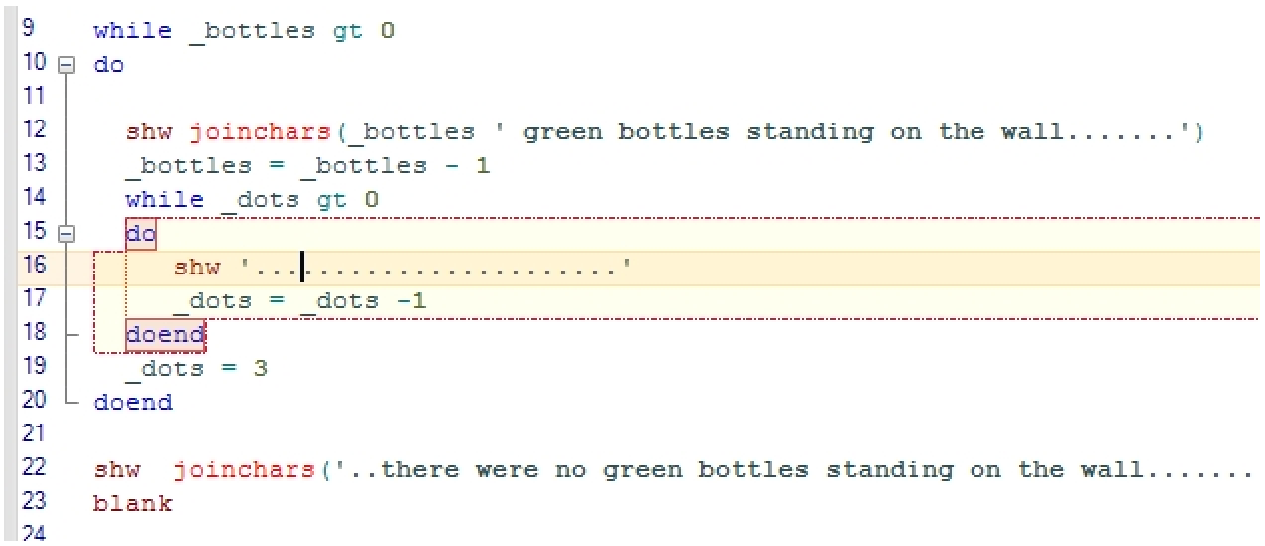
Whilst editing code it is important to quickly and easily be able to see the start and end of "While" "For" "Then Do/Doend". The Editor makes this easy by showing you which loop you are working within at any time:

In the following example we can see immediately the start and end of the "While" loop



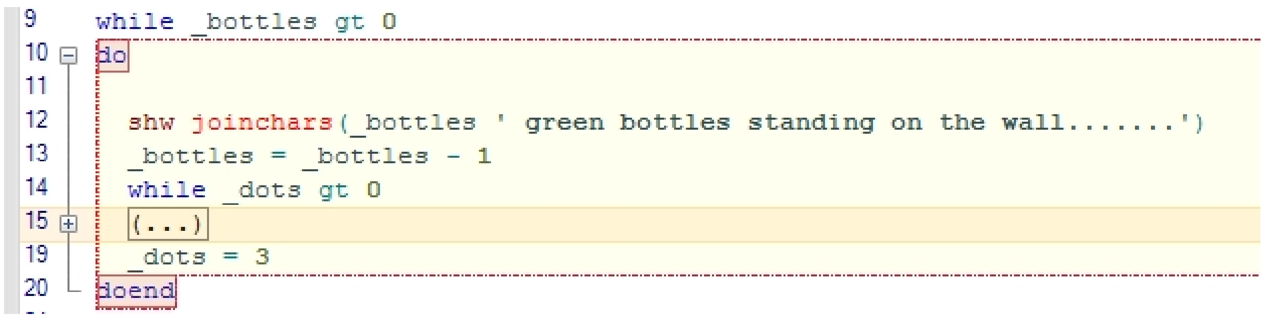
```
vrb     _bottles     integer

_bottles = 10

while _bottles gt 0
do

  shw joinchars(_bottles ' green bottles standing on the wall.......')
  _bottles = _bottles - 1

doend

shw  joinchars('..there were no green bottles standing on the wall.......')
blank
```

The editor handles nested loops with ease as shown below:



```
while _bottles gt 0
do

  shw joinchars(_bottles ' green bottles standing on the wall.......')
  _bottles = _bottles - 1
  while _dots gt 0
  do
     shw '.......................'
     _dots = _dots -1
  doend
  _dots = 3
doend

shw  joinchars('..there were no green bottles standing on the wall.......
blank
```

And we are able to hide the loops we are not interested in, as shown below; by pressing the + icon next to the start of the loop:

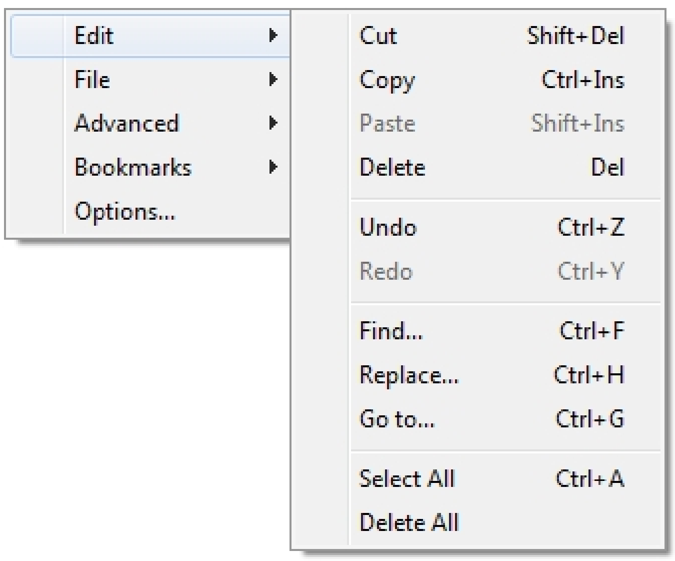


```
while _bottles gt 0
do

  shw joinchars(_bottles ' green bottles standing on the wall.......')
  _bottles = _bottles - 1
  while _dots gt 0
  (...)
  _dots = 3
doend
```

Right Click Options

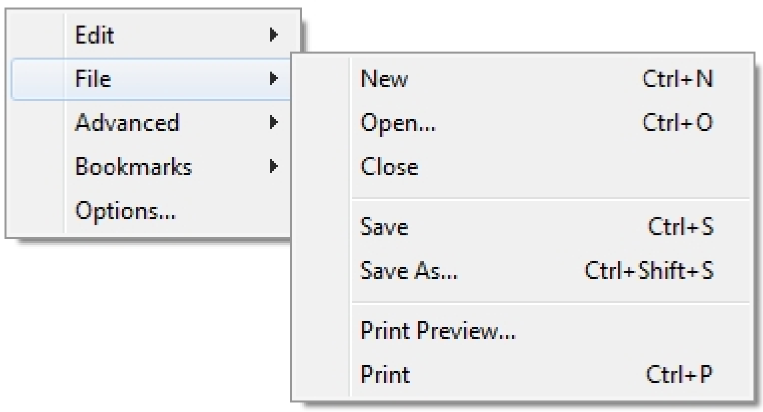
Edit

Standard options for text manipulation can be found under Edit:



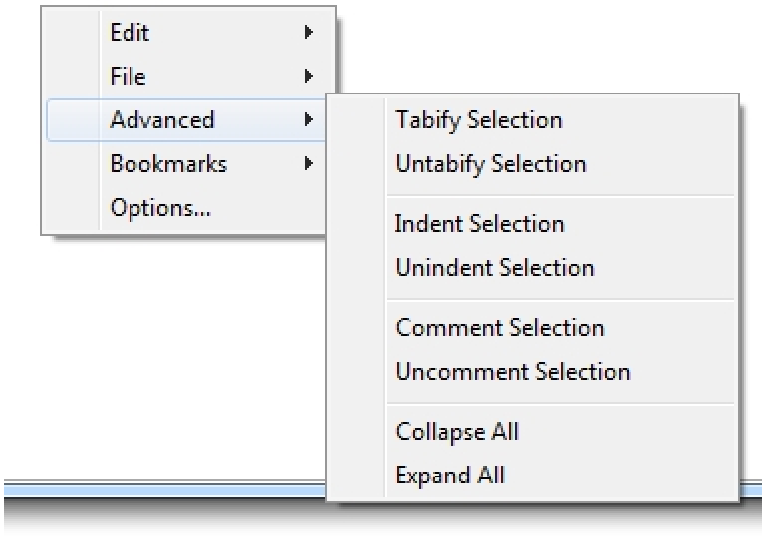
File

You may not always want to save the contents of your Editor to the database, you can Open, Save and Print from the File menu.   Saving and Opening allows you to save to a file on your local file system:



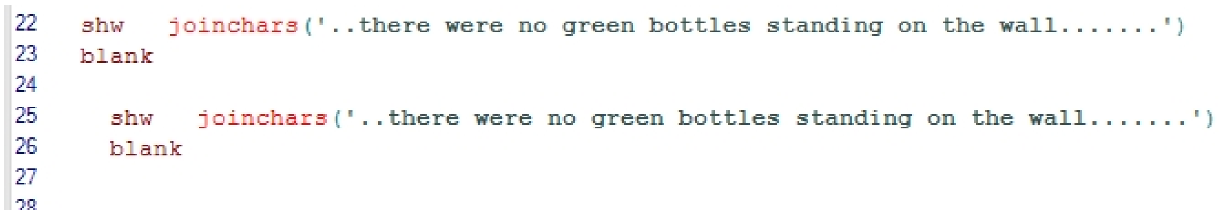
Advanced

Advanced allows you to carry out some tasks on multiple rows of code very quickly.



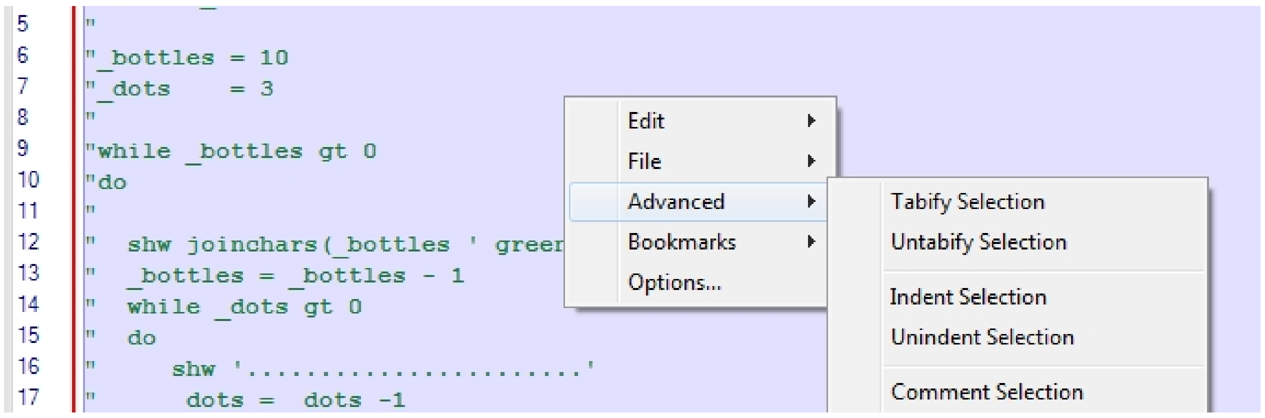
For example:

Indent the last two lines:



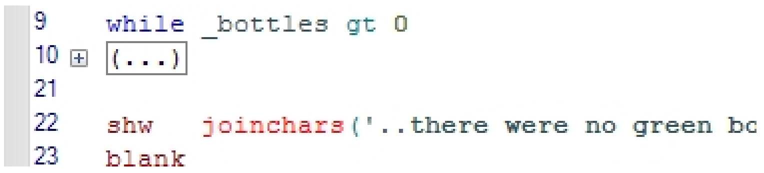
```
shw   joinchars('..there were no green bottles standing on the wall.......')
blank

  shw   joinchars('..there were no green bottles standing on the wall.......')
  blank

```

Comment (or uncomment) a lot of code at once:



```
"
"_bottles = 10
"_dots    = 3
"
"while _bottles gt 0
"do
"
"  shw joinchars(_bottles ' greer
"  _bottles = _bottles - 1
"  while _dots gt 0
"  do
"     shw '.......................'
"      dots =  dots -1
```

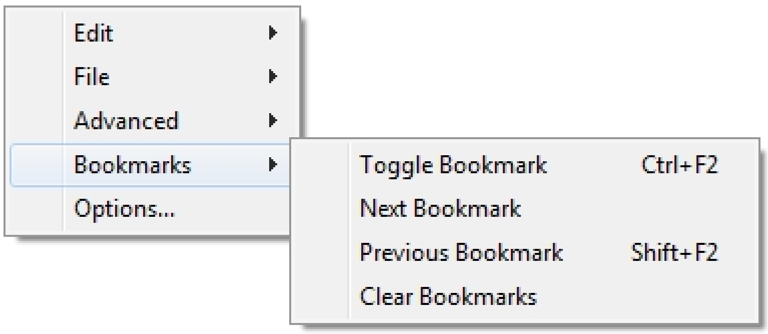
Collapse all your loops:



```
while _bottles gt 0
(...)

shw   joinchars('..there were no green bc
blank
```

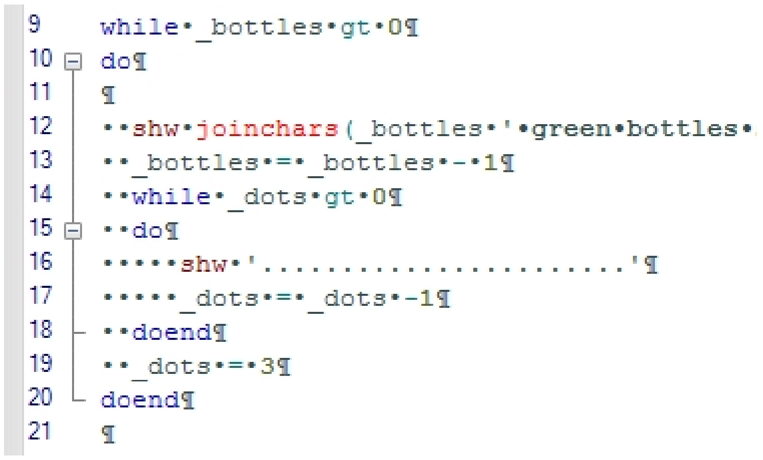
Bookmarks

Bookmarks allow you to manage your bookmarks, either ones put there automatically through the Find tool or manual bookmarks you have added to the code editor via Ctrl+F2



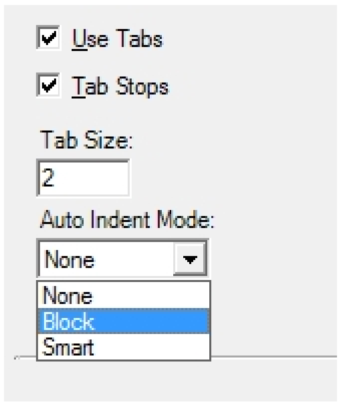
Options

Options allow you to fine tune your editor window, such as View Whitespace:



```
while•_bottles•gt•0¶
do¶
¶
••shw•joinchars(_bottles•'•green•bottles•
••_bottles•=•_bottles•-•1¶
••while•_dots•gt•0¶
••do¶
•••••shw•'......................'¶
•••••_dots•=•_dots•-1¶
••doend¶
••_dots•=•3¶
doend¶
¶
```

or to fine-tune the Auto-indent feature:



[✓] Use Tabs

[✓] Tab Stops

Tab Size: 2

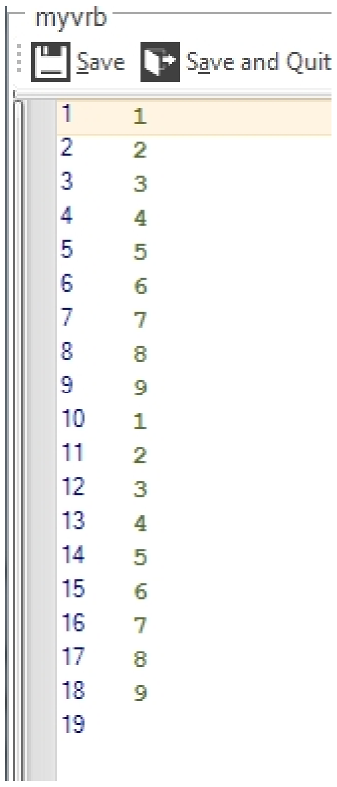
Auto Indent Mode: None

EditorV

EDTV is used to edit variables as shown below:

```
> dfn myvrb vrb text

> myvrb = NA

> edtv myvrb
```

```
Enter some text and Save and Quit:
```



```
1
2
3
4
5
6
7
8
9
1
2
3
4
5
6
7
8
9
```

```
> shw myvrb
1
2
3
4
5
6
7
8
9
1
2
3
4
5
6
7
8
9
```

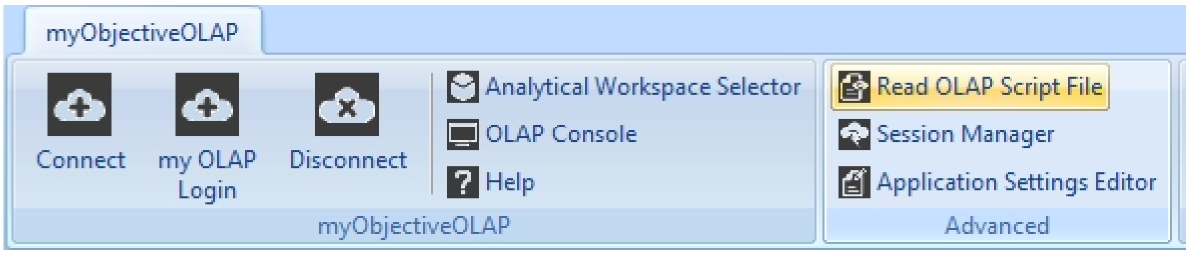
Read OLAP Script File

Read OLAP Script File

The Read OLAP Script File menu is found within the Advanced myObjectiveOLAP menu group.

Read OLAP Script File enables a user, developer or DBA to execute a list of OLAP DML statements from a text file stored locally on the users PC.

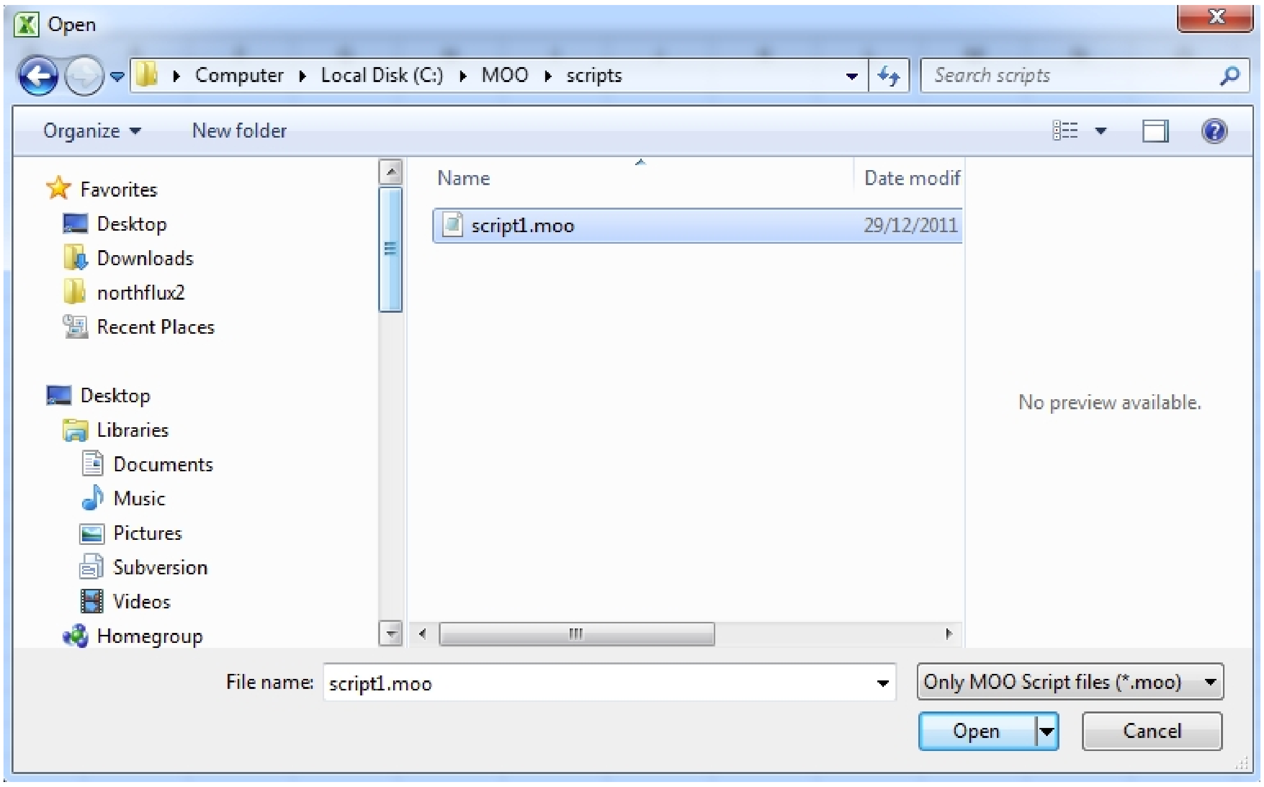
Files should be stored with a .moo extension.



Use of Read OLAP Script File.

Selecting the Read OLAP Script File menu item opens the File Open dialog window.

Select a .moo file and press Open.



script1.moo 29/12/2011

File name: script1.moo

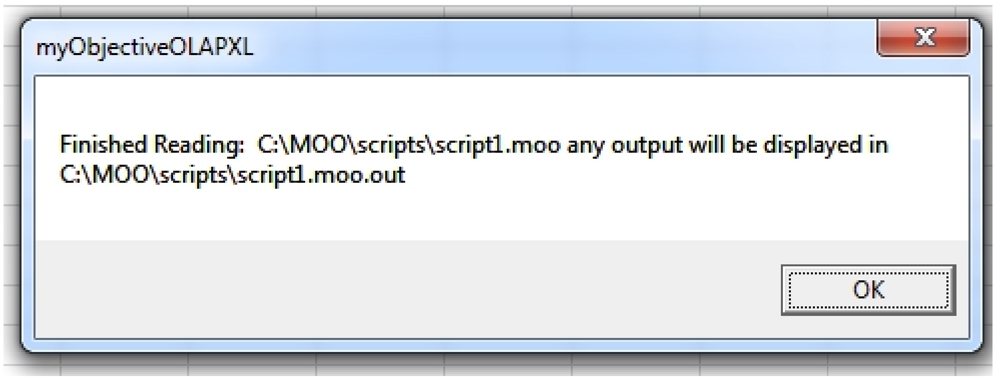
Only MOO Script files (*.moo)

In this example our script1.moo file contains the following:

```
shw tod
aw list
aw attach express ro first
shw lmt(name to first 10)
```

All commands are sent to the myObjectiveOLAP Command Preprocessor and executed against Oracle OLAP.

On completion the following dialog box is presented.



Finished Reading: C:\MOO\scripts\script1.moo any output will be displayed in C:\MOO\scripts\script1.moo.out

Output from the Oracle OLAP engine is saved in a filename.moo.out file in the same directory as the source script.

```
      shw tod
      16:00:33
      aw list
      EXPRESS    R/O UNCHANGED SYS.EXPRESS
      aw attach express ro first

      shw lmt(name to first 10)

_XLTID_XLTABLE_SHADOW_LNTYPEPRGTRACEBADLINE_DUMPSYNTAX_DUMPCODEINF_STOP_ON_ERR
_OBJECTPROTECT
```

Warning - myObjectiveOLAP sends each line of a source filename.moo file to the myObjectiveOLAP preprocessor for execution in the Oracle OLAP environment.
Processing of the file does not stop even if there is an error in the source DML.

You should review the filename.moo.out file before issuing any update; commit statements.
Test your code throughly before placing update; commit statements in a source filename.moo file.

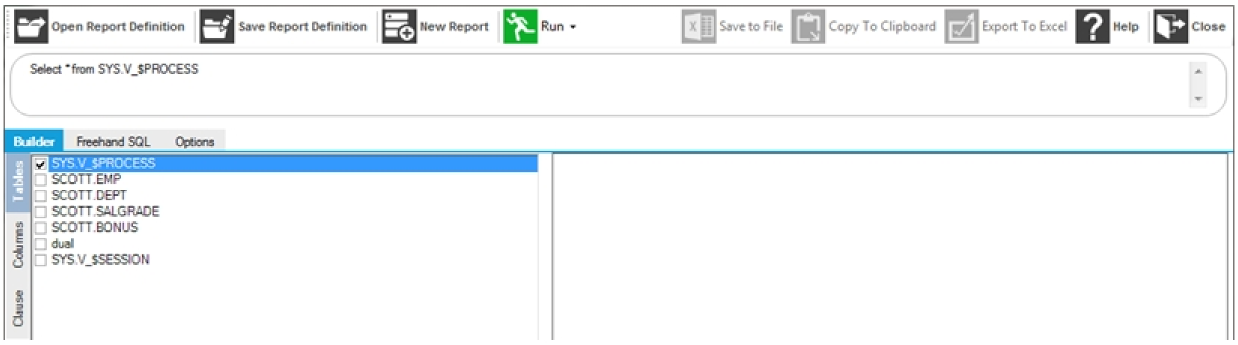
Relational Explorer

Relational Explorer

Relational Explorer

Relational Reporter allows you to build a query which will extract data from a Table or View from within your myObjectiveOLAP enabled Oracle Database.

The Builder topic shows you how to do this using the graphical tools, and the Freehand SQL topic will show you how to do the same thing using SQL (Structured Query Language).



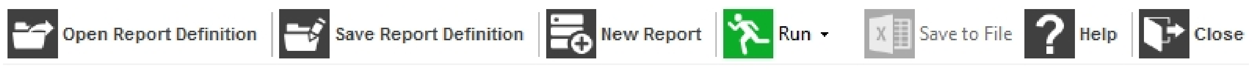
```
Select * from SYS.V_$PROCESS
```

- [✓] SYS.V_$PROCESS
- SCOTT.EMP
- SCOTT.DEPT
- SCOTT.SALGRADE
- SCOTT.BONUS
- dual
- SYS.V_$SESSION

Using Relational Explorer

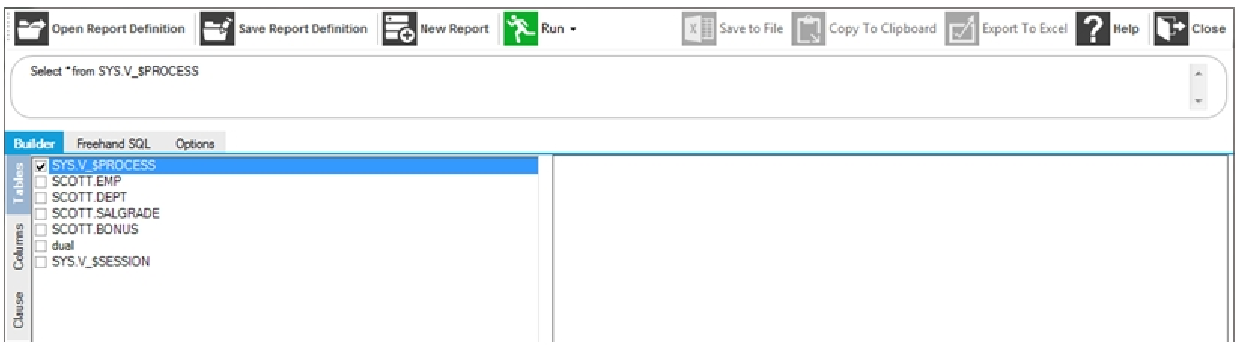
Using Relational Explorer

Ribbon Menu



Ribbon Menu Options	Purpose
Open Report Definition	Open a previously saved report definition
Save Report Definition	Save the current query as a report definition
New Report	Discard the current query and start a new one
Run	Run the current query
Save to File	Save the report data to a file in Excel format
Help	Get Help information
Close	Exit to myObjectiveOLAP

Relational Explorer initially opens with a New Report



```
Select * from SYS.V_$PROCESS
```

[✓] SYS.V_$PROCESS
SCOTT.EMP
SCOTT.DEPT
SCOTT.SALGRADE
SCOTT.BONUS
dual
SYS.V_$SESSION

Immediately below the Ribbon menu is a panel showing a SQL query (Select …). This will be dynamically generated as you work. SQL is covered in more detail in the Freehand SQL topic.

Below the SQL panel are two large panels. The left panel allows you to select your data for the report. The right panel is a work area known as the Canvas, in which you can build your query.

The main tool is broken into four work areas:

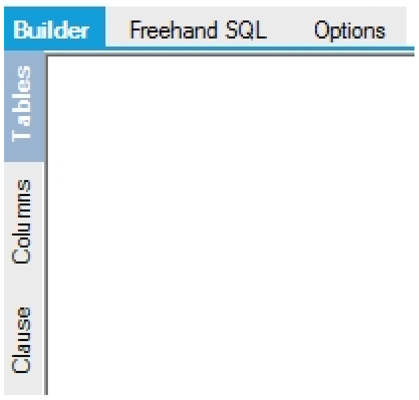
Work Area	Purpose
Builder	Provides a graphical tool for building your report.
Freehand SQL	Allows you to type in your query directly, using SQL.
Options	Allows you to include header and footer information.
My Report	Displays the report after you click Run

Builder

Builder

Builder is a graphical environment to aide you in creating reports without requiring any prior knowledge of SQL or relational databases.

When you select the Builder panel from the menu, your Selection Work Area is further divided into sub panels:

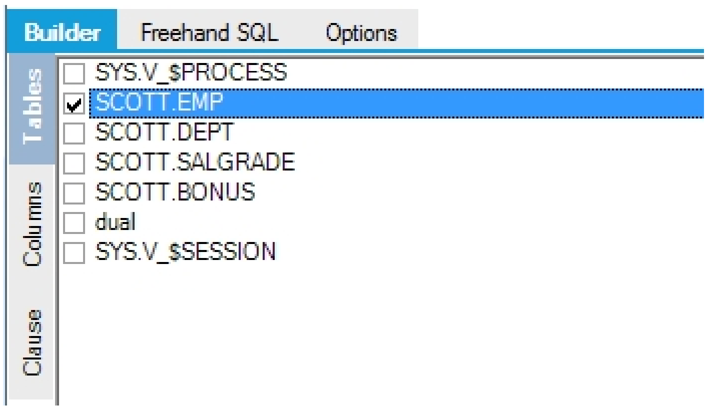


Panel	Purpose
Tables	Select a single table from which to read data
Columns	Select one or many columns of data from the selected table. To select columns, drag and drop them individually into the Canvas area to the right.
Clauses	Provide tools to refine your selection

Tables

Tables are used to store your data in specific logical areas. A table consists of Columns and Rows of data. Columns may contain different types of data, such as text, numbers and dates, and this affects the kinds of selections you can perform.

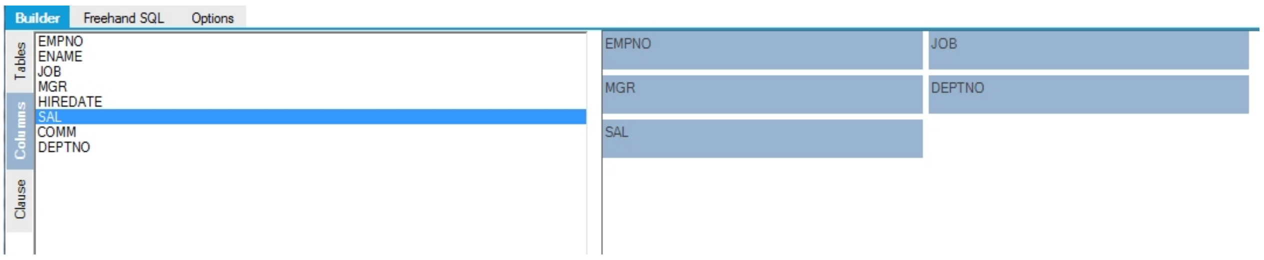
Your report will consist of data from one Table. Select a table by checking the appropriate box.



Columns

The Columns panel allows you to select columns in the table for displaying as columns in your report. Use drag and drop to pull selected columns into the Canvas area to the right.

When you have selected columns, you can rearrange the order of your selection within the Canvas area by dragging and dropping.



To remove a column, drag it to the bin symbol below the Canvas.

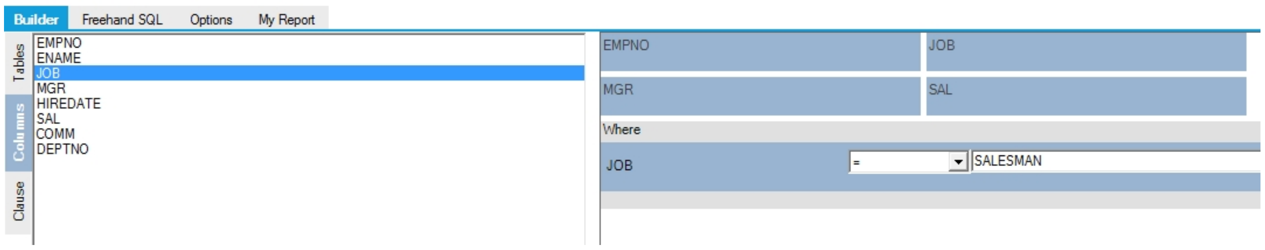
Clause

Clause allows you to further refine the appearance of your report. There are 3 choices available.

- 'Where' is the main tool for filtering rows from the table, based on the content of a selected column.
- 'Or' allows you to extend a "where" phrase to include selections from more than one column.
- 'Order' allows you to sort the rows of the report based on values in one or more columns.

'Where' allows you to limit your data according to the contents of any column you choose. Choose the 'Where' clause by dragging it into the Canvas pane. Then switch to the Columns tab, choose a column and drag it into the 'Where' part of the Canvas. You will then see two fields appear, a comparison operator drop-down choice field and a free text field.

In the following example, the ‘Equal To' operator is used with the Value of "SALESMAN" for column JOB.



The drop-down field offers these choices (comparison operators):

Value	Operator	Search Field
=	Equal to	Value
In	In the list	List of values
>	Greater than	Numerical value
<	Less than	Numerical value
<>	Not equal to	Numerical value
Not in	Is not in the list	List of values
!=	Not equal to	Value
Like	Matches a search string	Search string
Is null	Is an empty field	
Is not null	Is not an empty field	

To add a further selection, select the Clause sub-tab and drag and drop the 'Or' operator into the 'Where' panel (drop into the grey part of the 'where' box, not the blue part relating to the chosen field). Then choose another column and comparison operator as before. You can have more than one 'Or' phrase in a 'Where' clause.

To sort your report, drag and drop the 'Order' operator into the Canvas. Then in the Columns sub-tab select a column and drag and drop it into the 'Order' box in the Canvas. You can include more than one column in the Order box. The first in the list is the primary sort column.

The value you type in the Search field must match the type of data in the chosen column.

The 'In' operator allows you to select values which match a list. To input the list, use the [Enter] key to put values on separate lines.

The 'Like' operator allows you to find inexact matches by incorporating ‘wild-cards' into your search string.

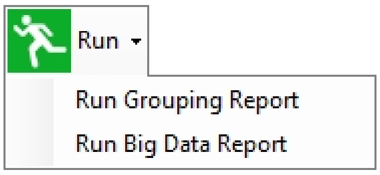
- % matches any group of characters (including none at all).
  For example LIKE Jon% matches Jon, Jones and Jonathan but not John. LIKE %es% matches any text containing the sequence es.
- _ (underscore) matches any single character. For example, Like _123% matches A123456 and B1236 but not 1237.

Less than (<) and Greater than (>) can be used with text columns, and refer to alphabetically before and after.

Viewing your report

Viewing your report

To view the result of the query you have built, use the Run option from the main menu.



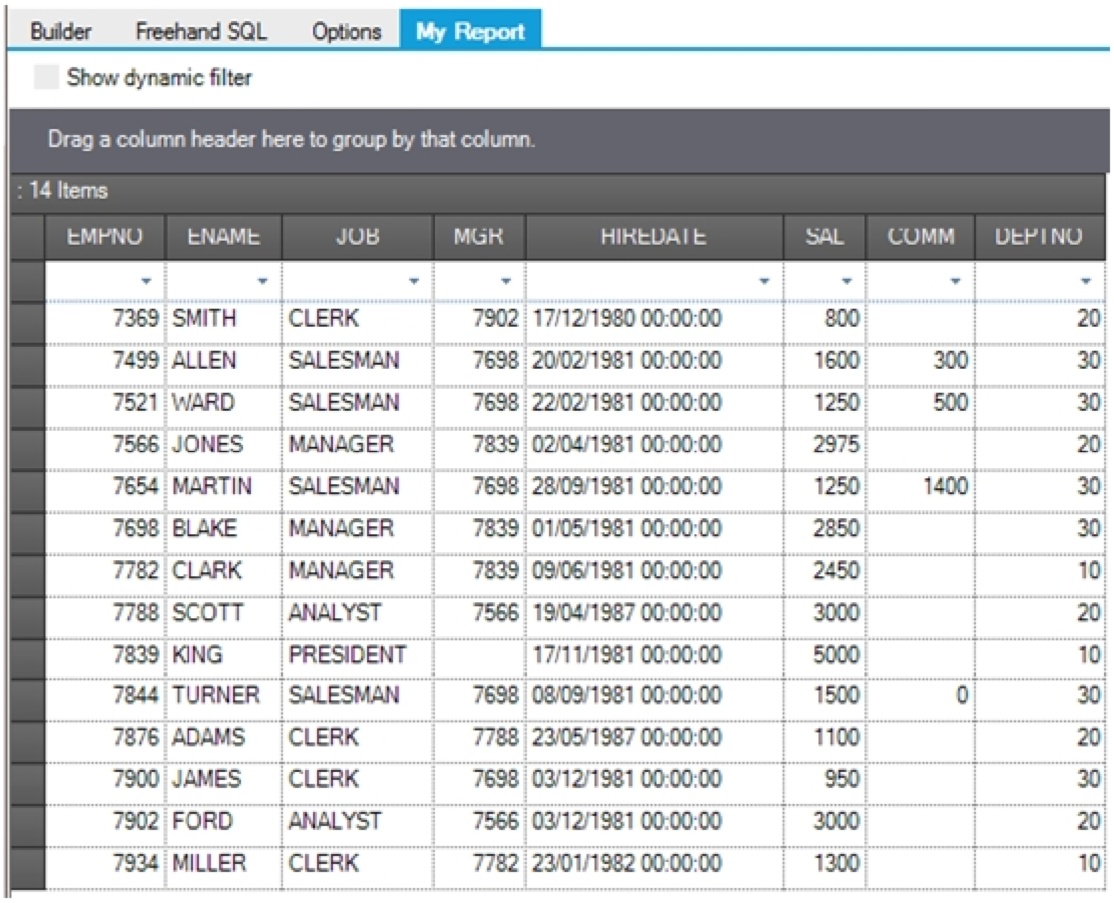
There are two types of report layout available:

- **Grouping Report**
 This provides more flexibility, and allows you to manipulate the report layout when viewing the report.

- **Big Data Report**
 This has less flexibility but is more suitable for large volumes of data.

These are covered in more detail below

The example below is a Grouping report from a table of employee data:



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/1980 00:00:00	800		20
7499	ALLEN	SALESMAN	7698	20/02/1981 00:00:00	1600	300	30
7521	WARD	SALESMAN	7698	22/02/1981 00:00:00	1250	500	30
7566	JONES	MANAGER	7839	02/04/1981 00:00:00	2975		20
7654	MARTIN	SALESMAN	7698	28/09/1981 00:00:00	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/1981 00:00:00	2850		30
7782	CLARK	MANAGER	7839	09/06/1981 00:00:00	2450		10
7788	SCOTT	ANALYST	7566	19/04/1987 00:00:00	3000		20
7839	KING	PRESIDENT		17/11/1981 00:00:00	5000		10
7844	TURNER	SALESMAN	7698	08/09/1981 00:00:00	1500	0	30
7876	ADAMS	CLERK	7788	23/05/1987 00:00:00	1100		20
7900	JAMES	CLERK	7698	03/12/1981 00:00:00	950		30
7902	FORD	ANALYST	7566	03/12/1981 00:00:00	3000		20
7934	MILLER	CLERK	7782	23/01/1982 00:00:00	1300		10

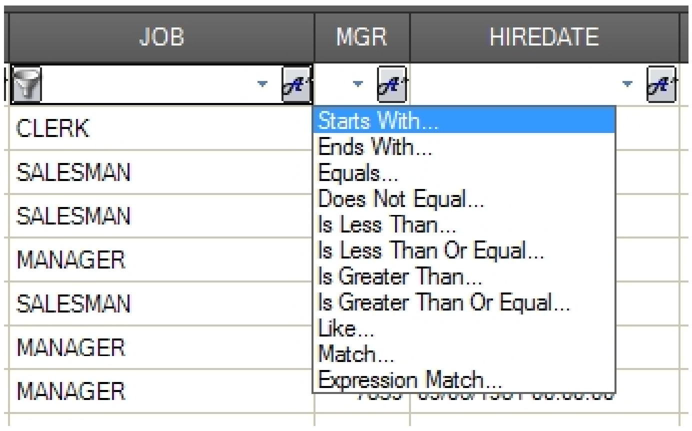
The report is in the **My Report** panel. You can switch between this and the Builder tab by clicking on the panel headings.

You can manipulate the report in the following ways:

- Adjust column widths by dragging the boundaries between column headers.
- Change the order of columns by dragging the column header and dropping it between other

columns.

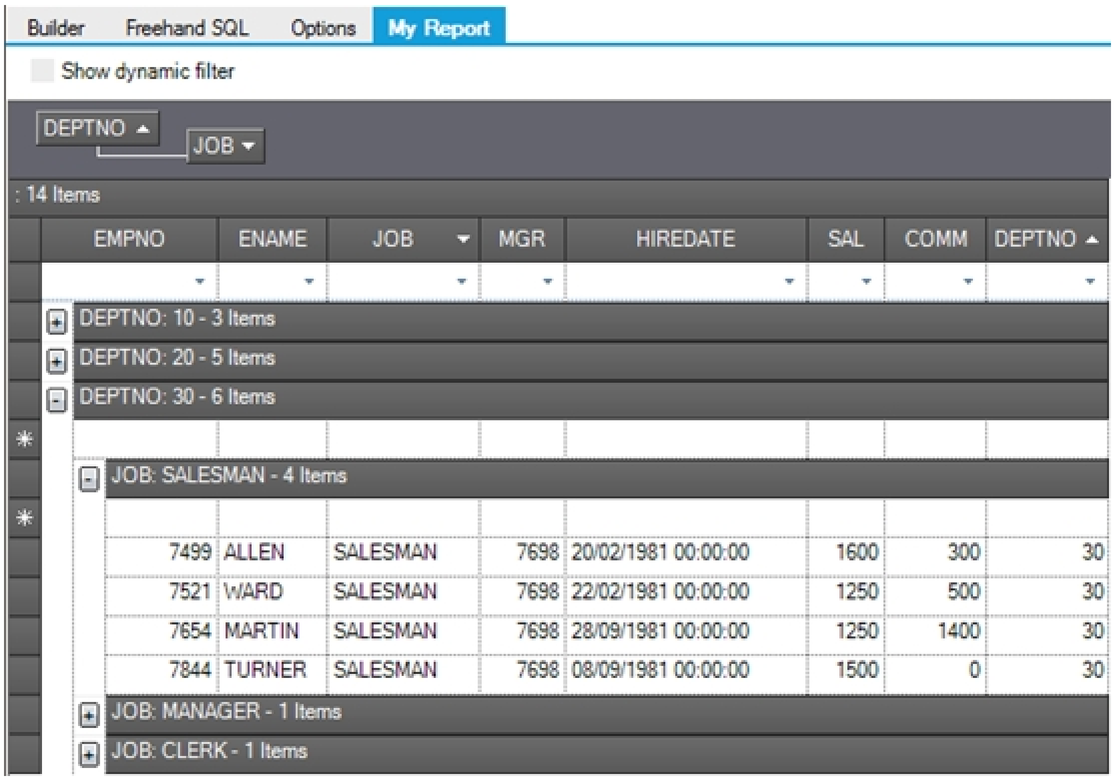
- Sort the data by clicking once on the column header. Click again to sort the data in the reverse order.
- Select data using the drop-down boxes just below the column headers. This allows you to select all occurrences of one instance, including the occurrence of empty fields.
- Check the 'Show Dynamic Filter' box to provide further selection features. These appear as additional filter tools in the column headers.



JOB	MGR	HIREDATE
CLERK		
SALESMAN		
SALESMAN		
MANAGER		
SALESMAN		
MANAGER		
MANAGER		

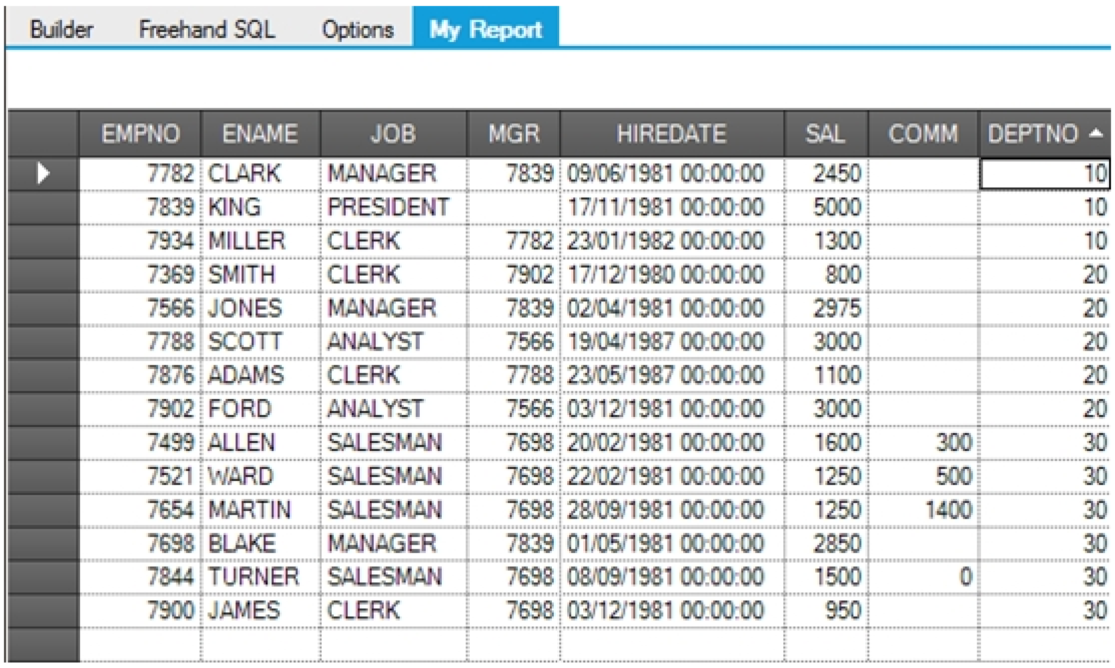
- Drag a column heading into the Grouping box above the column headers. This organises the rows into groups. Click on the [+] symbol by the group to expand it into its constituents.
- You can include more than one group.

The example below shows the employee table grouped by job within Department (DEPTNO). The Salesman job is expanded into its 4 employees.



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
DEPTNO: 10 - 3 Items							
DEPTNO: 20 - 5 Items							
DEPTNO: 30 - 6 Items							
JOB: SALESMAN - 4 Items							
7499	ALLEN	SALESMAN	7698	20/02/1981 00:00:00	1600	300	30
7521	WARD	SALESMAN	7698	22/02/1981 00:00:00	1250	500	30
7654	MARTIN	SALESMAN	7698	28/09/1981 00:00:00	1250	1400	30
7844	TURNER	SALESMAN	7698	08/09/1981 00:00:00	1500	0	30
JOB: MANAGER - 1 Items							
JOB: CLERK - 1 Items							

The example below shows the same data displayed as a Big Data Report.



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	09/06/1981 00:00:00	2450		10
7839	KING	PRESIDENT		17/11/1981 00:00:00	5000		10
7934	MILLER	CLERK	7782	23/01/1982 00:00:00	1300		10
7369	SMITH	CLERK	7902	17/12/1980 00:00:00	800		20
7566	JONES	MANAGER	7839	02/04/1981 00:00:00	2975		20
7788	SCOTT	ANALYST	7566	19/04/1987 00:00:00	3000		20
7876	ADAMS	CLERK	7788	23/05/1987 00:00:00	1100		20
7902	FORD	ANALYST	7566	03/12/1981 00:00:00	3000		20
7499	ALLEN	SALESMAN	7698	20/02/1981 00:00:00	1600	300	30
7521	WARD	SALESMAN	7698	22/02/1981 00:00:00	1250	500	30
7654	MARTIN	SALESMAN	7698	28/09/1981 00:00:00	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/1981 00:00:00	2850		30
7844	TURNER	SALESMAN	7698	08/09/1981 00:00:00	1500	0	30
7900	JAMES	CLERK	7698	03/12/1981 00:00:00	950		30

There are very few formatting options, but the Big Data report will give better performance when used with very large volumes of data.

The formatting options available are:

- Clicking on the column header sorts the report by that column.
- You can adjust column widths by dragging the boundaries between column headers.

Freehand SQL

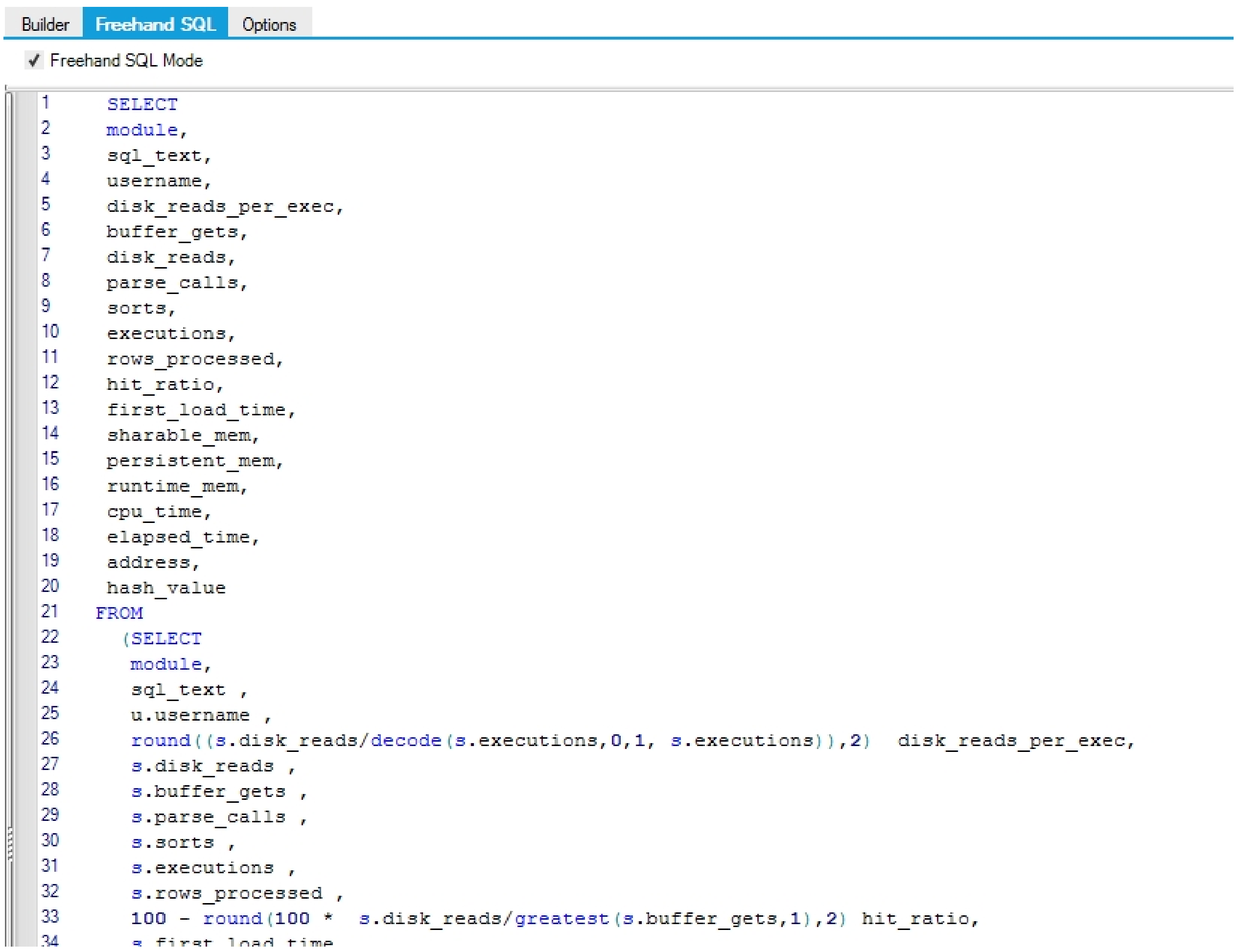
Freehand SQL

You can gain an understanding of SQL by checking the SQL panel of the Relational Explorer while using the Builder tool.

An experienced SQL user can develop reports more quickly by crafting a SQL query directly into the Freehand SQL panel, accessed from **Freehand SQL**.

In the Freehand SQL panel, check the Freehand SQL box to indicate to Relational Explorer that it should use this method only (the Builder tool is disabled until this box is unchecked).

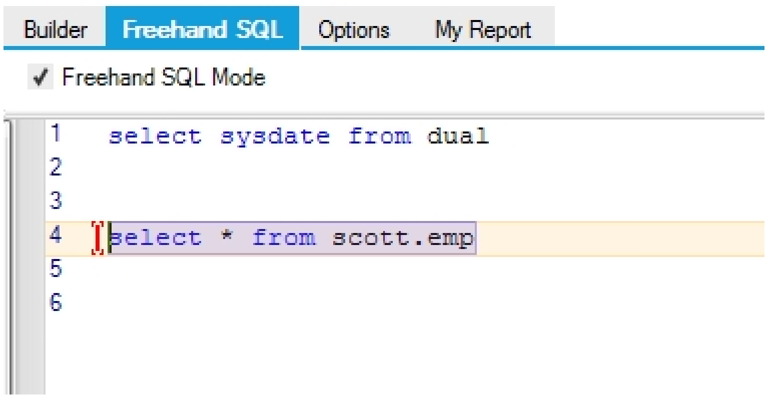
Type a SQL statement into the editor panel. Use the Run button as normal to run your report.



✓ Freehand SQL Mode

```
 SELECT
 module,
 sql_text,
 username,
 disk_reads_per_exec,
 buffer_gets,
 disk_reads,
 parse_calls,
 sorts,
 executions,
 rows_processed,
 hit_ratio,
 first_load_time,
 sharable_mem,
 persistent_mem,
 runtime_mem,
 cpu_time,
 elapsed_time,
 address,
 hash_value
FROM
  (SELECT
   module,
   sql_text ,
   u.username ,
   round((s.disk_reads/decode(s.executions,0,1, s.executions)),2)  disk_reads_per_exec,
   s.disk_reads ,
   s.buffer_gets ,
   s.parse_calls ,
   s.sorts ,
   s.executions ,
   s.rows_processed ,
   100 - round(100 *  s.disk_reads/greatest(s.buffer_gets,1),2) hit_ratio,
   s.first_load_time
```

You can create multiple statements within the Freehand SQL panel, if multiple statements are entered Relational Explorer will run the selected statement:



✓ Freehand SQL Mode

```
select sysdate from dual


select * from scott.emp
```

Saving Data to a file

Saving to data to a file

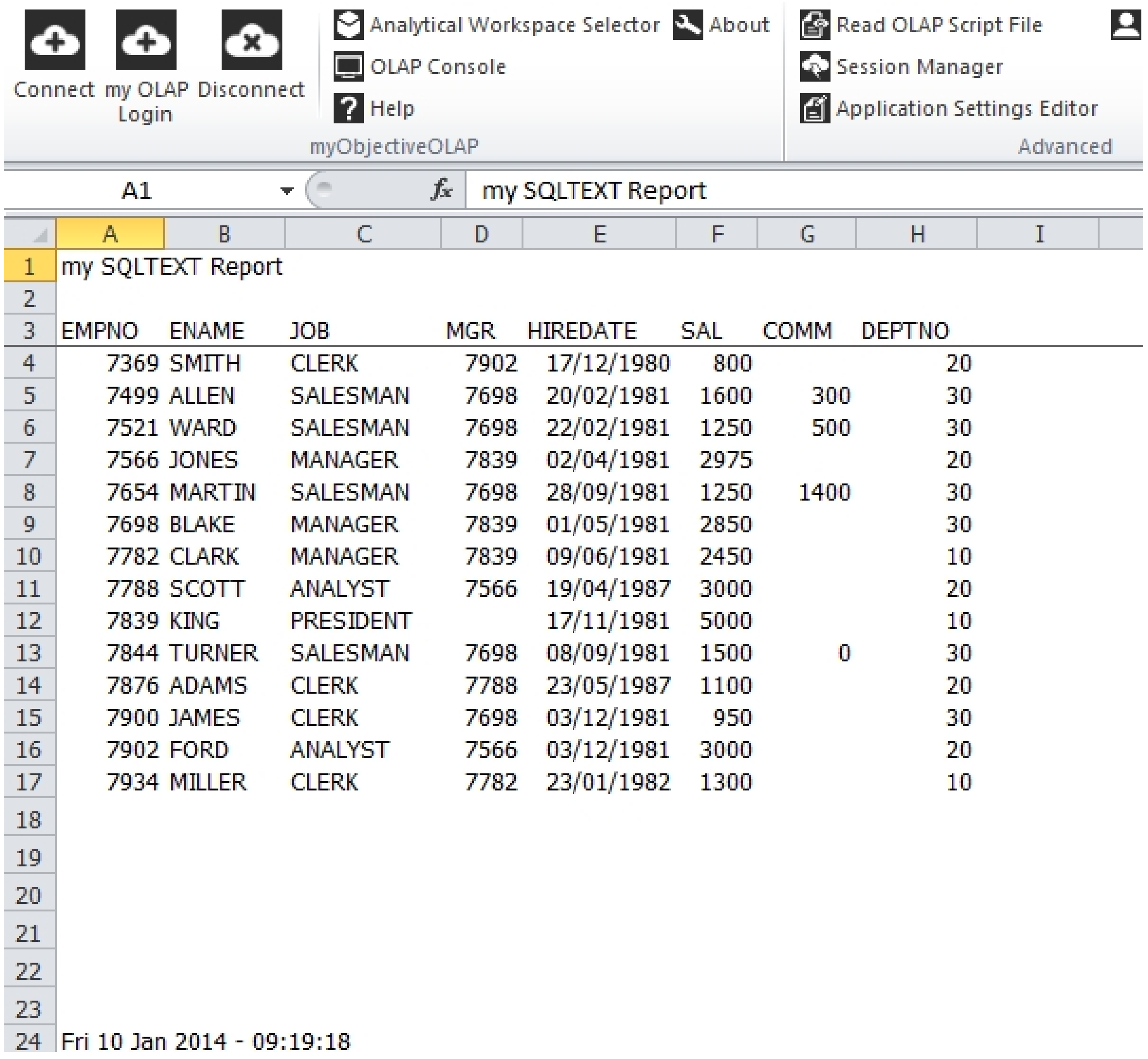
You can export the results of your query by using the **Save to File** Ribbon menu item:

Save to File

Before saving your data, you must use the Run option; this retrieves the data into Relational Explorer

Based on the type of report you have run, and the size of the data-set retrieved you will be able to export directly to a Microsoft Excel file form or a comma separated (csv) file.

Saving to a file opens a Windows file dialog box, and saves the data to the location you have specified and the filename you have entered. This can then be opened in Excel.



my SQLTEXT Report

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/1980	800		20
7499	ALLEN	SALESMAN	7698	20/02/1981	1600	300	30
7521	WARD	SALESMAN	7698	22/02/1981	1250	500	30
7566	JONES	MANAGER	7839	02/04/1981	2975		20
7654	MARTIN	SALESMAN	7698	28/09/1981	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/1981	2850		30
7782	CLARK	MANAGER	7839	09/06/1981	2450		10
7788	SCOTT	ANALYST	7566	19/04/1987	3000		20
7839	KING	PRESIDENT		17/11/1981	5000		10
7844	TURNER	SALESMAN	7698	08/09/1981	1500	0	30
7876	ADAMS	CLERK	7788	23/05/1987	1100		20
7900	JAMES	CLERK	7698	03/12/1981	950		30
7902	FORD	ANALYST	7566	03/12/1981	3000		20
7934	MILLER	CLERK	7782	23/01/1982	1300		10

Fri 10 Jan 2014 - 09:19:18

Options

The options panel allows you to specify a Header and Footer descriptive text field for your report when extracting to a file.   By default, the footer shows the date and time when you retrieved the data from the database.

Saving your report definition

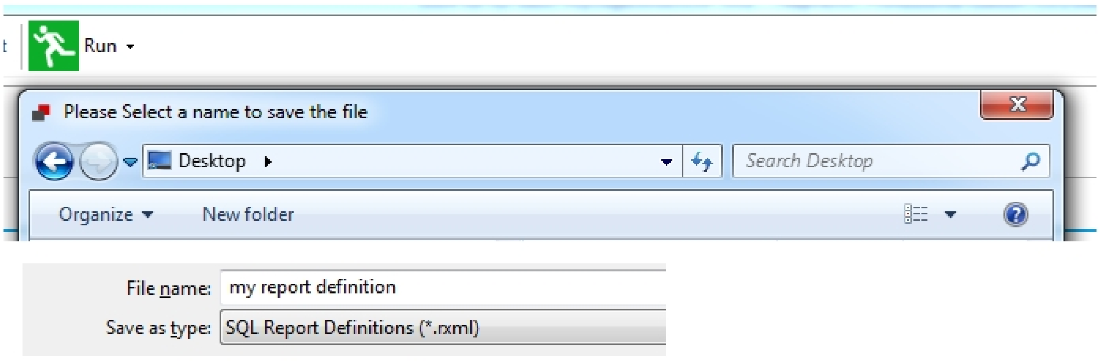
Saving and opening your report definition

These options allow you to save or retrieve your reports as definition files for use later. They do not save the data, only the selection and layout of the report.

The following options are available on the main menu of the Relational Explorer.

Save Report Definition	Save your current work to a report definition file (the file extension is ..rxml).
Open Report Definition	Open a previously saved report for display or to continue developing it.

These options open a standard Windows dialog box which will allow you to choose a file and location or locate an existing file:



File name: my report definition
Save as type: SQL Report Definitions (*.rxml)

You can freely distribute the report definition rxml file, in the above case "my report definition.rxml" which I have saved to the Desktop to any of your colleagues with access to the same myObjectiveOLAP Server installation. Provided their access credentials allow them access to the data objects defined within the report they will be able to Open and run the report.

The following shows the contents of a report definition file used to construct a report on the demo SCOTT.EMP table. No report result data is stored in the report definition file.

```
<?xml version="1.0" encoding="utf-8"?>
<VSQL_SaveReport xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SelectedTable>SCOTT.EMP</SelectedTable>
  <AllColumns>
    <VSQL_ColumnDT>
      <ColName>EMPNO</ColName>
      <ColDataType>System.Decimal</ColDataType>
    </VSQL_ColumnDT>
    <VSQL_ColumnDT>
      <ColName>ENAME</ColName>
      <ColDataType>System.String</ColDataType>
    </VSQL_ColumnDT>
    <VSQL_ColumnDT>
      <ColName>JOB</ColName>
      <ColDataType>System.String</ColDataType>
    </VSQL_ColumnDT>
    <VSQL_ColumnDT>
      <ColName>MGR</ColName>
      <ColDataType>System.Decimal</ColDataType>
    </VSQL_ColumnDT>
    <VSQL_ColumnDT>
      <ColName>HIREDATE</ColName>
      <ColDataType>System.String</ColDataType>
    </VSQL_ColumnDT>
    <VSQL_ColumnDT>
      <ColName>SAL</ColName>
      <ColDataType>System.Decimal</ColDataType>
    </VSQL_ColumnDT>
    <VSQL_ColumnDT>
      <ColName>COMM</ColName>
      <ColDataType>System.Decimal</ColDataType>
    </VSQL_ColumnDT>
    <VSQL_ColumnDT>
      <ColName>DEPTNO</ColName>
      <ColDataType>System.Decimal</ColDataType>
    </VSQL_ColumnDT>
  </AllColumns>
  <SelectedColumns>
    <string>EMPNO</string>
    <string>ENAME</string>
    <string>JOB</string>
  </SelectedColumns>
  <WhereClauses>
    <VSQL_OperandsDT>
      <ColumnDataType>System.String</ColumnDataType>
      <IsColumn>true</IsColumn>
      <Symbol>=</Symbol>
      <Enclose>false</Enclose>
      <Quote>false</Quote>
      <Values>
        <string>SALESMAN</string>
      </Values>
      <Value>SALESMAN</Value>
      <Column>JOB</Column>
    </VSQL_OperandsDT>
  </WhereClauses>
  <OrderClauses>
    <VSQL_OperandsDT>
      <IsColumn>false</IsColumn>
      <Enclose>false</Enclose>
      <Quote>false</Quote>
      <Values />
      <Value />
      <Column>JOB</Column>
    </VSQL_OperandsDT>
  </OrderClauses>
  <GroupClauses />
  <ManualSQLChecked>false</ManualSQLChecked>
  <IncludeHeader>true</IncludeHeader>
  <IncludeFooter>true</IncludeFooter>
  <IncludeColumnHeader>false</IncludeColumnHeader>
  <ReportHeader>my SQLTEXT Report</ReportHeader>
  <ReportFooter>Fri 10 Jan 2014 - 09:19:18</ReportFooter>
</VSQL_SaveReport>
```

Microsoft Excel Functions

Microsoft Excel Functions

The following functions enable the end user to directly reference data stored within an Analytic Workspace variable from within an Excel worksheet function.

All examples use Oracle Corporation's GLOBAL Analytic Workspace demo which can be downloaded from the Oracle OTN website.

mooDesc

=mooDesc(“[dim]” “[dimval]”)

Returns the Long Description of a dimension value for a given dimension.

Requires the Analytic Workspace meta-data to conform to Oracle OLAP standard form definition. Alternatively, this function can be used after the application DBA defines a formula text variable with the name [DIMENSION]_LONG_DESCRIPTION which references the non-standard named description variable.

Syntax

```
            =mooDesc("[dim]", "[dimval]")
```

Return Value

```
            STRING
```

Example

```
            =mooDesc("CUSTOMER", "ACCOUNT_BAVARIAN IND")
```

Example Output

Bavarian Industries

mooCellQDR

=mooCellQDR("[CUBE]", "[dim1]", "[dim1value]", "[dim2]", "[dim2value]", etc....)

Returns the numeric result of a qualified data reference from a numerical variable within an Analytic Workspace.

Syntax

```
            =mooCellQDR("[CUBE]", "[dim1]", "[dim1value]", "[dim2]",
"[dim2value]", etc....)
```

Return Value

```
            DECIMAL
```

Example

=mooCellQDR("UNITS_CUBE_COST", "CUSTOMER", "ACCOUNT_BAVARIAN IND", "TIME", "MONTH_2006.02", "CHANNEL", "TOTAL_TOTAL", "PRODUCT", "TOTAL_TOTAL")

Example Output

41822.97

Date Type

Windows Data Type	Nominal storage allocation	Value range
System.Decimal	16 bytes	0 through +/-79,228,162,514,264,337,593,543,950,335 with no decimal point; 0 through +/-7.9228162514264337593543950335 with 28 places to the right of the decimal; smallest nonzero number is +/-0.0000000000000000000000000001 (+/-1E-28).

Limitations

- Can only be used to return numerical data, TEXT / STRING data must be returned using the mooCellQDRTs() function.

- mooCellQDR cannot be seen interacting with the OLAP engine through the Oracle OLAP RECAP DML statement.

- myObjectiveOLAP supports retrieving data using mooCellQDR on cubes with between 1 and 14 dimensions.

mooCellQDRT

=mooCellQDRT(“[CUBE]”, “[dim1]”, “[dim1value]”, “[dim2]”, “[dim2value]”, etc....)

Returns the text result of a qualified data reference from a text variable within an Analytic Workspace.

Syntax

```
            =mooCellQDRT("[CUBE]", "[dim1]", "[dim1value]", "[dim2]",
"[dim2value]", etc....)
```

Return Value

```
STRING
```

Example

=mooCellQDRT(SYS.CFG, SYS.ROW, "FAILED_PASSWORD_LOCK", SYS.COL, "VALUE")

Example Output

YES

Limitations

- Can only be used to return TEXT data, numerical data must be returned using the mooCellQDR() function.

- mooCellQDRT cannot be seen interacting with the OLAP engine through the Oracle OLAP RECAP DML statement.

- myObjectiveOLAP supports retrieving data using mooCellQDRT on cubes with between 1 and 14 dimensions.

mooQT

=mooQT(“[TEXT_VARIABLE_NAME]([DIMENSION_NAME] '[DIMENSION_VALUE]')”)

Returns textual data from an Oracle OLAP TEXT cube or variable.

When you know the data type of a variable is TEXT you should use mooQT instead of mooQ in order to maximize performance of retrieved data.

Syntax

```
=mooQT("[TEXT_VARIABLE_NAME]([DIMENSION_NAME]
'[DIMENSION_VALUE]')")
```

Return Value

```
STRING
```

Example

=mooQT("CUSTOMER_LONG_DESCRIPTION(CUSTOMER 'ACCOUNT_CICI-D')")

Example Output

CiCi Douglas

Limitations

- Can only be used to return TEXT data, numerical data must be returned using the mooCellQN or mooQ function.

Data Types permissible with mooQN

TEXT

- mooQT cannot be seen interacting with the OLAP engine through the Oracle OLAP RECAP DML statement.
  If you are unsure as to why a specific mooQT retrieve is failing, the end-user can switch to mooQ in order to see the interaction with the Oracle OLAP option.

- myObjectiveOLAP supports retrieving data using mooQT on Oracle OLAP cubes of all dimension numbers.

mooQN

=mooQN(“[NUMERIC_VARIABLE_NAME]([DIMENSION1_NAME] '[DIMENSION_VALUE]')”)

Returns numerical data from an Oracle OLAP numerical cube or variable.

When you know the data type of a variable is numeric you should use mooQN instead of mooQ in order to maximize performance of retrieved data.

Syntax

```
=mooQN("[TEXT_VARIABLE_NAME]([DIMENSION_NAME]
'[DIMENSION_VALUE]')")
```

Return Value

```
DECIMAL
```

Example

=mooQN("UNITS_CUBE_COST (customer 'ACCOUNT_BAVARIAN IND' TIME 'MONTH_2006.02' channel 'TOTAL_TOTAL' product 'TOTAL_TOTAL')")

Example Output

41,823

Limitations

- Can only be used to return NUMERIC data, numerical data must be returned using the mooCellQN or mooQ function.

Data Types permissible with mooQN

```
INTEGER
SHORTINTEGER
LONGINTEGER
DECIMAL
SHORTDECIMAL
NUMBER
```

- mooQN cannot be seen interacting with the OLAP engine through the Oracle OLAP RECAP DML statement.
  If you are unsure as to why a specific mooQN retrieve is failing, the end-user can switch to mooQ in order to see the interaction with the Oracle OLAP
  option.

- myObjectiveOLAP supports retrieving data using mooQN on Oracle OLAP cubes of all dimension numbers.

mooQ

=mooQ(“[VARIABLE_NAME]([DIMENSION1_NAME] '[DIMENSION_VALUE]')”)

Returns numerical data (as String) or text data from an Oracle OLAP cube or variable.

When you know the data type of a variable is numeric or text you should use mooQN or mooQT instead of mooQ in order to maximize performance of retrieved data.

Syntax

```
=mooQ("[TEXT_VARIABLE_NAME]([DIMENSION_NAME]
'[DIMENSION_VALUE]')")
```

Return Value

```
STRING
```

Example

=mooQ("UNITS_CUBE_COST (customer 'ACCOUNT_BAVARIAN IND' TIME 'MONTH_2006.02' channel 'TOTAL_TOTAL' product 'TOTAL_TOTAL')")
=mooQ("CUSTOMER_LONG_DESCRIPTION(CUSTOMER 'ACCOUNT_CICI-D')")

Example Output

41823
CiCi Douglas

Notes

- Slower than data type specific myObjectiveOLAP Excel formula

Data Types permissible with mooQ

ALL (with the exception of `RAW`)

- mooQ can be seen interacting with the OLAP engine through the Oracle OLAP RECAP DML statement.

- myObjectiveOLAP supports retrieving data using mooQ on Oracle OLAP cubes of all dimension numbers.

mooW

=mooW(“[VARIABLE_NAME]([DIMENSION1_NAME] '[DIMENSION_VALUE]')”, [VALUE], [OPTION])

Enables cell based write back of data to an Oracle OLAP variable within an Analytic Workspace.

Syntax

```
        =mooW("[VARIABLE_NAME]([DIMENSION1_NAME] '[DIMENSION_VALUE]')",
[VALUE], [OPTION])
```

Return Value

```
        OBJECT
```

Arguments

The mooW function is called through three arguments:

Qualified Data Reference

The first component is a qualified data reference indicating a unique triangulated coordinate within an Oracle OLAP Analytic Workspace variable.

=mooW("UNITS_CUBE_COST (customer 'ACCOUNT_BAVARIAN IND' TIME 'MONTH_2006.02' channel 'TOTAL_TOTAL' product 'TOTAL_TOTAL')", 10, 0)

In the example above we are triangulating within the UNITS_CUBE_COST to a single cell, the intersection of the specified Customer, Channel, Time and Product dimensions.

User Supplied value.

This is the value you wish to upload to the specified intersecting co-ordinates within the OLAP variable.

=mooW("UNITS_CUBE_COST (customer 'ACCOUNT_BAVARIAN IND' TIME 'MONTH_2006.02' channel 'TOTAL_TOTAL' product 'TOTAL_TOTAL')", 10, 0)

This can be a cell reference to a value within another Excel cell, worksheet or workbook.

Option

=mooW("UNITS_CUBE_COST (customer 'ACCOUNT_BAVARIAN IND' TIME 'MONTH_2006.02' channel 'TOTAL_TOTAL' product 'TOTAL_TOTAL')", 10, 0)

The third argument tells the mooW function what you want to do:

Option 0

Passing 0 does nothing on re-calculation of the Excel formula, other than display the value passed as the User Supplied value.
This can be useful when working disconnected from the Oracle OLAP data source.

Option 1

Passing 1 writes the User Supplied value back to the Oracle Analytic workspace variable to the intersection supplied in the Qualified Data
Reference.

Option 2

Passing 2 essentially converts the mooW function into the mooQ function and retrieves the data supplied by the Qualified Data Reference
from the intersecting co-ordinates within the Oracle OLAP Analytic Workspace variable.

Data Types permissible with mooQ

ALL (with the exception of `RAW`)

Example

Option 0 Show the local user supplied value

```
=mooW("UNITS_CUBE_COST ( customer 'ACCOUNT_BAVARIAN IND' TIME 'MONTH_2006.02'  channel 'TOTAL_TOTAL' product 'TOTAL_TOTAL')", 10, 0)
```

Option 1 write the user supplied value back to the database

```
=mooW("UNITS_CUBE_COST ( customer 'ACCOUNT_BAVARIAN IND' TIME 'MONTH_2006.02'  channel 'TOTAL_TOTAL' product 'TOTAL_TOTAL')", 10, 1)
```

Option 2 shows the database value

```
=mooW("UNITS_CUBE_COST ( customer 'ACCOUNT_BAVARIAN IND' TIME 'MONTH_2006.02'  channel 'TOTAL_TOTAL' product 'TOTAL_TOTAL')", 10, 2)
```

Example Output

```
10
10
10
```

Notes

- mooW can be seen interacting with the OLAP engine through the Oracle OLAP RECAP DML statement.

- myObjectiveOLAP supports retrieving and writing of data using mooW on Oracle OLAP cubes of all dimension numbers.

- No mechanism is supplied to permanently store the written data within the Analytic workspace. This could easily be accomplished by the end-user or
developer by using a VBA macro which makes a call through the
run_nonQ function to attach the AW RW, followed by the data upload, an update; commit and a call to mooAWDetach.
Alternative controlled and audited write back is available as part the mooServer or Escendo server side products.

Manipulating Oracle OLAP from Microsoft Excel VBA

myObjectiveOLAP Object Orientated VBA Model

myObjectiveOLAP exports functions to Microsoft Excel which can be taken advantage of by users of Microsoft's Excel VBA model.

Common Functions

Common functions enable the end user to interact either with the myObjectiveOLAP library itself or to execute commands or retrieve output from the Oracle OLAP database server.

Setting OLAP Options

myObjectiveOLAP allows the VBA user to set Options within the Oracle OLAP environment

myObjectiveOLAP Graphical API

The myObjectiveOLAP Graphical API allows the VBA user or developer to integrate standard myObjectiveOLAP windows forms into their own application.

Common Functions

Common Functions

Common functions enable the end user to interact either with the myObjectiveOLAP library itself or to execute commands or retrieve output from the Oracle OLAP database server.

This includes a number of low level API's that do minimal checking before attempting to execute within the server side environment. It is best practice to only use these API's if myObjectiveOLAP does not offer a standard function which meets your requirement.
By using the myObjectiveOLAP functions in your code additional pre-execution checks are performed and enhanced error trapping is available to you.

Handling Connections

connect

connect()

Initiates a connection to an Oracle OLAP instance. Requires a valid XML connection file to have been created in advance.

Syntax

```
connect()
```

Return Value

```
BOOLEAN
```

Example

```
Public Sub connect()
'Actually does the connection

If Not oregistered Then:   boo = regQ:
boo = o.connect

If boo = True Then
  MsgBox "Connected OK"
Else
  MsgBox "Not Connected"
End If

End Sub
```

connected

connected()

Returns TRUE if a current connection is open to an Oracle OLAP instance.

Syntax

```
connected
```

Return Value

```
BOOLEAN
```

Example

```
Public Sub connected()
'Am I connected

If Not oregistered Then:   boo = regQ:
boo = o.connected

If boo = True Then
  MsgBox "Yes, Connected"
Else
  MsgBox "No, Not Connected"
End If

End Sub
```

connectSpec

connectSpec(“[host]” “[sid]” “[port]” “[user]” “[password]”)

Initiates a connection to an Oracle OLAP instance. Unlike connect a valid XML connection file is not required. However the developer must provide the necessary connection information during the function call.

Syntax

```
connectspec("[host]" "[sid]" "[port]" "[user]" "[password]")
```

Return Value

```
BOOLEAN
```

Example

```
Public Sub connectSpec()
'Create a manual connection without a connection xml file

Dim hostname  As String
Dim sid As String
Dim port As String
Dim username As String
Dim password As String

If Not oregistered Then:   boo = regQ:

hostname = "yourHostName"
sid = "yourSid-orcl"
port = "yourPort-1521?"
username = "yourUserName"
password = "yourUserPassword"

boo = o.connectSpec(hostname, sid, port, username, password)

If boo = True Then
 MsgBox "Connected OK"
Else
 MsgBox "Not Connected"
End If

End Sub
```

disconnect

disconnect()

Closes the current connection to an Oracle OLAP instance. No further Analytic Workspace operations are carried out including DML detaching the analytic workspace.

Syntax

```
            disconnect
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub disconnect()
'Disconnects Excel From Oracle OLAP

If Not oregistered Then:   boo = regQ:
boo = o.disconnect

If boo = True Then
  MsgBox "Disconnected OK"
Else
  MsgBox "Not Disconnected"
End If

End Sub
```

AW Operations

mooAWAttach

mooAWAttach("[aw name]", "[position]")

Attaches an Analytic Workspace in the specified position if it exists

Syntax

```
mooAWAttach("[aw name]", "[position]")
```

where [aw name] is the fully referenced analytic workspace name
[position] is the position the referenced aw should be attached.

FIRST makes the aw you are attaching the current one.
LAST makes the aw you are attaching the last in the list excluding the express aw.
BEFORE puts the aw you are attaching before an aw which is already in the list.
AFTER puts the aw you are attaching after an aw which is already in the list.

Return Value

```
BOOLEAN
```

Example

```
Public Sub mooAwAttach()
'Attach Analytic Workspaces

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

If boo = True Then
 boo = o.mooAwAttach("GLOBAL.GLOBAL", "FIRST")
 boo = o.mooAwAttach("GLOBAL.GLOBAL", "AFTER EXPRESS")
Else
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

mooAWAttached

mooAWAttached(“ [aw name] “)

Returns TRUE if the specified analytic workspace is attached (open)

Syntax

```
mooAWAttached("[aw name]")
```

Return Value

```
BOOLEAN
```

Example

```
Public Sub mooAwAttached()
'Check if an Aw is attached
```

```
Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

boo = o.mooAwAttached("EXPRESS")
If boo = False Then
   MsgBox "No Not Attached"
Else
    MsgBox "Yes Attached"
End If

End Sub
```

mooAWDetach

mooAWDetach(“ [aw name] “)

Detaches an analytic workspace. MooAWDetach does not perform an update.

Syntax

```
            mooawdetach("[aw name]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooAwDetach()
'Detach Analytic Workspaces

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

If boo = True Then
 boo = o.mooAWDetach("GLOBAL.GLOBAL ")
Else
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

Oracle OLAP Executing Commands

mooexecute

mooExecute(“ [OLAP DML] “)

An API that allows a client developer to execute Oracle OLAP DML statements directly within the Oracle OLAP environment and return any output from the Oracle OLAP engine

Syntax

mooexecute(“ [OLAP DML] “)

Return Value

```
STRING
```

Example

```
Sub mexecute()
'Example of MOOEXECUTE

On Error GoTo ErrorH

  Set moo = Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object

  Debug.Print moo.mooexecute("shw tod")

  Exit Sub

ErrorH:

Debug.Print Error(Err)

End Sub
```

wrap_runNonQ

wrap_runNonQ (“ [OLAP DML] “)

A low level API that allows a client developer to execute Oracle OLAP DML statements directly within the Oracle OLAP environment.   wrap_runNonQ does not request any output from the Oracle OLAP environment on execution.

Because no output is requested wrap_runNonQ is often used in time sensitive operations when no output is expected as its execution time is approximately half that of calling mooExecute.

Syntax

wrap_runnonq(“ [OLAP DML] “)

Return Value

```
BOOLEAN
```

Example

```
Public Sub wrap_runNonQ()
'Example of wrap_runNonQ and wrap_GetDML


Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

If boo = True Then
 boo = o.wrap_runNonQ("shw tod")
   If boo = True Then
     'You Could also use .mooGetDML here
      Debug.Print (o.wrap_getDML)
   Else
      Debug.Print o.getlastmooerr
   End If
Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

wrap_GetDML

wrap_GetDML

A low level API that retrieves the output from the Oracle OLAP environment after execution of a DML statement via the wrap_runNonQ function.

Because no output is requested wrap_GetDMML is often used in conjunction with wrap_runNonQ but only called when an error condition is detected within the Visual Basic for Applications client module.

Syntax

```
            wrap_getdml
```

Return Value

```
            STRING
```

Example

```
Public Sub wrap_runNonQ()
'Example of wrap_runNonQ and wrap_GetDML


Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

If boo = True Then
   boo = o.wrap_runNonQ("shw tod")
   If boo = True Then
      Debug.Print (o.wrap_getDML)
   Else
   Debug.Print o.getLastMooErr
   End If
Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
```

```
End If

End Sub
```

Functions

mooAllStat

mooAllStat

Opens the status of all dimensions within the currently attached Analytic Workspace.

Syntax

```
            mooAllStat
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooAllstat()
'Limits all dimensions in the current AW to all

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
   boo = o.mooAwAttach("EXPRESS", "FIRST")
   boo = o.wrap_runNonQ("lmt INTL.MLANG to 1")
   Debug.Print o.mooStatlen("INTL.MLANG")
   boo = o.mooAllstat
   Debug.Print o.mooStatlen("INTL.MLANG")
Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
```

mooAnalyzeCube

mooAnalyzeCube("[cube name]")

Returns a one-dimension array object each value of the array is a string value of the dimensions of the specified Analytic Workspace cube.

Syntax

```
            mooAnalyzeCube("[cube name]")
```

Return Value

```
            STRING() ARRAY
```

Example

```
Public Sub AnalyzeCube()
'returns a single dimension array of all dimensions of the variable passed to
mooAnalyzeCube

Dim str() As String

'Check Im connected if not connect
If Not   .connected Then
   boo = Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule")
Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object.Object
.connect
   boo =
Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object.connected
   Else
   boo = True
End If

str =
Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object.mooAnalyzeC
ube("GLOBAL.GLOBAL!UNITS_CUBE_COST")

For i = 0 To UBound(str)
  Debug.Print (str(i))
Next

End Sub
```

Return Value Example

```
TIME
CHANNEL
CUSTOMER
PRODUCT
```

mooClearAnalyzeCube

mooClearAnalyzeCube

Clears the internal myObjectiveOLAP array storing the result of any mooAnalyzeCube call. This destroys the internal array and release memory.

Syntax

```
            mooClearAnalyzeCube
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub ClearAnalyzeCube()
'returns a single dimension array of all dimensions of the variable passed to
mooAnalyzeCube

Dim str() As String
Dim boo as Boolean
```

```
'Check Im connected if not connect
If Not
Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object.connected
Then
   boo =
Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object.connect
   boo =
Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object.connected
   Else
   boo = True
End If

str =
Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object
.mooAnalyzeCube("GLOBAL.GLOBAL!UNITS_CUBE_COST")

For i = 0 To UBound(str)
  Debug.Print (str(i))
Next

boo =
Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object.mooClearAna
lyzeCube

if boo then
   debug.print("Clear of internal array complete")
End If

End Sub
```

Return Value Example

```
TIME
CHANNEL
CUSTOMER
PRODUCT
Clear of internal array complete
```

mooFreePages

mooFreePages

Reports the FREEPAGES of the current analytic workspace. If no analytic workspace is attached zero will be returned.

Syntax

```
            mooFreePages()
```

Return Value

```
            STRING
```

Example

```
Public Sub mooFreePages()
'Shows the freepages of the current AW

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If
```

```
'Connected so execute
If boo = True Then
    boo = o.mooAwAttach("EXPRESS", "FIRST")
    Debug.Print o.mooFreePages()
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

mooHost

mooHost()

Returns the hostname of the server which the Oracle OLAP environment is hosted.

Syntax

```
            mooHost()
```

Return Value

```
            STRING
```

Example

```
Public Sub mooHost()
'Shows the hostname of the server which you are connect to

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
    Debug.Print o.mooHost()
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

mooInstance

mooInstance()

Returns the instance name of the Oracle OLAP environment.

Syntax

```
            mooInstance()
```

Return Value

```
            STRING
```

Example

```
Public Sub mooInstance()
'Shows the SID of the Oracle database you are connect to

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
     Debug.Print o.mooInstance()
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

mooSeconds

mooSeconds()

Returns the value of Seconds from the Oracle OLAP environment.

Syntax

```
            mooseconds()
```

Return Value

```
            LONG
```

Example

```
Public Sub mooSeconds()

'Return the current value of seconds within Oracle OLAP
Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Get the value of seconds
If boo = True Then
  Debug.Print (o.mooSeconds)
Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

mooSysTimeStamp

mooSysTimeStamp()

Returns the value of SysTimeStamp within the Oracle OLAP environment.

Syntax

```
            mooSysTimeStamp
```

Return Value

```
            STRING
```

Example

```
Public Sub mooSysTimeStampANDmooSysDate()
'Shows the value of SysTimeStamp and sysDate

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
     Debug.Print o.mooSysTimeStamp()
     'returns i.e. 11-MAY-10 17.58.17.851963 +01:00
     Debug.Print o.mooSysDate()
     'returns i.e. 11-MAY-10

    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

mooUser

mooUser()

Returns the name of the user currently connected to Oracle OLAP.

Syntax

```
            mooUser()
```

Return Value

```
            STRING
```

Example

```
Public Sub mooUser()
'Shows the current database user
```

```
Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
     Debug.Print o.mooUser()
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

olapQDR

olapQDR(“ [valid QDR statement] “)

An end user orientated function that can be used to return a value from an Oracle OLAP array by fully qualifying the coordinates within the array.

Syntax

```
olapqdr("[valid QDR statement]")
```

Where QDR statements in the the form
cube(dim1 dimval1 dim2 dimval2 ….dimx dimvalx)
The QDR must be fully referenced for the stated cube otherwise an error will be returned.

Return Value

```
STRING
```

Example

```
Public Sub olapQDR()
'Pass a QDR to Oracle OLAP and return the result

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so bring up the command line
If boo = True Then
   boo = o.mooAwAttach("EXPRESS", "FIRST")
        'INTL.MLANGMAP is a variable within the Express AW
        'You can pass any valid OLAP qdr to olapqdr
          Debug.Print (o.olapQDR("INTL.MLANGMAP(INTL.MLANG 'ENB')"))
End If

End Sub
```

mooSysDate

mooSysDate()

Returns the value of SysDate within the Oracle OLAP environment.

Syntax

```
            mooSysDate()
```

Return Value

```
            STRING
```

Example

```
Public Sub mooSysTimeStampANDmooSysDate()
'Shows the value of SysTimeStamp and sysDate

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
     Debug.Print o.mooSysTimeStamp()
     'returns ie. 11-MAY-10 17.58.17.851963 +01:00
     Debug.Print o.mooSysDate()
     'returns ie. 11-MAY-10

    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

mooGetDimList

mooGetDimList ([Dimension Name], True/False)

The mooGetDimList function returns a one dimensional array containing dimension values from a dimension within Oracle OLAP.

The contents of the array passed back can then either be further processed in VBA or passed directly to a worksheet within Excel for reporting.

mooGetDimList observes the current status of the dimension passed to the function if the second boolean argument is either omitted or FALSE is passed.

If TRUE is passed as the second functional argument then mooGetDimList will temporarily open the dimension to all values by issuing a LIMIT [DIMENSION_NAME] to all statement before retrieving the list of dimension values. mooGetDimList will encapsulate the LIMIT ALL and dimension retrieval within a PUSH and POP statement ensuring the current dimensional limit is unaffected by the users request.

If the status of the dimension being requested is null then NA will be returned to VBA.

If the dimension does not exist or mooGetDimList detects that it is not connected to an Oracle OLAP enabled database then the 0 element of the array will contain any error information.

Syntax

```
            mooGetDimList ([Dimension Name], True / False)
```

Return Value

```
            STRING ()
```

Example

```
            Sub listDimensions()

              Dim d As Double 'Might be a long dimension
              Dim arr() As String

              Set moo =
Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object

              arr = moo.mooGetDimList("ACCOUNT", True)

              For d = 0 To UBound(arr)
               Debug.Print (arr(d))
              Next

            End Sub
```

Error Handling

getLastMooErr

getLastMooErr()

Returns the text of any error trapped by the myObjectiveOLAP library. This includes errors that were not handed over to the Oracle OLAP environment as the API determined the construct was invalid.

Syntax

```
            getLastMooError()
```

Return Value

```
            STRING
```

Example

```
Public Sub mooClearErr()
'Get Last Recorded Error

If Not oregistered Then:   boo = regQ:
   Debug.Print o.getlastmooerr    'Show the last Error
   Debug.Print o.mooClearErr      'Clear all error messages
   Debug.Print o.getlastmooerr    'show that all error messages have been
cleared
End Sub
```

mooClearErr

mooClearErr

Clears the last error trapped by the myObjectiveOLAP library. Subsequent calls to getLastMooErr would result in a NULL being returned until the next error.

Syntax

```
            mooClearErr()
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooClearErr()
  'Clear Errors

  If Not oregistered Then:   boo = regQ:
  Debug.Print o.getlastmooerr   'Show the last Error
  Debug.Print o.mooClearErr     'Clear all error messages
  Debug.Print o.getlastmooerr   'show that all error messages have been
cleared

End Sub
```

mooServErr

mooServErr()

Returns the text of any error trapped by the Oracle OLAP environment.

Syntax

```
            mooServErr()
```

Return Value

```
            STRING
```

Example

```
Public Sub mooServErr()

'Get the last Oracle OLAP error

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
   boo = o.wrap_runNonQ("shw GenerateAnError")
     Debug.Print o.mooServErr()
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

Working with objects

mooDimLen

mooDimLen("[dim1"])

Returns the maximum length of an Oracle OLAP dimension as supplied to the function.

The result is the same as executing an obj(dimmax 'DIMNAME') within the Oracle OLAP environment.

Syntax

```
            mooDimLen("[dim1]")
```

Return Value

```
            INTEGER
```

Example

```
Public Sub mooDimLen()

'Show the Total Number of Dimension Values of a given dimension
Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

If boo = True Then
 Debug.Print o.moodimlen("INTL.MLANG")
Else
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

mooExists

mooExists("[object name]")

Enables the developer to identify if a given object exists within an Analytic Workspace in the current Oracle OLAP session

Syntax

```
            mooExists("[object name]")
```

Return Value

```
            STRING
```

Example

```
Public Sub mooExists()

If Not oregistered Then:   boo = regQ:
boo = o.mooExists("ALLCOMPILE")

If boo = True Then
  MsgBox "Yes, ALLCOMPILE Exists"
Else
  MsgBox "No, ALLCOMPILE Does Not Exist"
  Debug.Print o.getlastmooerr
End If

End Sub
```

mooObjType

mooObjType("[object name]")

Enables the developer to identify the data type of an Oracle OLAP object

Syntax

```
            mooObjType("[object name]")
```

Return Value

```
            STRING
```

Example

```
Public Sub mooObjType()

If Not oregistered Then:   boo = regQ:

If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

If boo = True Then
   Debug.Print o.mooObjType("ALLCOMPILE")
   Debug.Print o.mooObjType("INTL.MLANGMAP")
End If
```

mooOpenDim

mooOpenDim("[dim1]"

Limits a specified dimension within the Oracle OLAP environment to a status of ALL.

Syntax

```
            mooOpenDim("[dim1 ]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooOpenDim()
'Limits a specified dimension to all

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
   boo = o.mooAwAttach("EXPRESS", "FIRST")
   boo = o.wrap_runNonQ("lmt INTL.MLANG to 1")
   Debug.Print o.mooStatlen("INTL.MLANG")
   boo = o.mooOpenDim("INTL.MLANG")
   Debug.Print o.mooStatlen("INTL.MLANG")
Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

mooPushDims

mooPushDims("[object name]")

Executes a PUSH of an Oracle OLAP Dimension if the dimension exists.

Syntax

```
            mooPushDims("[object name]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooPushMooPopExample()
'Example Using mooPush mooPop and mooStatlen

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not   o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
   boo = o.mooAwAttach("EXPRESS", "FIRST")
   'show the current length of the INTL.MLANG dimension
```

```
    Debug.Print o.mooStatlen("INTL.MLANG")
    'push the dimension
    boo = o.mooPushDims("INTL.MLANG")
    'limit INTL.MLANG dimension to 1 value
    boo = o.wrap_runNonQ("lmt INTL.MLANG to 1")
    'show the current length of the INTL.MLANG dimension
    Debug.Print o.mooStatlen("INTL.MLANG")
    'pop the dimension
    boo = o.mooPopDims("INTL.MLANG")
    Debug.Print o.mooStatlen("INTL.MLANG")
Else
    'Something went wrong print any error information
    Debug.Print "Unable to Connect"
    Debug.Print o.getlastmooerr
End If

End Sub
```

mooPopDims

mooPopDims("[object name]")

Executes a POP of an Oracle OLAP Dimension if the dimension exists.

Syntax

```
            mooPopDims("[object name]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooPushMooPopExample()
'Example Using mooPush mooPop and mooStatlen

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
    boo = o.connect
    boo = o.connected
    Else
    boo = True
End If

'Connected so execute
If boo = True Then
    boo = o.mooAwAttach("EXPRESS", "FIRST")
    'show the current length of the INTL.MLANG dimension
    Debug.Print o.mooStatlen("INTL.MLANG")
    'push the dimension
    boo = o.mooPushDims("INTL.MLANG")
    'limit INTL.MLANG dimension to 1 value
    boo = o.wrap_runNonQ("lmt INTL.MLANG to 1")
    'show the current length of the INTL.MLANG dimension
    Debug.Print o.mooStatlen("INTL.MLANG")
    'pop the dimension
    boo = o.mooPopDims("INTL.MLANG")
    Debug.Print o.mooStatlen("INTL.MLANG")
Else
    'Something went wrong print any error information
    Debug.Print "Unable to Connect"
    Debug.Print o.getlastmooerr
End If
End Sub
```

mooStatlen

mooStatlen("[dim1"])

Returns the current length of a specified dimension if the dimension exists within the current Oracle OLAP session.

Syntax

```
            mooStatlen("[dim1]")
```

Return Value

```
            LONG
```

Example

```
Public Sub mooPushMooPopExample()

'Example Using mooPush mooPop and mooStatlen

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
   boo = o.mooAwAttach("EXPRESS", "FIRST")
   'show the current length of the INTL.MLANG dimension
   Debug.Print o.mooStatlen("INTL.MLANG")
   'push the dimension
   boo = o.mooPushDims("INTL.MLANG")
   'limit INTL.MLANG dimension to 1 value
   boo = o.wrap_runNonQ("lmt INTL.MLANG to 1")
   'show the current length of the INTL.MLANG dimension
   Debug.Print o.mooStatlen("INTL.MLANG")
   'pop the dimension
   boo = o.mooPopDims("INTL.MLANG")
   Debug.Print o.mooStatlen("INTL.MLANG")
Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

Local library functions

mooEncrypt

mooEncrypt("[PASSWORD]")

Accepts a string and returns an encrypted version of the string that can be used in constructing a valid XML connection file.

The mooEncrypt function can be used to generate a password that can be copied and pasted into an XML connection file.

Note there is no decrypt equivalent function as it is anticipated that all Oracle account passwords could be reset by the local DBA.

Syntax

```
            mooencrypt("[password]")
```

Return Value

```
            STRING
```

Example

```
Public Sub mooEncrypt()

'Generate an encrypted password for a connection xml file
'The Output in the VBA immediate window can be pasted in to a connection file
If Not oregistered Then:   boo = regQ:
Debug.Print o.mooEncrypt("myPasswordHere")

End Sub
```

mooSetLang

mooSetLang("[EN|FR]")

Sets the language that any internal messages that are generated by the moo library or menu items used by the graphical user interface. Default is EN.

Syntax

```
            mooSetLang("[EN|FR]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooSetLang()

If Not oregistered Then:   boo = regQ:

'Switch the text in the graphical forms to French
o.mooSetLang ("FR")
o.mooShowConnFrm

'Switch back to English
o.mooSetLang ("EN")
o.mooShowConnFrm

End Sub
```

loadSavedScript

loadSavedScript()

The loadSavedScript file acts as VBA available API to the Read OLAP Script myObjectiveOLAP functionality.

The API opens a standard Windows file system dialog box enabling the end-user to select a saved text file containing one or more valid OLAP DML statements. The contents of the file are immediately executed when selection of the file is complete.

loadSavedScript returns the directory path and filename of the file being opened.

Notes

All text files should be saved with a ".moo" file extension.

An example of the loadSavedScript function is seen in the 2012-myObjectiveOLAP-FastExample example Excel workbook.

Syntax

```
            loadSavedScript()
```

Return Value

```
            STRING
```

Example

```
Private Sub openSavedBTN_Click()

On Error GoTo EH

Set moo = Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object

debug.print moo.loadSavedScript

Exit Sub

EH:
 MsgBox Err & ": " & Error(Err)
End Sub
```

loadSavedScriptFile

loadSavedScriptFile("filename", [TRUE][FALSE])

The loadSavedScript file acts as VBA available API to the Read OLAP Script myObjectiveOLAP functionality.

The API expects a valid directory and file string argument to be passed. The contents of the file are immediately executed when selection of the file is complete.

loadSavedScript returns the directory path and filename of the file being opened. If the file the calling application is attempting to open does not exist or can not opened due to other considerations (permissions etc.) the output from the loadSavedScriptFile function will be: `"ERR: No File specified"`

The second boolean argument determines if a .moo.out file recording OLAP IO is produced in the same directory as the source file.

Notes

All text files should be saved with a ".moo" file extension.

An example of the loadSavedScript function is seen in the 2012-myObjectiveOLAP-FastExample example Excel workbook.

Syntax

```
loadSavedScriptFile("filename", [TRUE][FALSE])
```

Return Value

```
STRING
```

Example

```
Private Sub openSavedBTN_Click()

On Error GoTo EH

Set moo = Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object

debug.print moo.loadSavedScript("c:\myfile.moo", TRUE)

Exit Sub

EH:
 MsgBox Err & ": " & Error(Err)
End Sub
```

moo Fast Reporting

Fast Reporting Functions

myObjectiveOLAP contains a number of functions which can enable large amounts of information to be transferred from your Oracle OLAP Database server to the client in a network efficient manner.

Three functions are provided:

- mooFR
  Returns a two dimensional array from a variable in Oracle OLAP
- mooFRDescDown
  Returns a one dimensional array on the Y axis from Oracle OLAP
- mooFRDescAcross
  Returns a one dimensional array on the X axis from Oracle OLAP

An example reporting GUI application is available for download which utilises many of the functions of the myObjectiveOLAP API and includes use of the Fast reporting functions. This is available from the myObjectiveOLAP Downloads page.

Warning
In the available example the use of the Fast Reporting functions are included in a subroutine not a function, this is intentional.
You can not return an array from an external application to a VBA array within a function, instead you must use a subroutine. Error trapping and control can still be accomplished by the use of Public or Global variables.

mooFR

mooFR ([DOWN_DIM], [ACROSS_DIM], [OLAP VARIABLE])

The mooFR function returns a two dimensional array from a variable in Oracle OLAP. The contents of the array (table) passed back can then either be further processed in VBA or passed directly to a worksheet within Excel for reporting.

To ensure valid data is returned by this function it is the responsibility of the calling application to ensure that only the DOWN and ACROSS dimensions have one or more values in status.

If you are using mooFR against a variable with more than two dimensions you should limit the paging dimensions to only one value each.

mooFR can not be used against one dimensional variables, however mooFRDescDown or mooFRDescAcross can be used in these circumstances, even for numeric data.

mooFR observes the current status of the DOWN and ACROSS dimensions within the Oracle OLAP database and it is the responsibility of the calling application or user to set these appropriately.

Syntax

```
mooFR ([DOWN_DIM], [ACROSS_DIM], [OLAP VARIABLE])
```

Return Value

```
STRING
```

Example

```
array = (o.mooFR("CUSTOMER", "TIME", "UNITS_CUBE_COST")
```

mooFRDescDown

mooFRDescDown ([DIMENSION], [OLAP VARIABLE])

The mooFRDescDown function returns a one dimensional array on the Y axis from Oracle OLAP. The contents of the array (table) passed back can then either be further processed in VBA or passed directly to a worksheet within Excel for reporting.

mooFRDescDown is primarily used to return either row or column descriptive data, however, it can be used to return any one dimensional array back to the calling desktop application.

mooFRDescDOWN observes the current status of the DOWN dimension within the Oracle OLAP database and it is the responsibility of the calling application or user to set these appropriately.

mooFRDescDown should only be used against one dimensional variables

Syntax

```
mooFRDescDown ([DIMENSION], [OLAP VARIABLE])
```

Return Value

```
STRING
```

Example

```
array = (o.mooFRDescDown("CUSTOMER", "CUSTOMER.DESC")
```

mooFRDescAcross

mooFRDescAcross ([DIMENSION], [OLAP VARIABLE])

The mooFRDescAcross function returns a one dimensional array on the X axis from Oracle OLAP. The contents of the array (table) passed back can then either be further processed in VBA or passed directly to a worksheet within Excel for reporting.

mooFRDescAcross is primarily used to return either row or column descriptive data, however, it can be used to return any one dimensional array back to the calling desktop application.

mooFRDescAcross observes the current status of the Across dimension within the Oracle OLAP database and it is the responsibility of the calling application or user to set these appropriately.

mooFRDescAcross should only be used against one dimensional variables

Syntax

```
mooFRDescAcross ([DIMENSION], [OLAP VARIABLE])
```

Return Value

```
STRING
```

Example

```
array = (o.mooFRDescAcross("TIME", "TIME.DESC")
```

Example Application

Example application.

An example reporting GUI application is available for download. It utilises many of the functions of the myObjectiveOLAP API and includes use of the Fast reporting functions. This is available from the myObjectiveOLAP Downloads page.

The VBA application can be used against any native 10g or 11g Oracle OLAP standard Analytic Workspace without any change to the VBA, and can be used against non-standard Analytic Workspaces with little change to the VBA.

Using the example application

In order to use the example application you should connect to your Oracle OLAP environment using the connection method appropriate for your server installation.

You should then ensure you have the necessary Analytic Workspace attached by your database session, this can be accomplished through the Analytic Workspace Selector

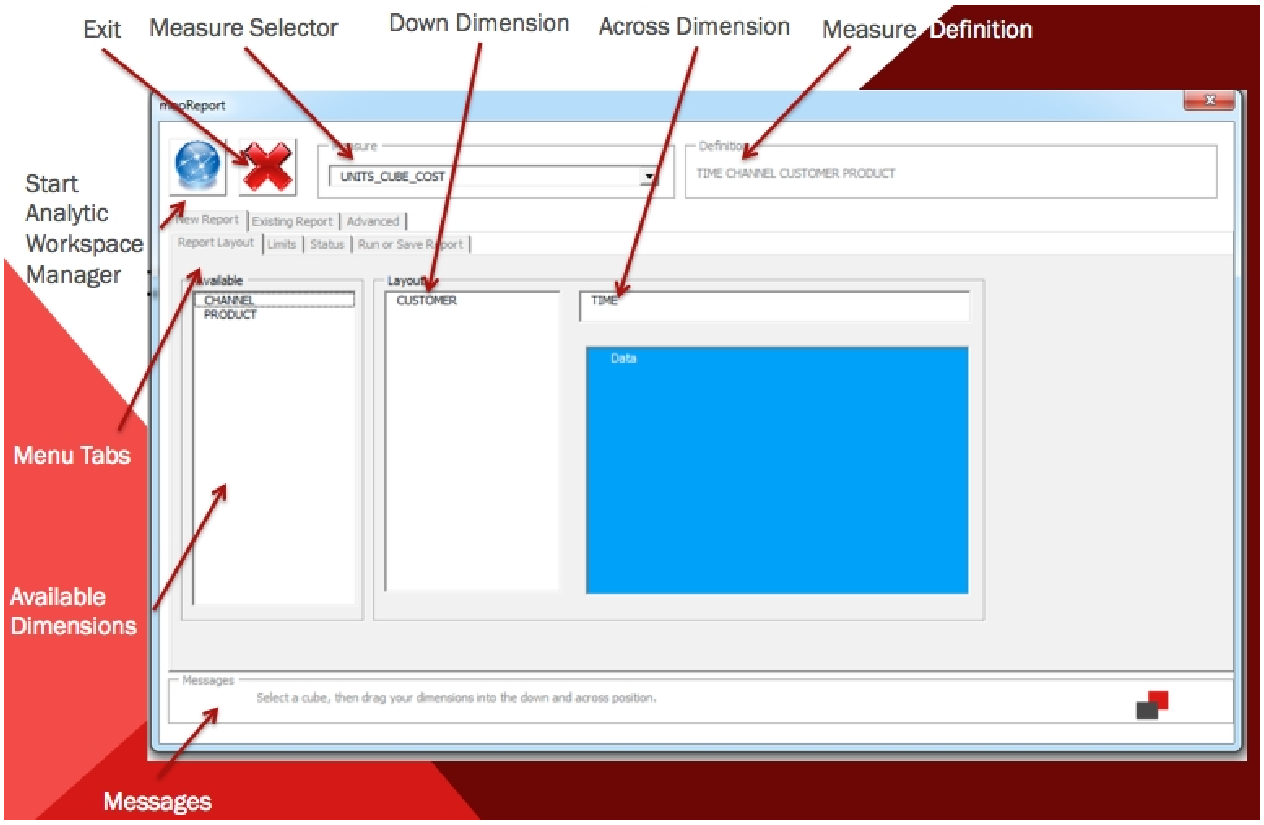
The application is designed as an example and can be easily customised for your specific installation requirement.

Hint

All of the screen-shots shown below use the Oracle provided GLOBAL example Analytic Workspace which can be downloaded from the Oracle website.

Main Window

When you start the example myObjectiveOLAP application the following New Report screen is displayed.



Select a cube, then drag your dimensions into the down and across position.

The following Main Tabs allow the user to:

Tab	Purpose
New Report	Define a new report, choose dimensional orientation, set limits, apply formatting, save the report, run the report
Existing Report	Apply an existing report, run an existing report, run all existing reports, delete existing reports
Advanced	Display the hidden worksheet which holds meta-data in relation to existing reports.

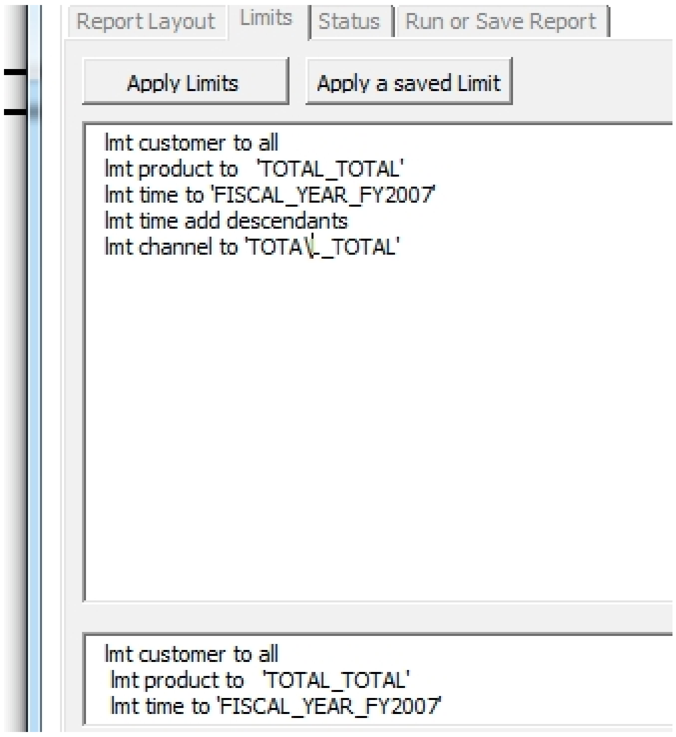
Table: myObjectiveOLAPHelp-2012-moo Fast Reporting-Example Application:1.0

New Report tab

When you want to create a new report you should select the measure you want to report on from the Measure drop-down list, choosing a measure will populate the Available Dimensions list. You can then drag your dimensions to the Down Dimension and Across Dimension layout area.

Limits tab

After selecting the Measure and orientation of your new report you should move over to the Limits tab. In the limits window you can type free form OLAP limit statements. Once you have entered your limit statement you should press the Apply Limits button. Any errors in your limit statement will be displayed in the window below. Alternatively you could choose a pre-saved text file with valid OLAP limit statements in it. To select a pre-saved file press the Apply a Saved Limit button, this will enable you to browse local file-systems available to you for a valid file. Pressing Open on a selected file will Apply the limits and any errors will be reported to you.

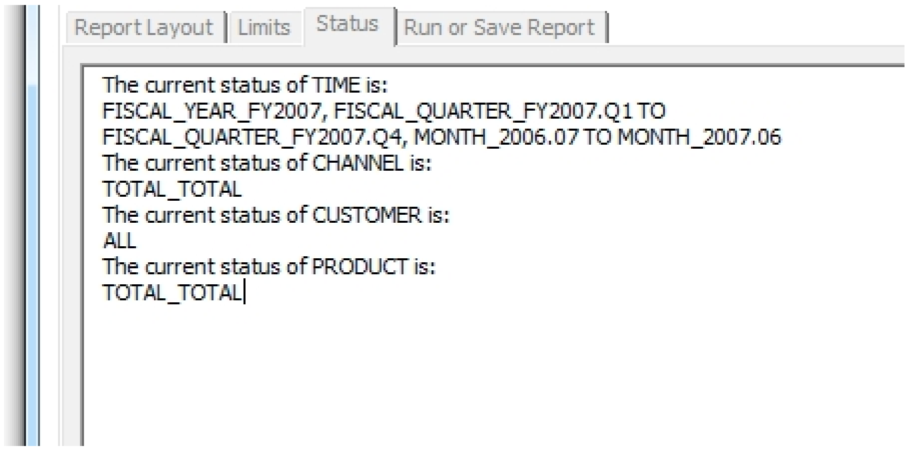


```
lmt customer to all
lmt product to  'TOTAL_TOTAL'
lmt time to 'FISCAL_YEAR_FY2007'
lmt time add descendants
lmt channel to 'TOTAL_TOTAL'
```

```
lmt customer to all
 lmt product to  'TOTAL_TOTAL'
 lmt time to 'FISCAL_YEAR_FY2007'
```

Status tab

After applying your limits you can check the Status of your selected Measure by navigating to the Status tab.

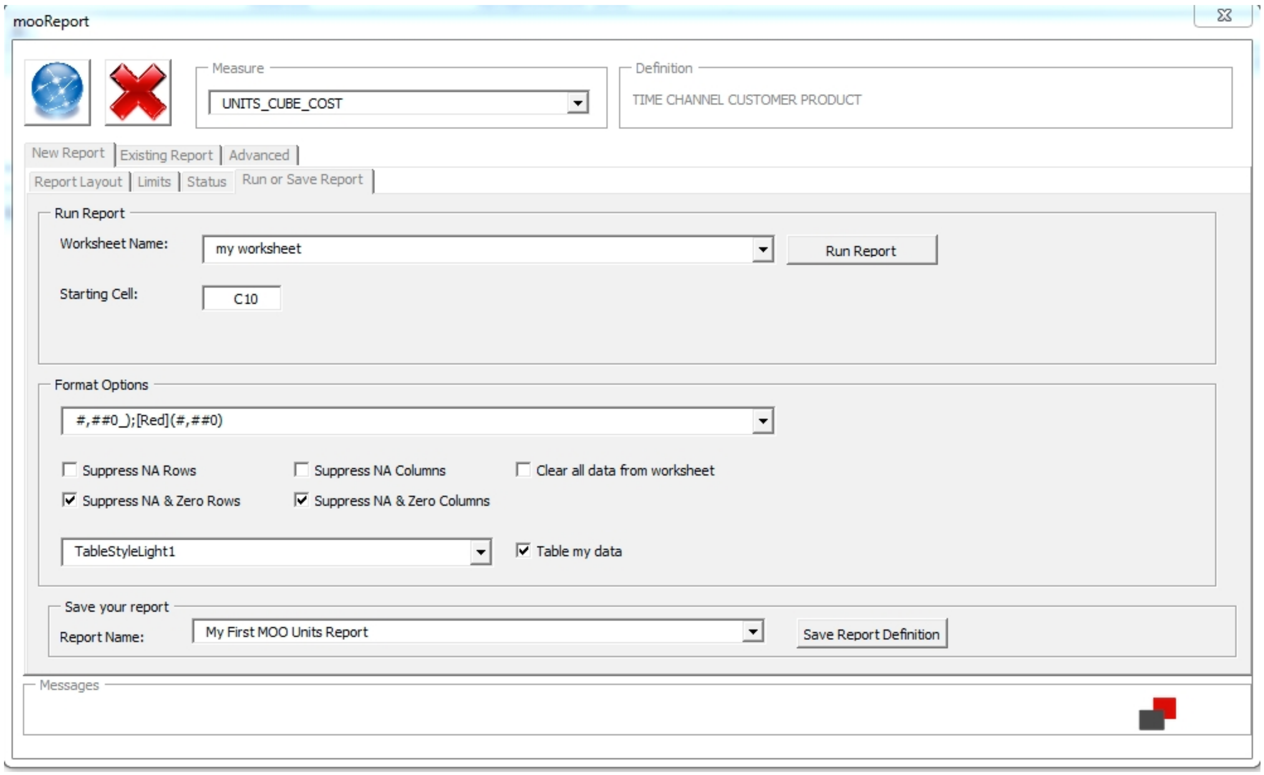


```
The current status of TIME is:
FISCAL_YEAR_FY2007, FISCAL_QUARTER_FY2007.Q1 TO
FISCAL_QUARTER_FY2007.Q4, MONTH_2006.07 TO MONTH_2007.06
The current status of CHANNEL is:
TOTAL_TOTAL
The current status of CUSTOMER is:
ALL
The current status of PRODUCT is:
TOTAL_TOTAL
```

Run or Save Report tab

After you are satisfied with your new report's orientation and applied limits you should select the Run or Save Report tab.

This tab enables you to select the destination Excel worksheet of your report, the starting position of your data, and a number of formatting options. You can then Run your report or choose to Save it.



Measure: UNITS_CUBE_COST
Definition: TIME CHANNEL CUSTOMER PRODUCT

Run Report
Worksheet Name: my worksheet
Starting Cell: C10

Format Options
#,##0_);[Red](#,##0)
☐ Suppress NA Rows ☐ Suppress NA Columns ☐ Clear all data from worksheet
☑ Suppress NA & Zero Rows ☑ Suppress NA & Zero Columns
TableStyleLight1 ☑ Table my data

Save your report
Report Name: My First MOO Units Report

The following table gives you an overview of the fields available in the Run or Save Report tab.

Field / Option	**Purpose**
Worksheet Name	This determines the destination Microsoft Excel worksheet for your report. This does not have to pre-exist.
Starting Cell	This is the cell of the first field of data from your measure. In most cases you will have dimensional descriptions in the X and Y axis on your report, so should choose cell B2 at a minimum.
Format Options Drop-down	Enables you to select the numerical cell format applied to your data. This only affects data retrieved from your measure and does not apply to your dimensional descriptions.
Suppress NA Rows	Adds an additional limit statement to keep only non-NA rows before running your report.
Suppress NA and Zero Rows	Adds an additional limit statement to keep only non-NA and non-zero rows before running your report
Suppress NA Columns	Adds an additional limit statement to keep only non-NA columns before running your report.
Suppress NA and Zero Columns	Adds an additional limit statement to keep only non-NA and non-zero columns before running your report
Clear all data from worksheet	This wipes the destination worksheet before publishing your report into it. This is useful if you have the row and column suppression switched on, as the scale of your report could significantly grow and shrink.
Table my Data Drown Down and Tick	This is only seen if the VBA behind the example application identifies that the Excel version is greater than or equal to Microsoft Excel 2010. If the "Table my data" box is selected then the selected Table Style from the drop down box will be applied to your published report.
Report Name	This is the name that refers to this specific orientation, limit, and formatting definition. Multiple reports could exist on a single

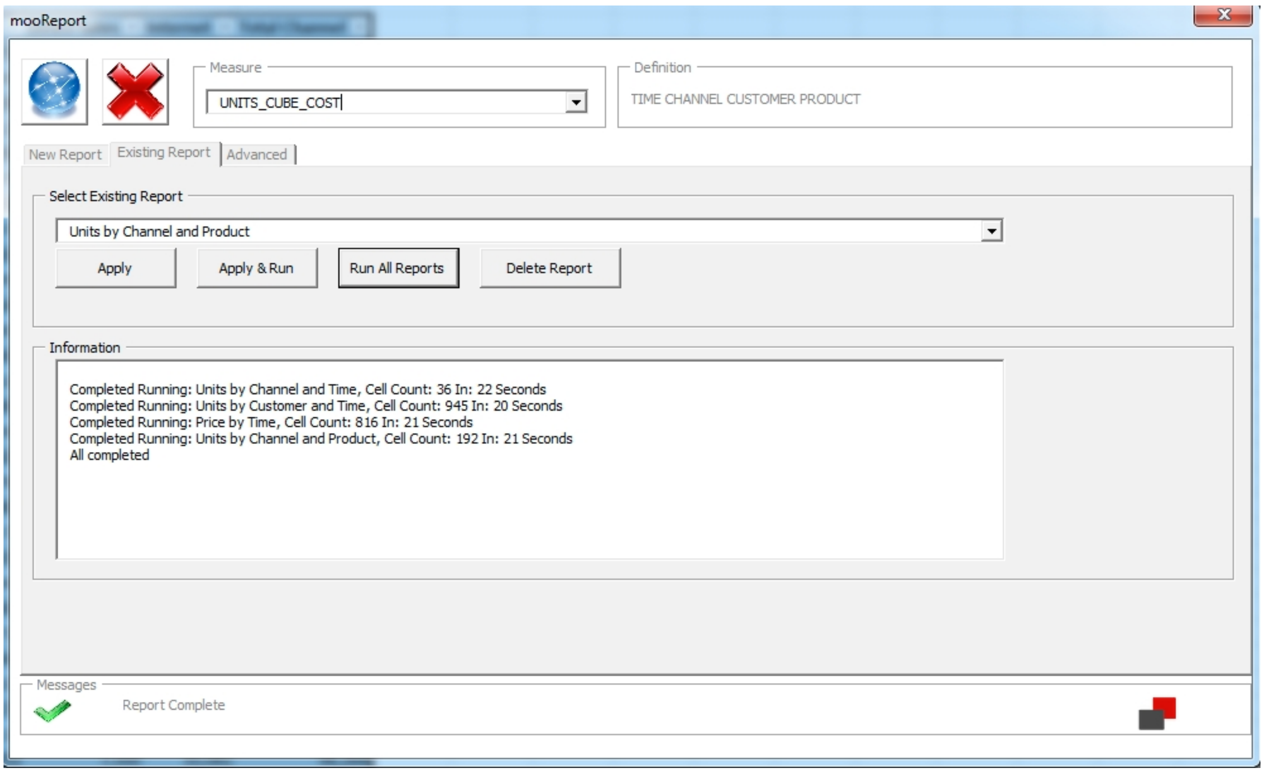
Table: myObjectiveOLAPHelp-2012-moo Fast Reporting-Example Application:2.0

Existing Report tab

You can select and run pre-saved reports from the Existing Report tab. Select the report you want to run or review and then select your desired option.

Option	Action
Apply	Apply settings of selected report and navigate me to the Run or Save Report tab
Apply & Run	Apply settings of selected report and run selected report.
Run All Reports	Run all reports available in this workbook.
Delete Report	Delete the selected report.

Table: myObjectiveOLAPHelp-2012-moo Fast Reporting-Example Application:3.0



UNITS_CUBE_COST

TIME CHANNEL CUSTOMER PRODUCT

Select Existing Report

Units by Channel and Product

Information

Completed Running: Units by Channel and Time, Cell Count: 36 In: 22 Seconds
Completed Running: Units by Customer and Time, Cell Count: 945 In: 20 Seconds
Completed Running: Price by Time, Cell Count: 816 In: 21 Seconds
Completed Running: Units by Channel and Product, Cell Count: 192 In: 21 Seconds
All completed

Report Complete

Advanced tab

The advanced tab enables you to unhide the hidden Excel worksheet which stores the meta-data definitions of your saved reports.

The example report uses a mixture of cell-contents and comment fields to store information relating to your saved report. The meta-data definitions are documented in the below *table (myObjectiveOLAPHelp-2012-moo Fast Reporting-Example Application:4.0)* and stored in a worksheet called mooFRStore. Each column holds the definition of one saved report.

Field	Data Type	Purpose
Row 1	STRING	Report name
Row 2	BOOLEAN	External Saved file. This will be TRUE if your limit statement is saved in an external text file, otherwise it will be FALSE
Row 3	STRIN	If your limit file is saved in an external text file this field will hold the directory location

	G	and filename.
Row 4	STRIN G	If your limit does not use an external file, but limit statements entered into the Limits tab your limit statements will be saved here.
Row 5	STRIN G	Worksheet name. The name of the worksheet which acts as a container for your report.
Row 6	STRIN G	Start Cell. The cell the data element of your report should start in.
Row 7	STRIN G	Down Dimension. The dimension whose orientation you have selected for the X axis of your report.
Row 8	STRIN G	Across Dimension. The dimension whose orientation you have selected for the Y axis of your report.
Row 9	STRIN G	Measure. The measure selected.
Row 10	BOOL EAN	Suppress NA rows.
Row 11	BOOL EAN	Suppress NA and Zero rows
Row 12	STRIN G	The selected cell formatting for the data element of your saved report.
Row 13	BOOL EAN	Suppress NA columns.
Row 14	BOOL EAN	Suppress NA and Zero columns
Row 15	BOOL EAN	Clear Worksheet before publishing new report
Row 16	BOOL EAN	Table my data, If you are using Excel 2010 or greater and this is set to TRUE then the example application will apply the Table Style as specified in Row 17.
Row 17	STRIN G	Table Style. If you are using Excel 2010 or greater and you have selected Table My Data then the Table Style will be saved in this field.

myObjectiveOLAPHelp-2012-moo Fast Reporting-Example Application:4.0

Setting OLAP Options

Common Options

The following functions can be used to set Oracle OLAP Server side options.

They are protected functions which use the myObjectiveOLAP sense-checking algorithm before being executed within the Oracle OLAP environment. Any errors are reported via: getLastMooErr

Documentation on the use of these options can be found in the “Oracle OLAP DML Reference Guide”

mooSetAwWaitTime

mooSetAwWaitTime(“ [integer value] ”)

Sets the Oracle OLAP AWWAITIME option

Syntax

```
mooSetAwWaitTime("[integer value]")
```

Where the [integer value] is the number of seconds required.
If zero is entered the value will be set to the default of 20.

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooSetAWWaitTime()

'Sets the AWWaitTime Option in Oracle OLAP
'If 0 is passed then the default 20 is applied

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not   o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
    boo = o.mooSetAWWaitTime("0")   ' seconds passed here
    boo = o.wrap_runNonQ("show AWWaitTime"): Debug.Print (o.mooGetDML)
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

Return Value Example

```
20
```

mooSetBadLine

mooSetBadLine("[yes|no]")

Sets the Oracle OLAP BADLINE option

Syntax

```
            mooSetBadline("[yes|no]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooSetBadLine()

'Sets the Badline Option in Oracle OLAP

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not   o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If
```

```
'Connected so execute
If boo = True Then
    boo = o.mooSetBadLine("YES")
    boo = o.wrap_runNonQ("show BadLine"): Debug.Print (o.mooGetDML)
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

Return Value Example

```
yes
```

mooSetCommas

mooSetCommas(“ [yes|no] ”)

Sets the Oracle OLAP COMMAS option

Syntax

```
            mooSetCommas("[yes|no]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooSetCommas()

'Sets the COMMAS Option in Oracle OLAP

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not   o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
    boo = o.mooSetCommas("NO")   ' YES or NO
    boo = o.wrap_runNonQ("show Commas"): Debug.Print (o.mooGetDML)
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

Return Value Example

```
no
```

mooSetDateFormat

mooSetDateFormat("[valid date format]")

Sets the Oracle OLAP DATEFORMAT option

Syntax

```
mooSetDateFormat("[valid date format]")
```

Return Value

```
BOOLEAN
```

Example

```
Public Sub mooSetDateFormat()

'Sets the DateFormat Option in Oracle OLAP

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not   o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
    boo = o.mooSetDateFormat("<DD><MTXT><YY>")   ' Format should be
<DD><MTXT><YY>
    boo = o.wrap_runNonQ("show DateFormat"): Debug.Print (o.mooGetDML)
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

Return Value Example

```
<DD><MTXT><YY>
```

mooSetDecimals

mooSetDecimals(“ [integer value] “)

Sets the Oracle OLAP DECIMALS option.

Syntax

```
mooSetDecimals("[integer value]")
```

Where the [integer value] represents the number of decimals places required

Return Value

```
BOOLEAN
```

Example

```
Public Sub mooSetDecimals()

'Sets the Decimals Option in Oracle OLAP
```

```
Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not   o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
    boo = o.mooSetDecimals("0")
    boo = o.wrap_runNonQ("show DECIMALS"): Debug.Print (o.mooGetDML)
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

Return Value Example

0

mooSetLikeCase

mooSetLikeCase(“ [yes|no] “)

Sets the Oracle OLAP LIKECASE option.

Syntax

```
            mooSetLikeCase("[yes|no]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooSetLikeCase()
'Sets the likecase Option in Oracle OLAP

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not   o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
    boo = o.mooSetLikeCase("NO")
    boo = o.wrap_runNonQ("show LIKECASE"): Debug.Print (o.mooGetDML)
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

Return Value Example

no

mooSetNASpell

mooSetNASpell

Sets the Oracle OLAP NASPELL option

Syntax

```
            mooSetNaSpell("[text|NA]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooSetNASpell()

'Sets the NASpell Option in Oracle OLAP

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
    boo = o.mooSetNASpell("0")
    boo = o.wrap_runNonQ("show na"): Debug.Print (o.mooGetDML)
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

Return Value Example

```
0
```

mooSetNASkip

mooSetNASkip("[yes|no]")

Sets the Oracle OLAP NASKIP option

Syntax

```
            mooSetNASkip("[yes|no]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooSetNASkip()

'Sets the NASpell Option in Oracle OLAP


Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not   o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
    boo = o.mooSetNASkip("YES")
    boo = o.wrap_runNonQ("show NASKIP"): Debug.Print (o.mooGetDML)
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

Return Value Example

```
yes
```

mooSetNASkip2

mooSetNASkip2("[yes|no]")

Sets the Oracle OLAP NASKIP2 option

Syntax

```
            moosetnaskip2("[yes|no]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooSetNASkip2()

'Sets the NASpell Option in Oracle OLAP

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not   o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
```

```
End If

'Connected so execute
If boo = True Then
    boo = o.mooSetNASkip2("NO")
    boo = o.wrap_runNonQ("show NASKIP2"): Debug.Print (o.mooGetDML)
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

Return Value Example

```
no
```

mooSetParens

mooSetParens(“ [yes|no]“)

Sets the Oracle OLAP PARENS option.

Syntax

```
            mooSetParens("[yes|no]")
```

Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooSetParens()
'Sets the parens Option in Oracle OLAP

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not   o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so execute
If boo = True Then
    boo = o.mooSetParens("no")
    boo = o.wrap_runNonQ("show PARENS"): Debug.Print (o.mooGetDML)
    Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

Return Value Example

no

Standard OLAP Graphical API

Standard OLAP Graphical API

The following functions can be called via the Microsoft Excel VBA model to display myObjectiveOLAP graphical components and forms.

CommandBar

commandBar

Displays a floating menu containing icons used to connect to or disconnect from a host, attach an analytic workspace via the AW Manager, open the OLAP Console and the help system.

Syntax

```
      commandBar
```

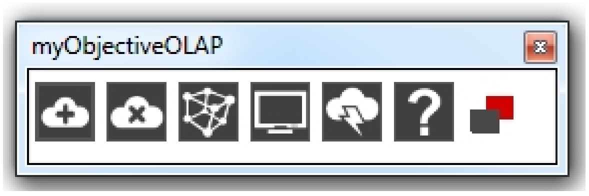
Return Value

```
            BOOLEAN
```

Example

```
Public Sub mooCommandBar()
'show the myObjectiveOLAP command bar

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:
boo = o.commandBar

End Sub
```

GUI Displayed:



mooCmd_line

mooCmd_line

Displays a command line interface that can be used to interact directly with the Oracle OLAP environment in a similar way to Oracle OLAP Worksheet or Oracle OX products.
The command line interface also offers access to MooScript which enables the developer to interact both with internal myObjectiveOLAP structures and Oracle OLAP structures through an automation layer.

Syntax

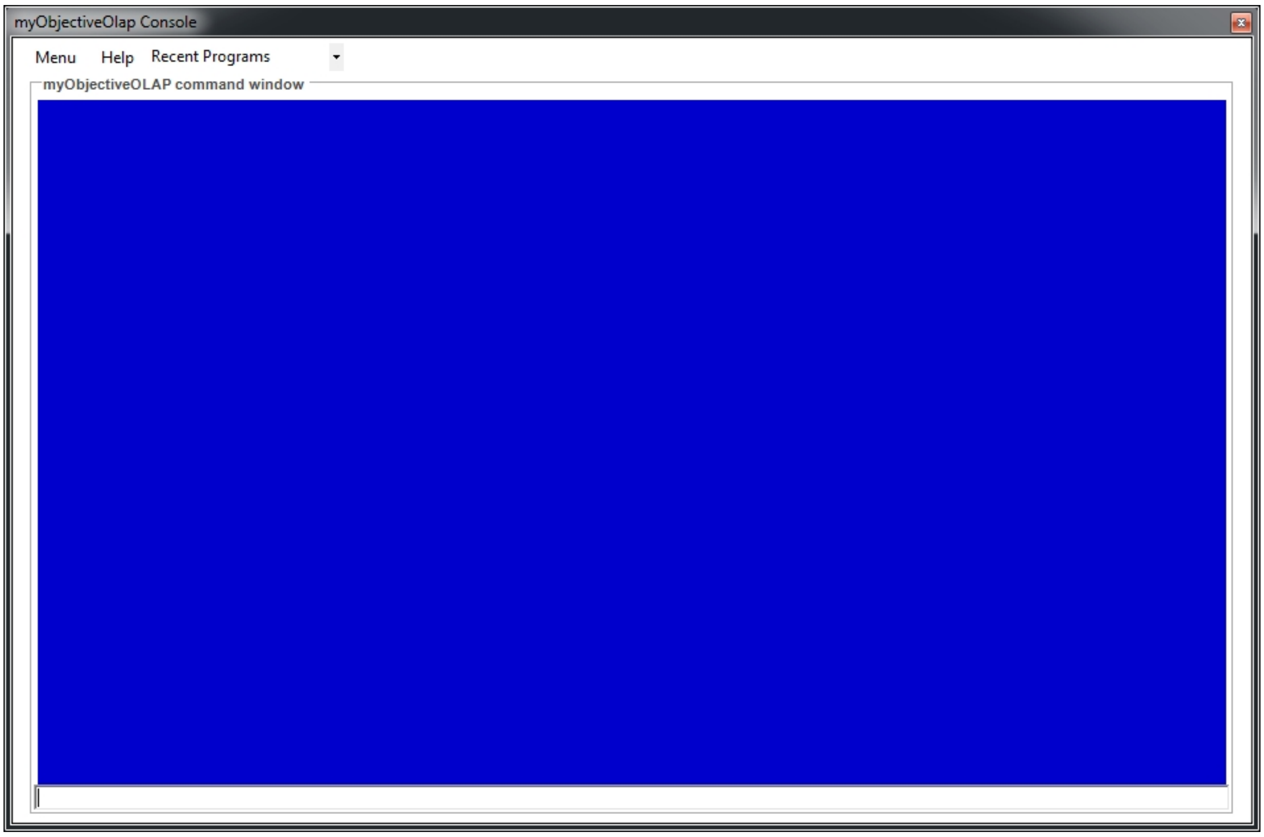
```
      mooCmd_line()
```

Example

```
Public Sub moo_CmdLine()
'Bring up the OLAP Console

Dim boo As Boolean
```

```
If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If


'Connected so bring up the command line
If boo = True Then
  boo = o.mooCmd_Line
Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If

End Sub
```

GUI Displayed:



mooShowConnFrm

mooShowConnFrm

Displays the myObjectiveOLAP Connection Editor Screen, this is used in creating a valid connection XML file for use with either the GUI connect or the connect function.

Syntax

```
mooShowConnFrm
```

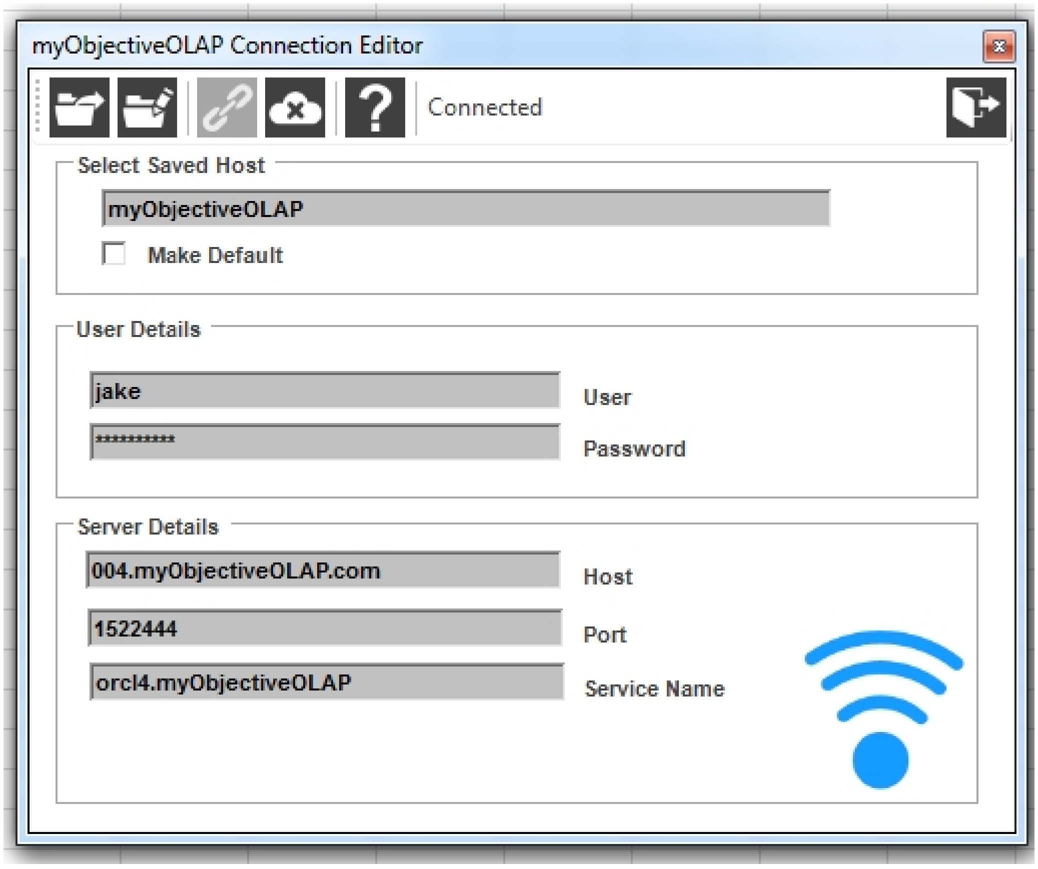
Return Value

```
      BOOLEAN
```

Example

```
Public Sub mooShowConnFrm()
'show the myObjectiveOLAP Connection Editor

    Application.COMAddIns.Item("myObjectiveOLAPXL.AddinModule").Object.mooShow
ConnFrm

End Sub
```

GUI Displayed:



myObjectiveOLAP

jake

****

004.myObjectiveOLAP.com

1522444

orcl4.myObjectiveOLAP

ShowAvailAW

ShowAvailAW

Displays a Selector style graphical attach tool that can be used to attach or detach available analytic workspaces.

Syntax

```
            showAvailAW()
```

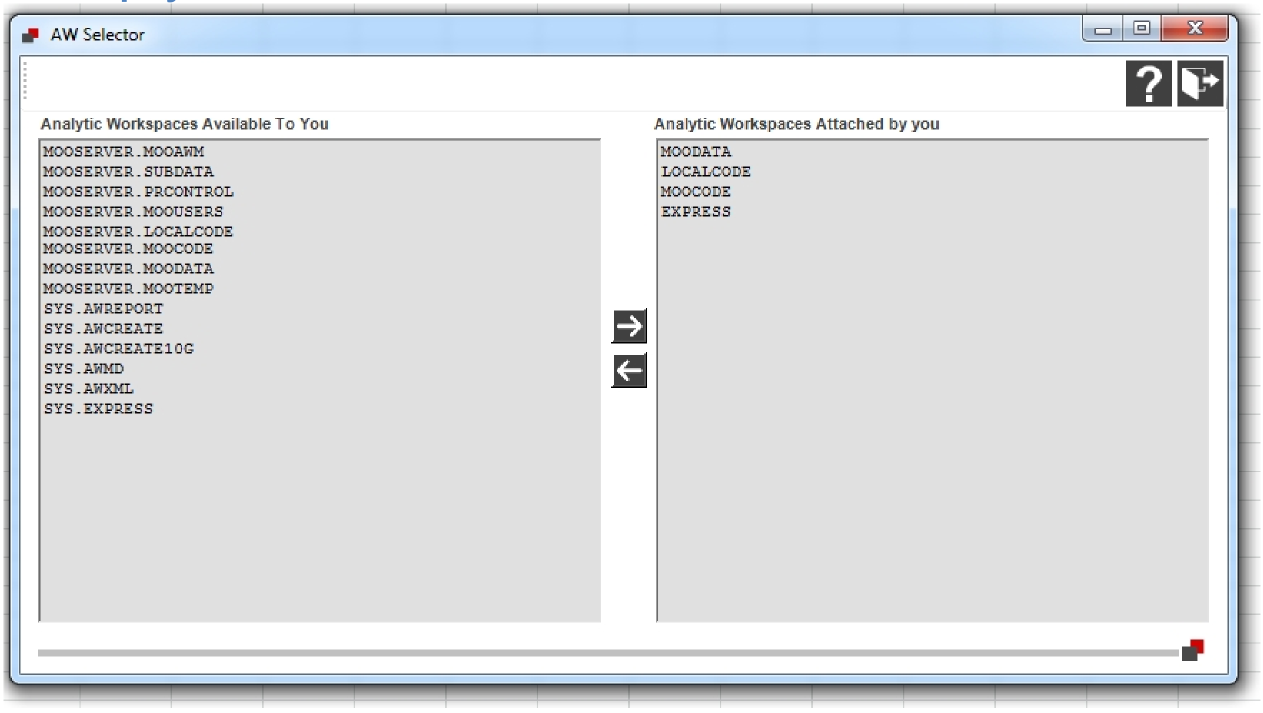
Return Value

```
            BOOLEAN
```

Example

```
Public Sub showAvailAW()
'Display the Analytic Workspace Selector

Dim boo As Boolean

If Not oregistered Then:   boo = regQ:

'Check Im connected if not connect
If Not o.connected Then
   boo = o.connect
   boo = o.connected
   Else
   boo = True
End If

'Connected so bring up the Analytic Workspace selector
If boo = True Then
  boo = o.showAvailAW
Else
   'Something went wrong print any error information
   Debug.Print "Unable to Connect"
   Debug.Print o.getlastmooerr
End If
End Sub
```

GUI Displayed:



Analytic Workspaces Available To You

```
MOOSERVER.MOOAWM
MOOSERVER.SUBDATA
MOOSERVER.PRCONTROL
MOOSERVER.MOOUSERS
MOOSERVER.LOCALCODE
MOOSERVER.MOOCODE
MOOSERVER.MOODATA
MOOSERVER.MOOTEMP
SYS.AWREPORT
SYS.AWCREATE
SYS.AWCREATE10G
SYS.AWMD
SYS.AWXML
SYS.EXPRESS
```

Analytic Workspaces Attached by you

```
MOODATA
LOCALCODE
MOOCODE
EXPRESS
```

myObjectiveOLAP Server

myObjectiveOLAP
Server

myObjectiveOLAP Server Overview

An optional server component can be installed to be used with the myObjectiveOLAP client.

The myObjectiveOLAP Server component offers a framework for migration of legacy Express, OFA, and OSA applications to Oracle OLAP 11g.
In addition it can be used by green-field Oracle OLAP application installations, or existing Oracle OLAP applications which would benefit from the following functionality.

Data Submission	Excel templates using the myObjectiveOLAP client can submit data back to Oracle OLAP, triggering post-load events such as aggregation, modeling or report generation. Full auditing of data submission is easily reportable and the API also allows for rollback of specific or all submissions. (1)
AW Data Loading	Data can be loaded directly into the Analytic Workspace environment without having to pass through the RDBMS layer. Data loads can be scripted and included as Local Processes within the myObjectiveOLAP work-flow Process Manager.
Meta-data management	Dimensional and structural maintenance can be scripted or manually maintained directly within the Analytic Workspace engine, without having to pre-populate the RDBMS layer.
User management	myObjectiveOLAP Server, handles user creation and rights management specific to an OLAP application.
Process Management	A robust and fully auditable Process Manager executes either Standard or locally defined processes.
OLAP DML Execution	Local processes can be created using the full Oracle OLAP DML programming syntax to manipulate, load or export your data. Many legacy applications can take advantage of this with minimal changes to the core business process logic. Maximizing your ROI.
Data Scoping	Data can be fully scoped to enable or disable access to your business data.
Work Flow	Business processes can be defined based on input actions.
Oracle OLAP standard computability	Once you have built your cubes and meta data, you can enable them for Oracle OLAP standard compatibility for SQL client access, or access from other tools, such as OBIEE, Oracle BI Spreadsheet Addin, or APEX.

Notes.

```
1)     Clients who have previously defined templates for use with Oracle
Express or Oracle Financial Analyzer
       using the "SDMC OFA Connect" software, can easily update their templates
for use with Oracle OLAP and
       myObjectiveOLAP Server.


2)     It is recommended that clients use Oracle OLAP 11.2.0.3 or greater.
```

Whilst the myObjectiveOLAP Server components are certified for use on 11.2.0.2 and below, clients may encounter issues with multiple level based hierarchies when attempting to maintain them within AWM. This issue has been resolved in 11.2.0.3 of the Oracle database.

Architecture

Installing myObjectiveOLAP Server

The following diagram shows a high level overview of the myObjectiveOLAP Server architecture

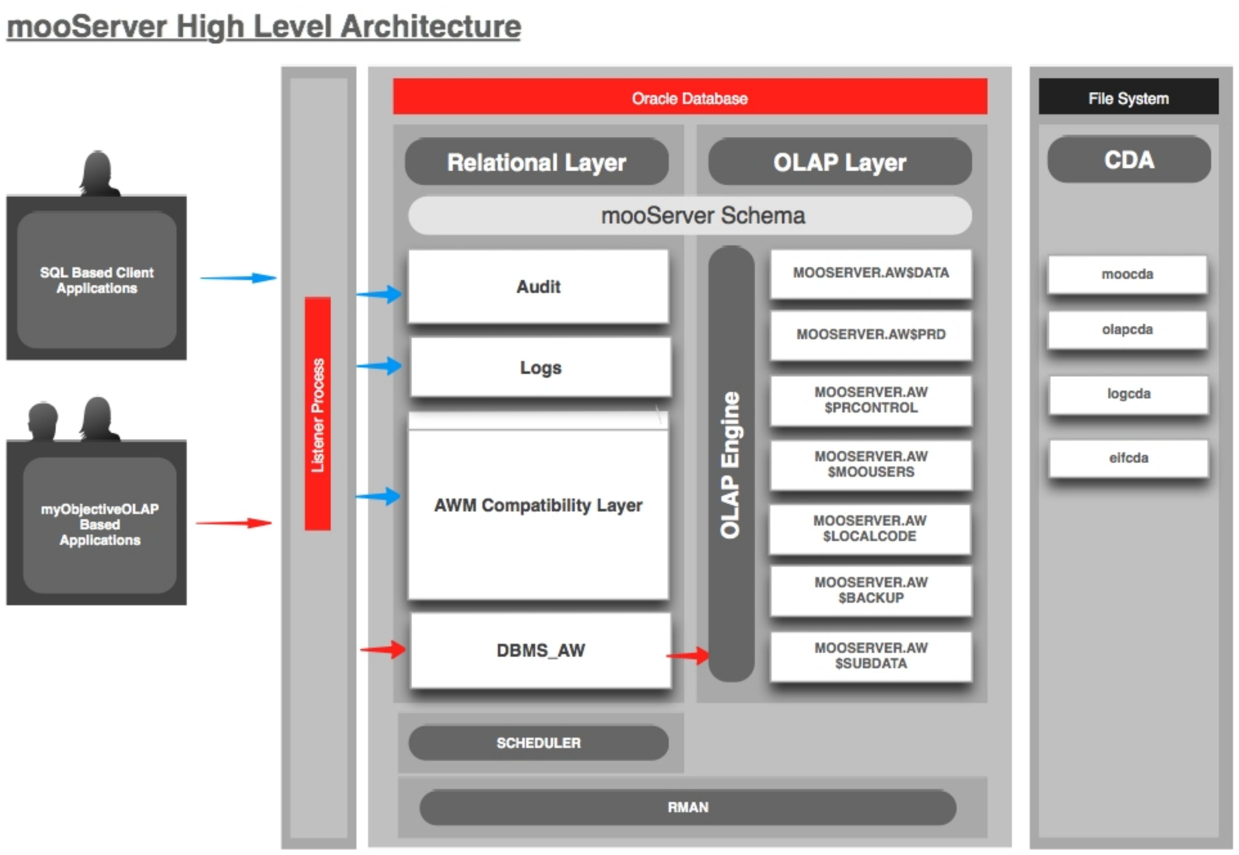


Table: myObjectiveOLAPHelp-2012-moo Fast Reporting-Example Application:1.0

Installing

Installing myObjectiveOLAP Server

Requirements

Database software

Oracle Database 11g with OLAP option.

Access

sqlPlus access as SYS (SYSDBA)

File system access on the same machine as sqlPlus.

```
File system access which is visible to the Oracle RDBMS and which Read Write
Directory aliases can be defined.
```

Overview

Two sql scripts are provided to install a base myObjectiveOLAP Server install:

Script	Purpose
mooSetupUser.sql	Defines two RDBMS users: MOOUSER and MOOSERVER. Grants the correct roles and responsibilities to these users.
mooSetupEnvironment.sql	Defines the base myObjectiveOLAP Server Analytic Workspace environment and imports the core structures from a series of supplied binary EIF files.

[myObjectiveOLAP-Server-Install-Table-1]

Getting Ready for Installation.

Before executing the following mooSetupUser.sql you should enable the OLAPSYS Oracle Database user:

If this is not a new Oracle OLAP installation, and you are adding to an existing environment then you may not need to complete this step.

This can be done by logging into the database as SYS as shown below:

```
moo12$ $ORACLE_HOME/bin/sqlplus / as SYSDBA

SQL*Plus: Release 11.2.0.3.0 Production on Wed Feb 1 20:10:04 2012

Copyright (c) 1982, 2011, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing
options

SQL>
SQL> GRANT CREATE SESSION       TO "OLAPSYS";
SQL> GRANT ALTER SYSTEM         TO "OLAPSYS";
SQL> commit;
SQL> EXIT
```

mooSetupUser.sql Overview

Users

mooSetupUser.sql defines two Oracle Database users:

Username	Purpose
MOOSERVER	This is the account used by the myObjectiveOLAP application administrator as the Oracle Database account, this is separate from the myObjectiveOLAP application user of the same name, although both are often used in conjunction. This user is assigned a custom Oracle Database profile called MOOSERVER as discussed below. This user is granted access to objects a normal user would not be able to access such as some of the directory aliases shown below.
MOOUSER	This is the primary Oracle Database account used as part of the connection by all normal end-users. This user is assigned a custom Oracle Database profile called

	MOOSERVER as discussed below.

[myObjectiveOLAP-Server-Install-Table-2]

Passwords

mooSetupUser.sql defines the passwords for the two Oracle Database users MOOSERVER and MOOUSER. You should change the password section of the script (RED) below before execution of the installation script:

Username	Purpose
MOOSERVER	-- Change the password below -- This is the username you use in an application DBA user db connection screen CREATE USER "MOOSERVER" PROFILE MOOSERVER IDENTIFIED BY "myObjectiveOLAP4321" ACCOUNT UNLOCK;
MOOUSER	-- Change the password below -- This is the username you use in a normal user db connection screen CREATE USER "MOOUSER" PROFILE MOOSERVER IDENTIFIED BY "myObjectiveOLAP4321" ACCOUNT UNLOCK;

[myObjectiveOLAP-Server-Install-Table-3]

Profiles

The following profile, based on the DEFAULT Oracle Database profile is created during the execution of the mooUserSetup.sql script. Differences to the DEFAULT profile are show in RED.

Profile	Adaptations From DEFAULT Profile
CREATE PROFILE "MOOSERVER" LIMIT CPU_PER_SESSION UNLIMITED CPU_PER_CALL UNLIMITED CONNECT_TIME UNLIMITED IDLE_TIME UNLIMITED SESSIONS_PER_USER UNLIMITED LOGICAL_READS_PER_SESSION UNLIMITED LOGICAL_READS_PER_CALL UNLIMITED PRIVATE_SGA UNLIMITED COMPOSITE_LIMIT UNLIMITED PASSWORD_LIFE_TIME UNLIMITED PASSWORD_GRACE_TIME 7 PASSWORD_REUSE_MAX UNLIMITED PASSWORD_REUSE_TIME UNLIMITED PASSWORD_LOCK_TIME DEFAULT FAILED_LOGIN_ATTEMPT	User expiration and failed attempt locking is delegated to the MOOSERVER application

S UNLIMITED PASSWORD_VERIFY_FUNCTION NULL;	

[myObjectiveOLAP-Server-Install-Table-4]

It is permissible to adapt the mooUserSetup.sql script to use the DEFAULT Oracle Database profile instead of the MOOSERVER profile, however, you must ensure that a new connection.xml file is distributed to the user base every time the password times-out. It would also be possible for an individual user to disable all users access if the DISABLE_PASSWORD_CHANGE tag is not enabled in your mooApplicationSettings.xml file.

Roles, Responsibilities and Permissions

The following roles, responsibilities and permissions are granted during execution of the mooUserSetup.sql installation script. Where appropriate a reason is included.

USER	Change	Reason
MOOUSER	CONNECT	
MOOUSER	OLAP_DBA	
MOOUSER	OLAP_USER	
MOOUSER	CREATE CUBE	
MOOUSER	CREATE SESSION	
MOOSERVER	UNLIMITED TABLESPACE	
MOOSERVER	CONNECT	
MOOSERVER	OLAP_DBA	
MOOSERVER	OLAP_USER	
MOOSERVER	CREATE CUBE	
MOOSERVER	CREATE SESSION	
MOOSERVER	ALTER SYSTEM	Required for OLAP Session Management
MOOSERVER	SELECT on DBA_SCHEDULER_JOBS	Required for OLAP Session Management
MOOSERVER	SELECT on DBA_SYS_PRIVS	
MOOSERVER	READ ON DIRECTORY moocda	
MOOSERVER	WRITE ON DIRECTORY moocda	
MOOSERVER	EXECUTE ON DIRECTORY moocda	
MOOSERVER	READ ON DIRECTORY logcda	
MOOSERVER	WRITE ON DIRECTORY logcda	
MOOSERVER	READ ON DIRECTORY olapcda	
MOOSERVER	WRITE ON DIRECTORY olapcda	
MOOSERVER	READ ON DIRECTORY eifcda	
MOOSERVER	WRITE ON DIRECTORY logcda	
MOOUSER	SELECT ON MOOSERVER.AW$MOODATA	
MOOUSER	SELECT ON MOOSERVER.AW$MOOUSER	
MOOUSER	SELECT ON MOOSERVER.AW$MOOCODE	
MOOUSER	SELECT ON MOOSERVER.AW$LOCALCODE	
MOOUSER	SELECT ON MOOSERVER.AW$MOORUNNING	
MOOUSER	SELECT ON MOOSERVER.AW$MOOUSERS	

MOOUSER	SELECT ON MOOSERVER.AW$MOOUSERS	
MOOUSER	SELECT ON MOOSERVER.AW$MOOAWM	
MOOUSER	SELECT ON MOOSERVER.AW$MOOBACKUP	
MOOUSER	SELECT ON MOOSERVER.AW$SUBDATA	
MOOUSER	SELECT ON DBA_SCHEDULER_JOBS	Enables the end-user to view if the Process Manager is running.
MOOUSER	Update ON MOOSERVER.AW$MOODATA	
MOOUSER	Update ON MOOSERVER.AW$MOOUSERS	
MOOUSER	Update ON MOOSERVER.AW$PRCONTROL	
MOOUSER	Update ON MOOSERVER.AW$MOOAWM	
MOOUSER	Update ON MOOSERVER.AW$SUBDATA	

[myObjectiveOLAP-Server-Install-Table-5]

** Unlike other parts of the installation script, whilst it is permissible to GRANT additional permissions to either MOOSERVER or MOOUSER it is NOT permissible to detract from the list above. Doing so may result in unexpected operation or failure of your myObjectiveOLAP Server based installation*

Directories

You should create at least one directory on the server for use by myObjectiveOLAP, the recommendation is to create three.
Your system or database administrator must ensure that the operating system directory has the correct permissions for the Oracle Database process.

The default location for the directory aliases is shown below:

Directory	**Default Location**	**Purpose**
moocda	/u01/ moocda	myObjectiveOLAP stores certain components on the file system temporarily during the execution of business logic, such as data submission.
olapcda	/u01/ logcda	Used by myObjectiveOLAP server when debugging is enabled.
logcda	/u01/ logcda	Various levels of logging can be enabled, optionally myObjectiveOLAP can output these logs as text to the logcda directory.
eifcda	/u01/ eifcda	This directory is required during the install, but it is also used as part of a number of standard tasks, such as automated backup.

[myObjectiveOLAP-Server-Install-Table-6]

** It is permissible for all the above directory aliases to physically be one file system directory, although it is recommended to separate them out for ease of analysis and maintenance.*

You should edit the following section of the mooSetupUser.sql script prior to installation to match your local environment:

```
-- Change the directory file system locations as required.
CREATE OR REPLACE DIRECTORY moocda as  '/u01/moocda';
CREATE OR REPLACE DIRECTORY olapcda as '/u01/logcda';
CREATE OR REPLACE DIRECTORY logcda as  '/u01/logcda';
CREATE OR REPLACE DIRECTORY eifcda as  '/u01/eifcda';
```

Running mooSetupUser.sql

Complete the following steps:

Step
Adjust the mooSetupUser.sql script as documented above
Copy the mooSetupUser.sql file to the directory you are going to start sqlPlus in.
Start sqlPlus as Oracle user
Execute the mooSetupUser.sql script
If there are no errors then type COMMIT; [ENTER] or if you wish to exit without committing ROLLBACK; [ENTER]

[myObjectiveOLAP-Server-Install-Table-7]

Example below:

```
moo12$ $ORACLE_HOME/bin/sqlplus / as SYSDBA

SQL*Plus: Release 11.2.0.3.0 Production on Wed Feb 1 20:10:04 2012

Copyright (c) 1982, 2011, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing
options

SQL>
SQL> @mooSetupUser.sql
Grant succeeded.
SQL>
Grant succeeded.

SQL>
Grant succeeded.

SQL>
Grant succeeded.
....[repeated]...
SQL>
SQL> commit;
SQL> EXIT
```

mooSetupEnvironment.sql Overview

The mooSetupEnvironment.sql installation script creates the Analytic Workspaces required to install the base myObjectiveOLAP Server objects and programs.

You should extract the contents of myObjectiveOLAPServer2012-R[XX].zip and transfer them to the physical file system location you defined for your eifcda Directory Alias as part of mooSetupUser.sql installation step.

Files used during this part of the installation:

File	Purpose
mooSetupEnvironement.sql	Installation script for the myObjectiveServer framework. Can also be used to revert an existing application to a new install state.
2012-Release-[XX]-MOODATA.eif	Data and meta-data for use within the main data storage and reporting component of the environment.

2012-Release-[XX]-PRD.eif	Objects and programs relating to the Process Manager daemon.
2012-Release-[XX]-PRCONTROL.eif	Data and meta-data for use by the Process Manager framework
2012-Release-[XX]-MOOUSERS.eif	Data and meta-data for use by the User Management framework.
2012-Release-[XX]-MOOCODE.eif	Meta-data and Oracle OLAP DML. This is the core program component of the myObjectiveOLAP Server framework.
2012-Release-[XX]-LOCALCODE.eif	Mata-data for housing localized code.
2012-Release-[XX]-MOOBACKUP.eif	Data and meta-data for automated backup of the environment.

[myObjectiveOLAP-Server-Install-Table-8]

Re-running mooSetupEnvironment.sql

The mooSetupEnvironment.sql installation script can be run at anytime to set your myObjectiveOLAP Severer environment to a new install state.
!IMPORTANT! -- This will result in the loss of all data.

Objects and Workspaces Created during Installation mooSetupEnvironment.sql

The following table lists the objects and workspaces created during the execution of the mooSetupEnvironment.sql installation script.

Workspace / Object	Purpose
MOOSERVER.AW$DATA	Analytic Workspace designed to store data and act as the main reporting repository Meta-data relating to data objects created within the
MOOSERVER.AW$PRD	Analytic Workspace holding objects and OLAP DML programs relating to the myObjectiveOLAP Process Manager daemon
MOOSERVER.AW$PRCONTROL	Analytic Workspace holding objects to store control and audit data for the Process Manager
MOOSERVER.AW$MOOUSERS	Analytic Workspace holding objects relating to users, user control, audit and data scoping.
MOOSERVER.AW$MOOCODE	Analytic Workspace holding the core myObjectiveOLAP Server OLAP DML programs
MOOSERVER.AW$LOCALCODE	Empty Analytic Workspace in which custom local programs can be stored. This workspace is not impacted by the myObjectiveOLAP upgrade process.
MOOSERVER.AW$MOOBACKUP	Analytic Workspace holding objects and OLAP DML relating to automated backups of the myObjectiveOLAP Server environment.
MOOSERVER.AW$SUBDATA	Empty Analytic Workspace used during the processing of user submissions of data and holds data audit and rollback data.
MOOSERVER.AW$MOOTEMP	Empty Analytic Workspace used temporarily during execution of business processes also used as a staging area during upgrade of myObjectiveOLAP Server.

Running mooSetupEnvironment.sql

Complete the following steps:

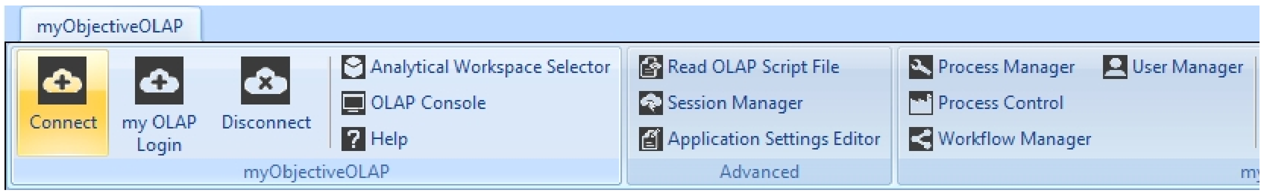
Step
Copy the EIF files listed in myObjectiveOLAP-Server-Install-Table-8 to the directory you defined for eifcda in table `myObjectiveOLAP-Server-Install-Table-6`
Copy the mooSetupUser.sql file to the directory you are going to start sqlPlus in.
Start sqlPlus as MOOSERVER user
Execute the mooSetupEnvironment.sql script
If there are no errors then type COMMIT; [ENTER] or if you wish to exit without committing ROLLBACK; [ENTER]

Example below:

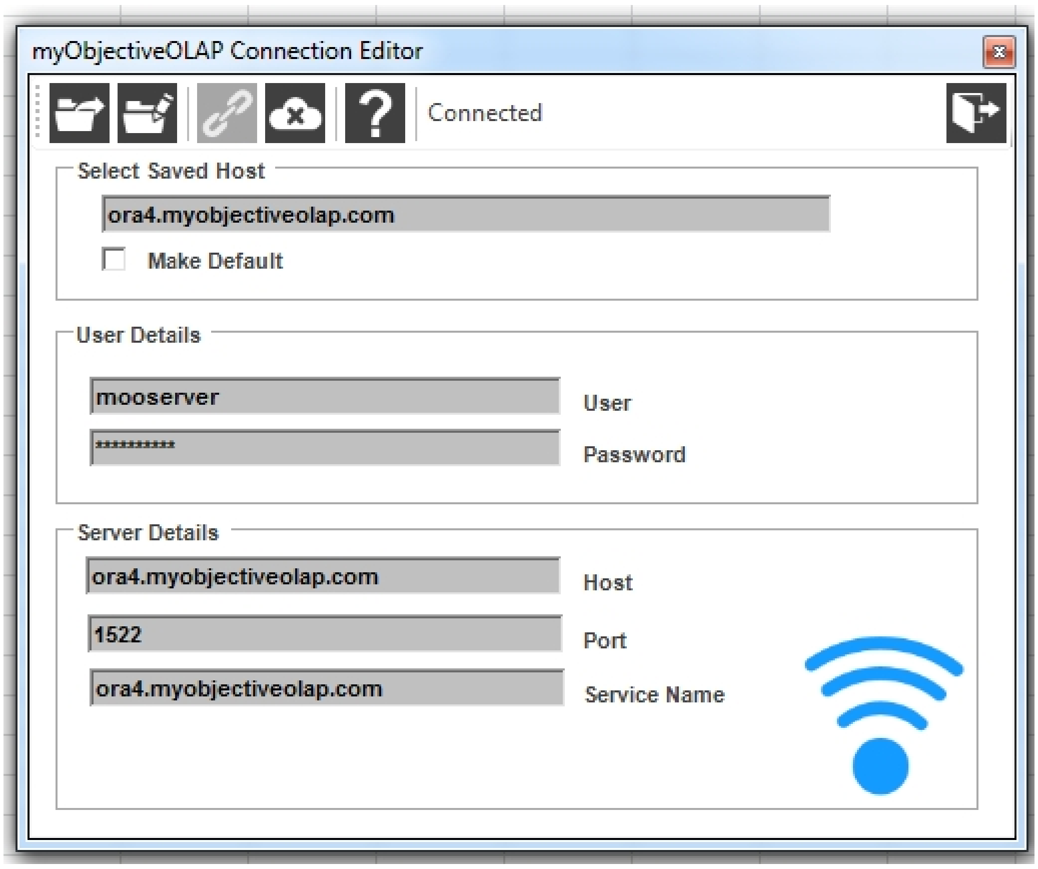
```
moo12$ $ORACLE_HOME/bin/sqlplus

SQL*Plus: Release 11.2.0.3.0 Production on Sun Feb 5 16:33:40 2012

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Enter user-name: mooserver
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing
options

SQL>
SQL> @mooSetupEnvironment.sql

{OUTPUT SHOWN}
{OUTPUT SHOWN}
{OUTPUT SHOWN}
{OUTPUT SHOWN}



....[repeated]...
SQL>

! MOOSERVER:>> If this is the first time you have run this you will see
Analytic workspace XXX does not exist. Ignore these errors
! MOOSERVER:>> If you did not see any Errors? You should now issue a COMMIT
or you could ROLLBACK
SQL> commit;
SQL> EXIT
```

Setting the initial mooserver application password.

Start Microsoft Excel and press the standard Connection Editor from the myObjectiveOLAP menu group



Complete the appropriate details for the server you have installed myObjectiveOLAP Server on.



Select Saved Host

ora4.myobjectiveolap.com

Make Default

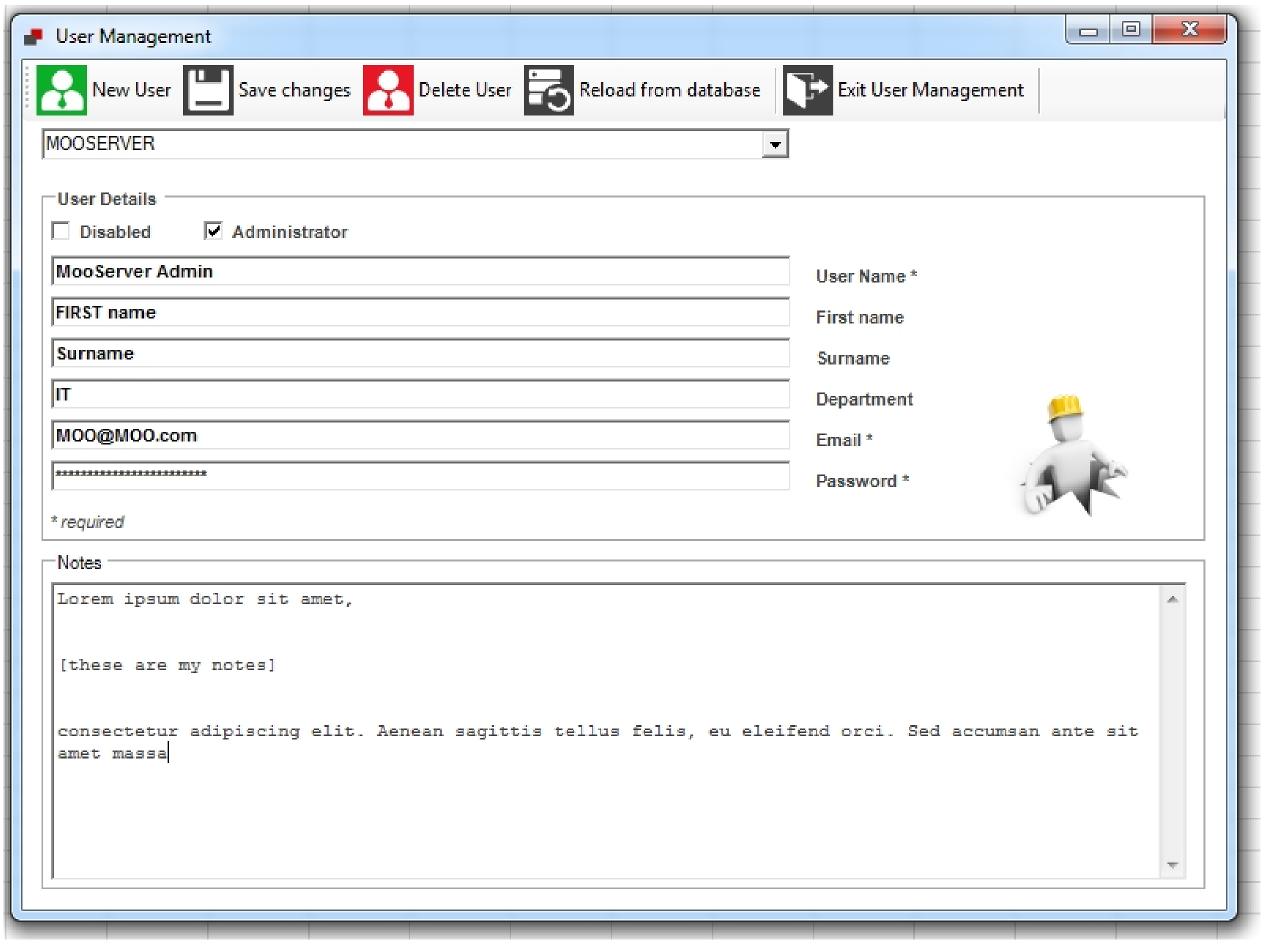
User Details

mooserver	User
**********	Password

Server Details

ora4.myobjectiveolap.com	Host
1522	Port
ora4.myobjectiveolap.com	Service Name

Select the User Manager menu item from the myObjectiveOLAP menu group.



MOOSERVER

User Details

Disabled [✓ Administrator]

User Name *	MooServer Admin
First name	FIRST name
Surname	Surname
Department	IT
Email *	MOO@MOO.com
Password *	************************

* required

Notes

Lorem ipsum dolor sit amet,

[these are my notes]

consectetur adipiscing elit. Aenean sagittis tellus felis, eu eleifend orci. Sed accumsan ante sit amet massa

Ensure the MOOSERVER user is selected, enter a new password and press Save Changes from the myObjectiveOLAP Ribbon menu.

Your myObjectiveOLAP Server installation is now complete.

Connecting

mooServer Connection

A myObjectiveOLAP mooServer Connection supports additional server side work flow, data submission and reporting tools.
This type of connection should only be used with a mooServer enabled environment.

For more information on connecting please see here.

Administration

myObjectiveOLAP Server Administration

myObjectiveOLAP Server contains a number of features to help you manage your application.

The following table's purpose is to give you a quick overview of the features.

Component	Purpose
Process Builder	Process Builder allows you to: Run and Schedule Processes Edit or delete existing Process

	Explain the Execution Plan of an existing Process
Process Manager	The Process Manager window controls the mooServer Process Manager.
Workflow Builder	Workflow Builder allows you to group Processes or Workflows together into a single workflow which can either be submitted for execution manually through Process Builder, or can be configured to respond to an event.
User Management	User manager allows you to carry out the following actions: Create new users, disable users, reset user passwords, delete users, update user details
Oracle OLAP Standard Compatibility	The AWM Compatibility Layer tool automatically builds Relational Views and Tables based on your myObjectiveOLAP multi-dimensional model. The relational model can then be used to easily access multi-dimensional information from native SQL tools such as SQL Developer, Toad, or more advanced reporting applications like Oracle BI Enterprise Edition (OBIEE).
Creating or modifying a dimension	myObjectiveOLAP Server enables the creation and maintenance of dimensions within the MOODATA analytic workspace.
Creating or modifying a cube	myObjectiveOLAP Server enables the creation and maintenance of cubes within the MOODATA analytic workspace.
Values, Hierarchies, Attributes	Whilst in most systems maintenance of Dimension Values and there associated meta-data is carried out through interfaces (Relational or OLAP), myObjectiveOLAP Server additionally allows for creation of this meta-data information through Excel, this enables both ad-hoc maintenance and easy bulk-creation or adaptation of this data.

Process Management

myObjectiveOLAP Server Process Management

myObjectiveOLAP Server contains a sophisticated Process Management model.

This section takes you through the steps required to understand the Process Management model, create and edit Processes, define and edit Workflows.

Process Builder

Process Builder

Process Builder

Process Builder allows you to:

Run and Schedule Processes

Edit or delete existing Process

Explain the Execution Plan of an existing Process

Refresh	Refreshes the list of processes from the database
Submit Process	Submits a process for execution by the Process Manager.
Save Changes	Saves any changes you have made to an existing Process.

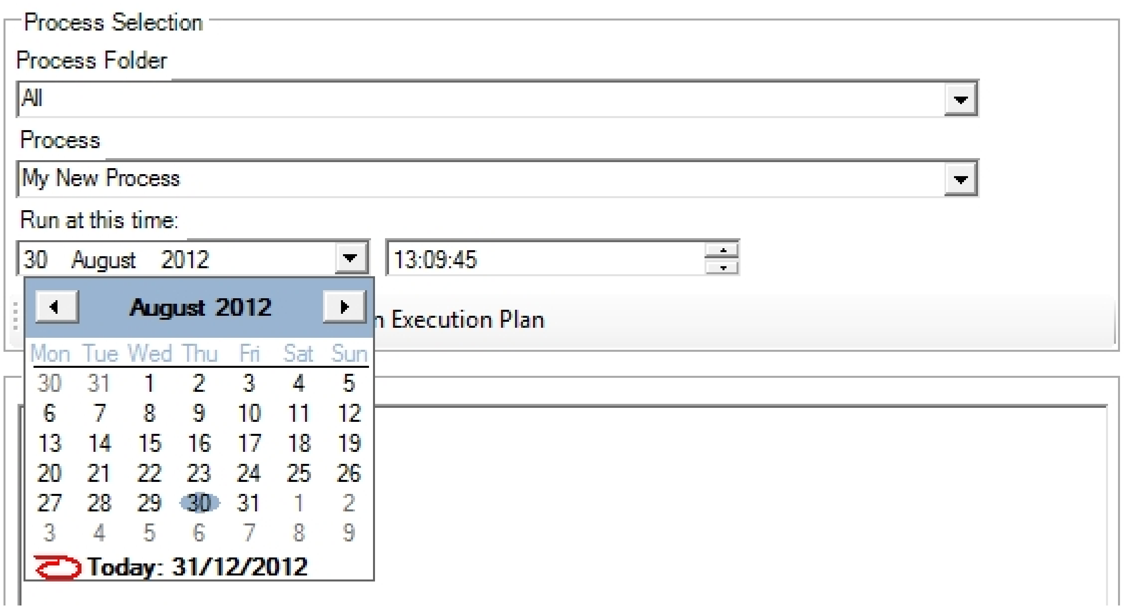
Copy Process	Allows you to create a new process based on an existing Process.
Delete Process	Deletes an existing Process.
New process	Allows you to define a new Process.
Process Manager	Opens the Process Manager window.
Workflow Builder	Opens the Workflow Builder window.
Help	Opens the myObjectiveOLAP Help file.
Close	Closes the Process Builder window.

Process Selection

Allows you to choose a process. Process can be stored in virtual folders to help you group related processes into easy to find Process Groups.

Once you have selected a process you wish to execute, you can schedule the process using the "Run at this time" Calendar and Clock interface.

If you want it to run immediately, you can reset the clock to "now" by just pressing the Set Time to Now button.



Process Selection

Process Folder: All

Process: My New Process

Run at this time: 30 August 2012 | 13:09:45

August 2012

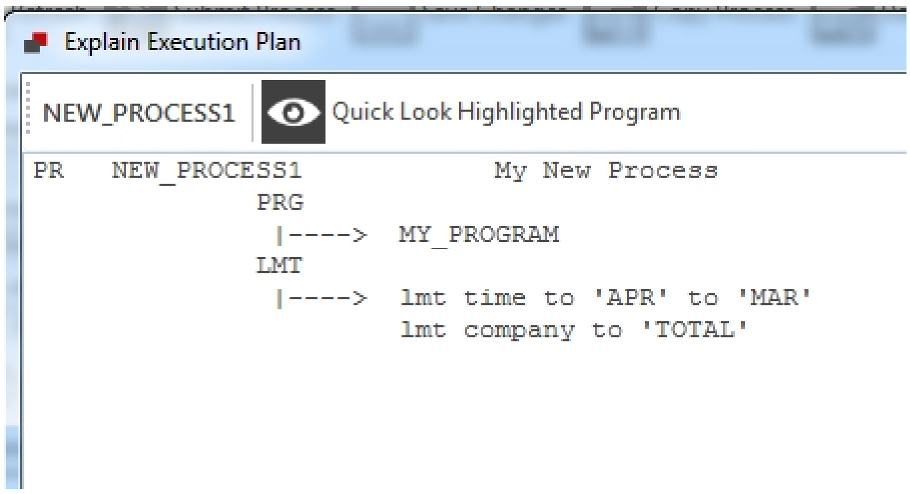
Mon	Tue	Wed	Thu	Fri	Sat	Sun
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Today: 31/12/2012

Execution Plan

Once you have selected a process you can view Execution Plan for the selected Process or Workflow by pressing the, Explain Execution Plan button.

The Execution Plan window allows you to see at a glance what is going to be executed by the selected Process. The viewer also shows you any Limits that will be applied to the Process.



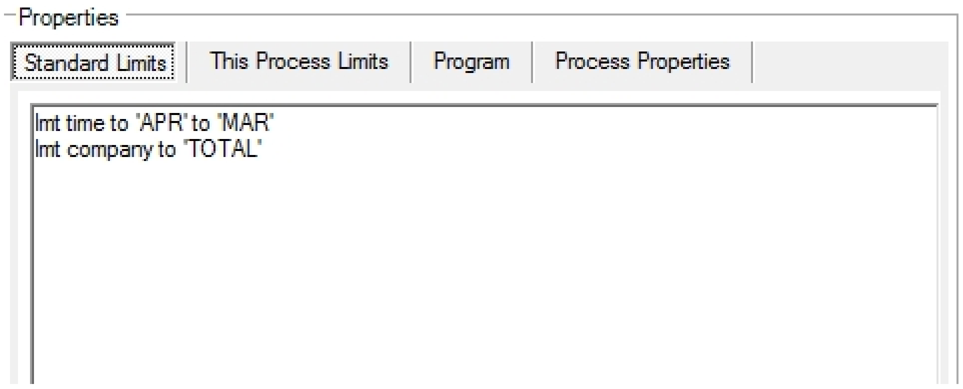
```
PR   NEW_PROCESS1            My New Process
               PRG
                |---->  MY_PROGRAM
               LMT
                |---->  lmt time to 'APR' to 'MAR'
                        lmt company to 'TOTAL'
```

If you are running a program as part of your process, you can highlight a program, and press the

will bring up the OLAP DML program viewer.

Properties

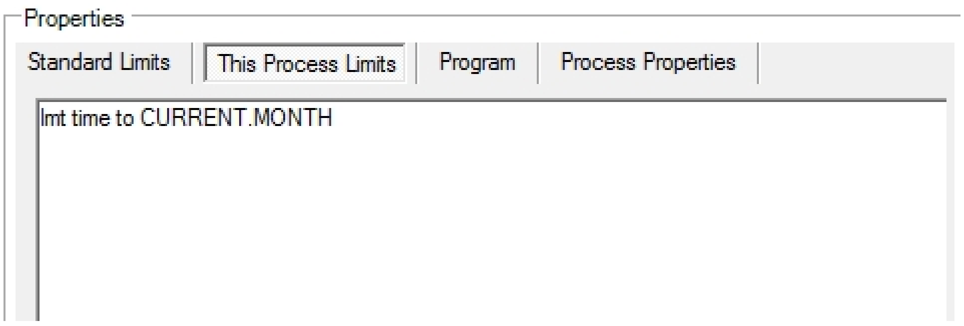
Standard Limits



lmt time to 'APR' to 'MAR'
lmt company to 'TOTAL'

Standard Limits, allows you to specify any Limits you wish to be applied when the Process is executed. As well as LIMIT statements you can enter any free-form OLAP DML statement you wish to be executed. Standard limits are part of the Process and usually are set as persistent.

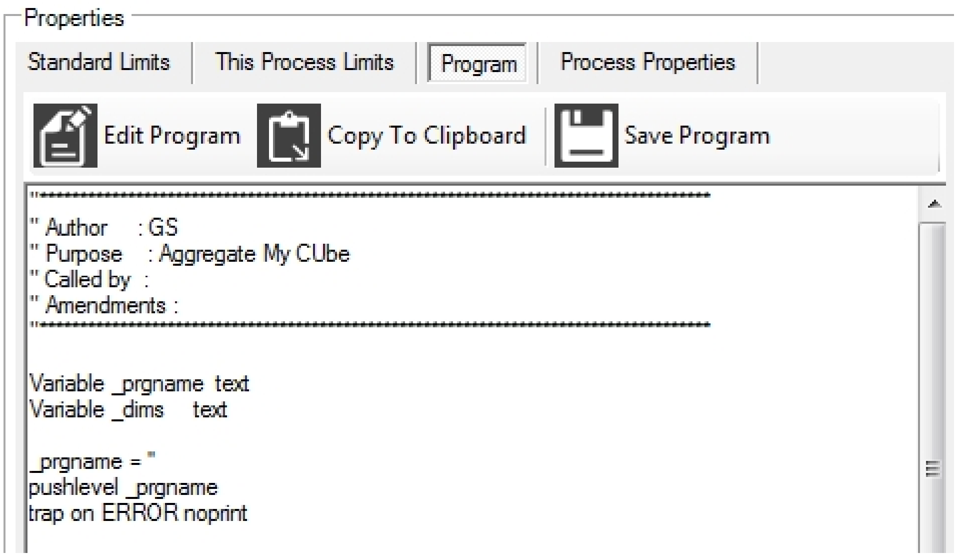
This Process Limits



lmt time to CURRENT.MONTH

This Process Limits, are non-persistent limits and are only executed once when you Submit the process this time. This Process Limits override any Standard Limits.

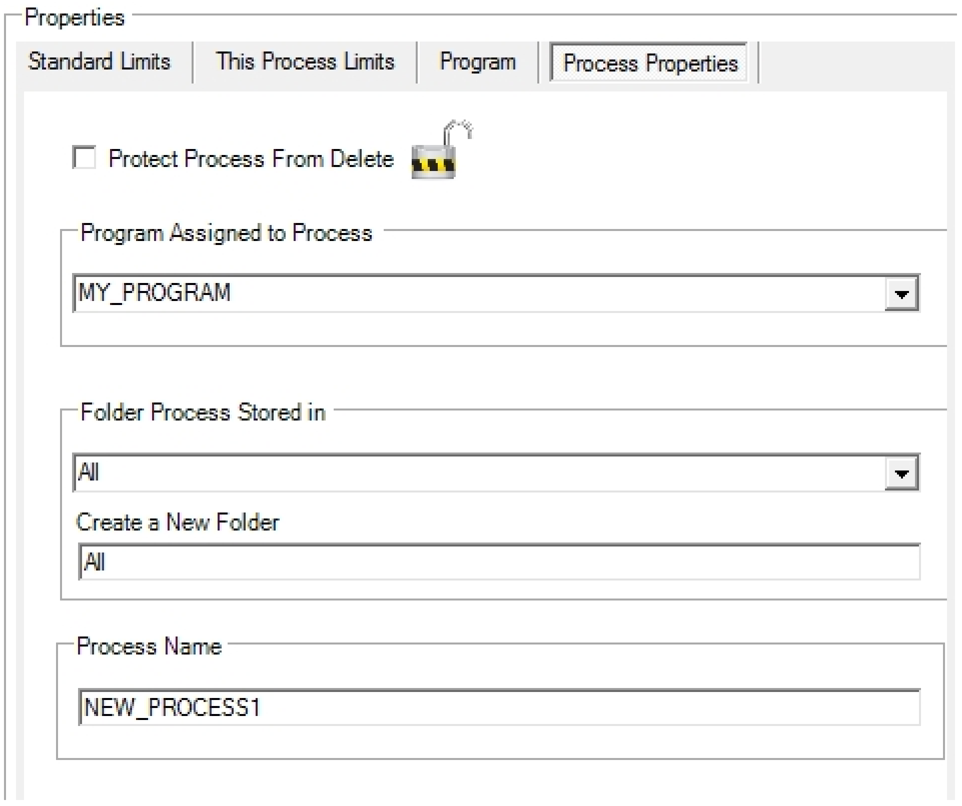
Program



```
"***************************************************************************
" Author     : GS
" Purpose    : Aggregate My CUbe
" Called by  :
" Amendments :
"***************************************************************************

Variable _prgname  text
Variable _dims     text

_prgname = ''
pushlevel _prgname
trap on ERROR noprint
```

Program allows you to View, Edit & Save, Copy to Clipboard a program attached to the selected Process.

Process Properties



☐ Protect Process From Delete

Program Assigned to Process: MY_PROGRAM

Folder Process Stored in: All

Create a New Folder: All

Process Name: NEW_PROCESS1

Protect Process From Delete

Processes protected from delete can not be deleted until the Lock is removed. This is designed to stop accidental deletion.

Program Assigned to Process

This shows you the OLAP DML program assigned to a Process and allows you to change the assigned program, press Save Changes after updating the assigned program.

Folder Process Stored In

Folder Process Stored In, is the name of a virtual Folder which the process is stored in. All Processes are always stored in the "All" virtual folder.

Create a New Folder

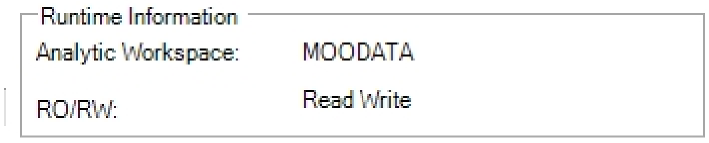
Create a New Folder, allows you to specify a name of a new Folder and assign it to a Process. If no processes remain in a Folder, the folder automatically is purged. A folder can be created again and processes assigned to it.

Process Name

The name of the process. This can not be changed, to change the name of a Process, copy the process, creating it with the desired name and then delete the original Process.

Runtime Information

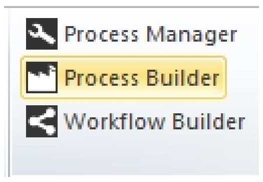
The runtime information panel indicates which Analytic Workspace the Process or Workflow is designed to be executed within.


Runtime Information
Analytic Workspace: MOODATA
RO/RW: Read Write

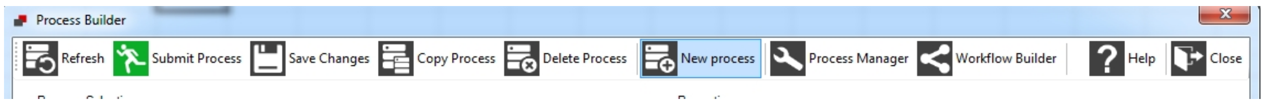
Defining a New Process

Defining a New Process

To define a new Process, select Process Builder from the myObjectiveOLAP tool bar.



From within Process Builder select "New Process"



Complete the following information:

Process ID
A unique identifier of your choice for the process you are creating. This will be used as the PR.ROW value within AW$PRCONTROL

Process Description

A description for your new process

Program Name

The name of either a MOOSERVER standard program such as MOO.AGGREGATE.CUBE (see the mooServer API) or a custom program of your own.

Analytic Workspace

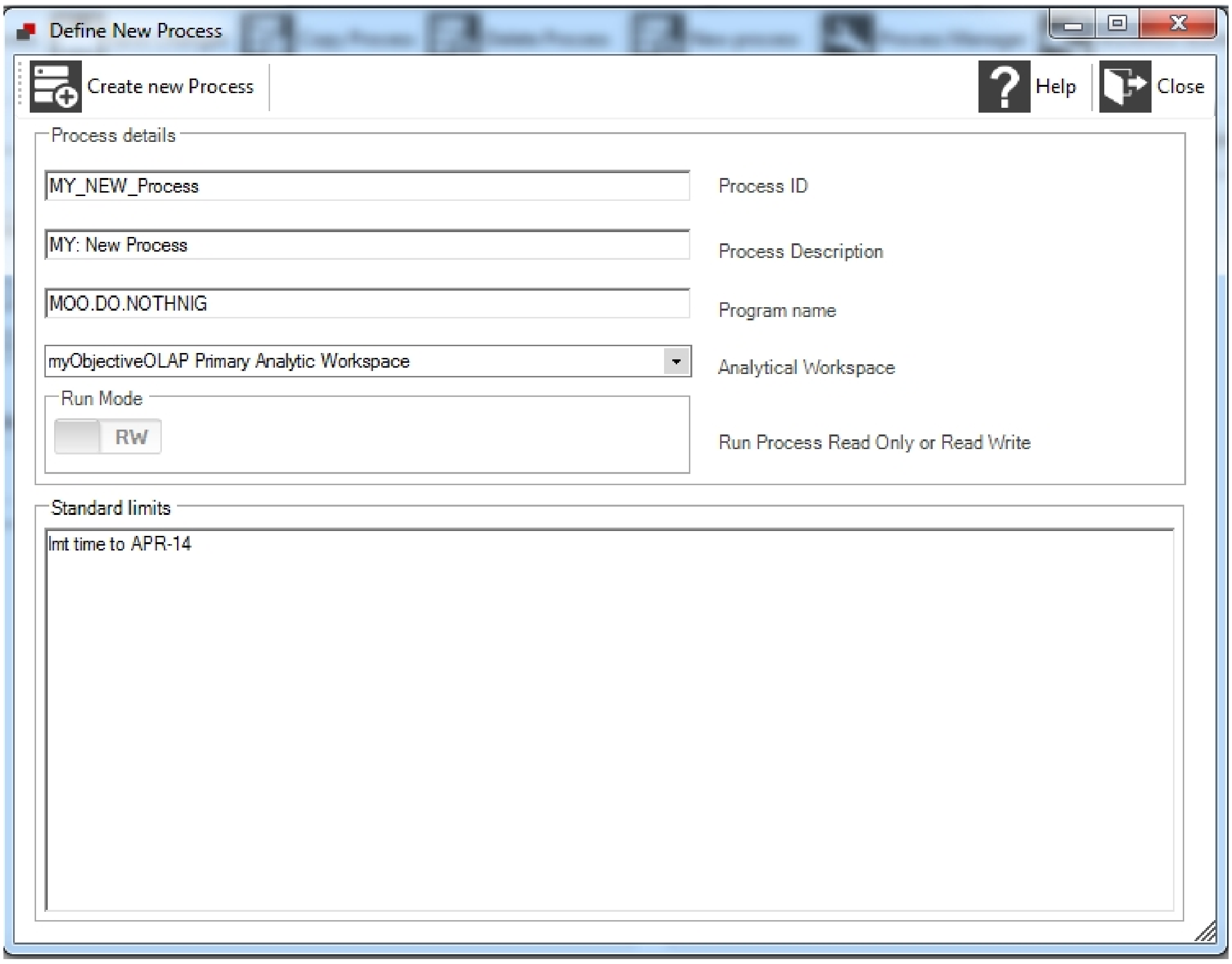
This is the name of the Oracle OLAP Analytic Workspace which the process will execute in. Either the primary myObjectiveOLAP Server MOODATA AW, or and Sub-AW's you have created

Run Mode

This is the mode your Process will run in. Read Only process can be run in parallel mode, whilst Read Write processes will run in a serial as soon as possible based on the availability of the specified Analytic Workspace.

Standard Limits

Any limits you want to apply to your data, prior to execution of the process.



Process details

MY_NEW_Process	Process ID
MY: New Process	Process Description
MOO.DO.NOTHNIG	Program name
myObjectiveOLAP Primary Analytic Workspace	Analytical Workspace
Run Mode: RW	Run Process Read Only or Read Write

Standard limits

```
lmt time to APR-14
```

Example of a Standard Limits to aggregate a cube.

The following is an example of Standard Limit using the MOO.AGGREGATE.CUBE program to aggregate a cube.

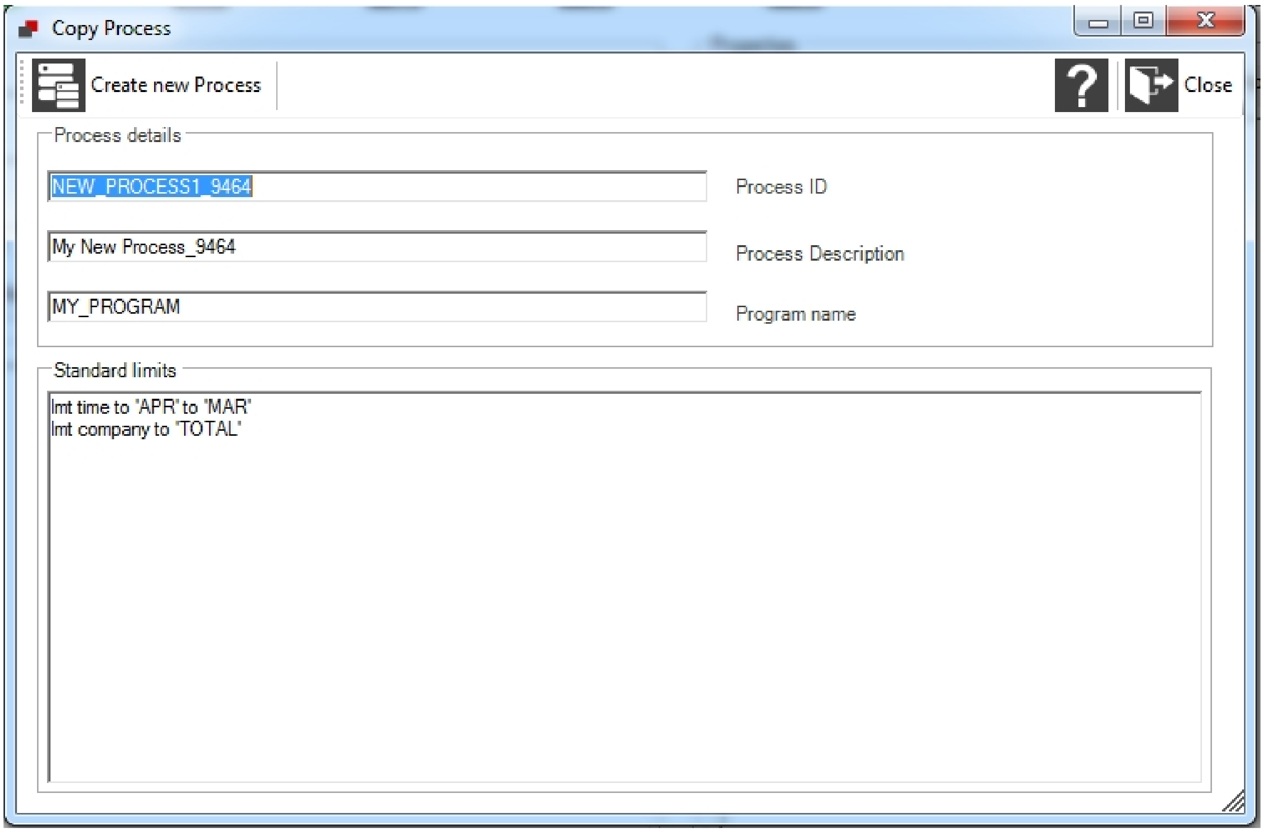
```
lmt cube.row to 'MY_CUBE'                    Tells the API which cube we
want to aggregate.
lmt dim.row to 'ACCOUNT COST_CENTRE MY_TIME'    Tells the API which dimensions
```

```
to aggregate over

call p.set.mycube.status                                Tells the API to call a custom
program which sets my dimension value limits and hierarchy
lmt my_time to 'PAPR-12'                                Limits a dimension which I
didn't want to set in my p.set.mycube.status program
```

Copy Process

Copy Process

Pressing the Copy Process button from with Process Builder allows you to create a new process based on an existing selected Process from within Process Builder



NEW_PROCESS1_9464
My New Process_9464
MY_PROGRAM

```
lmt time to 'APR' to 'MAR'
lmt company to 'TOTAL'
```

Process ID A unique identifier of your choice for the process you are creating. This will be used as the PR.ROW value within AW$PRCONTROL

Process Description A description for your new process

Program Name The name of either a MOOSERVER standard program such as MOO.AGGREGATE.CUBE (see the mooServer API) or a custom program of your own.

Standard Limits Any limits you want to apply to your data, prior to execution of the process.

Process Manager

Process Manager

Process Manager

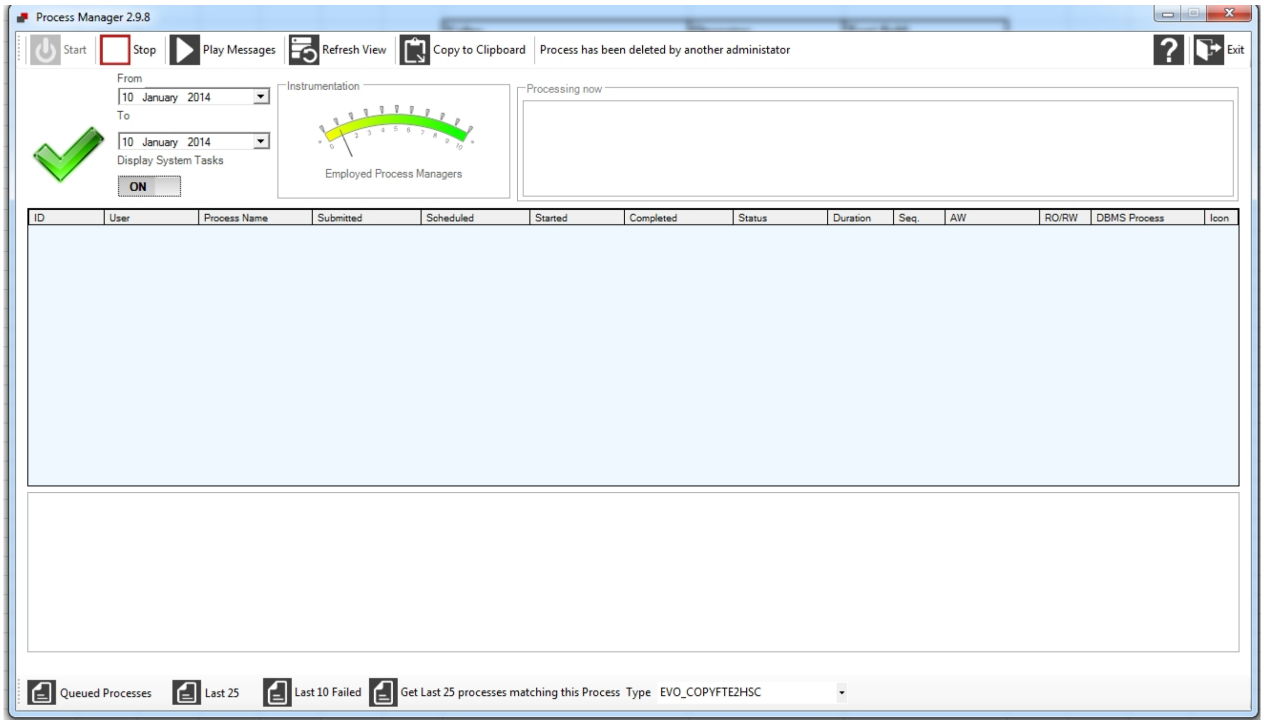
The Process Manager window controls the mooServer Process Manager.   The process manager executes processes in serial by default.

The controls available on the Process Manager window differ depending on the database user connected.

Users logged in as MOOUSER (normal users) have no control and are only able to view Processes being

Users logged in as MOOSERVER (Application DBA) users have full control of the Process Manager.

Process Manager (logged in as MOOSERVER)


Process Manager 2.9.8

Process has been deleted by another administator

From: 10 January 2014
To: 10 January 2014
Display System Tasks: ON

Get Last 25 processes matching this Process Type: EVO_COPYFTE2HSC

Controls

Start

Pressing the Start button will start the Process Manager and allow for execution of any jobs already in the queue.
This control is not available to users connected as MOOUSER.

Status is updated to: OLAP Scheduler is running

Quick-glance is updated to:Quick-glance is updated to:

Stop

Pressing the Stop button will stop the Process Manager immediately. Any processes already running will

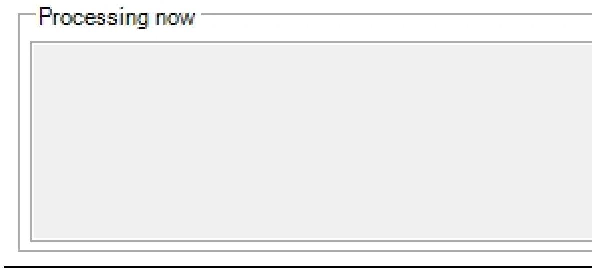
be terminated. If Play messages is running, pressing Stop will disable Play messages, it can be re-enabled on Start.
This control is not available to users connected as MOOUSER.

Status is updated to: Status is updated to OLAP Scheduler is not running

Quick-glance is updated to:

Play messages

Play messages will display the output of any Process or Workflow in the Processing now pane:
This control is not available to users connected as MOOUSER.



Play messages will spawn a second session on the database which will be visible from Session Manager, playing messages will not interfere with the Application DBA using other controls in the Process Manager window.   Exiting Process Manager will stop the second session. Messages are updated every 15 seconds.

Refresh View

Refreshes the queue-grid displaying the status of processes running, queued and completed.

Copy to Clipboard

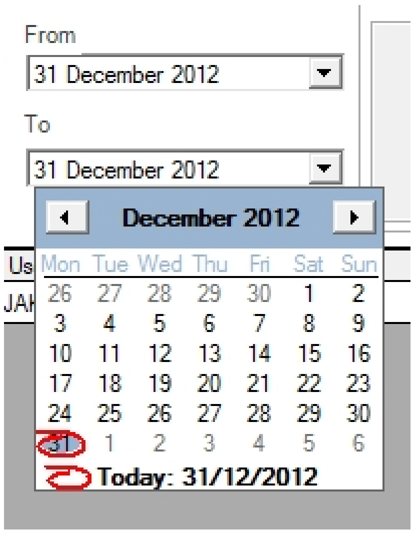
Copies the contents of the queue-grid to the Windows clip-board.

10 Apply Poll Settings

Changes the Poll duration that the Process Daemon (MOOSERVER.PRD$PRD) checks for Processes waiting to be executed.
This control is not available to users connected as MOOUSER.

Calendar Control

Enables you to set the From and To dates which you want to see displayed in the queue-grid. Select a From and To date and then press the Refresh View button.

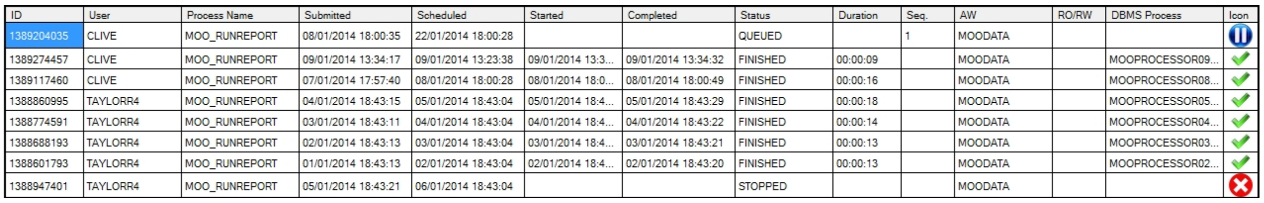


From: 31 December 2012

To: 31 December 2012

Queue grid

The queue grid displays information about Processes.



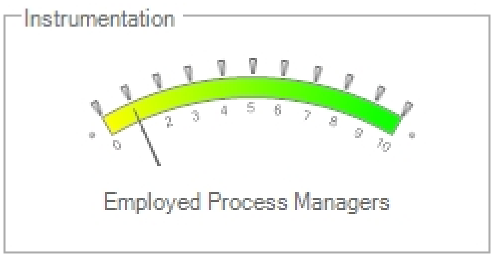
ID	User	Process Name	Submitted	Scheduled	Started	Completed	Status	Duration	Seq.	AW	RO/RW	DBMS Process	Icon
1389204035	CLIVE	MOO_RUNREPORT	08/01/2014 18:00:35	22/01/2014 18:00:28			QUEUED		1	MOODATA			⏸
1389274457	CLIVE	MOO_RUNREPORT	09/01/2014 13:34:17	09/01/2014 13:23:38	09/01/2014 13:3...	09/01/2014 13:34:32	FINISHED	00:00:09		MOODATA		MOOPROCESSOR09...	✓
1389117460	CLIVE	MOO_RUNREPORT	07/01/2014 17:57:40	08/01/2014 18:00:28	08/01/2014 18:0...	08/01/2014 18:00:49	FINISHED	00:00:16		MOODATA		MOOPROCESSOR08...	✓
1388860995	TAYLORR4	MOO_RUNREPORT	04/01/2014 18:43:15	05/01/2014 18:43:04	05/01/2014 18:4...	05/01/2014 18:43:29	FINISHED	00:00:18		MOODATA		MOOPROCESSOR05...	✓
1388774591	TAYLORR4	MOO_RUNREPORT	03/01/2014 18:43:11	04/01/2014 18:43:04	04/01/2014 18:4...	04/01/2014 18:43:22	FINISHED	00:00:14		MOODATA		MOOPROCESSOR04...	✓
1388688193	TAYLORR4	MOO_RUNREPORT	02/01/2014 18:43:13	03/01/2014 18:43:04	03/01/2014 18:4...	03/01/2014 18:43:21	FINISHED	00:00:13		MOODATA		MOOPROCESSOR03...	✓
1388601793	TAYLORR4	MOO_RUNREPORT	01/01/2014 18:43:13	02/01/2014 18:43:04	02/01/2014 18:4...	02/01/2014 18:43:20	FINISHED	00:00:13		MOODATA		MOOPROCESSOR02...	✓
1388947401	TAYLORR4	MOO_RUNREPORT	05/01/2014 18:43:21	06/01/2014 18:43:04			STOPPED			MOODATA			✖

The following table explains the data presented in the queue-grid.

Object	**Description**
ID	A unique ID that has been assigned to a specific Process. The unique ID is derived from POSIX time of submission. This ID can be used when analyzing the Audit log MOOSERVER.SUBDATA
User	The user who submitted the process, this is not the database user, but the application user stored in EXPRESS$ME_USER
Process Name	The Process name (PR.ROW) that has been submitted for processing.
Submitted	The date / time the process was submitted.
Scheduled	The date / time the process was scheduled to start.
Started	The date / time the process started.
Completed	The date / time the process completed.
Status	The Status: Queued, Started, Finished, Errored
Duration	The length of time the process took to complete.
Seq	The sequence (order (integer)) of jobs waiting to be processed.
AW	The name of the Oracle OLAP Analytic Workspace which the specified process or workflow is executing within
RO/RW	The mode; Read-only or Read-write for the specified process
DBMS Process	The name of the DBMS_SCHEDULER job which is running the specific Process or Workflow
Icon	Quick-glance, Queued [Pause Icon] Started [Play Icon], Completed Success [Tick Icon], Errored [Warning Icon], Unknown State [myObjectiveOLAP Icon]

Instrumentation

The instrumentation display shows the number of Employed Process Managers, The number of Process threads can be set within the system configuration. If no tasks are running then only the one monitoring Process will be running; stepping up to the maximum specified Process Managers.

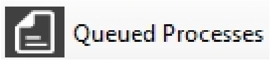
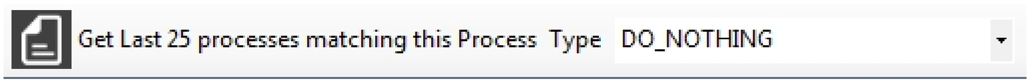


Display System Tasks

Internal tasks, for example: AW Syncing, Report Generators, Object Creation are hidden from the default view so that only business tasks are visible in the queue grid. Enabling this option means that all tasks are displayed in the queue grid.

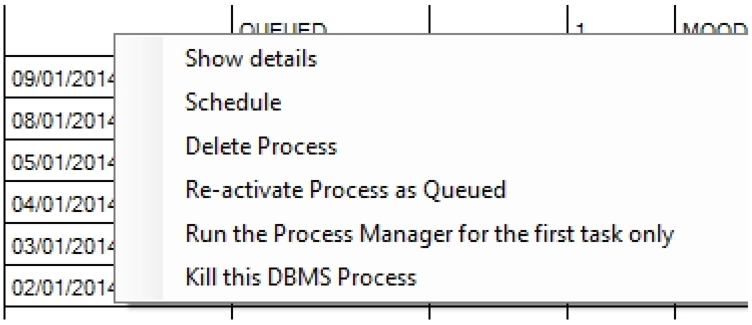
Quick Selection Reports

A number of predefined reports are available from the Quick Selection menu, all ignore the time selected in the Calendar control:

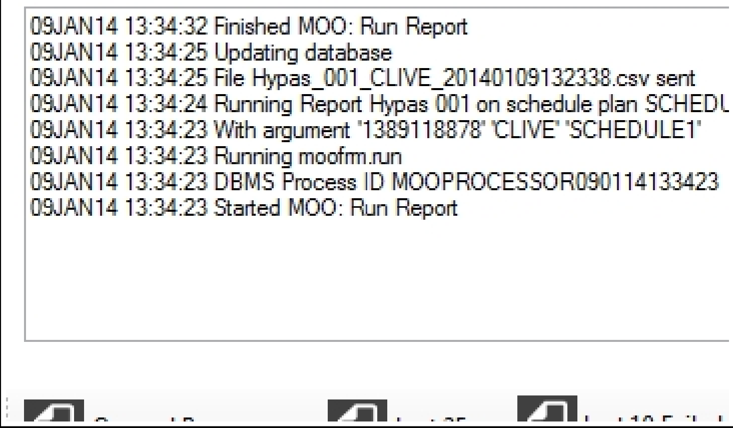
 Queued Processes	All processes that are queued
Last 25	Last 25 processes that have been processed
Last 10 Failed	Last 10 processes that have failed.
 Get Last 25 processes matching this Process Type DO_NOTHING	Displays the last 25 processes based on the selection in the **Type** drop-down list of Processes.

Individual Process Management

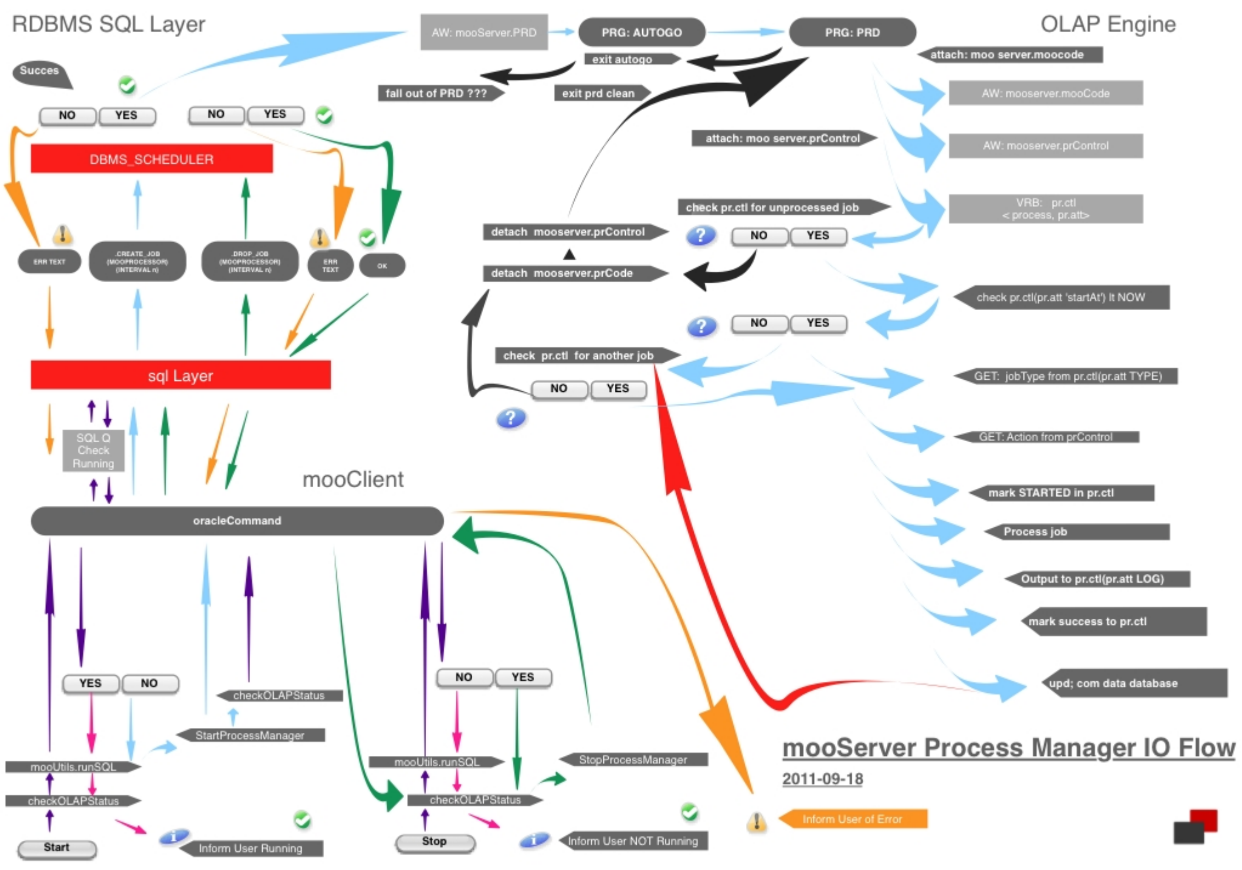
Right-clicking a process within the Process Management queue enables the Process Quick Access Menu:



Options within the QAM are summarised below:

QAM Option	Purpose
Show details	Displays details of the execution of a specific task within the Information panel:  09JAN14 13:34:32 Finished MOO: Run Report 09JAN14 13:34:25 Updating database 09JAN14 13:34:25 File Hypas_001_CLIVE_20140109132338.csv sent 09JAN14 13:34:24 Running Report Hypas 001 on schedule plan SCHEDU 09JAN14 13:34:23 With argument '1389118878' 'CLIVE' 'SCHEDULE1' 09JAN14 13:34:23 Running moofrm.run 09JAN14 13:34:23 DBMS Process ID MOOPROCESSOR090114133423 09JAN14 13:34:23 Started MOO: Run Report
Schedule	Enables an administrator to re-schedule a queued or completed task to run at a specified date and time.
Delete Process	Deletes a queued process from the queue.
Re-activate process as Queued	Re-enables a previously executed process to a queue state
Run the Process Manager for the first task only	If the Process Manager is stopped this will ask the Process Manager to create a single PM Thread and run the first waiting task in the queue.
Kill this DBMS Process	This enables an administrator to kill the PM thread running the selected process without having to completely shut down the Process Manager, which may be executing other parallel processes or workflows.

Technical Process Manager Flow Diagram



Technical Implementation

myObjectiveOLAP Server, Process Manager Technical Implementation

An Oracle Database administrator may want to understand the technical implementation of the myObjectiveOLAP Process Manager.   This note together with the Process Manager Flow Diagram found here, are designed to aide that understanding.

myObjectiveOLAP Server's Process Manager is based on the standard Oracle database RDBMS scheduler.

You must ensure that you connect to the Oracle Database as the MOOSERVER user before starting or stopping the Process Manager from sqlPlus.
Running the Process Manager as a different user is not supported.

Start the Process Manager from sqlPlus

When the Process Manager is started a statement similar to the following PLSQL statement is executed, passing the polling interval as set from the Process Manager graphical interface.

```
BEGIN DBMS_SCHEDULER.create_job ( job_name => 'MOOPROCESSOR', job_type => 'PLSQL_BLOCK',
job_action => 'BEGIN dbms_aw.execute(''aw attach MOOSERVER.PRD; PRD; aw detach
MOOSERVER.PRD''); END;', start_date => SYSTIMESTAMP, repeat_interval => 'freq= SECONDLY;
INTERVAL=[INTERGER_POLL_TIME_SECONDS],end_date => NULL, enabled => TRUE, comments =>
'myObjectiveOLAP Server Process Manager'); END;
```

The myObjectiveOLAP Process Manager graphical user interface also allows a "Run One Process" option, this executes the following which causes PRD (Process Manager daemon) to execute only one job before turning the Process Manager off.

```
BEGIN DBMS_SCHEDULER.create_job ( job_name => 'MOOPROCESSOR', job_type => 'PLSQL_BLOCK',
job_action => 'BEGIN dbms_aw.execute(''aw attach MOOSERVER.PRD; PRD; aw detach
MOOSERVER.PRD''); END;', start_date => SYSTIMESTAMP, repeat_interval => null ,end_date =>
NULL, enabled => TRUE, comments => 'myObjectiveOLAP Server Process Manager'); END;
```

By understanding how the Process Manager is started it would be possible to script the starting of the process manager daemon.

Stop the Process Manager from sqlPlus

When the Process Manager is asked to stop the following PLSQL is executed.

```
BEGIN DBMS_SCHEDULER.stop_job ('MOOPROCESSOR'); END;
BEGIN DBMS_SCHEDULER.disable('MOOPROCESSOR'); END;
BEGIN DBMS_SCHEDULER.drop_job ('MOOPROCESSOR'); END;
```

This may be useful in ensuring no jobs are running when you wish to bring down the Oracle Database or backup your mooData analytic workspace.

Note When asked to stop the myObjectiveOLAP Process Manager stops almost immediately, if it is currently processing a task the Process is terminated and a ROLLBACK executed.*

Checking the Current Status of the Process Manager from sqlPlus

The following sql select statement displays information from DBA_SCHEDULER_JOBS on the current status of the myOnjectiveOLAP Process Manager.

```
Select * from dba_scheduler_jobs WHERE JOB_NAME = 'MOOPROCESSOR';
```

Start the Process Manager from a myObjectiveOLAP Console Session

A myObjectiveOLAP Server application administrator may wish to run the Process Manager daemon directly from Oracle OLAP. To do this you should start an OLAP Console session as the MOOSERVER RDBMS user and detach all Analytic Workspaces either through the AW DETACH OLAP DML statement or the Analytic Workspace selector.   They should then attach the PRD Analytic Workspace either through the "AW ATTACH PRD ro" command followed by executing PRD: "call PRD". The Process Manager will process the next task and then exit.

Workflow Builder

Workflow Builder

Workflow Builder

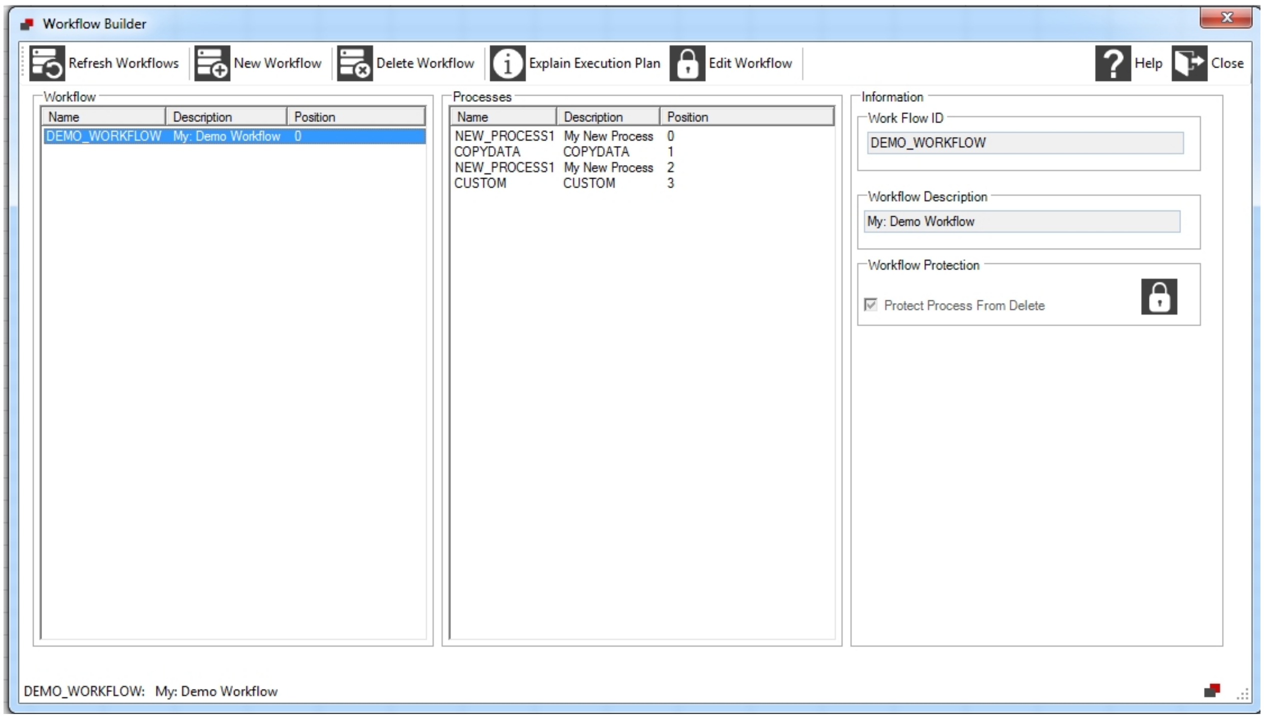
Workflow Builder allows you to group Processes or Workflows together into a single workflow which can either be submitted for execution manually through Process Builder, or can be configured to respond to an event.

Some Examples of Workflow Use

Example	Workflow
Interface file received from Oracle Retail.	Load Data --> Aggregate Data --> Model Data --> Generate Fixed Reports --> Generate and Email reconciliation reports.

Submit Data from Excel	Aggregate & Model Data --> Consolidate System --> Generate an interface file to Oracle Hyperion Financial Management
Refresh Cube	Load structures and data from a relational Star Schema --> Aggregate Data --> Model Data --> Email Users letting them know.

Workflow Builder, View Mode



Workflow

Name	Description	Position
DEMO_WORKFLOW	My: Demo Workflow	0

Processes

Name	Description	Position
NEW_PROCESS1	My New Process	0
COPYDATA	COPYDATA	1
NEW_PROCESS1	My New Process	2
CUSTOM	CUSTOM	3

Information

Work Flow ID: DEMO_WORKFLOW

Workflow Description: My: Demo Workflow

Workflow Protection: ☑ Protect Process From Delete

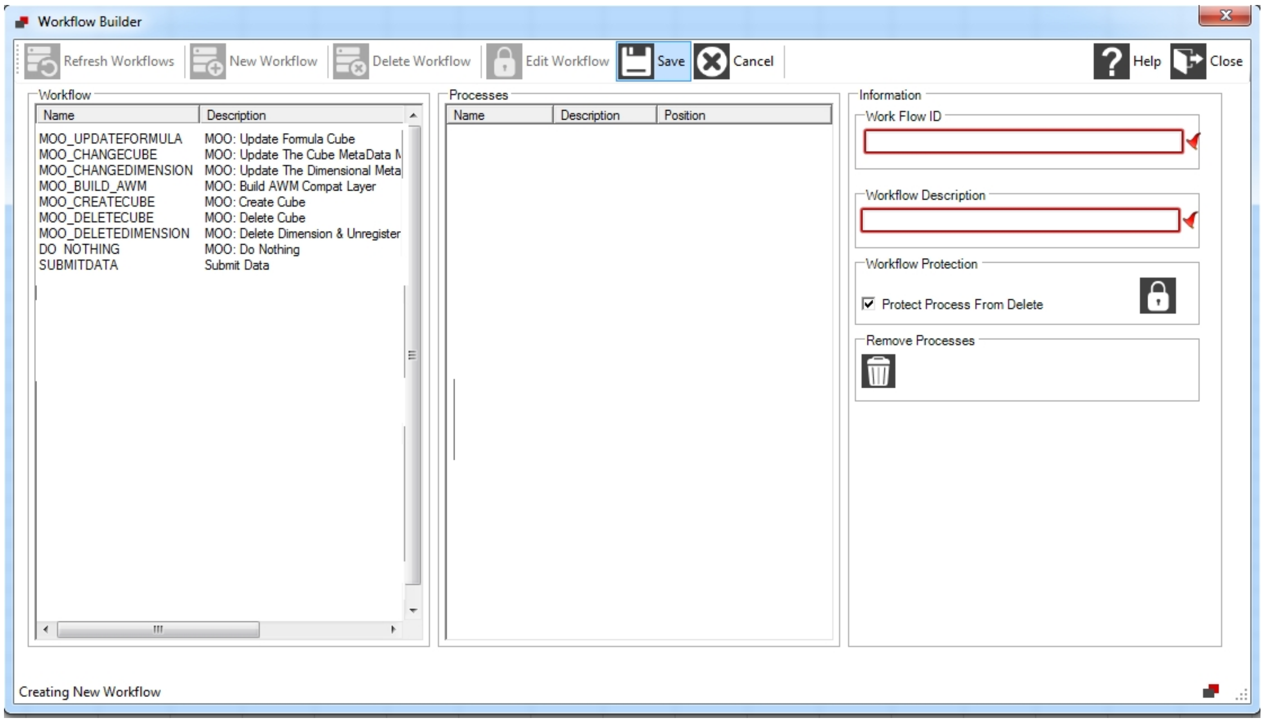
DEMO_WORKFLOW: My: Demo Workflow

Workflow Builder Window consists of the following important panes, when in Viewing Mode:

Pane	**Description**
Workflow	List of available Workflows
Processes	List of Processes or other Workflows assigned to the selected Workflow
Information	Information about the selected Workflow and its Lock status.

Workflow Builder, New Mode

Workflow Builder Window consists of the following important panes, when in New or Edit Workflow Mode:



Name	Description
MOO_UPDATEFORMULA	MOO: Update Formula Cube
MOO_CHANGECUBE	MOO: Update The Cube MetaData M
MOO_CHANGEDIMENSION	MOO: Update The Dimensional Meta
MOO_BUILD_AWM	MOO: Build AWM Compat Layer
MOO_CREATECUBE	MOO: Create Cube
MOO_DELETECUBE	MOO: Delete Cube
MOO_DELETEDIMENSION	MOO: Delete Dimension & Unregister
DO_NOTHING	MOO: Do Nothing
SUBMITDATA	Submit Data

Name	Description	Position

☑ Protect Process From Delete

Creating New Workflow

Pane	Description
Workflow	List of available Processes and Workflows
Processes	List of processes assigned to the Workflow
Information	Information about the selected Workflow and its Lock status.

Workflow Builder Controls

Control	Description
Refresh Workflows	Refreshes the list of available workflows.
New Workflow	Change Workflow Builder to New mode
Delete Workflow	Deletes the selected workflow, honors the prevent accidental deletion lock.
Explain Execution Plan	Generates an execution plan for the selected Workflow
Edit Workflow	Changes the Workflow Builder to Edit mode
Remove Processes	In New / Edit mode Processes can be removed from the workflow by dragging them into the bin.

Submission

Workflows are submitted to the Process Manager for execution via the Process Builder Window.

Workflows are automatically added to the Workflows virtual folder.

Hint: Many Processes and Workflows could have similar names. When defining Processes and Workflows, consider adding a (PR) and (WF) suffix to easily distinguish between them in the Process Builder window.

User Management

User Management

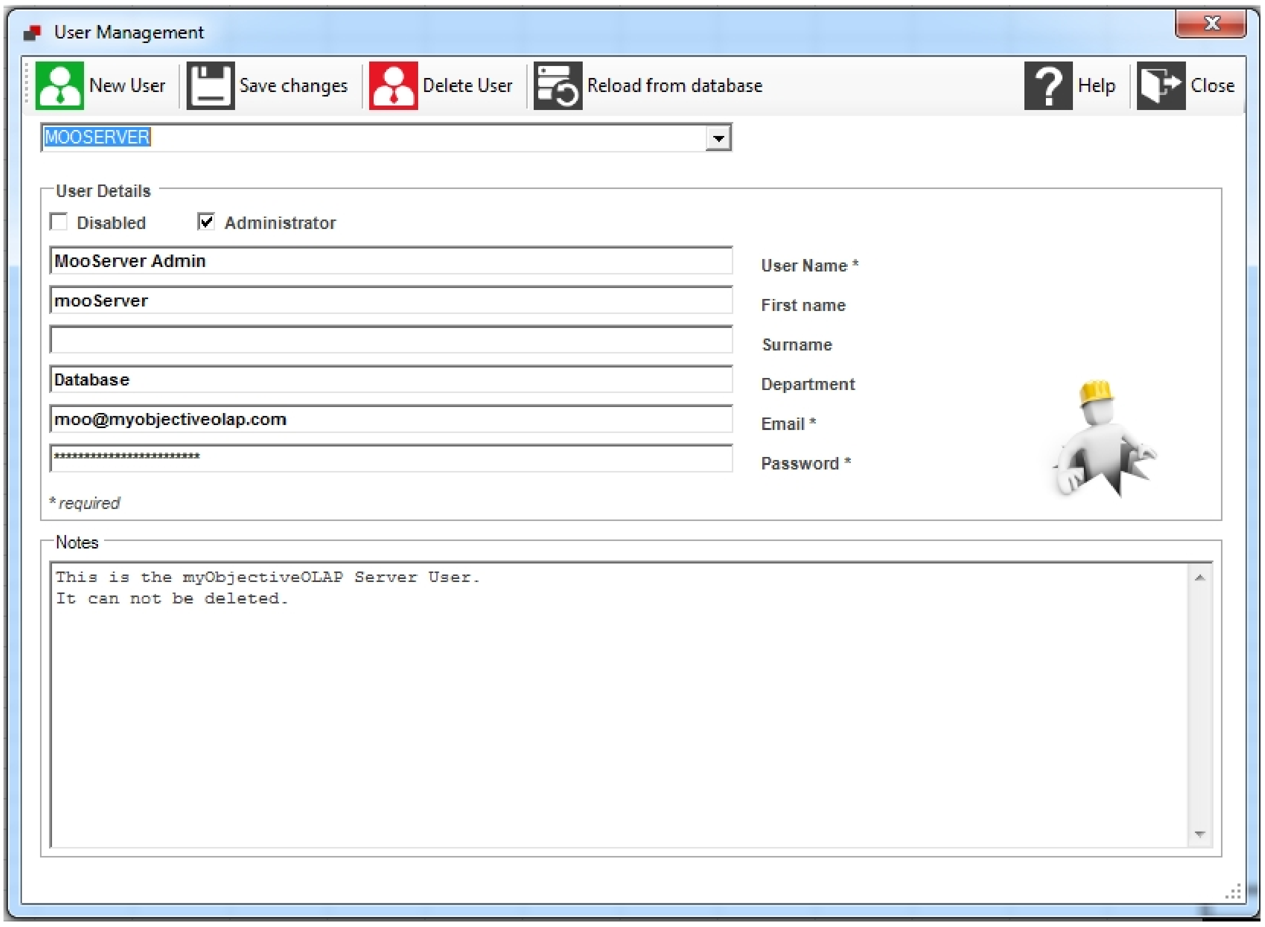
User Manager

User manager allows you to carry out the following actions:

Control	Actions
New User	Enter "New User" mode, complete all fields.
Create User	Creates the new user, validates user details
Save changes	Saves changes to an existing user.
Delete User	Deletes the selected user.
Reload from database	Reloads user information from myObjectiveOLAP Server
☐ Disabled	Enables disabling of a user account without deleting the account.
☑ Administrator	Enables specifying a user account as being capable of administrative functions

User Manager

The below image shows the User Management interface.



MOOSERVER

User Details

[ ] Disabled [✓ Administrator]

MooServer Admin	User Name *
mooServer	First name
	Surname
Database	Department
moo@myobjectiveolap.com	Email *
**************************	Password *

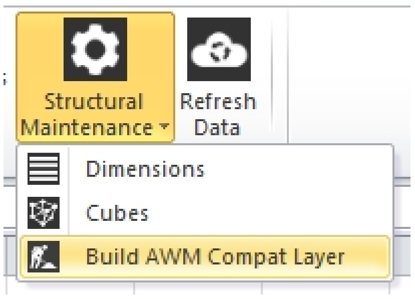
* required

Notes

```
This is the myObjectiveOLAP Server User.
It can not be deleted.
```

Oracle OLAP Standard Compatability

Build AWM Compat Layer



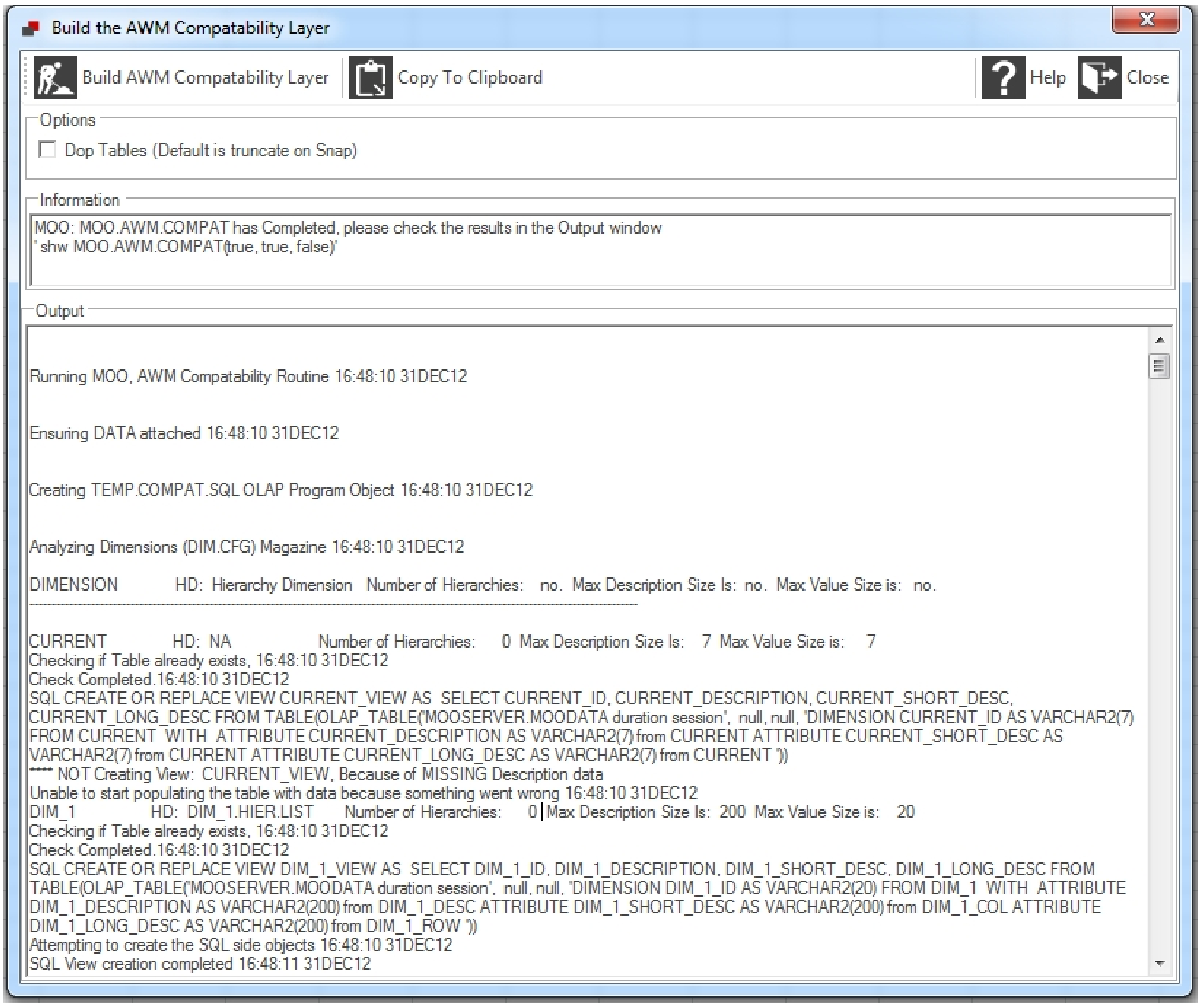
Building the AWM Compatibility Layer

The AWM Compatibility Layer tool automatically builds Relational Views and Tables based on your myObjectiveOLAP multi-dimensional model. The relational model can then be used to easily access multi-dimensional information from native SQL tools such as SQL Developer, Toad, or more advanced reporting applications like Oracle BI Enterprise Edition (OBIEE).

The compatibility layer initially creates a View of each multi-dimensional object and then snapshots that view into a table. This means you are able to use the MOOAWM Analytic Workspace to build formula variables within AWM based on star schema structural information sourced from the compatibility layer.

Note: Do not attempt to build AWM structural data based on the views themselves as this will cause erroneous errors as the OLAP engine attempts to retrieve data from myObjectiveOLAP server whilst building structures in another OLAP AW in the same session.

The below image shows the Build AWM Compatibility control.


Build the AWM Compatability Layer
Build AWM Compatability Layer
Copy To Clipboard
Help
Close
Options
Dop Tables (Default is truncate on Snap)
Information
Output

```
MOO: MOO.AWM.COMPAT has Completed, please check the results in the Output window
' shw MOO.AWM.COMPAT(true, true, false)'
```

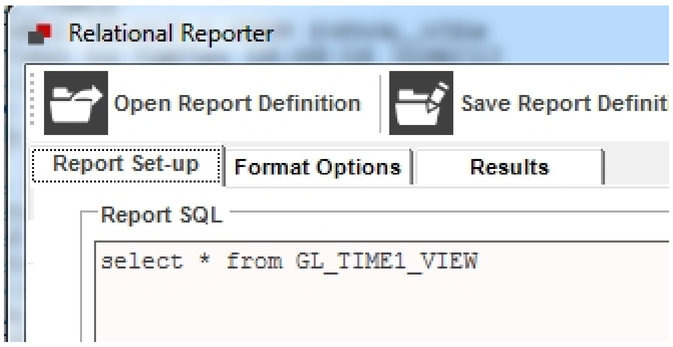
```
Running MOO, AWM Compatability Routine 16:48:10 31DEC12

Ensuring DATA attached 16:48:10 31DEC12

Creating TEMP.COMPAT.SQL OLAP Program Object 16:48:10 31DEC12

Analyzing Dimensions (DIM.CFG) Magazine 16:48:10 31DEC12

DIMENSION          HD:  Hierarchy Dimension   Number of Hierarchies:   no.  Max Description Size Is:  no.  Max Value Size is:   no.
--------------------------------------------------------------------------------------------------------------------------------

CURRENT            HD:  NA                 Number of Hierarchies:     0  Max Description Size Is:    7  Max Value Size is:     7
Checking if Table already exists, 16:48:10 31DEC12
Check Completed.16:48:10 31DEC12
SQL CREATE OR REPLACE VIEW CURRENT_VIEW AS  SELECT CURRENT_ID, CURRENT_DESCRIPTION, CURRENT_SHORT_DESC,
CURRENT_LONG_DESC FROM TABLE(OLAP_TABLE('MOOSERVER.MOODATA duration session',  null, null, 'DIMENSION CURRENT_ID AS VARCHAR2(7)
FROM CURRENT  WITH  ATTRIBUTE CURRENT_DESCRIPTION AS VARCHAR2(7) from CURRENT ATTRIBUTE CURRENT_SHORT_DESC AS
VARCHAR2(7) from CURRENT ATTRIBUTE CURRENT_LONG_DESC AS VARCHAR2(7) from CURRENT '))
**** NOT Creating View:  CURRENT_VIEW, Because of MISSING Description data
Unable to start populating the table with data because something went wrong 16:48:10 31DEC12
DIM_1            HD:  DIM_1.HIER.LIST      Number of Hierarchies:     0 Max Description Size Is:  200  Max Value Size is:    20
Checking if Table already exists, 16:48:10 31DEC12
Check Completed.16:48:10 31DEC12
SQL CREATE OR REPLACE VIEW DIM_1_VIEW AS  SELECT DIM_1_ID, DIM_1_DESCRIPTION, DIM_1_SHORT_DESC, DIM_1_LONG_DESC FROM
TABLE(OLAP_TABLE('MOOSERVER.MOODATA duration session',  null, null, 'DIMENSION DIM_1_ID AS VARCHAR2(20) FROM DIM_1  WITH  ATTRIBUTE
DIM_1_DESCRIPTION AS VARCHAR2(200) from DIM_1_DESC ATTRIBUTE DIM_1_SHORT_DESC AS VARCHAR2(200) from DIM_1_COL ATTRIBUTE
DIM_1_LONG_DESC AS VARCHAR2(200) from DIM_1_ROW '))
Attempting to create the SQL side objects 16:48:10 31DEC12
SQL View creation completed 16:48:11 31DEC12
```

The tool consists of the following controls and panes:

Control / Pane	Action
Build AWM Compatability Layer	Start the build of the compatability relational layer
Copy To Clipboard	Copies the contents of Output to the Windows clipboard
Options Dop Tables (Default is truncate on Snap)	Ask the compatibility layer program to Drop the relational tables and re-create instead of just truncating and re-populating from the created views.
Information	Information related to the build and any major errors. Also indicates the API call which the wizard uses to create the Relational layer.
Output	A report of the build including all the SQL executed during the build. This also includes any errors encountered during the build.

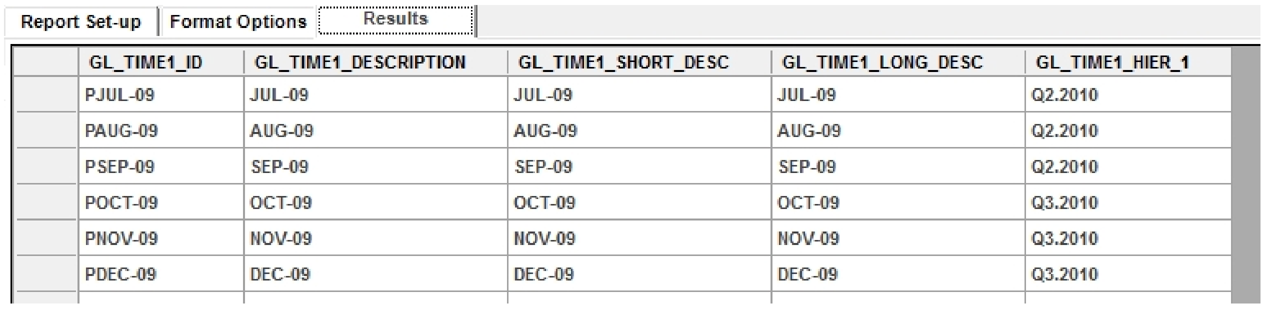
Looking at the output

Once complete you can run SQL from any client against the compatability layer:



```
select * from GL_TIME1_VIEW
```

Result:



GL_TIME1_ID	GL_TIME1_DESCRIPTION	GL_TIME1_SHORT_DESC	GL_TIME1_LONG_DESC	GL_TIME1_HIER_1
PJUL-09	JUL-09	JUL-09	JUL-09	Q2.2010
PAUG-09	AUG-09	AUG-09	AUG-09	Q2.2010
PSEP-09	SEP-09	SEP-09	SEP-09	Q2.2010
POCT-09	OCT-09	OCT-09	OCT-09	Q3.2010
PNOV-09	NOV-09	NOV-09	NOV-09	Q3.2010
PDEC-09	DEC-09	DEC-09	DEC-09	Q3.2010

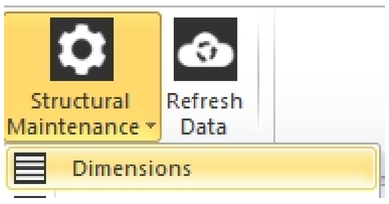
Structures

Structural Maintenance

myObjectiveOLAP Server enables the creation and maintenance of structures within the MOODATA analytic workspace.

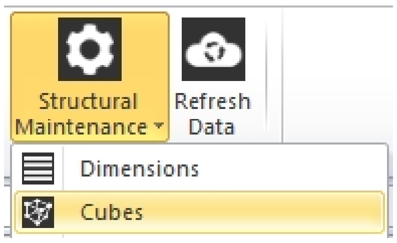
Dimensional maintenance

Dimensional maintenance is carried out through the Dimensions form:



Cube maintenance

Cube maintenance is carried out through the Cubes form:

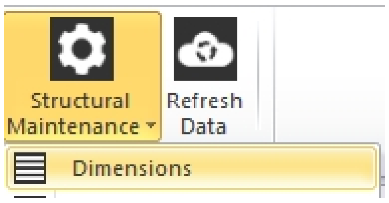


Values, hierarchies, attributes.

Whilst in most systems maintenance of Dimension Values and there associated meta-data is carried out through interfaces (Relational or OLAP), myObjectiveOLAP Server additionally allows for creation of this meta-data information through Excel, this enables both ad-hoc maintenance and easy bulk-creation or adaptation of this data.

Creating or modifying a dimension

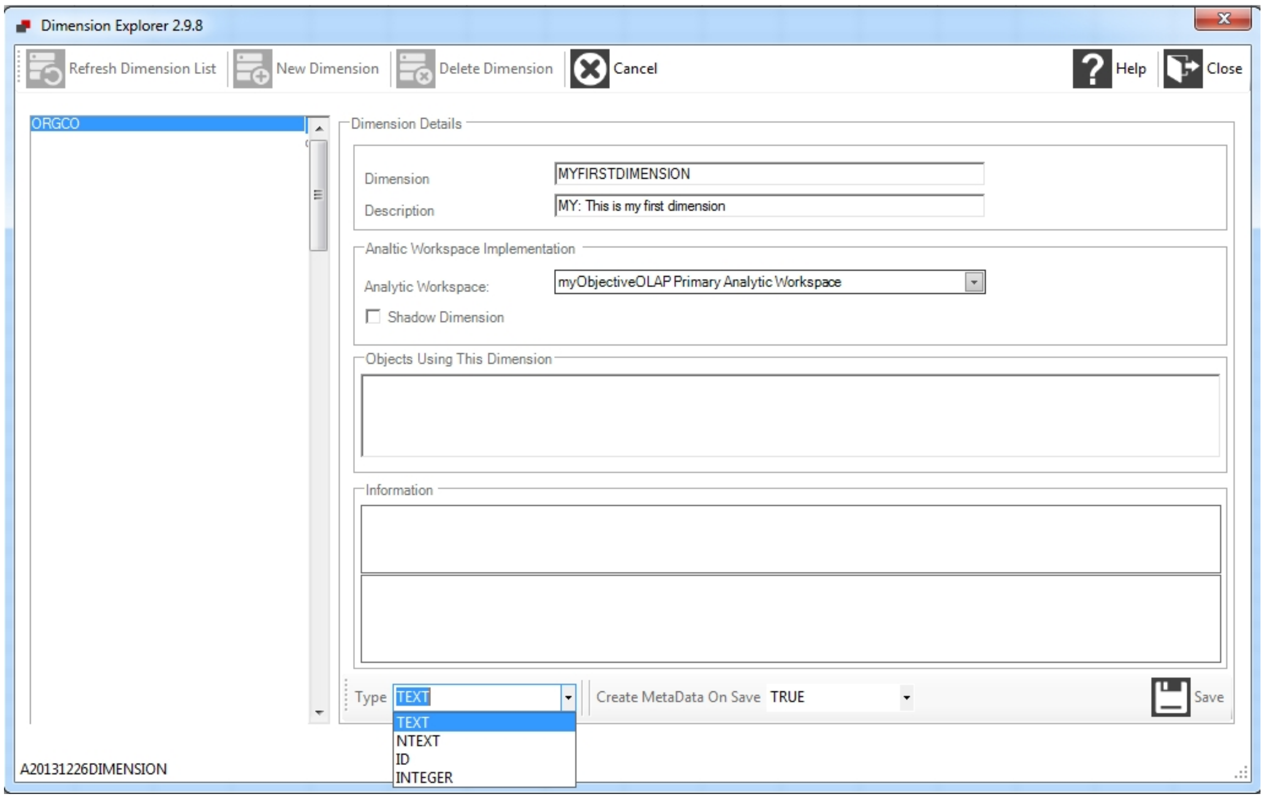
Dimensional Maintenance



myObjectiveOLAP Server enables the creation and maintenance of dimensions within the MOODATA analytic workspace.

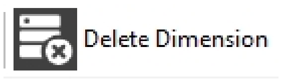
Dimensional maintenance

The Dimensional Explorer window is shown below.   You can select any existing dimension from the left Available Dimension pane.



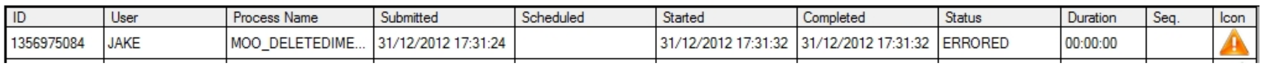
Dimension: MYFIRSTDIMENSION
Description: MY: This is my first dimension
Analytic Workspace: myObjectiveOLAP Primary Analytic Workspace
☐ Shadow Dimension
Type: TEXT
Create MetaData On Save: TRUE

Selecting a dimension shows you information about that dimension in the Dimension details pane, and lists all objects using that dimension.



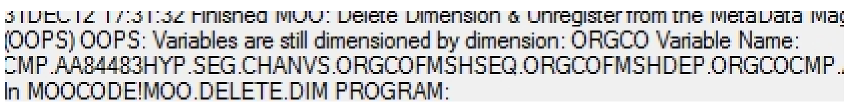
Pressing the Delete Dimension button will cause myObjectiveOLAP to ask if you sure?, if we say Yes to this myObjectiveOLAP will ask if you want to cascade this deletion. As you can see in the "Objects Using This Dimension" information pane, ORGCO is used by many variables and data cubes. If I were to say Yes at this point all of the objects listed above would be deleted.

If I say No to this question a MOO_DELETEDIMENSION Process will be submitted to the Process Manager but it will fail:



ID	User	Process Name	Submitted	Scheduled	Started	Completed	Status	Duration	Seq.	Icon
1356975084	JAKE	MOO_DELETEDIME...	31/12/2012 17:31:24		31/12/2012 17:31:32	31/12/2012 17:31:32	ERRORED	00:00:00		⚠

as the objects still exist.



```
31DEC12 17:31:32 Finished MOO: Delete Dimension & Unregister from the MetaData Ma
(OOPS) OOPS: Variables are still dimensioned by dimension: ORGCO Variable Name:
CMP.AA84483HYP.SEG.CHANVS.ORGCOFMSHSEQ.ORGCOFMSHDEP.ORGCOCMP..
In MOOCODE!MOO.DELETE.DIM PROGRAM:
```

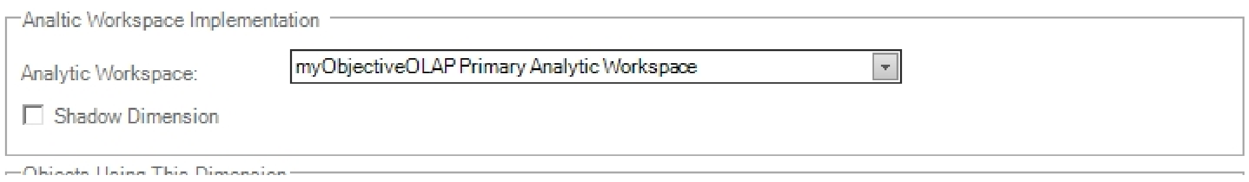
Delete dimension uses the MOO.DELETE.DIM API.

To create a new dimension press the New Dimension button, enter a name for your dimension.

Enter a description for the Dimension.

Multi-AW Mode

Withing the Analytic Workspace Implementation pane you can select the Analytic Workspace you wish to physically create your dimension within.   Prior to myObjectiveOLAP Server 2.9.8 all data and data-dictionary components were created within the primary myObjectiveOLAP Analytic Workspace (MOODATA).

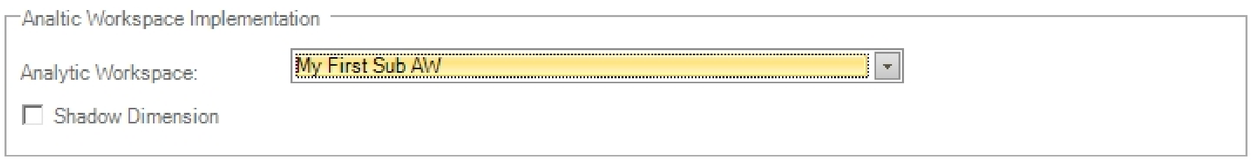


Analtic Workspace Implementation

Analytic Workspace: myObjectiveOLAP Primary Analytic Workspace

☐ Shadow Dimension

Post myObjectiveOLAP 2.9.8 it is possible to choose a specific child Analytic Workspace in which to create your dimension:



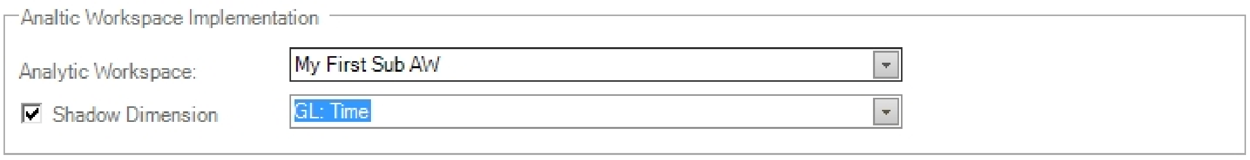
Analtic Workspace Implementation

Analytic Workspace: My First Sub AW

☐ Shadow Dimension

Shadow Dimensions allow you to maintain a dimension in a child Analytic Workspace which is mirrored from a dimension which is maintained within a different Analytic Workspace.   myObjectiveOLAP Server will automatically ensure synchronisation of a shadow dimension to its master dimension.   Additional values may be added to the shadow dimension which do not exist in the master, however, these will not be synchronized back to the master.

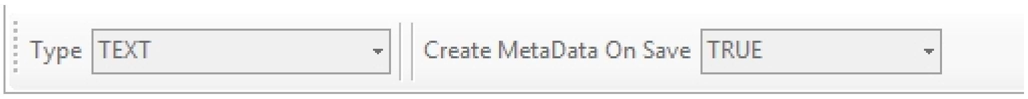
In the following example a new dimension will be created in the Analytic Workspace "My First Sub AW" and will be mastered from the master dimension "GL: Time". myObjectiveOLAP will ensure that all values, descriptions, hierarchies created and registered against the master dimension are mirrored within the child:


Analtic Workspace Implementation

Analytic Workspace: My First Sub AW

☑ Shadow Dimension: GL: Time

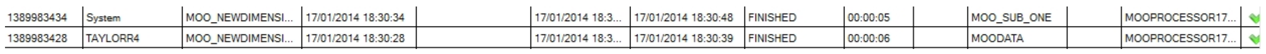
Once you have chosen to create a shadow dimension it is no longer possible to choose the dimension data-type or to specify if you want the meta-data created. The dimension will be completely mastered from the original and that includes its data-type:


Type: TEXT

Create MetaData On Save: TRUE

Press Save to start the creation and synchronisation process:

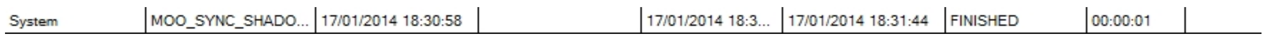
myObjectiveOLAP will create two Processes in the Process Manager:



1389983434	System	MOO_NEWDIMENSI...	17/01/2014 18:30:34		17/01/2014 18:3...	17/01/2014 18:30:48	FINISHED	00:00:05		MOO_SUB_ONE		MOOPROCESSOR17...	✓
1389983428	TAYLORR4	MOO_NEWDIMENSI...	17/01/2014 18:30:28		17/01/2014 18:3...	17/01/2014 18:30:39	FINISHED	00:00:06		MOODATA		MOOPROCESSOR17...	✓

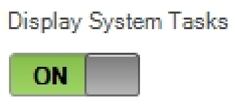
The first process creates and registers your new dimension in the MOODATA primary Analytic Workspace. The second dimension defines the physical object within the child Analytic Workspace.

myObjectiveOLAP server will also spawn a synchronisation process for each analytic workspace defined within the application:



System	MOO_SYNC_SHADO...	17/01/2014 18:30:58		17/01/2014 18:3...	17/01/2014 18:31:44	FINISHED	00:00:01	

Note; in order to view these system processes the Display switch must be toggled to ON.


Display System Tasks

ON

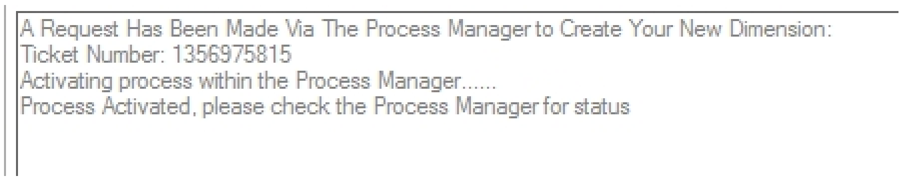
Non Multi-AW Mode

If you are creating a master dimension you should choose the data-type for your new dimension from the drop down Type list.

Choose if you want myObjectiveOLAP to create the meta-data variables or just register them. Typically you would leave this as TRUE unless you are merging existing structures from an Oracle Financial Analyzer (OFA), Oracle Sales Analyzer (OSA) or legacy Oracle Express database into Oracle OLAP, in which case you would complete the MetaData Information box with your legacy variable names and set the Create MetaData on Save to FALSE, then import your OFA structures from an EIF file.

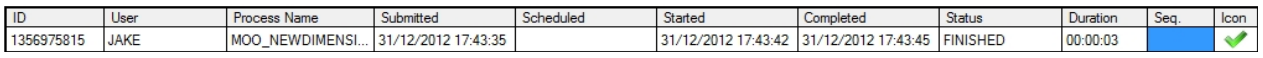
Press Save to save your dimension.

myObjectiveOLAP will give you a "ticket" for the dimension creation process



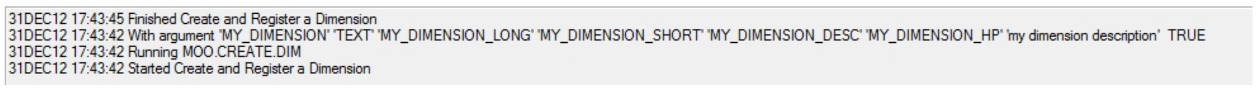
A Request Has Been Made Via The Process Manager to Create Your New Dimension:
Ticket Number: 1356975815
Activating process within the Process Manager......
Process Activated, please check the Process Manager for status

Your dimensional creation will be processed through the Process Manager

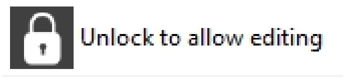


ID	User	Process Name	Submitted	Scheduled	Started	Completed	Status	Duration	Seq.	Icon
1356975815	JAKE	MOO_NEWDIMENSI...	31/12/2012 17:43:35		31/12/2012 17:43:42	31/12/2012 17:43:45	FINISHED	00:00:03		✓

Information and the exact arguments to the MOO.CREATE.DIM API are displayed in the Process Manager Details information pane.

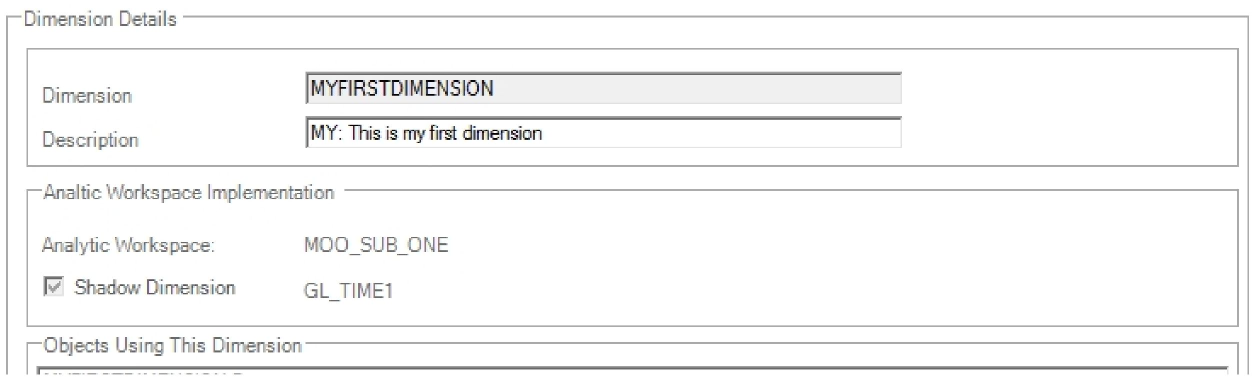


```
31DEC12 17:43:45 Finished Create and Register a Dimension
31DEC12 17:43:42 With argument 'MY_DIMENSION' 'TEXT' 'MY_DIMENSION_LONG' 'MY_DIMENSION_SHORT' 'MY_DIMENSION_DESC' 'MY_DIMENSION_HP' 'my dimension description'  TRUE
31DEC12 17:43:42 Running MOO.CREATE.DIM
31DEC12 17:43:42 Started Create and Register a Dimension
```



Unlock to allow editing

Pressing the Unlock to allow editing will allow you to edit the description of your dimension.



Dimension Details

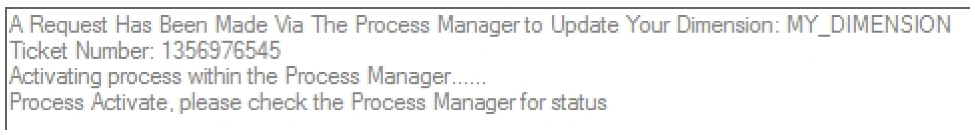
Dimension	MYFIRSTDIMENSION
Description	MY: This is my first dimension

Analtic Workspace Implementation

Analytic Workspace:	MOO_SUB_ONE
☑ Shadow Dimension	GL_TIME1

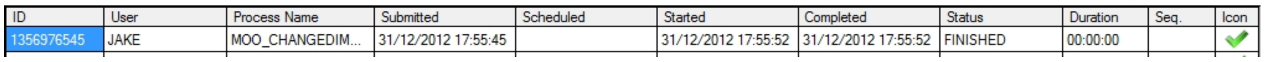
Objects Using This Dimension

Press Save to save your changes, you will be given a "ticket" for your change:



A Request Has Been Made Via The Process Manager to Update Your Dimension: MY_DIMENSION
Ticket Number: 1356976545
Activating process within the Process Manager......
Process Activate, please check the Process Manager for status

Your MOO_CHANGEDIMENSION Process will be processed by the Process Manager



ID	User	Process Name	Submitted	Scheduled	Started	Completed	Status	Duration	Seq.	Icon
1356976545	JAKE	MOO_CHANGEDIM...	31/12/2012 17:55:45		31/12/2012 17:55:52	31/12/2012 17:55:52	FINISHED	00:00:00		✓

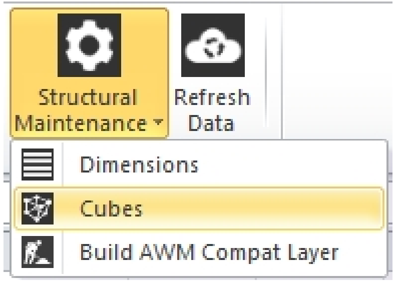
Details, including the API call arguments to MOO.CHANGE.DIM will be displayed in the Process Manager details pane:

31DEC12 17:55:52 Finished MOO: Update The Dimensional MetaData Magazine
31DEC12 17:55:52 With argument 'MY_DIMENSION' 'MY_DIMENSION_LONG' 'MY_DIMENSION_SHORT' 'MY_DIMENSION_DESC' 'MY_DIMENSION_HP' 'my dimension description CHANGED' TRUE
31DEC12 17:55:52 Running MOO.CHANGE.DIM
31DEC12 17:55:52 Started MOO: Update The Dimensional MetaData Magazine

If you do not Exit and Re-open the Dimensional configuration window, you can see your new dimension added to the Available Dimensions pane after it has progressed through the Process Manager by pressing the Refresh Dimension List button.

Creating or modifying a cube

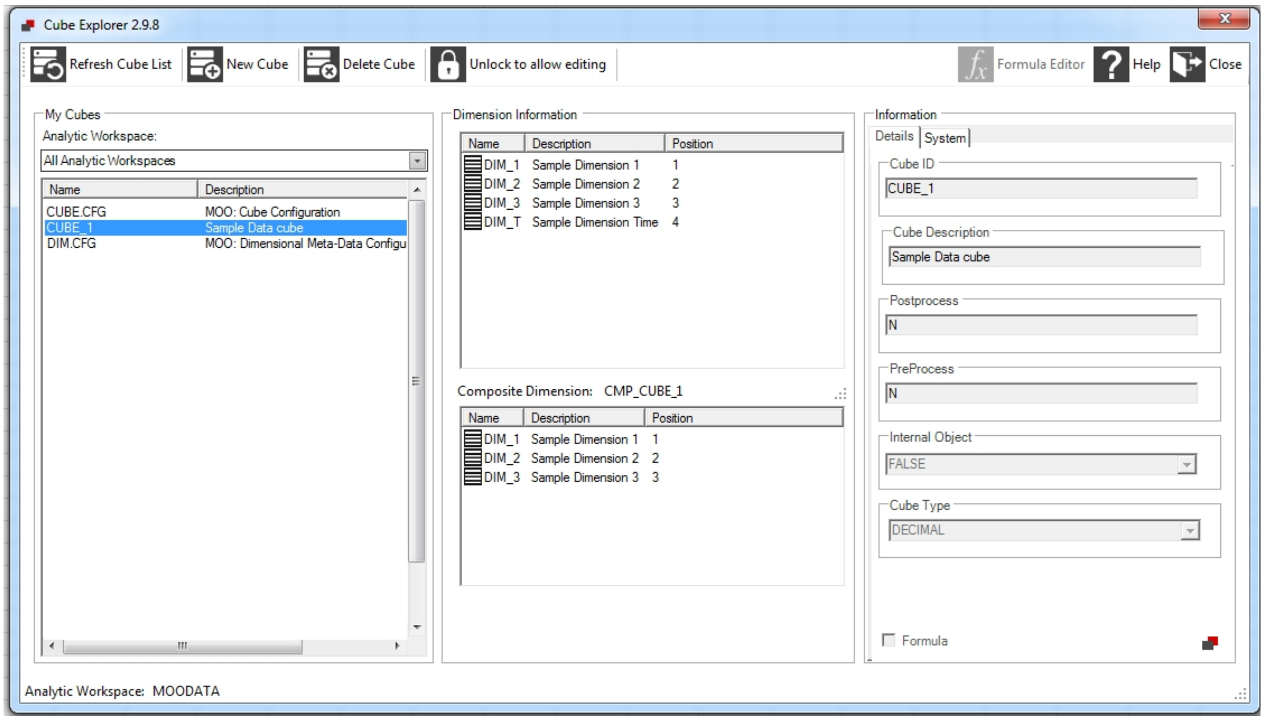
Cube Maintenance



myObjectiveOLAP Server enables the creation and maintenance of cubes within the MOODATA analytic workspace.

Cube maintenance

The Cube Explorer window is shown below.   You can select any existing cube from the left available Cubes pane.



My Cubes

Analytic Workspace: All Analytic Workspaces

Name	Description
CUBE.CFG	MOO: Cube Configuration
CUBE_1	Sample Data cube
DIM.CFG	MOO: Dimensional Meta-Data Configu

Dimension Information

Name	Description	Position
DIM_1	Sample Dimension 1	1
DIM_2	Sample Dimension 2	2
DIM_3	Sample Dimension 3	3
DIM_T	Sample Dimension Time	4

Composite Dimension: CMP_CUBE_1

Name	Description	Position
DIM_1	Sample Dimension 1	1
DIM_2	Sample Dimension 2	2
DIM_3	Sample Dimension 3	3

Information

Details | System

Cube ID: CUBE_1

Cube Description: Sample Data cube

Postprocess: N

PreProcess: N

Internal Object: FALSE

Cube Type: DECIMAL

☐ Formula

Analytic Workspace: MOODATA

Selecting a cube displays the following information:

Object	Details
Dimension Information	Lists the dimensions, dimension descriptions, dimension order

Composite	Names the composite and the dimensions which form part of the a composite dimension on the selected cube.

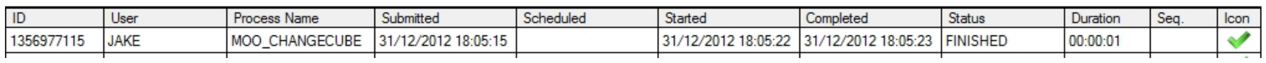
Advanced information is shown in the Information pane:

Object	Details
Cube ID	Unique identifier for the internal object name (CUBE.ROW)
Cube Description	Description for the Cube
PostProcess	Name of an OLAP DML program which you want to be executed before data submission is processed.
PreProcess	Name of an OLAP DML program which you want to be executed after data submission is processed
Internal Object	Identifier if the cube is classed as "Protected" internal.
Cube Type	Cube Data-type
Formula	Boolean, is the cube a formula?

In the lower-left corner the name of the Analytic Workspace where the physical object has been created is displayed.

Delete Cube

Pressing the Delete Cube button will cause myObjectiveOLAP to ask if you sure?, if we say Yes to this myObjectiveOLAP will submit a MOO_DELETECUBE Process to the Process Manager.



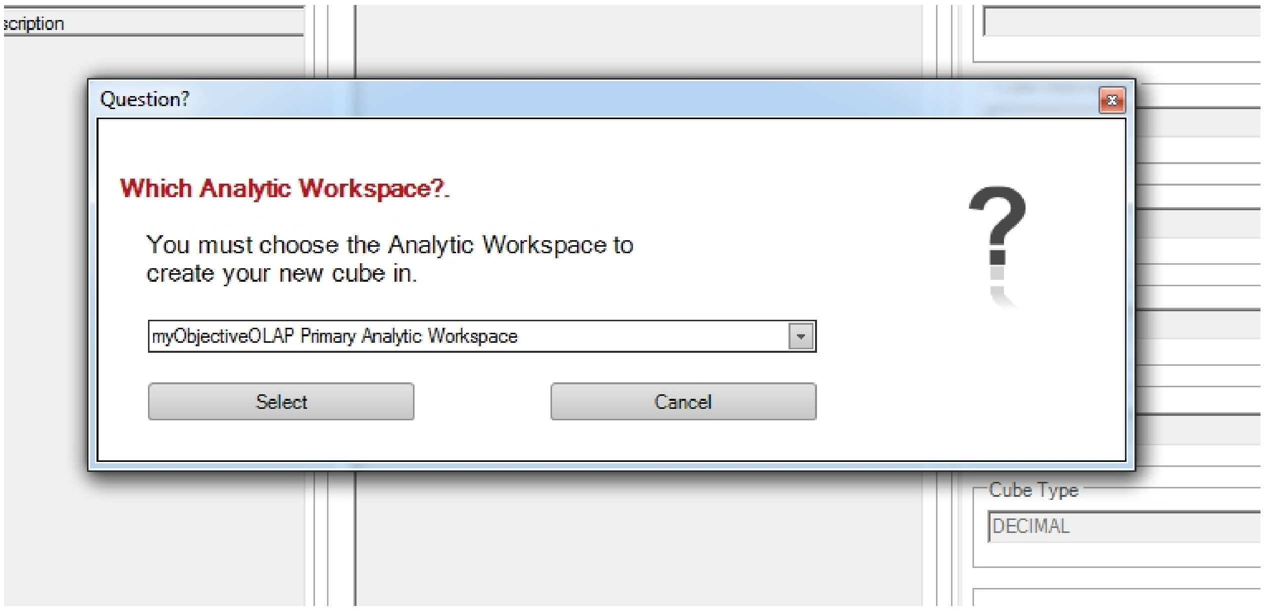
ID	User	Process Name	Submitted	Scheduled	Started	Completed	Status	Duration	Seq.	Icon
1356977115	JAKE	MOO_CHANGECUBE	31/12/2012 18:05:15		31/12/2012 18:05:22	31/12/2012 18:05:23	FINISHED	00:00:01		✓

Information about the deletion will logged and available in the Process Manager details pane:

```
31DEC12 18:03:24 Finished MOO: Delete Cube
31DEC12 18:03:22 With argument 'JUSTCUBE'
31DEC12 18:03:22 Running MOO.DELETE.CUBE
31DEC12 18:03:22 Started MOO: Delete Cube
```

New Cube

When you press New Cube, you are immediately asked which Analytic Workspace you want to create your cube in:


Question?

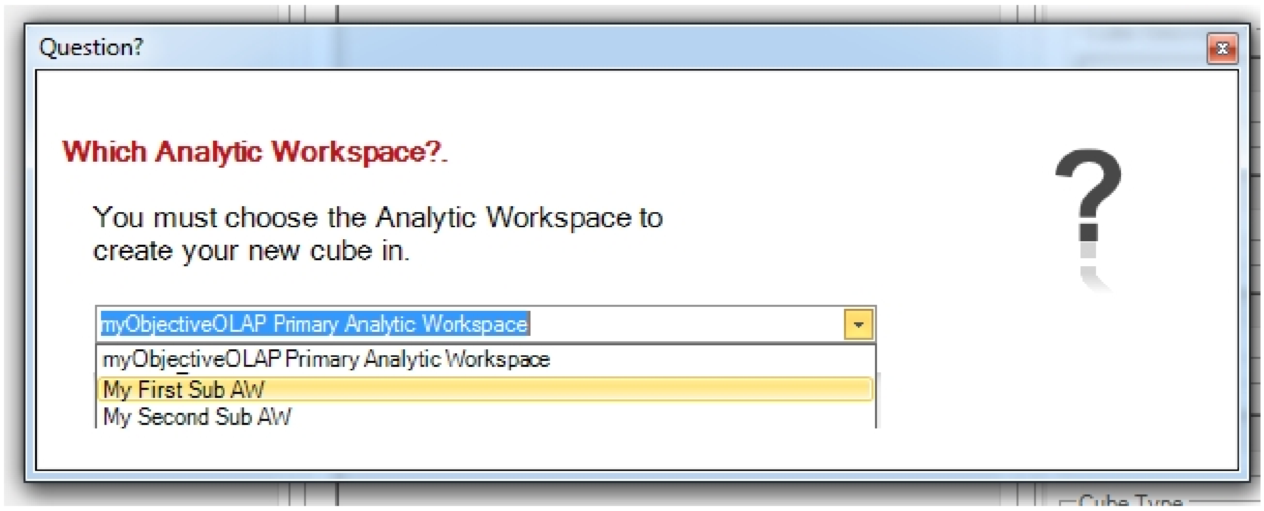
Which Analytic Workspace?.

You must choose the Analytic Workspace to create your new cube in.

myObjectiveOLAP Primary Analytic Workspace

You can either choose the myObjectiveOLAP Primary Analytic Workspace (MOODATA), which was the default before myObjectiveOLAP 2.9.8.

Or you can chose a child Analytic Workspace which has been created through Analytical Workspace Control


Question?

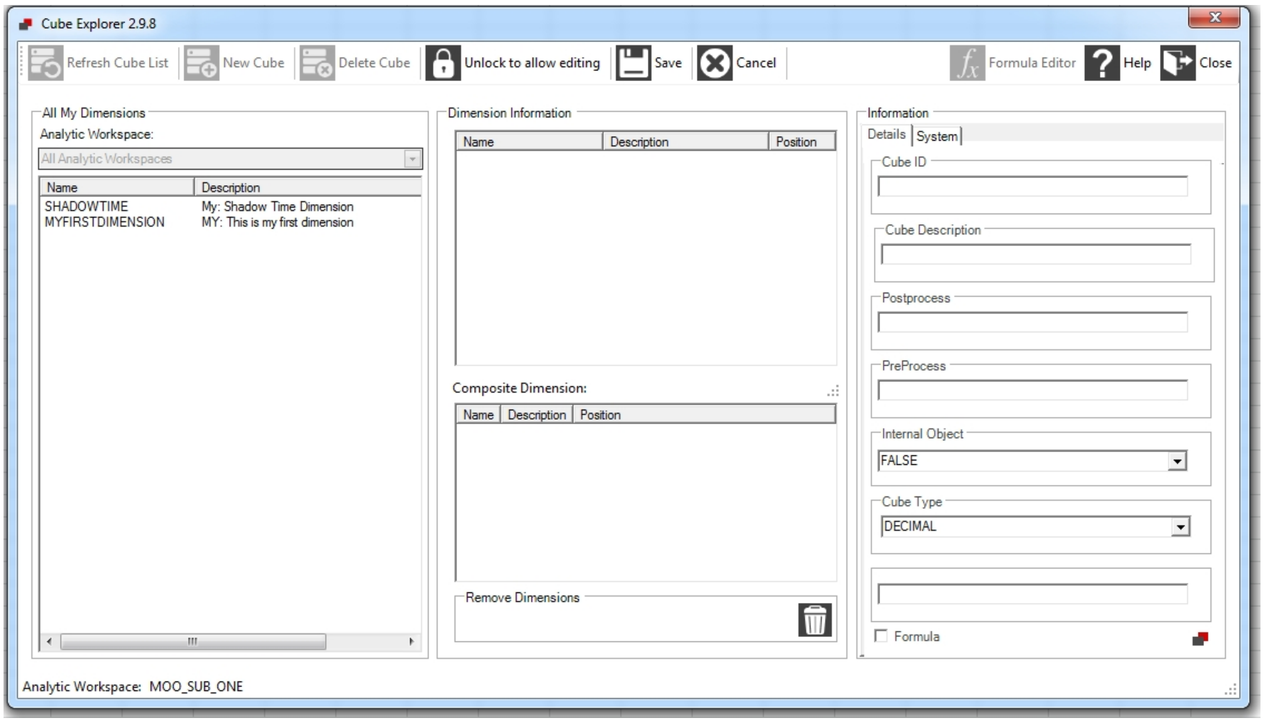
Which Analytic Workspace?.

You must choose the Analytic Workspace to create your new cube in.

myObjectiveOLAP Primary Analytic Workspace

myObjectiveOLAP Primary Analytic Workspace
My First Sub AW
My Second Sub AW

Once you have chosen which Analytic Workspace to create your cube within, only the dimensions which have been defined within the Analytic Workspace will be available to you for selection:



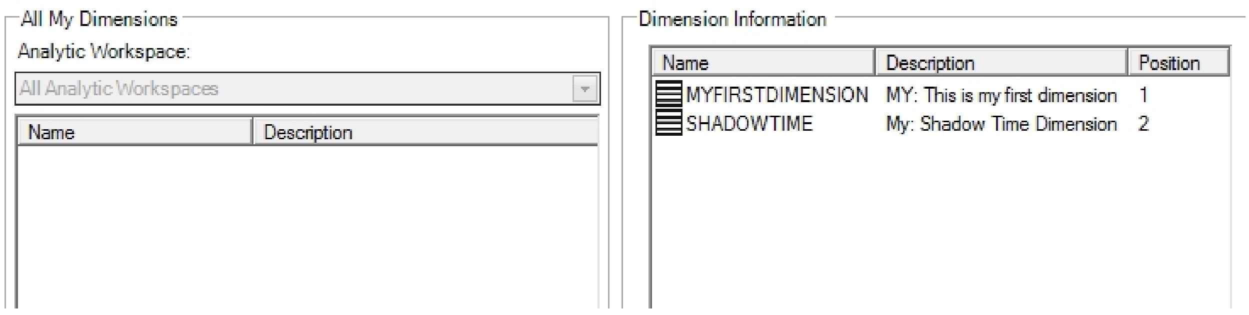
Name	Description
SHADOWTIME	My: Shadow Time Dimension
MYFIRSTDIMENSION	MY: This is my first dimension

Internal Object: FALSE

Cube Type: DECIMAL

Analytic Workspace: MOO_SUB_ONE

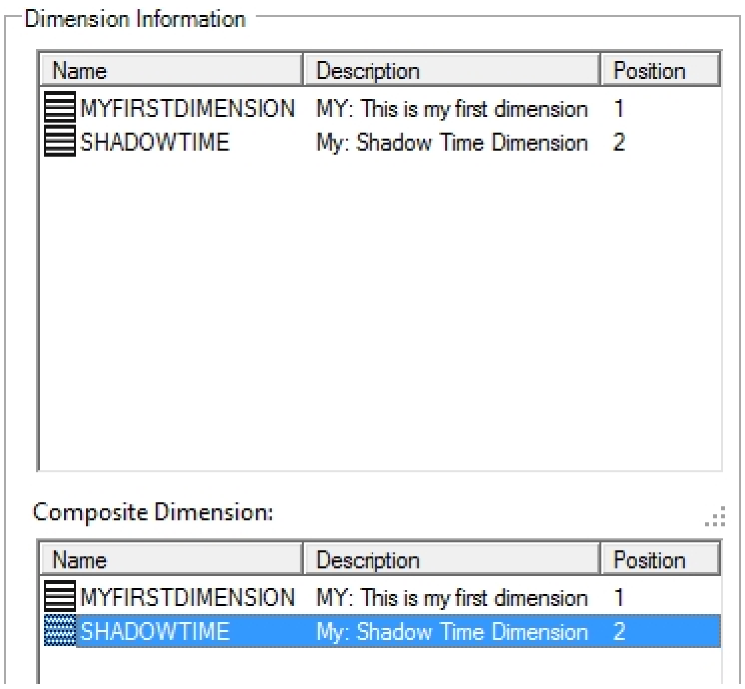
To define the dimensionality of your cube drag your dimensions from the All My Dimensions panel to the Dimension Information panel:



Name	Description	Position
MYFIRSTDIMENSION	MY: This is my first dimension	1
SHADOWTIME	My: Shadow Time Dimension	2

The order in which your dimensions are shown will control the order in which they are defined within the Oracle OLAP Analytic Workspace.

To composite your cube, you should drag the dimensions from the Dimension Information panel to the Composite Dimensions panel:

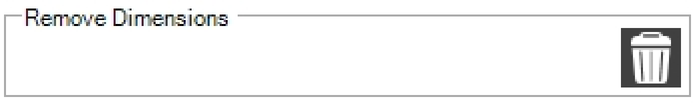

Dimension Information

Name	Description	Position
MYFIRSTDIMENSION	MY: This is my first dimension	1
SHADOWTIME	My: Shadow Time Dimension	2

Composite Dimension:

Name	Description	Position
MYFIRSTDIMENSION	MY: This is my first dimension	1
SHADOWTIME	My: Shadow Time Dimension	2

To remove a dimension from either the cube definition or the composite drag and drop the dimension from the appropriate panel onto the rubbish-bin icon:



Values, hierarchies, attributes.

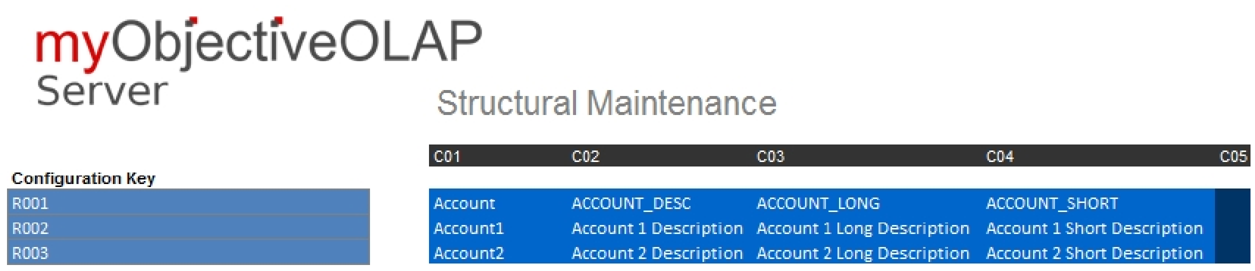
Values, Hierarchies and Attribute Maintenance

Whilst in most systems maintenance of Dimension Values and there associated meta-data is carried out through interfaces (Relational or OLAP), myObjectiveOLAP Server additionally allows for creation of this meta-data information through Excel, this enables both ad-hoc maintenance and easy bulk-creation or adaptation of this data.

Open a myObjectiveOLAP Structural upload template.

Column 1,	Row 1	Enter the Dimension name you wish to maintain.
Column 2 ---> N,	Row 1	Enter the names of any Variables, Attributes or Hierarchies you want to maintain.
Column 1,	Row 2 --> N	Enter the values of any new dimension values, or existing dimension values you wish to maintain.
Column 2 --> N,	Row 2 --> N	Enter the meta-data information you wish to apply to the dimension value on the same row.

In the following example we are going to create two dimension values on the Account dimension and populate the description information associated with that dimension.


myObjectiveOLAP Server

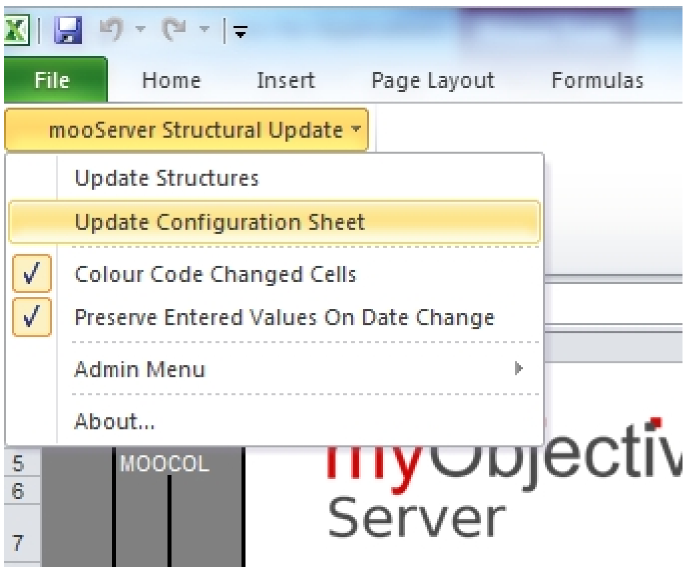
Structural Maintenance

Configuration Key

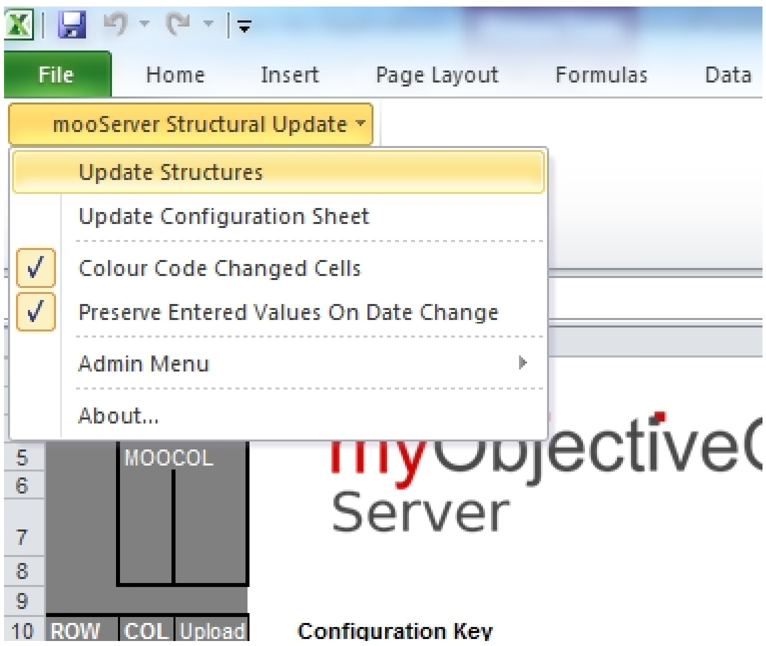
R001
R002
R003

C01	C02	C03	C04	C05
Account	ACCOUNT_DESC	ACCOUNT_LONG	ACCOUNT_SHORT	
Account1	Account 1 Description	Account 1 Long Description	Account 1 Short Description	
Account2	Account 2 Description	Account 2 Long Description	Account 2 Short Description	

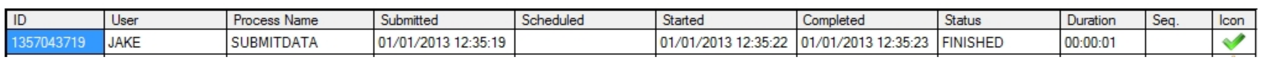
Now we prime the Structural upload template by running the Update Configuration Sheet menu item.


File
Home
Insert
Page Layout
Formulas
mooServer Structural Update
Update Structures
Update Configuration Sheet
Colour Code Changed Cells
Preserve Entered Values On Date Change
Admin Menu
About...
MOOCOL
myObjectiv
Server

Now we submit our changes to the database by running the Update Structures menu item.

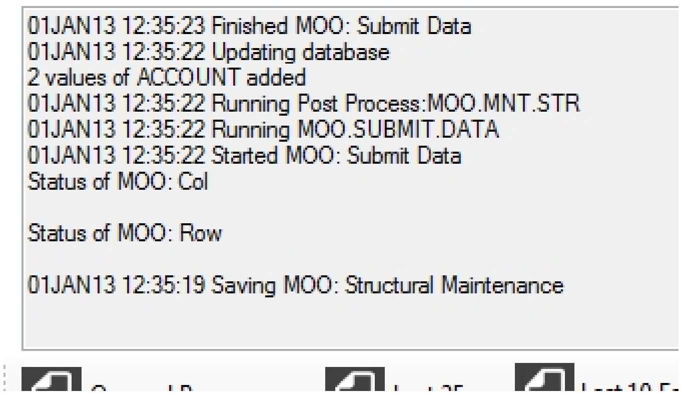


This places a SUBMITDATA Process in the Process Manager:



ID	User	Process Name	Submitted	Scheduled	Started	Completed	Status	Duration	Seq.	Icon
1357043719	JAKE	SUBMITDATA	01/01/2013 12:35:19		01/01/2013 12:35:22	01/01/2013 12:35:23	FINISHED	00:00:01		✓

Details of the Process are shown in the detail view:



```
01JAN13 12:35:23 Finished MOO: Submit Data
01JAN13 12:35:22 Updating database
2 values of ACCOUNT added
01JAN13 12:35:22 Running Post Process:MOO.MNT.STR
01JAN13 12:35:22 Running MOO.SUBMIT.DATA
01JAN13 12:35:22 Started MOO: Submit Data
Status of MOO: Col

Status of MOO: Row

01JAN13 12:35:19 Saving MOO: Structural Maintenance
```

01JAN13 12:35:19 Saving MOO: Structural Maintenance

If we open a console session we can see our dimension values have been created and our meta-data updated.

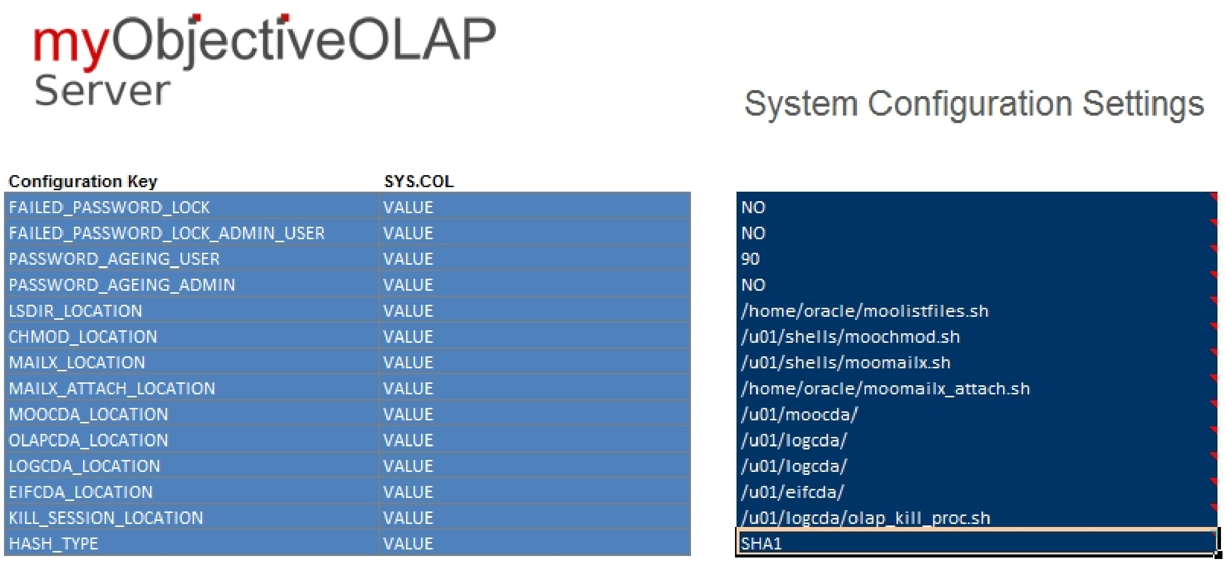
```
> rpr w 30 down account w 34 account_desc

ACCOUNT                                   ACCOUNT_DESC
------------------------------ ----------------------------------
Account1                       Account 1 Description
Account2                       Account 2 Description
```

You can either re-use the existing Structural Maintenance Template, or save them as dimension specific and add them to your structural maintenance library.

System Configuration

System Configuration

myObjectiveOLAP Server enables the application administrator to control global system options through the System Configuration upload template.



myObjectiveOLAP Server

System Configuration Settings

Configuration Key	SYS.COL	
FAILED_PASSWORD_LOCK	VALUE	NO
FAILED_PASSWORD_LOCK_ADMIN_USER	VALUE	NO
PASSWORD_AGEING_USER	VALUE	90
PASSWORD_AGEING_ADMIN	VALUE	NO
LSDIR_LOCATION	VALUE	/home/oracle/moolistfiles.sh
CHMOD_LOCATION	VALUE	/u01/shells/moochmod.sh
MAILX_LOCATION	VALUE	/u01/shells/moomailx.sh
MAILX_ATTACH_LOCATION	VALUE	/home/oracle/moomailx_attach.sh
MOOCDA_LOCATION	VALUE	/u01/moocda/
OLAPCDA_LOCATION	VALUE	/u01/logcda/
LOGCDA_LOCATION	VALUE	/u01/logcda/
EIFCDA_LOCATION	VALUE	/u01/eifcda/
KILL_SESSION_LOCATION	VALUE	/u01/logcda/olap_kill_proc.sh
HASH_TYPE	VALUE	SHA1

System Configuration Keys

The application administrator can change any of the options, detailed descriptions of the system configuration key values are shown in the table below:

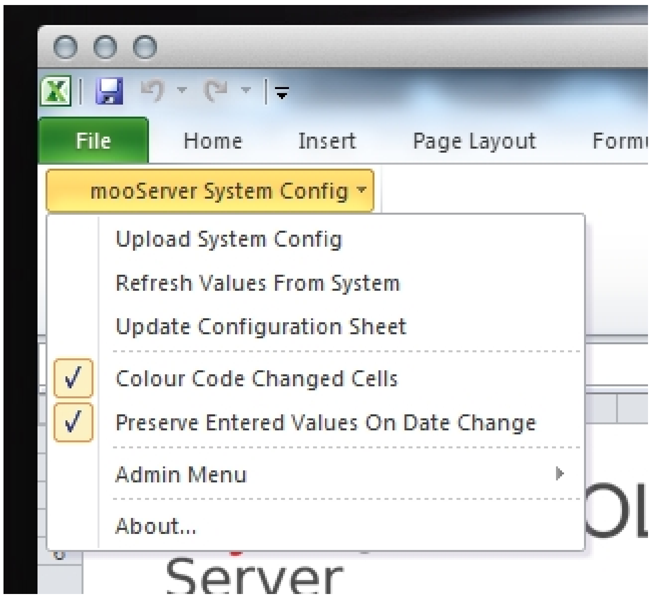
Key	Example	Description
FAILED_PASSWORD_LOCK	NO	Determines if a non-admin account is locked in USER.CTL if a user supplies an incorrect password on connecting to Oracle OLAP. If FAILED_PASSWORD_LOCK is set to NO then a user has unlimited password attempts. If an integer value is supplied then myObjectiveOLAP Server records the number of failed password attempts and will expire an account after the FAILED_PASSWORD_LOCK value is reached. This setting has no impact on user-accounts defined as Admin. Valid values: NO Integer [3]
FAILED_PASSWORD_LOCK_ADMIN_USER	NO	Determines if an admin account is locked in USER.CTL if a user supplies an incorrect password on connecting to Oracle OLAP. If FAILED_PASSWORD_LOCK is set to NO then a user has unlimited password attempts. If an integer value is supplied then myObjectiveOLAP Server records the number of failed password attempts and will expire an account after the FAILED_PASSWORD_LOCK value is reached. Valid values: NO Integer [3]
PASSWORD_AGEING_USER	90	Determines the number of days before a non-admin user is prompted to change their password Valid values: NO Integer [3]
PASSWORD_AGEING_ADMI	NO	Determines the number of days before an

N		admin user is prompted to change their password Valid values: NO Integer [3]
LSDIR_LOCATION	/home/oracle/ moolistfiles.sh	The physical location of the moolistfiles.sh shell script
CHMOD_LOCATION	/u01/shells/moochmod.sh	The physical location of the moochmod.sh shell script
MAILX_LOCATION	/u01/shells/moomailx.sh	The physical location of the moomailx.sh shell script
MAILX_ATTACH_LOCATION	/home/oracle/ moomailx_attach.sh	The physical location of the moomailx_attach.sh shell script
MOOCDA_LOCATION	/u01/moocda/	The file-system location of the moocda directory alias
OLAPCDA_LOCATION	/u01/logcda/	The file-system location of the olapcda directory alias. Note it is permissible to share file-system locations across directory alias.
LOGCDA_LOCATION	/u01/logcda/	The file-system location of the logcda directory alias
EIFCDA_LOCATION	/u01/eifcda/	The file-system location of the eifcda directory alias
KILL_SESSION_LOCATION	/u01/logcda/ olap_kill_proc.sh	The physical location of the olap_kill_proc.sh shell script
HASH_TYPE	SHA1	HASH_TYPE This is the Type of hashing applied to passwords stored and checked by mooserver. Default is TDES encryption and this will be used if no setting is applied. Other available valid settings are: SHA1 SHA256 SHA512 If an invalid setting is applied for example SHA666 then TDES will be applied. Changing this setting will invalidate all saved passwords. To reset passwords after a change use the User Manager tool, or to generate a standard password with a valid hash to update the USER.CFG magazine download the myObjectiveOLAP Hash Creation tool from the myObjectiveOLAP

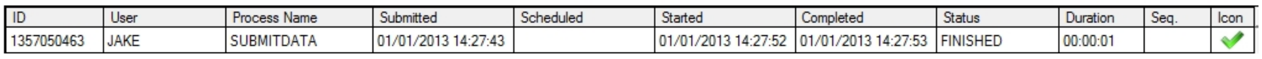
		support site.

Updating System Configuration Keys

Once you have updated any keys with settings specific to your installation, you should apply them to the system by choosing the "Upload System Config" from the mooServer System Config menu.

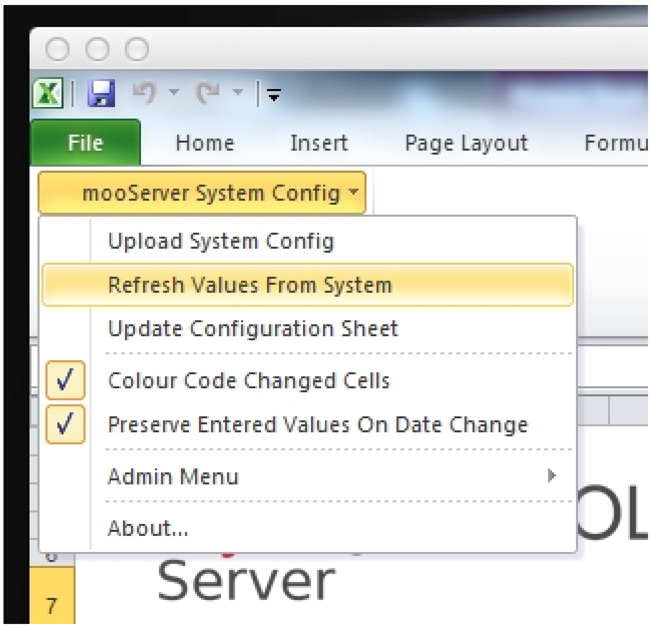

File
Home
Insert
Page Layout
Form
mooServer System Config
Upload System Config
Refresh Values From System
Update Configuration Sheet
Colour Code Changed Cells
Preserve Entered Values On Date Change
Admin Menu
About...
Server

Your changes will be sent to the Process Manager for validation, and will then be applied to the system.



ID	User	Process Name	Submitted	Scheduled	Started	Completed	Status	Duration	Seq.	Icon
1357050463	JAKE	SUBMITDATA	01/01/2013 14:27:43		01/01/2013 14:27:52	01/01/2013 14:27:53	FINISHED	00:00:01		✓

Once your SUBMITDATA Process has been processed by the Process Manager, you can check the current values in the system by running the "Refresh Values from System" menu item from mooServer System Config menu. This will populate the System Configuration upload template with that latest values stored in the database.


File
Home
Insert
Page Layout
Formu
mooServer System Config
Upload System Config
Refresh Values From System
Update Configuration Sheet
Colour Code Changed Cells
Preserve Entered Values On Date Change
Admin Menu
About...
Server

Health Check

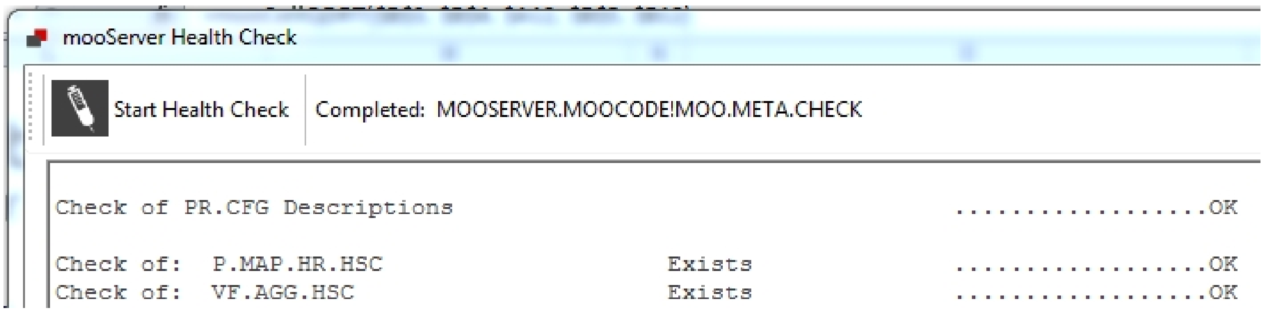
Health Check

mooServer Health Check

myObjectiveOLAP Server has a built in function to check the health of your system and identify any potential problems.

Access to the Health Check tool is from the myObjectiveOLAP Server menu.

To run the health check, press the Start Health Check button.



Completed: MOOSERVER.MOOCODE!MOO.META.CHECK

```
Check of PR.CFG Descriptions                                 ..................OK

Check of:  P.MAP.HR.HSC                 Exists               ..................OK
Check of:  VF.AGG.HSC                   Exists               ..................OK
```

Review the output and fix any issues which the Health Check tool reports, if you are unsure contact Support.

The Health Check tool runs the following OLAP DML MOOSERVER.MOOCODE!MOO.META.CHECK

MOO.META.CHECK is structured in a modular manner, and is pre-populated with standard checks, you are free to add any additional checks by adding the following template module:

```
call moo.attach.aw('MOOSERVER.AW_IN_WHICH_YOU_WANT_TO_RUN_THE_CHECK')
[
  Either your code or a call to your own program.
  If you are calling your own code you should place the following additional attach
  statements
  [
   call moo.attach.aw('MOOSERVER.LOCALCODE')
   call moo.detach.aw('MOOSERVER.LOCALCODE')
  ]
]
call moo.detach.aw('MOOSERVER.AW_IN_WHICH_YOU_WANT_TO_RUN_THE_CHECK')
```

You should backup your MOO.META.CHECK program text before any myObjectiveOLAP Server upgrades are completed and ensure that you add any additional custom modules code into the upgraded version. The upgrade process will automatically attempt to merge your changes into any new versions, but you should always check the result post upgrade.

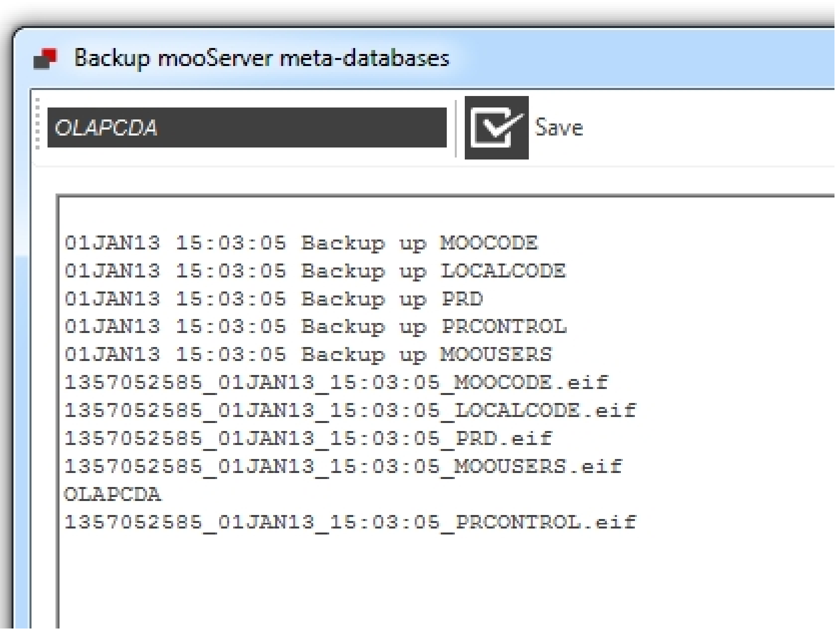
mooServer Backup

mooServer Backup

Backup mooServer Code AWs

myObjectiveOLAP Server enables you to quickly and easily snapshot your meta-data analytic workspaces.

To access the mooServer Backup utility access the tool from the myObjectiveOLAP Server ribbon menu.



```
01JAN13 15:03:05 Backup up MOOCODE
01JAN13 15:03:05 Backup up LOCALCODE
01JAN13 15:03:05 Backup up PRD
01JAN13 15:03:05 Backup up PRCONTROL
01JAN13 15:03:05 Backup up MOOUSERS
1357052585_01JAN13_15:03:05_MOOCODE.eif
1357052585_01JAN13_15:03:05_LOCALCODE.eif
1357052585_01JAN13_15:03:05_PRD.eif
1357052585_01JAN13_15:03:05_MOOUSERS.eif
OLAPCDA
1357052585_01JAN13_15:03:05_PRCONTROL.eif
```

Enter a valid Oracle database directory alias which the MOOSERVER database user has WRITE access to and press Save.

The backup tool will backup the following analytic workspaces:

Analytic Workspace	Contents
MOOCODE	myObjectiveOLAP standard server code
LOCALCODE	Your local OLAP DML programs
PRD	The Process Manager daemon
MOOUSERS	All the information and meta-data relating to your application users.
PRCONTROL	All the information and meta-data relating to your application Processes and Workflows

All the analytic workspaces will be exported in EIF format to the directory alias supplied. If an invalid directory alias or no alias is supplied, myObjectiveOLAP will report an error.

EIF files will be created in the following format:

```
[POSIX time]_[DDMMMYY]_[HH]:[MM]:[SS]_[AWNAME].eif
```

Backups can be queried either by creating a report, or looking at the MOOSERVER.MOOBACKUP$BACKUP.CTL variable:

```
> aw attach MOOBACKUP ro

> lmt BACKUP to last 1

> rpr w 40 down BACKUP.ATT w 50 BACKUP.CTL

                                         --------------------
BACKUP.CTL--------------------
                                         ----------------------
BACKUP----------------------
BACKUP.ATT                                                  1357052585
----------------------------------------
--------------------------------------------------
```

```
LOG                                         01JAN13 15:03:05 Backup up MOOCODE
                                            01JAN13 15:03:05 Backup up LOCALCODE
                                            01JAN13 15:03:05 Backup up PRD
                                            01JAN13 15:03:05 Backup up PRCONTROL
                                            01JAN13 15:03:05 Backup up MOOUSERS
MOOCODE_FILE                                1357052585_01JAN13_15:03:05_MOOCODE.eif
MOOLOCALCODE_FILE                           1357052585_01JAN13_15:03:05_LOCALCODE.eif
PRD_FILE                                    1357052585_01JAN13_15:03:05_PRD.eif
MOOUSERS_FILE                               1357052585_01JAN13_15:03:05_MOOUSERS.eif
CDA                                         OLAPCDA
PRCONTROL_FILE                              1357052585_01JAN13_15:03:05_PRCONTROL.eif


> aw detach MOOBACKUP
```

Backups of MOODATA would normally be carried out as part of your database backup and recovery strategy. RMAN etc....

Multi AW Mode

Multi AW Mode

myObjectiveOLAP enables multiple Analytic Workspaces to be created for end-user data.   Dimensions can be shared between the Analytic Workspaces, data transferred, and Processes and Workflows enabled for each Analytic Workspace.

myObjectiveOLAP Server enables full partitioning of the security model across cubes, dimensions or at the Analytic Workspace level.

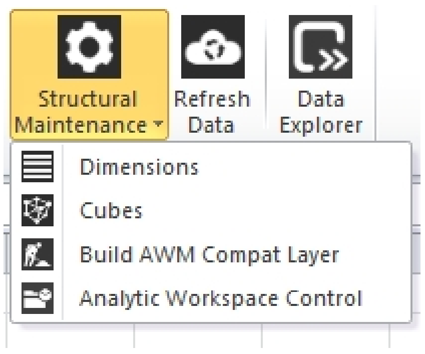
Reasons to consider Multi-AW

- Hard partition your security model.
- Enable parallel Process Manager.
- Segregate your data model.

The following topics take you through the management of Multi AW mode.

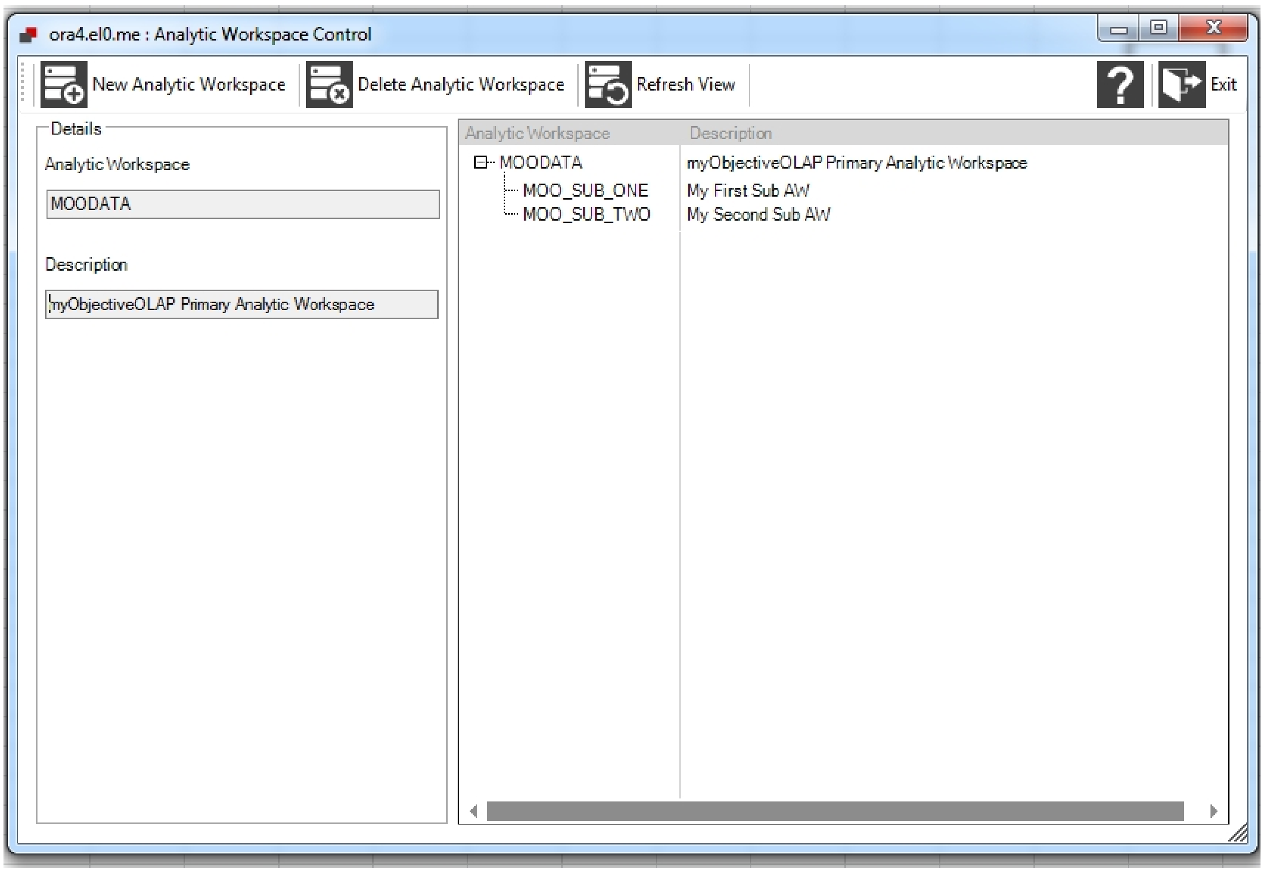
Managing Multi AW

Managing Multi AW Mode



myObjectiveOLAP Analytic Workspace Control enables you to create or delete new child Analytic

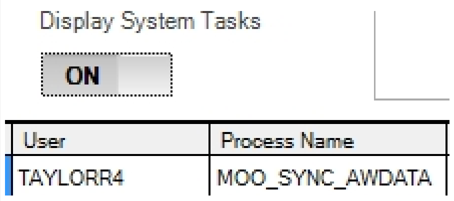
Workspaces. You will always have the primary Analytic Workspace MOODATA, which contains the data dictionary for your application, although should you choose no data needs to be stored within the primary analytic workspace. Multi AW mode was introduced in myObjectiveOLAP Server 2.9.8, applications created prior to 2.9.8 can take advantage of Multi AW by upgrading to 2.9.8.


ora4.el0.me : Analytic Workspace Control
New Analytic Workspace
Delete Analytic Workspace
Refresh View
Exit
Details

Analytic Workspace: MOODATA

Description: myObjectiveOLAP Primary Analytic Workspace

Analytic Workspace	Description
MOODATA	myObjectiveOLAP Primary Analytic Workspace
MOO_SUB_ONE	My First Sub AW
MOO_SUB_TWO	My Second Sub AW

Option	Purpose
New Analytic Workspace	Create a new Analytic Workspace and register it in the application configuration. As soon as you create a new Analytic Workspace users will have the new Analytic Workspace attached when they login. Dimensions and cubes can be created in the child Analytic Workspace. Processes and Workflows can be defined to execute within the Analytic Workspace. A synchronisation process will execute in the Process Manager to register the meta-data for the new Analytic Workspace:  Display System Tasks: ON; User: TAYLORR4; Process Name: MOO_SYNC_AWDATA
Delete Analytic Workspace	Deletes an Analytic Workspace and removes it from the data model.
Refresh View	Refreshes the view of all Analytic Workspaces

Once a new Analytic Workspace has been created you will be able to assign objects and processes to be assigned to it.

Workflows and Processes can be identified as running in a specific Analytic Workspace through the Process Manager

Submitting Data

Submitting Data

User data can be submitted to the database directly from within Microsoft Excel. This offers the end-user the comfort of a familiar environment and enables them to model data within Excel and submit the final result.   All the upload templates also act as reports retrieving the data already loaded into the system so the user can view previously submitted data.

Users of "SDMC OFA Connect" often localized to "{YourCompanyName} OFA Connect" can take advantage of the "myObjectiveOLAP Upload Upgrade Tool", please contact for support for a copy of the tool and instructions for use.

Excel Reporting Functions

mooServer Excel Functions

The following functions can only be used against mooServer enabled Oracle OLAP Analytic Workspaces.

They are used to meet ad-hoc or complex reporting requirements.

These can be used by users of mooServer in addition to the standard myObjectiveOLAP Microsoft Excel Functions.

mooDimDesc

=mooDimDesc(“[DIMENSION]”, “[DIMENSION_VALUE]”, [OPTION])

Returns the description for a given dimension value in a mooServer enabled Analytic Workspace

Syntax

```
=mooDimDesc("[DIMENSION]", "[DIMENSION_VALUE]", [OPTION])
```

Return Value

```
STRING
```

Option

The third option argument tells the mooDimDesc function what description you wish to retrieve:

Option   0
Passing 0 returns the column description

Option   1
Passing 1 returns the row description

Option   2
Passing 2 returns the long description description

Example 1

=mooDimDesc("CUSTOMER", "ACCOUNT_BAVARIAN IND", 0)

Example Output

```
ACCOUNT_BAVARIAN IND
```

Example 2

=mooDimDesc("CUSTOMER", "ACCOUNT_BAVARIAN IND", 1)

Example Output

```
Bavarian Industries
```

Example 3

=mooDimDesc("CUSTOMER", "ACCOUNT_BAVARIAN IND", 2)

Example Output

```
ACCOUNT_BAVARIAN IND Bavarian Industries
```

mooCellQDR

=mooCellQDR("[CUBE]", "[dim1]", "[dim1value]", "[dim2]", "[dim2value]", etc....)

Returns the numeric result of a qualified data reference from a numerical variable within an Analytic Workspace.

Syntax

```
=mooCellQDR("[CUBE]", "[dim1]", "[dim1value]", "[dim2]",
"[dim2value]", etc....)
```

Return Value

```
DECIMAL
```

Example

=mooCellQDR("UNITS_CUBE_COST", "CUSTOMER", "ACCOUNT_BAVARIAN IND", "TIME", "MONTH_2006.02", "CHANNEL", "TOTAL_TOTAL", "PRODUCT", "TOTAL_TOTAL")

Example Output

41822.97

Limitations

- Can only be used to return numerical data, TEXT / STRING data must be returned using the mooCellQDRString() function.

- mooCellQDR can not be seen interacting with the OLAP engine through the Oracle OLAP RECAP DML statement.

- myObjectiveOLAP supports retrieving data using mooCellQDR on cube between 1 and 14 dimensions in size.

Data Explorer

Data
Explorer

Data Explorer

Data Explorer provides a graphical tool for reporting on OLAP data from a myObjectiveOLAP Server enabled Oracle OLAP database.

OLAP data is stored in ‘Cubes'. The edges of the cubes are dimensions, and they represent business organisational elements such as Cost Centres, Products and Time Periods. A cube may have several dimensions of varying sizes, and dimensions may have hundreds or thousands of values.

An important aspect of dimensions is that they are usually hierarchical, and this allows the data in the cube to be aggregated into higher levels in the organisations, and it also allows a structured access route to the data (by ‘drilling down' through the hierarchy).

Cubes can therefore be very large, and commonly they are populated with data in particular regions but empty in others. The ‘emptiness' of the cube is referred to as ‘sparsity'. It is important therefore to have a good understanding of the dimensions used by the organisation when selecting data for reporting

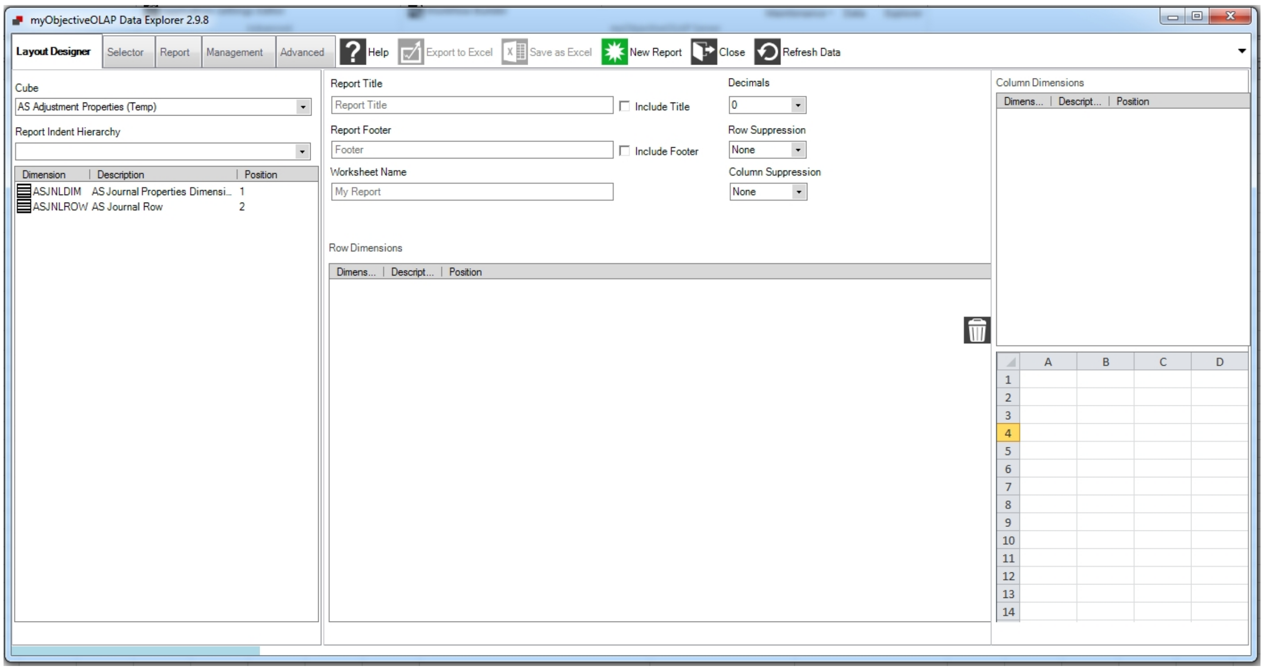
Using Data Explorer

Using Data Explorer

Select Data Explorer from the main myObjectiveOLAP menu.

Data
Explorer

This opens the Data Explorer Designer tool, initially at the Layout Designer panel.



Cube: AS Adjustment Properties (Temp)

Report Indent Hierarchy:

Dimension	Description	Position
ASJNLDIM	AS Journal Properties Dimensi...	1
ASJNLROW	AS Journal Row	2

Report Title: Report Title ☐ Include Title

Report Footer: Footer ☐ Include Footer

Worksheet Name: My Report

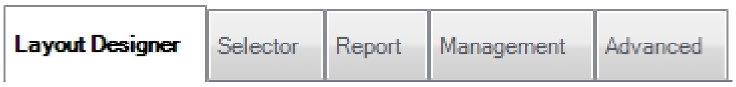
Decimals: 0

Row Suppression: None

Column Suppression: None

The top part of the screen shows the selection tabs to the Work Area panels of the Data Explorer tool.

Work Area

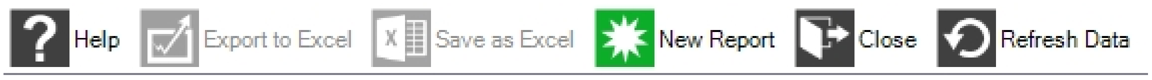


Each Work Area panel allows for different actions to be taken during the construction, running or maintenance of your report.

The Work Areas are summarised below:

Work Area	Summary
Layout Designer	Choose a cube and its associated dimensions, assign the dimensions to rows and columns of the report, and include header, footer and formatting.
Selector	Refine the dimensions to reduce the scope of the report
Report	Displays the report, and allows some fine-tuning of the report appearance.
Management	Enables you to Save your report definition, open a saved report or schedule to be run by the server and emailed to you at a later date and time.
Advanced	This panel can be used by your application administrator to understand the impact of your report on the Oracle OLAP data model

Common Controls



The Common Controls Ribbon Menu is always available to you, irrespective of which Work Area you are currently working in.

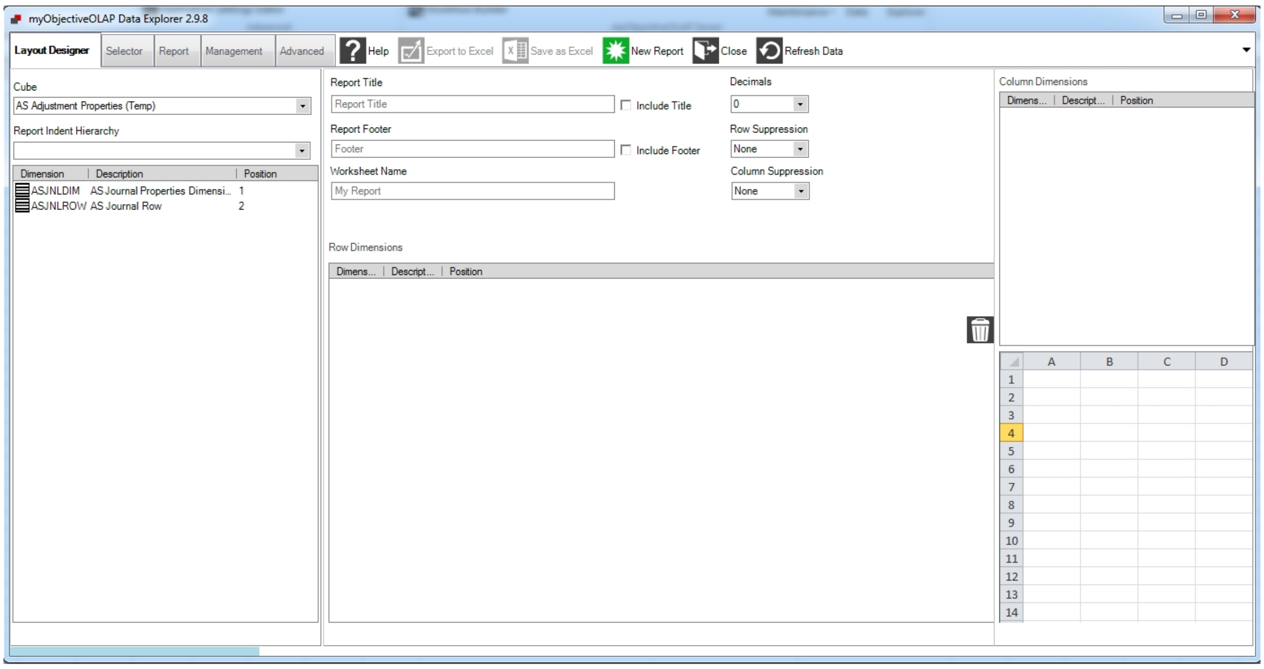
The purpose of the Common Controls are summarised below:

Common Control	Summary
Help	Access the myObjectiveOLAP Help system.
Export to Excel	Exports the current data selection to Microsoft Excel
Save as Excel	Saves the current Data Selection to Microsoft Excel
New Report	Discards the current report and start a new one.
Close	Exit Data Explorer
Refresh Data	Resets your connection to the database, allowing you to see the latest data-set. Note this will reset your report selections

Layout Designer

Layout Designer

Layout Designer allows you to select the Cube you want to view and the row, column and paging dimension axis for your report.



Cube: AS Adjustment Properties (Temp)

Report Indent Hierarchy:

Dimension	Description	Position
ASJNLDIM	AS Journal Properties Dimensi...	1
ASJNLROW	AS Journal Row	2

Report Title: Report Title ☐ Include Title

Report Footer: Footer ☐ Include Footer

Worksheet Name: My Report

Decimals: 0

Row Suppression: None

Column Suppression: None

Row Dimensions

Dimens...	Descript...	Position

Column Dimensions

Dimens...	Descript...	Position

Use the Cube drop-down selector to choose the cube you wish to view.

The Report Indent Hierarchy drop-down list, if populated, allows you to choose a hierarchy (a dimension may have many hierarchies). If the dimension forms the lowest level of rows of your report, the rows will be indented according to the level within the selected hierarchy. This enables you to see the relationship between rows for a hierarchical dimension. If the dimension has no hierarchy, this box will be empty. This drop-down list is only populated after you have selected one or more dimensions for the rows of your report.

Below this will appear a list of the dimensions of the cube. These are available to become the rows, columns and pages of the report.

To assign a dimension to the columns of the report, drag the dimension name to the Column Dimensions box (far right).

To assign a dimension to the rows of the report, drag the dimension name to the Row Dimensions box (lower centre).

Rows and columns can comprise more than one dimension, and any remaining dimensions not assigned will form the paging dimensions of the report.

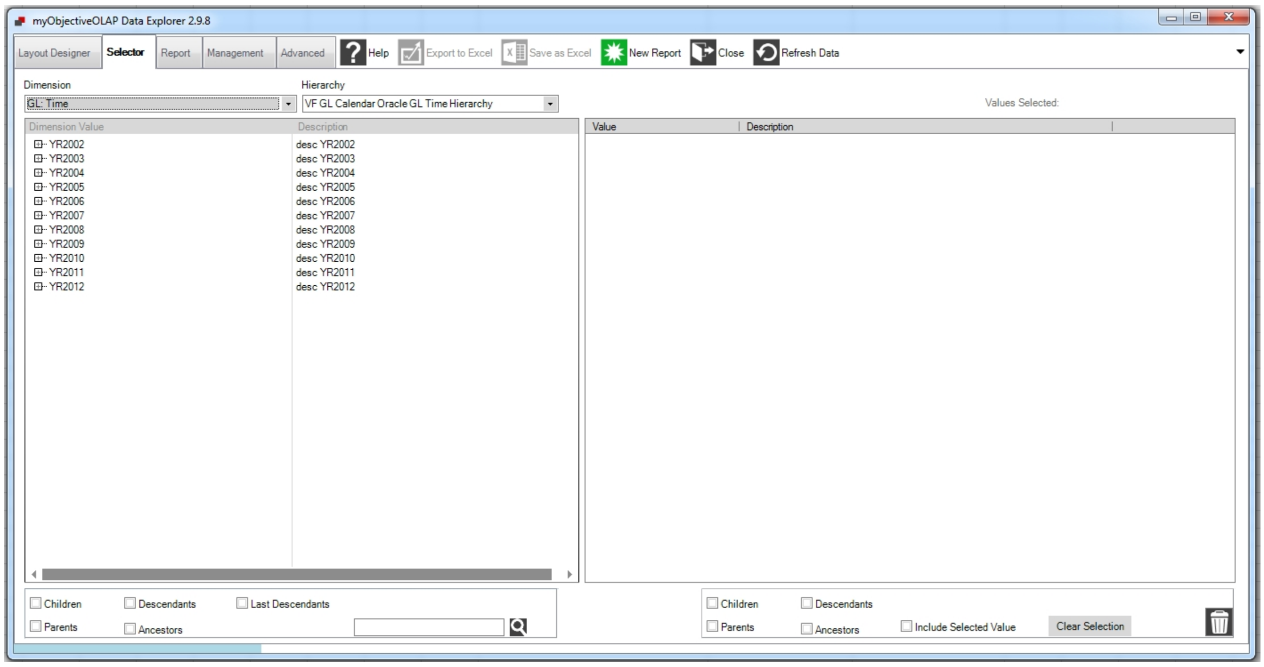
The upper centre portion of the Work Area allow you to give the report a text header and footer, provide a worksheet name, and to set the display options.

Row and Column suppression allow rows and columns to be omitted from the report if they contain no data, or if the data contains only zeros. 'NA' which means that no data has ever existed against the specified combination.

Selector

Selector

The purpose of the Selector tab is to reduce the scope of the report, by selecting only those dimension values that are appropriate for the report.



Dimension: GL: Time

Hierarchy: VF GL Calendar Oracle GL Time Hierarchy

Values Selected:

Dimension Value	Description
YR2002	desc YR2002
YR2003	desc YR2003
YR2004	desc YR2004
YR2005	desc YR2005
YR2006	desc YR2006
YR2007	desc YR2007
YR2008	desc YR2008
YR2009	desc YR2009
YR2010	desc YR2010
YR2011	desc YR2011
YR2012	desc YR2012

Value	Description

Overview

The left half of the screen contains the Available Panel, this allows you to locate the dimension values you require, using a hierarchy if appropriate. The right half of the screen contains the Selected Panel, which displays a list of dimension values which you have selected. To move dimension values from Available to Selected just drag and drop selected dimension values from left to right.

The Dimension drop-down selector allows you to select a dimension of the cube. Select each of these in turn, and make your selection from the pane below. You can also make multiple selections by using the Shift or Ctrl keys. To highlight a block of dimension values, highlight the first value, then highlight the last while holding down the Ctrl key. The whole block should be highlighted. To highlight several separated dimension values, hold down the Ctrl key while clicking on each required value. You can also remove selected values by clicking on them again. Drag and drop the highlighted multiple selection into the right panel.

Row and Column dimensions will usually have multiple dimension values. For other dimensions, only the first value in the selection list will be visible in your report, so it is best to select only one value.

Hierarchies are initially displayed in the ‘collapsed' state. Expand a branch of the hierarchy by clicking on the [+] symbol. Drag any ‘branch' or ‘leaf' of the hierarchy into the right pane.

At the lower edge of the panel there are some check boxes that allow you to select values from within the hierarchy based on your selection. Highlight a dimension value and click one of the relation boxes. Then drag the dimension value into the Selected panel. The relations will also be added to the right Selected panel.

Search tool

To search for a dimension value, type a part of its name into the box and click the magnifying glass symbol. The Dimension value list will show a sub-set according to your search string. To revert to the full list, clear the search box and click the magnifying glass symbol again. You must turn off the Hierarchy to enable the Search Tool

Clear Selection

Clear Selection

Pressing Clear Selection will remove all values that have previously been Selected.

Removing individual values

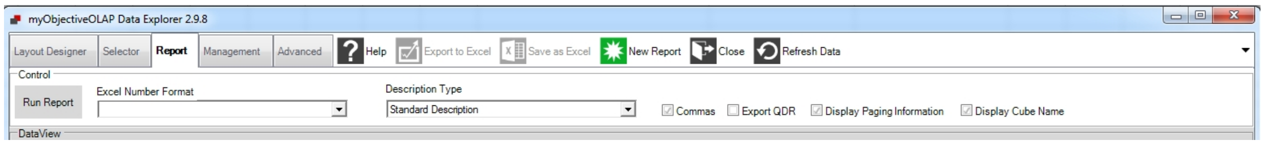
To remove individual unwanted values, you can drag and drop them into the bin at lower right.

Report

Report

This tab displays the report in a spreadsheet layout. Click the Run report button to re-display the report data.

At the top of the work area there are some options which control the display of data:



Excel Number Format:

Description Type: Standard Description

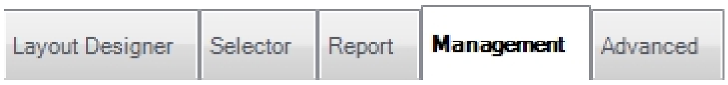
[✓] Commas [ ] Export QDR [✓] Display Paging Information [✓] Display Cube Name

The following tables summarises the purpose of the Report Options

Option	Summary
Excel Number Format	How the data will be formatted when exported to Excel
Description Type	How the dimension values are presented in the report.
Commas [check-box]	Whether commas are used as thousands separators
Export QDR [check-box]	Report cells will contain QDR formulas (see Export Excel)
Display Paging Information [check-box]	Displays values of Paging dimensions at the top of the report
Display Cube Name [check-box]	Displays the Cube name at the top of the report.

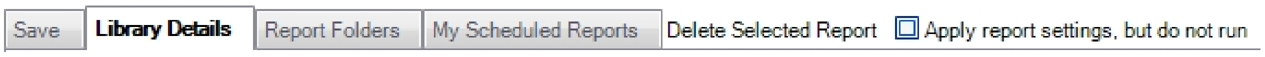
Management

Management



Management enables you to manage your reports library.

The following sub-menus group the management console:



Menu	Summary
Save	Save a created report definition to the database so it can be run at a later date and time.
Library Details	Contains all reports you have saved. You can open one of your reports by double-clicking the report id in the first column.
Report Folders	Allows you to organise your reports into folders. Also allows you to schedule

	reports for running at a later time, and to run them at regular intervals.
My Scheduled Reports	Displays reports which are scheduled to run later, and allow you to cancel a scheduled report.
Delete Selected Report	Allows deletion of reports in tabs where they are displayed.
Apply Report Settings	Allows a report to be loaded without loading data.

Saving your report definition

This allows you to save the current report definition as a named report.

OLAP Report Name : {enter a meaningful name for your report}

Excel Worksheet Name : {enter a worksheet name}   This will appear as the sheet name when you export your report to Excel.

Click the Save button.

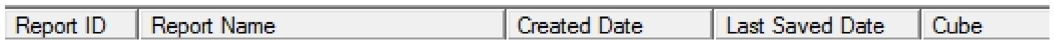
An important aspect of saving Reports in myObjectiveOLAP is that they are saved without data. Reports are saved in an OLAP workspace. Each time you open or run a report, it will be populated with the current data (as at the time you logged in or Refreshed Data - see below).

The panel below the Save button provides a technical report on details of the saved report (this can sometimes be useful for support purposes).   The final line of the Output will display the assigned Report ID when saving a report:

```
Report Saved Successfully.  Report ID: 1389257738
```

Library Details

This provides a list of your reports, displayed in tabular format with the following columns:



Report ID	Report Name	Created Date	Last Saved Date	Cube

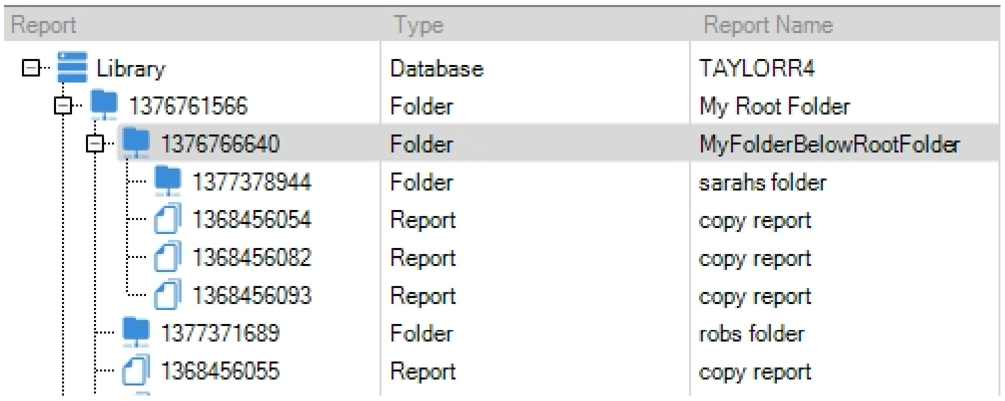
You can sort the list of reports by any of the columns by clicking on the column name tile. Click again to reverse the sort order.

To run a report, double-click on the Report ID. This is affected by the Apply Report Settings check-box. If the box is checked, the report definition is loaded but the report is not run. This allows you to modify the report before running it.

To delete a report, highlight the Report ID and click the Delete Selected Report button above the column titles. You will be asked to verify your intention to delete the report.

Report Folders

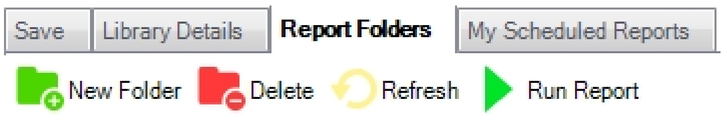
Your report library can be organised into folders to enable you to manage them more easily.



Report	Type	Report Name
Library	Database	TAYLORR4
1376761566	Folder	My Root Folder
1376766640	Folder	MyFolderBelowRootFolder
1377378944	Folder	sarahs folder
1368456054	Report	copy report
1368456082	Report	copy report
1368456093	Report	copy report
1377371689	Folder	robs folder
1368456055	Report	copy report

The list of reports is presented in a hierarchical format, arranged into folders that you can create according to your requirements.

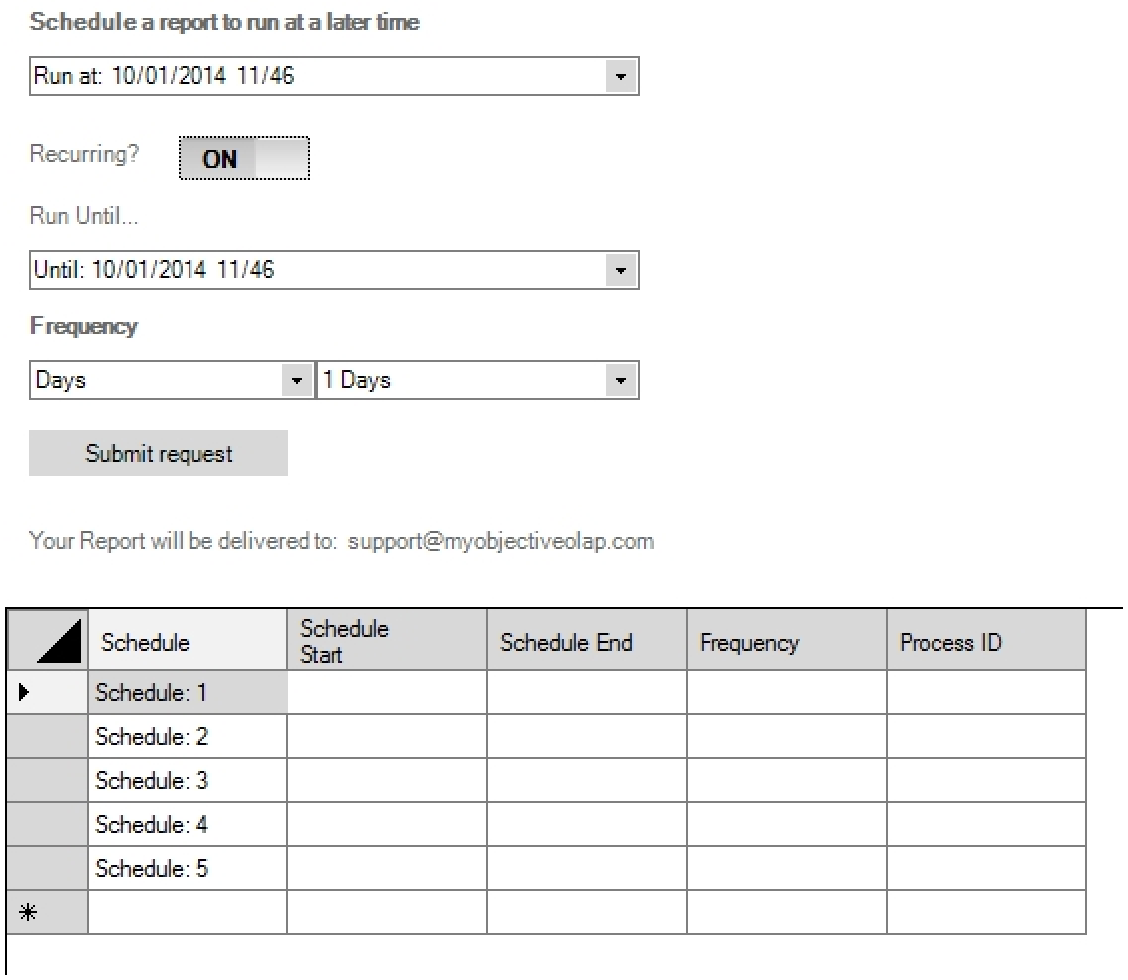
The following mini Ribbon menu options are available:



Ribbon Item	Summary
New Folder	**Create a new folder.** Highlight the place where you want to create the folder (Library, or another folder), and click the New Folder Tab. The dialog box requests a folder name
Delete	**Delete a report or folder** Highlight the report or folder for deletion and click the Delete button.
Refresh	Refresh the library from the database
Run Report	Run the selected report

Scheduled Reports

You can schedule a report to run at a later time, or at intervals. Highlight the Report ID in the Library hierarchy. The fields in the schedule panel will then be available


Schedule a report to run at a later time

Run at: 10/01/2014 11/46

Recurring? ON

Run Until...

Until: 10/01/2014 11/46

Frequency

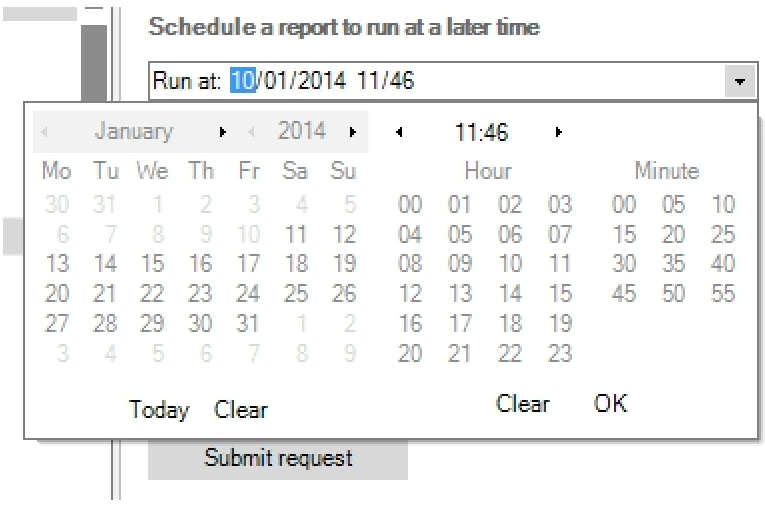
Days | 1 Days

Submit request

Your Report will be delivered to: support@myobjectiveolap.com

	Schedule	Schedule Start	Schedule End	Frequency	Process ID
▶	Schedule: 1				
	Schedule: 2				
	Schedule: 3				
	Schedule: 4				
	Schedule: 5				
*					

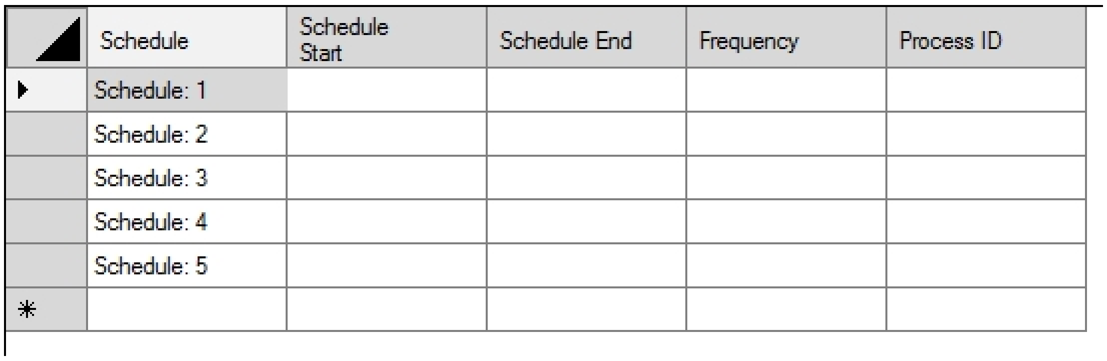
Click on the field marked ‘Schedule a report to run at a later time' to open a Graphical Date/Time selector tool. Select a future date and time.


Schedule a report to run at a later time

Run at: 10/01/2014 11/46

If you wish to run your report at repeating intervals, click the Recurring button so that it displays ON.

Then click on the ‘Run Until … ' field to choose an end Date and Time. Select the Frequency in Hours, Days or Weeks. When you are happy with your settings, click Submit Request. Your scheduled report will appear in the table below.



	Schedule	Schedule Start	Schedule End	Frequency	Process ID
▶	Schedule: 1				
	Schedule: 2				
	Schedule: 3				
	Schedule: 4				
	Schedule: 5				
✱					

Each report can be scheduled with up to five schedule slots.

Your report will be emailed to the email address shown below the Submit Request button.   This is the email address which has been defined by your myObjectiveOLAP Server administrator and cannot be changed by the end-user.

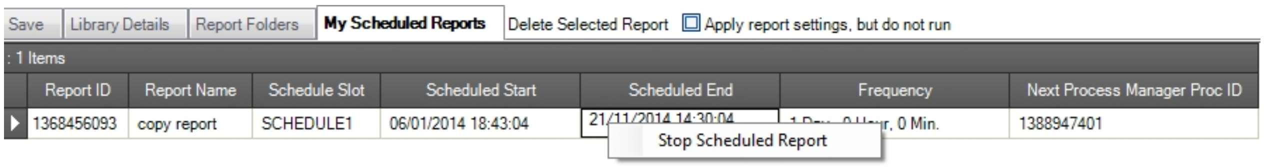
Submit request

Your Report will be delivered to: support@myobjectiveolap.com

If you want to delete a scheduled report, right-click on the row in the table and choose ‘Stop Scheduled Report'.

My Scheduled Reports

My Scheduled Reports lists all reports scheduled under your user ID. Right-clicking on an individual scheduled report allows you to stop an individual report schedule.



: 1 Items

	Report ID	Report Name	Schedule Slot	Scheduled Start	Scheduled End	Frequency	Next Process Manager Proc ID
▶	1368456093	copy report	SCHEDULE1	06/01/2014 18:43:04	21/11/2014 14:30:04	1 D... 0 H...r, 0 Min.	1388947401

Delete Selected Report button

When using the Library Details or the Report Folders, you can delete a report by highlighting the report name and clicking on the Delete Selected Report button. A dialog box will ask you to verify your intention.

Apply Report Settings check-box

In Library Details you can run a report by double-clicking on the report ID. In Report Folders, you can run a report by highlighting the report and clicking the Run button.

In either case, if the Apply Report Setting box is checked, the report will be loaded but the report will not be run. This sets the selections of dimensions and other report choices, but does not load new data from the OLAP workspace.

Advanced tab

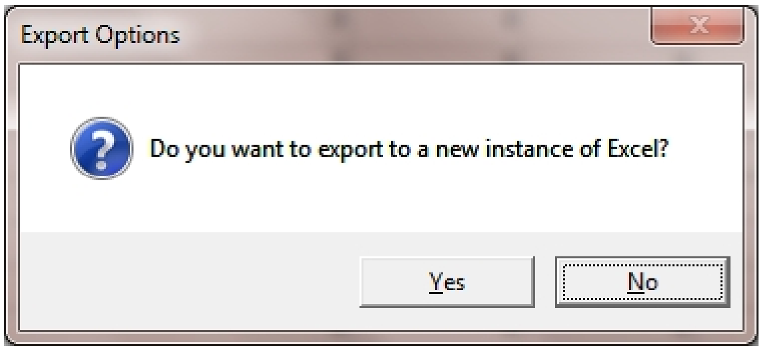
This work area provides OLAP DML command lists, mainly of interest to developers.

Export Excel

Export to Excel

Export to Excel

Use this button to copy your report to an instance of Excel. You will be presented with a choice:



Export Options

Do you want to export to a new instance of Excel?

Yes

The report will be exported to a workbook in a new instance of Excel which is independent of the instance running myObjectiveOLAP.

No

The report will be exported to a workbook which is in the same instance of Excel which running my ObjectiveOLAP.
The workbook cannot be closed until after the Data Explorer is closed.

Export QDR

When this option is ticked (in the header area of the Report tab), the Export to Excel option writes formulas in the worksheet cells instead of the data. These formulas use myObjectiveOLAP functions to refer directly to the data in the OLAP analytic workspace, for example:

```
     =mooCellQDR("ABCOH","ABCACC","E870TO895","ABCCC","CCA000000","GL_TIME1","
YR2012")
```

If the instance of Excel is correctly logged in to OLAP, the formula provides the required data. If the instance is not logged in, the formula returns an error message:

```
     You are not connected to an Oracle OLAP database
```

You can now run the Excel linked report directly to the database without using Data Explorer.

Save as Excel button

Save as Excel

Saves the current data as an Excel worksheet file. A normal file selector window is opened so that you can create a new file or choose to over-write an existing one. The file type is .xls (excel spreadsheet), and can be opened with Excel.

Close button

Close

Exit from the Data Explorer and return to the myObjectiveOLAP main menu.

New Report button

New Report

Discard the current report and start a new one.

Refresh Data button

Refresh Data

Refresh the connection with the OLAP database in order to see any recent changes to the data.

Background information: When you log in to myOracleOLAP, you connect to the OLAP workspace, and the data that you see during your session reflects the saved state of the data at that time. If the OLAP workspace data changes during your session, you do not normally see those changes, and this ensures that you see a consistent view of the data during your session. If you want to see an updated view of the OLAP workspace data, you can use the Refresh Data button; this has a similar effect to logging in again.

Your report settings will be lost when you do this. To avoid this, save your report first, then re-open it after refreshing.

API

mooServer API

The following API calls are supported. They reside in MOOSERVER.MOOCODE. You are free to make use of these API calls in your own application. We at myObjectiveOLAP will ensure that we provided backwards compatibility for these APIs.

Important Note.

We have made the decision not to HIDE the source code for these supported APIs to help you understand their execution and use, and should you have issues using them enable PRGTRACE.

However:

- The source code remains subject to SDMC Consulting Limited copyright.
- You may not publish on the Internet any part of the source code. Only exception is to the support.myobjectiveolap.com support portal.
- You may not take any part or all of the source code and use it in any other application.
- You may not copy, distribute any part or all of the source code by any medium.

Failure to understand and comply could result in any or all of the following actions.

- Removal of your license to use myObjectiveOLAP no license fee will be returned.
- Disposal of your support agreement no outstanding support fees will be returned.

We do not limit our remedies to the above and reserve the right to seek damages for any breech.

At the very least we will start hiding the API source code.

That said, if you notice any bugs we'd be happy to hear from you.

Warning and Information.

Other API's exist in MOOSERVER.MOOCODE use of APIs specifically not listed in this API Help Chapter is not supported or recommended as they are liable to change.

We are constantly working to ensure are APIs are stable and the Input / Output can be guaranteed.

APIs will be moved regularly into this stable and supported list.

If you believe we should prioritise an API to stable/supported to resolve an application development issue please log a support request and we will do our best to fast-track the specific API.

MOO.ATTACH.AW

MOO.ATTACH.AW

.Attach a named aw in a given state and position in the database order

Syntax

```
              moo.attach.aw('[aw name]','[RW|RO]','[pos]')
```

Return Value

```
              none
```

Example

```
              call moo.attach('my_aw','ro','last')
```

MOO.CHANGE.DIM

MOO.CHANGE.DIM

Changes the meta-data objects for a given dimension and re-registers.

Syntax

```
              call moo.change.dim([DimensioNameToChange],
[RowDescriptionVarianle]
                                    ,[ColDescriptionVariable],
[GeneralDescriptionVariable]
                                    ,[NameOfVariableStoringParentChileRelationship]
```

```
                              ,[DescriptionOfDimension],  Bool:CreateObjects
True/False )
```

Return Value

```
            SUCCESS or error values
```

Example

```
            call moo.change.dim('EXISTING_DIM',
                                'NEW_DIM_ROW','NEW_DIM_COL',
                                'NEW_DIM_DESC',
                                'NEW_DIM_HIER',
                                'New Dimension Description',
                                'FALSE')
```

MOO.CHANGE.PASSWORD

MOO.CHANGE.PASSWORD

Changes the password for the current user

Syntax

```
            moo.change.aw('[old password hash]','[new password hash]')
```

Return Value

```
            "OK" or {ERROR Code}
```

Example

```
            call
MOO.CHANGE.PASSWORD('KYLsbLM1INV17zrItythxnB34yU=','bF5VLKIeX8m4sgzmCe3p3seTlr
U=')
```

MOO.CREATE.CUBE

MOO.CREATE.CUBE

Create a new cube and its associated metadata and re-register it.

Syntax

```
         call moo.create.cube([CubeObjectNameToCreate],[Dimensions],
[CompositeDimensions],
                                 [CompositeObjectName],[DescriptionOfCube],
                                 [DataTypeOfCube],
                                 [PostUpdateProcess],[PreUpdateProcess],
                                 [IsInternalObject])
```

Definitions

Argument	Description
CubeObjectNameToCreate	Object name of new cube

Dimensions	List of dimension names separated by a space
CompositeDimensions	List of dimensions of the composite separated by a space
CompositeObjectName	Object name of the composite
DescriptionOfCube	Description of the cube
DataTypeoOfCube	Data type of the cube
PostUpdateProcess	Name of the process to run after the cube has been updated
PreUpdateProcess	Name of the process to run prior to the cube being updated
IsInternalObject	Is this an internal object?

Return Value

```
              SUCCESS or error values
```

Example

```
              call moo.create.dim('NEW_CUBE','DIM_1 DIM_2 DIM_3 DIM_T',
                                   'DIM_1 DIM_2 DIM_3',
                                   'CMP_NEW_CUBE',
                                   'New Decimal Data Cube',
                                   'DECIMAL',
                                   'N',
                                   'N',
                                   'FALSE')
```

MOO.CREATE.DIM

MOO.CREATE.DIM

Create and new dimension and its associated metadata.

Syntax

```
              call moo.create.dim([DimensioNameToCreate],[DimensionType],
[RowDescriptionVariable]
                                  ,[ColDescriptionVariable],
[GeneralDescriptionVariable]
                                  ,[ParentChileRelationshipVariable]
                                  ,[DescriptionOfDimension],  Bool:CreateObjects
True/False )
```

Definitions

Argument	**Description**
DimesniontNameToCreate	Name of dimension to create
DataTypeoOfCube	Data type of new dimension
RowDesriptionVariable	Object holding the row descriptions of the dimension values
ColDesriptionVariable	Object holding the column descriptions of the dimension values
GeneralDesriptionVariable	Object holding the general descriptions of the dimension values
ParentChileRelationshipVariable	Object holding the parent/child relations for the dimension values

DescriptionOfDmension	Description of the dimension
IsInternalObject	Is this an internal dimension?

Return Value

```
            SUCCESS or error values
```

Example

```
            call moo.create.dim('NEW_DIM','TEXT',
                                'NEW_DIM_ROW','NEW_DIM_COL',
                                'NEW_DIM_DESC',
                                'NEW_DIM_HIER',
                                'New Dimension in AW',
                                'FALSE')
```

MOO.DATA.ENTRY

MOO.DATA.ENTRY

Processes changed data in a valid MOODATA registered cube or variable.

Syntax

```
            call moo.data.entry([CUBE.ROW Value])
```

Return Value

```
            SUCCESS or error values
```

Example

You have changed data in a variable called OPEX_ACT read-only in MOODATA. You have limited the dimensions of OPEX_ACT to just the values that have changed. To ask the Process Manager to Save this data you should execute:

```
      shw MOO.DATA.ENTRY('OPEX_ACT')
```

The Process Manager will execute a SUBMITDATA process, it will log the data to the SUBDATA audit analytic workspace, check the data is valid and the user has the correct rights to update the OPEX_ACT variable. Finally it will write the data to MOODATA and commit the change.

MOO.DELETE.CUBE

MOO.DELETE.CUBE

Delete a cube in the MOODATA AW and associated metadata and de-register it.

Syntax

```
          call moo.delete.cube([CubeObjectName])
```

Definitions

CubeObjectName	Object name of cube to be deleted

Return Value

```
SUCCESS or error values
```

Example

```
call moo.create.dim('NEW_CUBE')
```

MOO.DETACH.AW

MOO.DETACH.AW

Detach a named AW.

Update if requested.

Syntax

```
moo.detach.aw('[aw name]',{yes|no})
```

Definitions

Argument	Description
AW Name	Name of the Analytic Workspace to detach
Update First	Boolean YES / NO {Default NO} Update the AW and commit any changes before detaching

Return Value

```
none
```

Examples

```
call moo.detach('my_aw')

call moo.detach('my_aw',y)
```

MOO.DELETE.DIM

MOO.DELETE.DIM

Deletes a dimension in the MOODATA Analytic Workspace and associated metadata and de-registers it from DIM.CFG.

Syntax

```
call moo.delete.cube('[DimObjectName]',[Cascade{TRUE}])
```

Definitions

Argument	Description

DimObjectName	Object name of dimension to be deleted
Cascade	Boolean YES / NO {Default NO} If YES all objects dimensioned by the DimObjectName will be deleted

Return Value

```
SUCCESS or error values
```

Examples

```
call moo.delete.dim('DIM_1')

call moo.delete.dim('DIM_1',TRUE)
```

MOO.EXTERNAL.CALL

MOO.EXTERNAL.CALL

Process to run an external task to the Oracle DBMS_Scheduler. This may be one of a number of predefined tasks or a shell script held on the server.

Syntax

```
call moo.external.call([Job{LSDIR|CHMOD|MAIL|MAIL_ATTACH}],
                [Argument1],[Argument2],[Argument3],[Argument4],
               [Enable],[AddBit],
                       [Execute])
```

where	LSDIR	Lists the files in a provided directory
	CHMOD	Changes the permissions on a given file
	MAIL	Send an email from OLAP provided that mailx has been installed
	MAIL_ATTACH	Send an email from OLAP provided that mails has been installed

Definitions

Job	Name of the predefined job
Argument[1-4]	Individual arguments to be passed to the job
Enable	Object holding the row descriptions of the dimension values
AddBit	Object holding the column descriptions of the dimension values
Execute	Name of a shell script to be run

Return Value

```
none
```

Examples

List all files in database directory alias MOOCDA and send to a file [MOOCDA]/tmp.txt

```
            call moo.external.call('LSDIR', 'MOOCDA' ,  '  ' ,
'listfile.txt',  NA, TRUE, FALSE, NA)
```

Changes the file permissions of a given file. Example, Change /tmp/tmp.sh to 777

```
            call moo.external.call('CHMOD', '777', '/tmp/tmp.sh', NA,NA, TRUE,
FALSE, NA)
```

To send an email directly from Oracle OLAP. Example, Send an email to me@myobjectiveolap.com

```
            call moo.external.call('MAIL', 'This is the content', 'Subject:
Message from OLAP', 'me@myobjectiveolap.com', NA, TRUE, FALSE, NA)
```

To send an email with attachment directly from Oracle OLAP, Example, Send an email to me@myobjectiveolap.com and attach a copy of the script which sends it

```
            call moo.external.call('MAIL_ATTACH', '/home/oracle/moochmod.sh',
'moochmod.sh',
                                  'Subject Sending a file from olap',
'me@myobjectiveolap.com', TRUE, FALSE, NA)
```

Running Your own Jobs

You can pass the name of your own shell script to moo.external.call
Essentially you could build a script dynamically through an OLAP DML OTF statement and then call this program to run it.
Often you should change the permissions first. This is done by setting the REQUIRE_ADDBIT argument to TRUE as below.
Permissions are set to 775. This is only supported on Unix-like and otherwise POSIX-compliant systems.
Do not try and set the REQUIRE_ADDBIT on Windows

```
            call moo.external.call('MyJob', 'myArg1stToMyJob',
'myArg2ndToMyJob', 'myArg3rdToMyJob', TRUE, TRUE, '/tmp/myShellScript.sh')
```

Example

```
            call moo.external.call('myJob', '/u01', '/tmp/myjob.out', NA, NA,
TRUE, FALSE, '/home/oracle/moolistfiles.sh' )
```

MOO.FIND.CMP

MOO.FIND.CMP

Returns the composite of a given variable

Syntax

```
            call moo.find.cmp('[object name]')
```

Return Value

Composite value

Example

```
            shw moo.find.cmp('CUBE_1')

            CMP_CUBE_1
```

MOO.SUBMIT.DATA

MOO.SUBMIT.DATA

Run a data submission. If a data submission has been placed in PRCONTROL by calling MOO.DATA.ENTRY, the Submit Data Process can be run through the Process Manager, or manually by calling MOO.SUBMIT.DATA passing the PROCESS id assigned to the SUBMITDATA Process.

Syntax

```
            call moo.submit.data([process number])
```

Return Value

```
            SUCCESS or error value
```

MOOMAN

mooMan

Provides online syntax information of moo API components. The syntax information is held within the program code'

Syntax

```
         mooman [program name]
```

Return Value

Appropriate syntax or an error message

Example

```
            mooman 'moo.create.dim'

            API:        MOO.CREATE.DIM
            Purpose:    Create Dimensions and associated meta data for storing
description and hierarchical data.
                        Register the created objects in the the metaData
magazine DIM.CFG


                        Arguments:

                        call moo.create.dim([DimensioNameToCreate],
[DimensionType], [RowDescriptionVariable]
                                              ,[ColDescriptionVariable],
[GeneralDescriptionVariable]
                                              ,
[NameOfVariableStoringParentChildRelationship]
                                              , [DescriptionOfDimension],
Bool:CreateObjects True/False )
```

MOO.LIST.DIMS

MOO.LIST.DIMS

Lists the dimensions of a given variable or composite

Syntax

```
call moo.dist.dims('[object name]')
```

Return Value

A list of dimensions or na

Example

```
shw moo.list.dims('cube_1')

DIM_1
DIM_2
DIM_3
DIM_T
```

MOO.NEW.PROCESS

MOO.NEW.PROCESS

Submit a new process to Process Control

Syntax

```
call moo.new.process([ProcesstType],[Limit],[ArgumentString])
```

Return Value

```
Process Id created
```

MOO.USER.CLOSE

MOO.CLOSE.USER

Called by user GUI at disconnect to close AWs down

Syntax

```
call moo.close.user
```

Return Value

```
OK or error message
```

MOO.USER.INIT

MOO.USER.INIT

Start-up a mooServer Session

You can manually start a mooServer session by calling the following Express program:

```
        MOOSERVER.MOOCODE!MOOUSER.INIT.
```

By calling this program, mooServer will validate the user details, and attach all necessary Oracle OLAP Analytic Workspaces.

Before calling this program you must have pre-populated the following variables:

```
       EXPRESS!ME_USER
       EXPRESS!HASH_STR
```

If this is a brand new session it may be necessary for your client to create these objects, below is the definition:

```
DEFINE ME_USER  EXPRESS VARIABLE TEXT TEMPORARY
DEFINE HASH_STR EXPRESS VARIABLE TEXT TEMPORARY
```

ME_USER

You should populate the ME_USER variable with a valid MOOSERVER.MOOUSERS!USER

HASH_STR

HASH_STR must be populated with a valid password hash.

MOO.CHANGE.CUBE

MOO.CHANGE.CUBE

Changes the metadata of an exiting cube and its associated metadata and re-register it in CUBE.CFG.

Syntax

```
          call moo.change.cube([CubeObjectName],
                               [DescriptionOfCube],
                               [PostUpdateProcess],
                               [PreUpdateProcess],
                               [IsInternalObject])
```

Definitions

Argument	**Definition**
CubeObjectName	Object name of an existing cube
DescriptionOfCube	New description of the cube
PostUpdateProcess	New name of the process to run after the cube has been updated
PreUpdateProcess	New name of the process to run prior to the cube being updated

IsInternalObject	New status of this being an internal object

Return Value

```
SUCCESS or error values
```

Example

```
call moo.create.dim('CUBE_1',
                    'New Description for Decimal Data Cube',
                    'N',
                    'N',
                    'FALSE')
```

MOO.COMP.SANE

MOO.COMP.SANE

Checks the dimensionality of an existing composite against a given set of dimensions.

Syntax

```
call moo.comp.sane('[list of dimesnions]','[object name]')
```

Return Value

```
Yes or no
```

Example

where,

```
dsc cube_1

DEFINE CUBE_1 VARIABLE DECIMAL <CMP_CUBE_1 <DIM_1 DIM_2 DIM_3>
DIM_T>

shw moo.comp.sane('DIM_1 DIM_2 DIM_3' 'CMP_CUBE_1')

yes

shw moo.comp.sane('DIM_1 DIM_2 DIM_X' 'CMP_CUBE_1')

no
```

MOO.LIST.DIM.DESC

MOO.LIST.DIM.DESC

Lists the dimension descriptions of a given variable or composite

Syntax

```
call moo.dist.dim.desc('[object name]')
```

Return Value

A list of dimension descriptions or NA

Example

```
shw moo.list.dim.desc('cube_1')

Sample Dimension 1
Sample Dimension 2
Sample Dimension 3
Sample Dimension Time
```

MOO.CREATE.USER

MOO.CREATE.USER

Create a user in MOOUSERS

Syntax

```
            shw moo.create.user([USERNAME],[FIRSTNAME],[SURNAME],[DEPARTMENT],
_                                    [EMAIL],[NOTES {multiline \n}],[PASSWORD],
[DISABLED{YES|NO},[IS_ADMIN{YES|NO})
```

Definitions

Username	New user ID
FirstName	New user's first name
Surname	New user's surname
Department	New user's department
EMail	New user's email address
Notes	Notes on specific user (use \n for multiline)
Password	New user's password
Disabled	Whether the user is disabled {YES|NO}
Is_Admin	Whether the users is a systems admin user {YES|NO}

Return Value

```
SUCCESS of error message
```

MOO.DELETE.USER

MOO.DELETE.USER

Deletes a user from MOOUSERS

Syntax

```
shw moo.create.user([USERNAME])
```

Definitions

Argument	Definition
Username	MOOSERVER.MOOUSERS$USER Dimension Value

Return Value

```
SUCCESS of error message
```

Examples

```
shw moo.delete.user('JAKE')
```

MOO.MNT.HI

MOO.MNT.HI

Populates the appropriate parent, sequence and depth objects for a given hierarchy based on the current parent/child relationships.
The objects populated will be those held in the MOOSERVER.MOODATA$HI.CFG magazine for the given hierarchy.

MOOSERVER.MOODATA$HI.CFG(MOOSERVER.MOODATA$HI.CFG$HI.COL 'PARENT ')
MOOSERVER.MOODATA$HI.CFG(MOOSERVER.MOODATA$HI.CFG$HI.COL 'SEQUENCE')
MOOSERVER.MOODATA$HI.CFG(MOOSERVER.MOODATA$HI.CFG$HI.COL 'DEPTH ')

Syntax

```
call moo.mnt.hi('[object name]')
```

Where object name is a value from the .HIER.LIST of the base dimension.

```
dsc dim_1_hier

DEFINE DIM_1_HIER RELATION DIM_1 <DIM_1 DIM_1.HIER.LIST>
```

Return Value

```
none
```

Example

```
call moo.mnt.hi('DIM_1_MAIN')
```

MOO.PR.ACTIVATE

MOO.PR.ACTIVATE

Activates a process that has been placed in the Process Manager queue.

Syntax

```
call moo.pr.activate([Process])
```

Definitions

Argument	Description
Process	An existing Process ID

Return Value

```
call moo.pr.activate('1357056773')
```

MOO.PR.DURATION

MOO.PR.DURATION

Calculates the duration of a process given the start and finish times.

Syntax

```
call moo.pr.duration([Start],[Finish])
```

Definitions

Start	Seconds value of the process start time
Finish	Seconds value of the process finish time

Return Value

A time represented in HH:MM:SS

Examples

```
shw moo.pr.duration('1352026146','1352026152')

00:00:06
```

MOO.PR.MGR

MOO.PR.MGR

Manages process execution in the Process Manager. Normally called by the Process Daemon (PRD). Can be called manually to process one Process execution.

Syntax

```
call moo.pr.mgr
```

Return Value

None or error message

MOO.PR.SEQUENCE

MOO.PR.SEQUENCE

Populate the sequence property in PR.CTL

Syntax

```
call moo.pr.sequence
```

Return Value

None or error message

MOO.PROCESS.LOG

MOO.PROCESS.LOG

Update the process log of a given process with a message.   Typically used in OLAP DML Programs designed to execute in the Process Manager.

Syntax

```
call moo.process.log([Process],[Message])
```

Definitions

Process	An existing Process ID
Message	Text message to go in process log

Return Value

None or error message

Examples

```
_msg = 'Duplicate values in dimension list. Job Canceled'
call moo.process.log(current.process, _msg)
```

MOO.REMOVE.WFPROCESS

MOO.REMOVE.WFPROCESS

Remove a process from all workflows

Syntax

```
call moo.remove.wfprocess([Process])
```

Definitions

Argument	Description
Process	The PR.ROW value of a valid Process

Return Value

SUCCESS or error message

Examples

```
call moo.remove.wfprocess('AGGREGATE_MYCUBE')
```

MOO.SPLIT

MOO.SPLIT

Convert a chr30 delimited input into a multi-line output.

Syntax

```
call moo.split('[Space delimited input]')
```

Return Value

Multi-line output

Example

```
shw moo.split('Text1 Text2 Text3 Text4')

Text1
Text2
Text3
Text4
```

MOO.UNTAR.FILE

MOO.UNTAR.FILE

Wrapper to MOO.EXTERNAL.CALL and userland TAR utility to un-TAR a file into a specific directory

UNTAR a file on the host file system. This is a wrapper to MOO.EXTERNAL.CALL API.
Location of the un-tar shell script used by this API is stored in the SYS.CFG Magazine.
Execution of the un-tar is managed by the Oracle DBMS Scheduler and requires the correct DBMS Scheduler Credentials to have been created

Syntax

```
      shw MOO.UNTAR.FILE( {FQ Directory to change to before Un-tar} , {Full Path and
Name of tar file}  )
```

Return Value

SUCCESS, ERROR, MOOMAN Entry

Example

```
      shw moo.untar.file('/u01/inbound_files' , '/u01/inbound_files/
ebiz_feed.tar')
```

MOO.AGGREGATE.CUBE

MOO.AGGREGATE.CUBE

Process control program used to run moo.aggregate. moo.aggregate.cube assumes that cube.row and dim.row have been limited to a cube and the appropriate dimensions respectively.

Syntax

```
            call moo.aggregate.cube
```

Return Value

```
            SUCCESS or error message
```

Examples

```
            call moo.aggregate.cube
```

MOO.AGGREGATE

MOO.AGGREGATE

Runs an aggregation on the cube passed to the program over the dimensions passed.

The aggregation uses default aggmap created based on the cube dimensions.

Syntax

```
            call moo.aggregate([Cube],[AggDims],[Status]{yes|no})
```

Definitions

Argument	**Description**
Cube	Object name of the cube to be aggregated
AggDims	A multi-line list of dimensions over which the cube will be aggregated
Status	Retain status of dimension set during the aggregation on exit.

Return Value

```
            None or error message
```

Examples

```
            call moo.aggregate('OPEX_CUBE' , 'ACCOUNT/nCOST_CENTRE/TIME' ,
YES)
```

MOO.AWM.COMPAT

MOO.AWM.COMPAT

Creates the AWM compatibility layer.

Syntax

```
call moo.awm.compat('[Verbose]{true|false}',
                    [ShwSQL]{true|false}',
                    [JustSQL]{true|false}',
                    [DropTable]{true|false}',)
```

Definitions

Argument	Description
Verbose	Show Output
ShwSql	Send the definition of the SQL to be executed to the current outfile
JustSQL	Just generate TEMP.COMPAT.SQL but do not actually execute it
dropTable	Drop and re-create Tables instead of just truncating and re-snapping them from the Relational Views

Return Value

Various {STRING}

MOO.AWM.COMPAT.WRAP

MOO.AWM.COMPAT.WRAP

Runs the moo.awm.compat process with all arguments set to false.

Syntax

```
call moo.awm.compat.wrap
```

Return Value

SUCCESS or error message