IBM Cognos Business Intelligence Series 7
IBM Cognos Series 7 PowerPlay Transformer

Version 7.5

Transformer MDL Reference

Information Management software

Product Information

This document applies to IBM Cognos Series 7 PowerPlay Transformer Version 7.5 and may also apply to subsequent releases. To check for newer versions of this document, visit the IBM Cognos Information Centers (http://publib.boulder.ibm.com/infocenter/cogic/v1r0m0/index.jsp).

Copyright

Table of Contents

Introduction

You can use this book as a reference to the model definition language (MDL) for PowerPlay Transformer Series 7 Version 4 and subsequent versions.

MDL keywords are compatible with later versions and this book clearly indicates when functionality is introduced or changed. For details of new features in each version, see the index entry for "version x.xx new features."

There are two types of MDL that define a Transformer model: verb MDL and structured MDL. This book is a complete reference to verb MDL. With the exception of Chapter 3, this book always assumes that MDL is verb MDL.

Note: Some Asian languages are not supported by IBM Cognos Impromptu, IBM Cognos Impromptu Web Reports, and IBM Cognos Visualizer.

Finding Information

To find the most current product documentation, including all translated documentation, access one of the IBM Cognos Information Centers at http://publib.boulder.ibm.com/infocenter/cogic/v1r0m0/index.jsp.

You can also read PDF versions of the product release notes and installation guides directly from IBM Cognos product disks.

Samples Disclaimer

The Great Outdoors Company, GO Sales, any variation of the Great Outdoors name, and Planning Sample, depict fictitious business operations with sample data used to develop sample applications for IBM and IBM customers. These fictitious records include sample data for sales transactions, product distribution, finance, and human resources. Any resemblance to actual names, addresses, contact numbers, or transaction values, is coincidental. Unauthorized duplication is prohibited.

Chapter 1: What Is MDL?

You can use MDL to define or change a Transformer model. Depending on your Transformer edition, you may be able to modify your .ini file to work more effectively in MDL.

MDL is the model definition language for Transformer. You can use MDL to create, manipulate, update, and store Transformer models outside the Transformer user interface, including on a server. You can set preferences globally and also automate Transformer functions for greater efficiency.

MDL represents Transformer models in text format and you can view the text in any text editor. The .py? file is a representation of the model, stored in default binary format. The question mark (?) in the extension .py? is replaced by the character used in your version of Transformer. Note that it is equivalent to the .mdl-format file, with the exception of any adjustments made to the user interface and, in some instances, passwords.

A model file stored MDL files contain statements, which can contain verbs, objects, and options. For example, in the statement

```
DataSourceUpdate 257 "All Staff" SourceType ExcelDatabase
```

- DataSourceUpdate is a verb specifying that a data source should be changed.
- 257 is the object identifier and All Staff is the object name that identify a specific data source object.
- SourceType ExcelDatabase is an option.

You can use MDL in two ways:

- to create an MDL model that completely defines a Transformer model
- to create an MDL script that manipulates or updates an existing Transformer model

These uses can be combined. For example, a script can be appended to the end of a model definition.

For examples of MDL scripts, see "Sample Scripts" (p. 47).

.py? and .mdl Formats

You can save Transformer models in two formats: .py? or .mdl. You can open models saved as .py? files in the Series 7 Windows interface. You can open models saved as .mdl files in the user interface or in a text editor.

The .mdl format is a plain text representation of the model that is compatible between versions of Transformer. It loads more slowly than .py? format because it recreates all of the Transformer objects when it loads. Models saved in .mdl format are machine-independent.

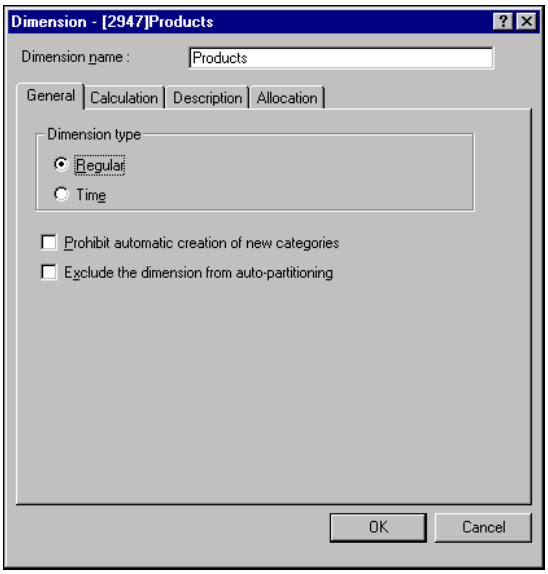
The .py? format is a binary representation of the model. It isn't compatible between versions of Transformer. It loads faster than .mdl format because the Transformer objects are not recreated when the model reloads.

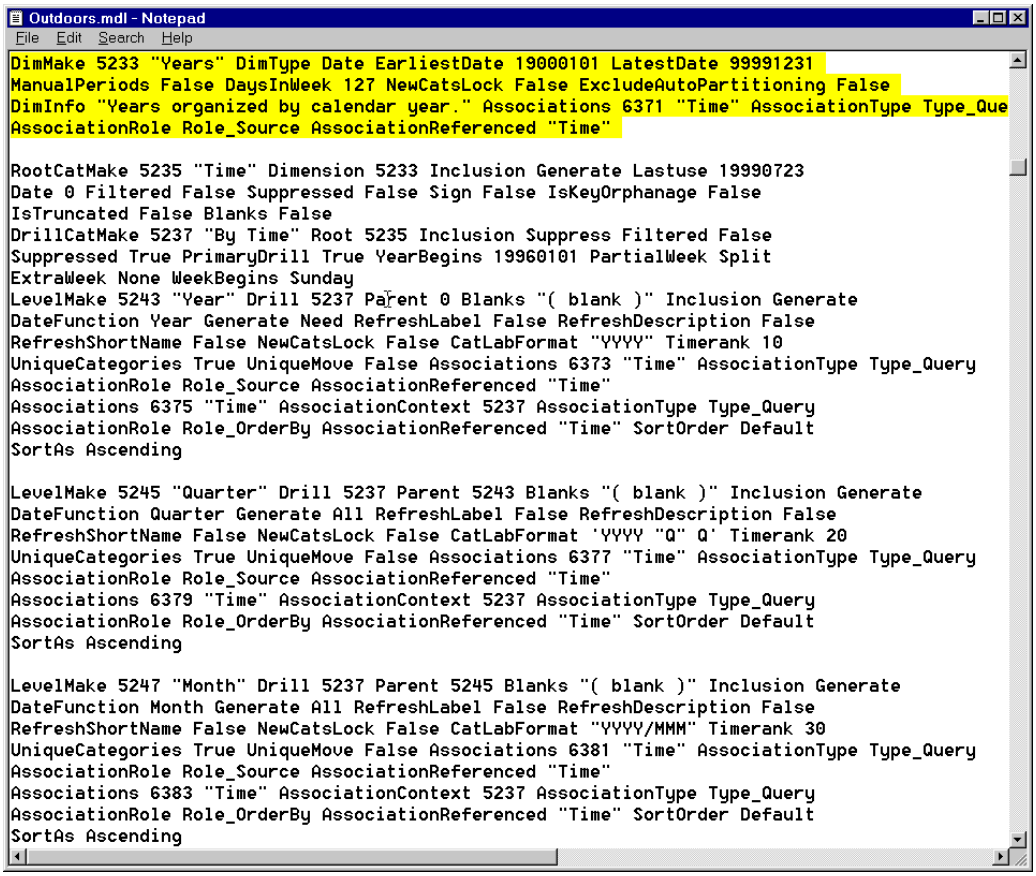
In general, .py? model files are larger than .mdl model files. Additionally, as you edit a model, the size of the associated .py? file increases. This increase occurs because Transformer maintains information on the operations performed during model editing. Transformer uses this information in a variety of contexts, such as client-server operations and incremental updates.

Tip: Storing model operations can cause model fragmentation within the binary model file. To eliminate this fragmentation, save .py? models periodically in .mdl format.

What MDL Looks Like

This is an example of a Dimension property sheet in the Transformer user interface (first diagram) and its corresponding MDL highlighted in a text editor (second diagram). It is helpful to use a text editor that provides easy access to line numbers, because Transformer error messages tell you the line on which an error occurred. You can use any text editor or word processor to edit an MDL file, so long as you can save the file in text-only format.





Both applications show a dimension from the same file (Outdoors.mdl). In the user interface the property sheet for dimension 5233 appears. In the text editor the find utility locates the MDL statement that defines that dimension.

In the property sheet, Dimension type Regular is selected and Prohibit automatic creation of new categories is not selected. This information is represented in the MDL statement by the options DimType Regular and NewCatsLock False.

For ease of use, Transformer inserts line breaks in lines over 70 characters to make MDL easier to read. This does not affect functionality.

How to Use MDL

In general, Transformer's Windows user interface is the easiest way to create and maintain Transformer models. MDL is more difficult to use, and in most cases does not provide greater functionality than the user interface.

However, there are situations where an experienced Transformer user will benefit from using MDL. Following are some of the ways you can use MDL, beginning with ways that require no knowledge of the language and moving to more advanced applications.

- You can save your Transformer model in MDL format by selecting Save As in the File menu, changing the extension of the file to .mdl, and then saving the file. You can now use the .mdl file to
  - open the model in future versions of Transformer.

    - move models between client and server applications of Transformer.
    - archive Transformer models efficiently. Models stored in MDL format are compact, and because they can be opened in future versions of Transformer, you don't have to worry about updating them.
    - improve the performance of Transformer models. Saving a .py? file to .mdl cleans up the file by integrating all changes into a clean model. It also prevents possible corruption of the model.
- You can open an MDL model in any text editor to perform global edits of your model, such as changing all directory paths from C:\ to D:\ or changing all instances of "Weeks" to "Wks."
- You can write scripts in MDL to manipulate Transformer models, whether the models are in .mdl or .py? format. This allows you to
    - store repetitive tasks in an MDL batch file that can be run at any time
    - perform mass edits quickly, using MDL commands
- You can use a text editor to work directly with Transformer models in .mdl format. This allows you to
    - change a model on the server
    - store parts of models in separate files to use as needed
    - create models as needed
    - create custom applications of Transformer

MDL is an interpreted language. There are several ways to process MDL files:

- open the .mdl file in the Transformer Windows user interface
- run the .mdl file with Trnsfrmr.exe
- access the .mdl file from the command line with *rsserver* and the command line option `-m` (Transformer on UNIX only)

Change Models with MDL

There are three basic ways that you can use MDL to modify an object in a Transformer model:

- update the definition with a script
- change the MDL definition directly
- update the definition within the model

Example

In the model Great Outdoors.mdl, you want to change the source file for the data source Products (CSV) from Prodinfo.csv to P0001.csv. This is the original definition of the data source:

```
DataSourceMake 103 "Products (CSV)" Separator "," SourceType FlatFile_ColNames

CharacterSet Multibyte DecimalSep "." Thousandsep ","
Columns True Timing PopYesCreateNo
Source ".\samples\powerplay\cubes and reports\prodinfo.csv"
Speed False
SetCurrent False ServerSource False Presummarized False
EnableMultiProcess False
```

You can update the definition with a script that changes the source file of the data source to P0001.csv. It opens the model, issues a DataSourceUpdate statement, and saves the model.

```
OpenMDL "Great Outdoors.mdl"
DataSourceUpdate "Products (CSV)" Source "c:\Production\P0001.csv"
SaveMDL "Great Outdoors (Admin).mdl"
```

For information about command line options you can use when running scripts, see the Transformer online help. For examples of MDL scripts, see "Sample Scripts" (p. 47).

Alternatively, you can change an object definition in the model. In this case, delete the word Prodinfo in the DataSourceMake statement and replace it with P0001. This is the new DataSourceMake statement:

```
DataSourceMake 103 "Products (CSV)" Separator "," SourceType FlatFile_ColNames

CharacterSet Multibyte DecimalSep "." Thousandsep ","
Columns True Timing PopYesCreateNo
Source "c:\Production\P0001.csv" Speed False
SetCurrent False ServerSource False Presummarized False
EnableMultiProcess False
```

You can also open the MDL model in a text editor and add the DataSourceUpdate statement to the end of the file. The next time you open the model in Transformer, the change will be implemented. The original DataSourceMake statement is not changed, and the change is in effect while the DataSourceUpdate statement remains attached to the model.

At the end of the model definition, type

```
DataSourceUpdate "Products (CSV)" Source "c:\Production\P0001.csv"
```

Note: If you update an object within the model, Transformer may be unable to complete the forward referencing that is required in the definition of some objects. Therefore, you should use the verb UpdateForwardReference before the statement that updates the object. For example, at the end of the model definition, type

```
UpdateForwardReference
DataSourceUpdate "Products (CSV)" Source "c:\Production\P0001.csv"
```

Set Transformer to Verb MDL

There are two types of MDL: Verb MDL and Structured MDL. Both types can completely define a Transformer model.

This book is a complete reference to Verb MDL. With the exception of Chapter 3, this book always assumes that MDL is Verb MDL.

By default, MDL that is generated by Transformer is in structured format. This ensures compatibility between versions of Transformer, because it is the original version. However, it is recommended that you use verb MDL. To have Transformer generate models in verb format, set the following Trnsfrmr.ini parameter:

```
VerbOutput=1
```

This setting only affects the format of model definitions generated by Transformer. Whichever setting you use, you can still use both verb and structured format.

For information about the history of MDL and a brief description of Structured MDL, see "Structured MDL" (p. 241).

Chapter 2: Transformer Objects

This chapter includes information about objects in Transformer models, how to set object identifiers and object names, and how to locate objects uniquely.

Objects are the building blocks of Transformer models. An object is a definition of a particular component of the model.

For example, the Locations data source object contains all the information in Transformer about the Locations data source. In the user interface, the Locations object is defined in the Data Source property sheet for Locations. In an MDL model definition, the same information is found in the MDL statement that defines the data source object Locations. The settings in the property sheet are represented in the MDL statement as options.

There are many types of objects in Transformer:

Model	User Class
Data Source	PowerCube
Column	PowerCubeGroup
Dimension	PowerCubeGroupCube
Level	Subdimension
Root Category	Signon
Drill Category	View
Regular Category	Currency Table
Special Category	Currency
Measure	Dimension Calculation Definition

Each Transformer model must have at least one model object. The model may also contain one or more of each of the other object types.

All Transformer objects have property sheets in the Transformer user interface. Only objects have property sheets.

Object Identifiers

By default, Transformer assigns a numeric object identifier to each object, and assigns an object name when possible. Alternatively, you can assign object identifiers and object names when you

create the object. For example, in the MDL statement DataSourceMake 103 "Natsmall," the data source object is identified by the object identifier 103 and the object name Natsmall. Every object has an object identifier and an object name, although it is possible to prevent the object identifier from appearing in MDL files. You can track objects in MDL and the Transformer user interface by using object identifiers and object names.

To view object identifiers and object names in Transformer, do the following:

- Select the Object Identifier and Object Name check boxes in the Titles tab of the Preferences dialog box to view them in the user interface.

Note: In early versions of Transformer, the object identifier was known as the Unique ID.

Object identifiers must be unique within a model. If they are not unique, Transformer issues an error message and the model doesn't load. When generated by Transformer, client object identifiers are odd numbers and server object identifiers are even numbers to guarantee that server and client models are synchronized. Object identifiers must be greater than 100 and less than 4,294,967,295. After an object identifier is assigned to an object, it cannot be changed, either by the user or by Transformer.

You can assign object identifiers and object names when you create the object, but these can be difficult to maintain. It is generally recommended that you leave the specification of object identifiers to Transformer. If you do not specify the object identifier, or if you choose incremental update, then Transformer assigns one.

If you copy and paste parts of models from various MDL files, use the .ini setting that prevents object identifiers from appearing in MDL files. Otherwise, you cannot guarantee that the object identifiers will be unique.

To have Transformer generate models with no object identifiers, change the Trnsfrmr.ini setting as follows:

```
ObjectIdOutput=0
```

Object Names

Transformer derives object names from the source data when possible. If you create a Transformer object manually using MDL, you must supply a name for it. Transformer object names

- must appear in single or double quotes
- must not contain a carriage return
- must not be over 256 characters in length

You can change object names, but not object identifiers. For example, one way to change the object name is to type the following:

```
DataSourceUpdate 103 "newname"
```

Because object names do not always have to be unique within a model, if you identify objects only by name you must often supply more information to uniquely identify the object.

Object names are identical in the user interface and MDL except in the case of categories and, in some circumstances, columns.

Uniqueness of Object Names

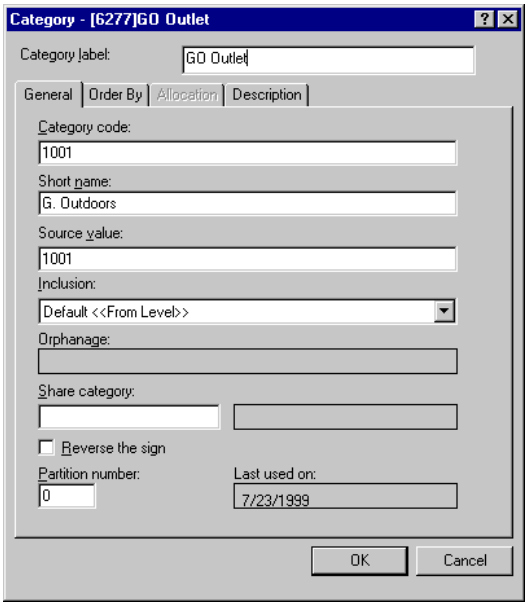
In certain contexts, object names for the same object type must be unique. These are the restrictions on the uniqueness of object names:

Object names	Must be unique
Category names	Within a dimension. This applies to regular categories, root categories, drill categories, and special categories.
Column names	Within a data source.
Currency names	Within a model.
Currency table names	Within a model.
Data Source names	Within a model.
DimCalcDef names	Within a dimension.
Dimension names	Within a model.
Level names	Within dimensions, if advanced subsets are used. Otherwise, never.
Measure names	Must be unique within a model.
PowerCube names	Within a model and cannot be the same as PowerCubeGroup names.
PowerCubeGroup names	Within a model and cannot be the same as PowerCube names.
PowerCubeGroupCube names	Within a group.
Signon names	Within a model.
Subdimension names	Within a dimension.
User Class names	As determined by Access Manager.
View names	Within a dimension; see "View Names" (p. 20).
FilterName	Architect data source.

Category Names

Categories have a category label and a category code instead of a single name. In the user interface, category object names are derived from category labels. In MDL, category object names are derived from category codes. Category names cannot contain a tilde (~) or an at sign (@).

For example, this is the property sheet for a category with object identifier 6277. The object name, which is shown in the title bar, is GO Outlet. It is derived from the category label.



In the corresponding MDL statement, the object identifier, 6277, is the same, but the object name is 1001. That is because the name is derived from the category code. Options in `bold` in this MDL statement are represented on the General tab of the Category property sheet:

```
CatMake 6277 "1001" Dimension 5695 Drill 5699 Levels
5707

Parent 6275 Label "GO Outlet" ShortName "G. Outdoors"

Lastuse 19990723 SourceValue "1001" Filtered False Suppressed
False

Sign False IsKeyOrphanage False IsTruncated False Blanks
False
```

Column Names

Column names are identical in the user interface and MDL unless they are manually changed. For more information, see "Column" (p. 193). Column names cannot contain an at sign (@).

View Names

In structured MDL, views are represented by the keyword `ViewName`, using the syntax `ViewName id "Name"` where `id` is the unique Transformer identifier and `"View Name"` is the name of the view.

Locate Objects Uniquely

Whenever an object is specified, it must be uniquely identified. If it is not identified by its object identifier, and if its object name does not uniquely identify it, then it must be identified by specifying its relationship within a hierarchy of objects.

For example, a column exists within a data source. When you reference a column, if you do not provide a unique object name or object identifier for the column, you must uniquely identify it by specifying the data source in which it exists.

A more complex example of the same thing is a category. A category exists within a dimension, drill-down path, level, and parent category. When the category is created, you must specify all those objects to precisely locate it. But when you reference an existing category, even if you do not uniquely identify it by its object identifier, you only need to specify the dimension name to uniquely identify it. This is because category names are unique within dimensions and dimension names are unique within models.

The following rules govern precise identification and location of objects:

- If the object already exists and is identified by its object identifier, then no other information is needed to locate the object in the model.
- If the object already exists and is identified only by its object name, then Transformer searches the model for an object with the same name and object type and uses the first one it finds.
- If the object does not already exist, then there must be sufficient information to precisely locate it in the model.

When both an object identifier and an object name are used to identify an object in MDL, it is the object identifier that determines whether the object exists. If there is a conflict between the object name and object identifier, it is the object identifier that is used to locate the object, and the name is changed.

If the object does not already exist or is not uniquely identified by its object name or object identifier, then additional information must be supplied to locate it. The information required to locate the object is supplied in the syntax for each verb. It is also summarized in the following two tables.

Identify Existing Objects

The following objects must be referenced to uniquely identify an existing object that is not uniquely identified by its object identifier.

Object	Objects that must be referenced
Currency, Dimension, Measure, Model, Data Source, PowerCube, PowerCube Group, Signon, User Class	None
Category	Dimension
Column	Data source

Object	Objects that must be referenced
DimCalcDef	Dimension
Drill Category	Dimension, Root Category
Level	Dimension, Drill Category
PowerCube Group Cube	PowerCube Group, Dimension, Category
Root Category	Dimension
Subdimension	Dimension, Drill Category, Level, Parent Category
Special Category	Dimension, Parent Category
View	Dimension
FilterName	Architect data source

Place New Objects

The following objects must be referenced to precisely locate an object that does not already exist.

Object	Objects that must be referenced
Currency, Dimension, Measure, Model, Data Source, PowerCube, PowerCube Group, Signon, User Class	None
Category	Dimension, Drill Category, Level, Parent Category
Column	Data Source
DimCalcDef	Dimension
Drill Category	Dimension, Root Category
Level	Dimension, Drill Category, Parent Category
PowerCube Group Cube	PowerCube Group, Dimension, Category
Root Category	Dimension

Object	Objects that must be referenced
Subdimension	Dimension, Drill Category, Level, Parent Category
Special Category	Dimension, Parent Category
View	Dimension

Set Global Preferences

You can use the first few lines of your MDL script or a special MDL header file to specify global environment variables and processing preferences.

Category Sorting Preferences

In Transformer Version 7.3 and subsequent versions, you can set the sorting preferences for all categories in the model. You can specify one of three possible `ModelCategoryOrderDefault` model header options in the first few lines of your MDL script:

- `OrderYes`
- `OrderNo`
- `OrderUsePreference`

For more information about each option, click the What's This? help on the Transformer Windows interface.

Chapter 3: Create an MDL Model

This chapter shows you how to create a complete Transformer model in MDL. Even if you never attempt to create a model with MDL, by following this brief tutorial you will learn about the structure and interdependencies of MDL models.

Create a Complete Model

To create a Transformer model in MDL, you need to create a separate file for each part of the model and then combine them with a script.

In the scripts, the "Add" form of verbs is used; for example, DataSourceAdd. You can also use the "Make" form; for example, DataSourceMake, with no other changes to the scripts. The "Add" form is used here only to illustrate the difference between what you must type to define an object, and how Transformer defines objects. In model definitions generated by Transformer, the "Make" form of the verbs is always used.

Only the necessary options are included to create each object.

Each Transformer statement starts with a verb. Many statements are quite long. They can wrap over the line, or you can include carriage returns between keywords.

Step 1: Create the Data Sources in Transformer

This script creates data sources and columns for the model.

One of the queries, Main (IQD), is an Impromptu query definition (IQD) data source. The SQL string originates from Impromptu and is usually copied from Impromptu by Transformer when Transformer connects to the data source. Since this example creates the entire model definition manually, the SQL string was copied from Impromptu and pasted into this script.

Que_Col.mdl

```
DataSourceAdd "Products (CSV)" Separator ","
SourceType FlatFile_ColNames
Source "c:\prodinfo.csv" ColumnAdd "Product Line"
DataSource "Products (CSV)"
Origin Generated Offset 0 Column "Product Line"
ColumnAdd "Product Type" DataSource "Products (CSV)"
Origin
Generated
Offset 1 Column "Product Type"
ColumnAdd "Product Id" DataSource "Products (CSV)" Origin Generated
Offset 2 Column "Product Id"
ColumnAdd "Product Name" DataSource "Products (CSV)"
Origin
Generated
Offset 3 Column "Product Name"
```

```
DataSourceAdd "Prod Line Plan" Separator ","

SourceType FlatFile_ColNames Source "c:\ProdPlan.asc"

ColumnAdd "YEAR" DataSource "Prod Line Plan" Origin Generated
Offset
0

Column "Time" Format Y DateLevel Year Class Date

ColumnAdd "PROD_LINE" DataSource "Prod Line Plan" Origin
Generated

Offset 1 Column "Product Line"

ColumnAdd "FORECAST" DataSource "Prod Line Plan" Origin
Generated

Offset 2 Column "Planned sales"

DataSourceAdd "MAIN (IQD)" Separator ","

SourceType DataSource

Source "c:\bsc_msrs.iqd"

SQL 'select T1."ORDER_DT" as c1,

    T2."PROD_NO" as c2,

    T1."REP_NO" as '

'c3,

    T1."CUST_NO" as c4,

    T2."QTY" as c5,

    (T2."QTY" * T2."PRI'

'CE") as c6,

    (T2."QTY" * T3."PROD_COST") as c7,

    (CASE WHEN ((((T'

'2."QTY" * T2."PRICE") - (T2."QTY" * T3."PROD_COST"))
/
(T2."QTY" * T2."'

'PRICE")) <= 0.19) THEN (' "'Under 20%') WHEN ((((T2."
'"QTY" * T2."PRICE") - (T2."QTY" * T3."PROD_COST")) / (T2."QTY"
*
T2."PR'

'ICE")) BETWEEN 0.2 AND 0.65) THEN (' "'20% - 65%') WHEN
((((T2." '"QTY" * T2."PRICE") - (T2."QTY" * T3."PROD_COST")) / (T2."QTY"
* T2."PR'

'ICE")) >= 0.66) THEN (' "'Over 65%') ELSE ('ERROR')
END)
as c8

from "

'"ORDER" T1,

 ("PRODUCT" T3 left outer join "ORDRDETL" T2 on T2."PROD_NO'

'" = T3."PROD_NO")

where (T2."ORDER_NO" = T1."ORDER_NO")

order by c1 asc'

",c3 asc,c4 asc,c2 asc

"

ColumnAdd "Order Dt" DataSource "MAIN (IQD)" Origin Source
Offset
0 Column "Time"
```

```
Storage Int32 Scale 0 Size 4 Decimals 0 Class Date InputScale
0
TimeArray Off

ColumnAdd "od-Prod No" DataSource "MAIN (IQD)" Origin
Source
Offset 1 Column "Product Id"

Storage Float64 Scale 0 Size 5 Decimals 0 Class Quantity
InputScale
0 TimeArray Off

ColumnAdd "Revenue" DataSource "MAIN (IQD)" Origin Generated

Offset 5 Column "Revenue"

ColumnAdd "Cost" DataSource "MAIN (IQD)" Origin Source
Offset
6 Column "Cost"

Storage Float64 Scale 0 Size 6 Decimals 0 Class Quantity
InputScale
0 TimeArray Off
```

Step 2: Create the Time Dimension

This script creates the time dimension.

When you create a dimension in the Transformer user interface, Transformer automatically creates a root category and drill category for the dimension. If you create a dimension in MDL, you must create those yourself. RootCatMake and DrillCatMake are used in the example because there is no RootCatAdd or DrillCatAdd.

The inclusion property is set to Suppress for the drill category so that the drill category does not appear in the dimension viewer in your Series 7 PowerPlay clients.

When you create a dimension in the user interface, Transformer also creates the default dimension views that are required for each dimension, All Categories, and Omit Dimension. In MDL, you must create them yourself or use the verb ModelEnsureCompleteness to have Transformer create them. In this example, ModelEnsureCompleteness is included in the script in step 12, so the default views are not added.

The definition of levels contains the specification of dimension, drill, and parent. This defines the structure and hierarchy of the dimension.

Dim_Years.mdl

```
DimAdd "Years" DimType Date

RootCatMake "Time" Dimension "Years"

DrillCatMake "By Time" Dimension "Years" Root "Time"

Inclusion Suppress

LevelAdd "Year" Dimension "Years" Drill "By Time" Parent
""

Source "Time" DateFunction Year UniqueCategories True

OrderBy Drill "By Time" Column "Time" SortOrder Default

SortAs Ascending

LevelAdd "Quarter" Dimension "Years" Drill "By Time"
Parent "Year"

Source "Time" DateFunction Quarter CatLabFormat 'YYYY
"Q" Q'
```

```
UniqueCategories True OrderBy Drill "By Time" Column
"Time"

SortOrder Default SortAs Ascending

LevelAdd "Month" Dimension "Years" Drill "By Time" Parent
"Quarter"

Source "Time" DateFunction Month CatLabFormat "YYYY/MMM"

UniqueCategories True OrderBy Drill "By Time" Column
"Time"

SortOrder Default SortAs Ascending
```

Step 3: Create the Products Dimension

This script adds a dimension called Products.

The level Product Id contains the setting UniqueCategories True. This makes the level unique. This is necessary because the level is associated with columns from multiple queries.

Dim_Products.mdl

```
DimAdd "Products" DimType Regular

RootCatMake "Product Line" Dimension "Products"

DrillCatMake "By Product Line" Dimension "Products"

Root "Product Line" Inclusion Suppress

LevelAdd "Product Line" Dimension "Products"

Drill "By Product Line" Source "Product Line"

Associations "Product Line" AssociationType Type_Query
AssociationRole Role_Source

AssociationReferenced "Product Line"

LevelAdd "Product Type" Dimension "Products"

Drill "By Product Line" Parent "Product Line" Source
"Product Type"

Associations "Product Type" AssociationType Type_Query

AssociationRole Role_Source AssociationReferenced "Product
Type"

LevelAdd "Product Id" Dimension "Products" Drill "By
Product Line"

Parent "Product Type" Source "Product Id" Label "Product
Name"

UniqueCategories True

Associations "Product Id" AssociationType Type_Query

AssociationRole Role_Source AssociationReferenced "Product
Id"
```

Step 4: Create the Product Cost Measure

This script creates the Product Cost measure. It specifies that the measure is taken from the Cost column.

Me_Prodcost.mdl

```
MeasureAdd "Product Cost"

Associations "Cost" AssociationType Type_Query
```

```
AssociationRole Role_Source AssociationReferenced "Cost"
```

Step 5: Create the Product Plan Measure

This script creates the Product Plan measure. It specifies that the measure is taken from the Planned Sales column.

Me_Prodplan.mdl

```
MeasureAdd "Product Plan"
Associations "Planned Sales" AssociationType Type_Query
AssociationRole Role_Source AssociationReferenced "Planned
sales"
```

Step 6: Create the Revenue Measure

This script creates the Revenue measure. It specifies that the measure is taken from the Revenue column.

Me_Revenue.mdl

```
MeasureAdd "Revenue"
Associations "Revenue" AssociationType Type_Query AssociationRole
Role_Source
AssociationReferenced "Revenue"
```

Step 7: Create the Signon

This script creates the signon required for access to the Impromptu data source. In this example, the password is disabled.

Signon.mdl

```
SignonAdd "I50_DBASE_NTV_PP60_SAMPLE" PromptForPassword
False
```

Step 8: Create Allocations

This script creates the default allocations required by each measure. (In Step 12, ModelEnsureCompleteness is used to generate default allocations, so they are provided here for illustration only.)

This script also creates two allocations of summary data. It allocates Product Plan data to the Product Line level and to the Year level according to the distribution of revenue.

Alloc_Me.mdl

```
AllocationAdd Measure "Revenue" Type Default
AllocationAdd Measure "Product Cost" Type Default
AllocationAdd Measure "Product Plan" Type Default
AllocationAdd Drill "By Product Line" Levels "Product
Line"
Measure "Product Plan" Type Allocate AllocationMeasure
"Revenue"
AllocationAdd Dimension "Years" Drill "By Time" Levels
"Year"
```

```
Measure "Product Plan" Type Allocate AllocationMeasure
"Revenue"
```

Step 9: Populate the Model

This script generates categories for the model.

Populate.mdl

```
Populatemodel
```

Step 10: Create the PowerCube

This script adds the PowerCube object to the model. It specifies where the .mdc file should be created.

Cube.mdl

```
CubeAdd "The Great Outdoors Co. Sales" MdcFile
"c:\Outdoors.mdc"
MeasureInclude 195 Yes
```

Step 11: Create .mdc Files

This script generates the PowerCube (the .mdc file).

Createcube.mdl

```
CreateFiles
```

Step 12: Create the Model by Combining the Scripts

This script combines all the previous scripts to create a complete Transformer model and generate a PowerCube.

NewModel.mdl

```
NewModel "ModelScript"
OpenMDL "C:\Que_Col.mdl"
OpenMDL "C:\Dim_Years.mdl"
OpenMDL "C:\Dim_Products.mdl"
OpenMDL "C:\Me_Revenue.mdl"
OpenMDL "C:\Me_Prodcost.mdl"
OpenMDL "C:\Me_Prodplan.mdl"
OpenMDL "C:\Signon.mdl"
OpenMDL "C:\Cube.mdl"
OpenMDL "C:\Alloc_Me.mdl"
OpenMDL "C:\Populate.mdl"
ModelEnsureCompleteness
OpenMDL "C:\Createcube.mdl"
SaveMDL "C:\FinalModel.mdl"
```

The Model as Generated by Transformer

Transformer generates the model after you run the script in Step 12. The next time you open the model FinalModel.mdl in a text editor, you will see Transformer's definition of it. This model is longer than the original scripts because

- Transformer adds default options to the object definitions
- the verb PopulateModel adds categories, all of which are defined
- the verb ModelEnsureCompleteness adds default views

Line numbers have been added to this model to make it easier to read. The .ini setting, ObjectIdOutput=0, is in effect to suppress object identifiers so that all the objects are identified by object name.

FinalModel.mdl

```
1          NewModel "ModelScript" AutoAccess False Synchronize False
           SynchroCycle 0 SynchroStamp 0 UpdateCycle 3 ModelStamp 996769120
           ClientStamp 996769120 ServerStamp 0 Generation 0 RootUser 0 Version
           "7.x.95.0 - Not For Resale" AccessManager False
           AccessManagerUserClasses False

2          DataSourceMake "Products (CSV)" Separator "," SourceType
           FlatFile_ColNames CharacterSet Ansi DecimalSep "." Thousandsep ","
           Columns True Timing PopYesCreateDefault Source "c:\prodinfo.csv"
           EnableMultiProcess False SetCurrent True ServerSource False Speed
           False Presummarized False

3          ColumnMake "Product Line" DataSource "Products (CSV)" Origin
           Generated Offset 0 Column "Product Line" Storage Default Scale 0
           Size 1 Decimals 0 InputScale 0 TimeArray Off

4          ColumnMake "Product Type" DataSource "Products (CSV)" Origin
           Generated Offset 1 Column "Product Type" Storage Default Scale 0
           Size 1 Decimals 0 InputScale 0 TimeArray Off

5          ColumnMake "Product Id" DataSource "Products (CSV)" Origin
           Generated Offset 2 Column "Product Id" Storage Default Scale 0
           Size 1 Decimals 0 InputScale 0 TimeArray Off

6          ColumnMake "Product Name" DataSource "Products (CSV)" Origin
           Generated Offset 3 Column "Product Name" Storage Default Scale 0
           Size 1 Decimals 0 InputScale 0 TimeArray Off
```

```
7          DataSourceMake "Prod Line Plan" Separator "," SourceType
           FlatFile_ColNames CharacterSet Ansi DecimalSep "." Thousandsep ","
           Columns True Timing PopYesCreateDefault Source "c:\ProdPlan.asc"
           EnableMultiProcess False SetCurrent True ServerSource False Speed
           False Presummarized False

8          ColumnMake "YEAR" DataSource "Prod Line Plan" Origin Generated
           Offset 0 Column "Time" Storage Default Format Y DateLevel Year
           Scale 0 Size 1 Decimals 0 Class Date InputScale 0 TimeArray Off

9          ColumnMake "PROD_LINE" DataSource "Prod Line Plan" Origin Generated
           Offset 1 Column "Product Line" Storage Default Scale 0 Size 1
           Decimals 0 InputScale 0 TimeArray Off

10         ColumnMake "FORECAST" DataSource "Prod Line Plan" Origin Generated
           Offset 2 Column "Planned sales" Storage Default Scale 0 Size 1
           Decimals 0 InputScale 0 TimeArray Off
```

```
11          DataSourceMake "MAIN (IQD)" Separator "," SourceType DataSource
            CharacterSet Ansi DecimalSep "." Thousandsep "," Columns True
            Timing PopYesCreateDefault

            Source "c:\bsc_msrs.iqd" SQL 'select T1."ORDER_DT" as c1,

            T2."PROD_NO" as c2,

            T1."REP_NO" as '

            'c3,

            T1."CUST_NO" as c4,

            T2."QTY" as c5,

            (T2."QTY" * T2."PRI'

            'CE") as c6,

            (T2."QTY" * T3."PROD_COST") as c7,

            (CASE WHEN ((((T'

            '2."QTY" * T2."PRICE") - (T2."QTY" * T3."PROD_COST")) / (T2."QTY"
            * T2."'

            'PRICE")) <= 0.19) THEN (' "'Under 20%') WHEN ((((T2." '"QTY" *
            T2."PRICE") - (T2."QTY" * T3."PROD_COST")) / (T2."QTY" * T2."PR'

            'ICE")) BETWEEN 0.2 AND 0.65) THEN (' "'20% - 65%') WHEN ((((T2."
            '"QTY" * T2."PRICE") - (T2."QTY" * T3."PROD_COST")) / (T2."QTY" *
            T2."PR'

            'ICE")) >= 0.66) THEN (' "'Over 65%') ELSE ('ERROR') END) as c8

            from "

            '"ORDER" T1,

            ("PRODUCT" T3 left outer join "ORDRDETL" T2 on T2."PROD_NO'

            '" = T3."PROD_NO")

            where (T2."ORDER_NO" = T1."ORDER_NO")

            order by c1 asc'

            ",c3 asc,c4 asc,c2 asc

            " Isolation 0 UserEditable False SourceSignonList "I70_DBASE_NTV_
            PP70_SAMPLE"

            EndList ImrName "C:\Program Files\Cognos\cern\samples\PowerPlay\
            Cubes and Reports\bsc_msrs.imr" Stamp 996601938 EnableMultiProcess
            False SetCurrent True ServerSource False Speed False Presummarized
            False

12          ColumnMake "Order Dt" DataSource "MAIN (IQD)" Origin Source Offset
            0 Column "Time" Storage Int32 Scale 0 Size 4 Decimals 0 Class Date
            InputScale 0 TimeArray Off
```

```
13          ColumnMake "od-Prod No" DataSource "MAIN (IQD)" Origin Source
            Offset 1 Column "Product Id" Storage Float64 Scale 0 Size 5
            Decimals 0 Class Quantity InputScale 0 TimeArray Off

14          ColumnMake "Revenue" DataSource "MAIN (IQD)" Origin Generated
            Offset 5 Column "Revenue" Storage Default Scale 0 Size 1 Decimals
            0 InputScale 0 TimeArray Off

15          ColumnMake "Cost" DataSource "MAIN (IQD)" Origin Source Offset 6
            Column "Cost" Storage Float64 Scale 0 Size 6 Decimals 0 Class
            Quantity InputScale 0 TimeArray Off

16          DimMake "Years" DimType NonStandard EarliestDate 19000101
            LatestDate 99991231 ManualPeriods False DaysInWeek 127 NewCatsLock
            False ExcludeAutoPartitioning False

17          RootCatMake "Time" Dimension "Years" Inclusion Generate Lastuse
            20010802 Date 0 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

18          DrillCatMake "By Time" Dimension "Years" Root "Time" Inclusion
            Suppress Filtered False Suppressed True PrimaryDrill True
            YearBegins 20010101 PartialWeek Split ExtraWeek None WeekBegins
            Sunday

19          LevelMake "Year" Dimension "Years" Drill "By Time" Parent "" Blanks
            "( blank )" DateFunction Year Generate Need RefreshLabel False
            RefreshDescription False RefreshShortName False NewCatsLock False
            CatLabFormat "YYYY" Timerank 10 UniqueCategories True UniqueMove
            False Associations "Time" AssociationContext "By Time"
            AssociationType Type_Query AssociationRole Role_OrderBy
            AssociationReferenced "Time" SortOrder Default SortAs Ascending
            Associations "Time" AssociationType Type_Query AssociationRole
            Role_Source AssociationReferenced "Time"

20          LevelMake "Quarter" Dimension "Years" Drill "By Time" Parent "Year"
            Blanks "( blank )" DateFunction Quarter Generate Need RefreshLabel
            False RefreshDescription False RefreshShortName False NewCatsLock
            False CatLabFormat 'YYYY "Q" Q' Timerank 20 UniqueCategories True
            UniqueMove False Associations "Time" AssociationContext "By Time"
            AssociationType Type_Query AssociationRole Role_OrderBy
            AssociationReferenced "Time" SortOrder Default SortAs Ascending
            Associations "Time" AssociationType Type_Query AssociationRole
            Role_Source AssociationReferenced "Time"
```

```
21          LevelMake "Month" Dimension "Years" Drill "By Time" Parent
            "Quarter" Blanks "( blank )" DateFunction Month Generate Need
            RefreshLabel False RefreshDescription False RefreshShortName False
            NewCatsLock False CatLabFormat "YYYY/MMM" Timerank 30
            UniqueCategories True UniqueMove False Associations "Time"
            AssociationContext "By Time" AssociationType Type_Query
            AssociationRole Role_OrderBy AssociationReferenced "Time" SortOrder
            Default SortAs Ascending Associations "Time" AssociationType
            Type_Query AssociationRole Role_Source AssociationReferenced "Time"

22          CatMake "19960101-19961231" Dimension "Years" Drill "By Time"
            Levels "Year" Parent "By Time" OrderBy Drill "By Time" Value "1996"
            Label "1996" Lastuse 20010802 SourceValue "1996" Date 19960101
            Filtered False Suppressed False Sign False IsKeyOrphanage False
            IsTruncated False Blanks False

23          CatMake "19960101-19960331" Dimension "Years" Drill "By Time"
            Levels "Quarter" Parent "19960101-19961231" OrderBy Drill "By Time"
            Value "19960101" Label "1996 Q 1" Lastuse 20010802 SourceValue
            "19960101" Date 19960101 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

24          CatMake "19960101-19960131" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19960101-19960331" OrderBy Drill "By Time"
            Value "199601" Label "1996/Jan" Lastuse 20010802 SourceValue
            "199601" Date 19960101 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

25          CatMake "19960201-19960229" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19960101-19960331" OrderBy Drill "By Time"
            Value "199602" Label "1996/Feb" Lastuse 20010802 SourceValue
            "199602" Date 19960201 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

26          CatMake "19960301-19960331" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19960101-19960331" OrderBy Drill "By Time"
            Value "199603" Label "1996/Mar" Lastuse 20010802 SourceValue
            "199603" Date 19960301 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

27          CatMake "19960401-19960630" Dimension "Years" Drill "By Time"
            Levels "Quarter" Parent "19960101-19961231" OrderBy Drill "By Time"
            Value "19960401" Label "1996 Q 2" Lastuse 20010802 SourceValue
            "19960401" Date 19960401 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False
```

```
28          CatMake "19960401-19960430" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19960401-19960630" OrderBy Drill "By Time"
            Value "199604" Label "1996/Apr" Lastuse 20010802 SourceValue
            "199604" Date 19960401 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

29          CatMake "19960501-19960531" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19960401-19960630" OrderBy Drill "By Time"
            Value "199605" Label "1996/May" Lastuse 20010802 SourceValue
            "199605" Date 19960501 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

30          CatMake "19960601-19960630" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19960401-19960630" OrderBy Drill "By Time"
            Value "199606" Label "1996/Jun" Lastuse 20010802 SourceValue
            "199606" Date 19960601 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

31          CatMake "19960701-19960930" Dimension "Years" Drill "By Time"
            Levels "Quarter" Parent "19960101-19961231" OrderBy Drill "By Time"
            Value "19960701" Label "1996 Q 3" Lastuse 20010802 SourceValue
            "19960701" Date 19960701 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

32          CatMake "19960701-19960731" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19960701-19960930" OrderBy Drill "By Time"
            Value "199607" Label "1996/Jul" Lastuse 20010802 SourceValue
            "199607" Date 19960701 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

33          CatMake "19960801-19960831" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19960701-19960930" OrderBy Drill "By Time"
            Value "199608" Label "1996/Aug" Lastuse 20010802 SourceValue
            "199608" Date 19960801 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

34          CatMake "19960901-19960930" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19960701-19960930" OrderBy Drill "By Time"
            Value "199609" Label "1996/Sep" Lastuse 20010802 SourceValue
            "199609" Date 19960901 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

35          CatMake "19961001-19961231" Dimension "Years" Drill "By Time"
            Levels "Quarter" Parent "19960101-19961231" OrderBy Drill "By Time"
            Value "19961001" Label "1996 Q 4" Lastuse 20010802 SourceValue
            "19961001" Date 19961001 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False
```

```
36          CatMake "19961001-19961031" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19961001-19961231" OrderBy Drill "By Time"
            Value "199610" Label "1996/Oct" Lastuse 20010802 SourceValue
            "199610" Date 19961001 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

37          CatMake "19961101-19961130" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19961001-19961231" OrderBy Drill "By Time"
            Value "199611" Label "1996/Nov" Lastuse 20010802 SourceValue
            "199611" Date 19961101 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

38          CatMake "19961201-19961231" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19961001-19961231" OrderBy Drill "By Time"
            Value "199612" Label "1996/Dec" Lastuse 20010802 SourceValue
            "199612" Date 19961201 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

39          CatMake "19970101-19971231" Dimension "Years" Drill "By Time"
            Levels "Year" Parent "By Time" OrderBy Drill "By Time" Value "1997"
            Label "1997" Lastuse 20010802 SourceValue "1997" Date 19970101
            Filtered False Suppressed False Sign False IsKeyOrphanage False
            IsTruncated False Blanks False

40          CatMake "19970101-19970331" Dimension "Years" Drill "By Time"
            Levels "Quarter" Parent "19970101-19971231" OrderBy Drill "By Time"
            Value "19970101" Label "1997 Q 1" Lastuse 20010802 SourceValue
            "19970101" Date 19970101 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

41          CatMake "19970101-19970131" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19970101-19970331" OrderBy Drill "By Time"
            Value "199701" Label "1997/Jan" Lastuse 20010802 SourceValue
            "199701" Date 19970101 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

42          CatMake "19970201-19970228" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19970101-19970331" OrderBy Drill "By Time"
            Value "199702" Label "1997/Feb" Lastuse 20010802 SourceValue
            "199702" Date 19970201 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

43          CatMake "19970301-19970331" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19970101-19970331" OrderBy Drill "By Time"
            Value "199703" Label "1997/Mar" Lastuse 20010802 SourceValue
            "199703" Date 19970301 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False
```

```
44          CatMake "19970401-19970630" Dimension "Years" Drill "By Time"
            Levels "Quarter" Parent "19970101-19971231" OrderBy Drill "By Time"
            Value "19970401" Label "1997 Q 2" Lastuse 20010802 SourceValue
            "19970401" Date 19970401 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

45          CatMake "19970401-19970430" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19970401-19970630" OrderBy Drill "By Time"
            Value "199704" Label "1997/Apr" Lastuse 20010802 SourceValue
            "199704" Date 19970401 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

46          CatMake "19970501-19970531" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19970401-19970630" OrderBy Drill "By Time"
            Value "199705" Label "1997/May" Lastuse 20010802 SourceValue
            "199705" Date 19970501 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

47          CatMake "19970601-19970630" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19970401-19970630" OrderBy Drill "By Time"
            Value "199706" Label "1997/Jun" Lastuse 20010802 SourceValue
            "199706" Date 19970601 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

48          CatMake "19970701-19970930" Dimension "Years" Drill "By Time"
            Levels "Quarter" Parent "19970101-19971231" OrderBy Drill "By Time"
            Value "19970701" Label "1997 Q 3" Lastuse 20010802 SourceValue
            "19970701" Date 19970701 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

49          CatMake "19970701-19970731" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19970701-19970930" OrderBy Drill "By Time"
            Value "199707" Label "1997/Jul" Lastuse 20010802 SourceValue
            "199707" Date 19970701 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

50          CatMake "19970801-19970831" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19970701-19970930" OrderBy Drill "By Time"
            Value "199708" Label "1997/Aug" Lastuse 20010802 SourceValue
            "199708" Date 19970801 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

51          CatMake "19970901-19970930" Dimension "Years" Drill "By Time"
            Levels "Month" Parent "19970701-19970930" OrderBy Drill "By Time"
            Value "199709" Label "1997/Sep" Lastuse 20010802 SourceValue
            "199709" Date 19970901 Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False
```

```
52        CatMake "19971001-19971231" Dimension "Years" Drill "By Time"
          Levels "Quarter" Parent "19970101-19971231" OrderBy Drill "By Time"
          Value "19971001" Label "1997 Q 4" Lastuse 20010802 SourceValue
          "19971001" Date 19971001 Filtered False Suppressed False Sign False
          IsKeyOrphanage False IsTruncated False Blanks False

53        CatMake "19971001-19971031" Dimension "Years" Drill "By Time"
          Levels "Month" Parent "19971001-19971231" OrderBy Drill "By Time"
          Value "199710" Label "1997/Oct" Lastuse 20010802 SourceValue
          "199710" Date 19971001 Filtered False Suppressed False Sign False
          IsKeyOrphanage False IsTruncated False Blanks False

54        CatMake "19971101-19971130" Dimension "Years" Drill "By Time"
          Levels "Month" Parent "19971001-19971231" OrderBy Drill "By Time"
          Value "199711" Label "1997/Nov" Lastuse 20010802 SourceValue
          "199711" Date 19971101 Filtered False Suppressed False Sign False
          IsKeyOrphanage False IsTruncated False Blanks False

55        CatMake "19971201-19971231" Dimension "Years" Drill "By Time"
          Levels "Month" Parent "19971001-19971231" OrderBy Drill "By Time"
          Value "199712" Label "1997/Dec" Lastuse 20010802 SourceValue
          "199712" Date 19971201 Current Filtered False Suppressed False
          Sign False IsKeyOrphanage False IsTruncated False Blanks False

56        ViewMake "All Categories" Dimension "Years" Type All ViewUserClass
          ""

57        ViewMake "Omit Dimension" Dimension "Years" Type Omit ViewUserClass
          ""

58        DimMake "Products" DimType Regular NewCatsLock False
          ExcludeAutoPartitioning False RootCatMake "Product Line" Dimension
          "Products" Inclusion Generate Lastuse 20010802 Filtered False
          Suppressed False Sign False IsKeyOrphanage False IsTruncated False
          Blanks False

59        DrillCatMake "By Product Line" Dimension "Products" Root "Product
          Line" Inclusion Suppress Filtered False Suppressed True
          PrimaryDrill True

60        LevelMake "Product Line" Dimension "Products" Drill "By Product
          Line" Parent "" Blanks "( blank )" DateFunction None Generate Need
          RefreshLabel False RefreshDescription False RefreshShortName False
          NewCatsLock False Timerank 0 UniqueCategories False UniqueMove
          False Associations "Product Line" AssociationType Type_Query
          AssociationRole Role_Source AssociationReferenced "Product Line"
```

```
61          LevelMake "Product Type" Dimension "Products" Drill "By Product
            Line" Parent "Product Line" Blanks "( blank )" DateFunction None
            Generate Need RefreshLabel False RefreshDescription False
            RefreshShortName False NewCatsLock False Timerank 0
            UniqueCategories False UniqueMove False Associations "Product Type"
            AssociationType Type_Query AssociationRole Role_Source
            AssociationReferenced "Product Type"

62          LevelMake "Product Id" Dimension "Products" Drill "By Product Line"
            Parent "Product Type" Blanks "( blank )" DateFunction None Generate
            Need RefreshLabel False RefreshDescription False RefreshShortName
            False NewCatsLock False Timerank 0 UniqueCategories True UniqueMove
            False Associations "Product Id" AssociationType Type_Query
            AssociationRole Role_Source AssociationReferenced "Product Id"
            Associations "Product Name" AssociationType Type_Query
            AssociationRole Role_Label AssociationReferenced "Product Name"

63          CatMake "Environmental Line" Dimension "Products" Drill "By Product
            Line" Levels "Product Line" Parent "By Product Line" Lastuse
            20010802 SourceValue "Environmental Line" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

64          CatMake "Alert Devices" Dimension "Products" Drill "By Product
            Line" Levels "Product Type" Parent "Environmental Line" Lastuse
            20010802 SourceValue "Alert Devices" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

65          CatMake "60100" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Alert Devices" Label "Pocket U.V. Alerter"
            Lastuse 20010802 SourceValue "60100" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

66          CatMake "60101" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Alert Devices" Label "Microwave Detector"
            Lastuse 20010802 SourceValue "60101" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

67          CatMake "60102" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Alert Devices" Label "Pocket Radon Alerter"
            Lastuse 20010802 SourceValue "60102" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False
```

```
68          CatMake "Bio-Friendly Soaps" Dimension "Products" Drill "By Product
            Line" Levels "Product Type" Parent "Environmental Line" Lastuse
            20010802 SourceValue "Bio-Friendly Soaps" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

69          CatMake "60300" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Bio-Friendly Soaps" Label "RiverKind Shampoo"
            Lastuse 20010802 SourceValue "60300" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

70          CatMake "60301" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Bio-Friendly Soaps" Label "RiverKind Soap"
            Lastuse 20010802 SourceValue "60301" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

71          CatMake "60302" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Bio-Friendly Soaps" Label "RiverKind
            Detergent" Lastuse 20010802 SourceValue "60302" Filtered False
            Suppressed False Sign False IsKeyOrphanage False IsTruncated False
            Blanks False

72          CatMake "Recycled Products" Dimension "Products" Drill "By Product
            Line" Levels "Product Type" Parent "Environmental Line" Lastuse
            20010802 SourceValue "Recycled Products" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

73          CatMake "60201" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Recycled Products" Label "Enviro-Kit" Lastuse
            20010802 SourceValue "60201" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False

74          CatMake "60200" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Recycled Products" Label "EnviroSak" Lastuse
            20010802 SourceValue "60200" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False

75          CatMake "60202" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Recycled Products" Label "Enviro-T" Lastuse
            20010802 SourceValue "60202" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False
```

```
76          CatMake "Sunblock" Dimension "Products" Drill "By Product Line"
            Levels "Product Type" Parent "Environmental Line" Lastuse 20010802
            SourceValue "Sunblock" Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

77          CatMake "60400" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Sunblock" Label "Sun Shelter-8" Lastuse
            20010802 SourceValue "60400" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False

78          CatMake "60401" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Sunblock" Label "Sun Shelter-15" Lastuse
            20010802 SourceValue "60401" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False

79          CatMake "60402" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Sunblock" Label "Sun Shelter-30" Lastuse
            20010802 SourceValue "60402" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False

80          CatMake "Water Purifiers" Dimension "Products" Drill "By Product
            Line" Levels "Product Type" Parent "Environmental Line" Lastuse
            20010802 SourceValue "Water Purifiers" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

81          CatMake "60500" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Water Purifiers" Label "Pro-Lite Water Filter"
            Lastuse 20010802 SourceValue "60500" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

82          CatMake "60501" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Water Purifiers" Label "Pocket Water Filter"
            Lastuse 20010802 SourceValue "60501" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

83          CatMake "GO Sport Line" Dimension "Products" Drill "By Product
            Line" Levels "Product Line" Parent "By Product Line" Lastuse
            20010802 SourceValue "GO Sport Line" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

84          CatMake "Carry-Bags" Dimension "Products" Drill "By Product Line"
            Levels "Product Type" Parent "GO Sport Line" Lastuse 20010802
            SourceValue "Carry-Bags" Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False
```

```
85        CatMake "50100" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Carry-Bags" Label "GO Sport Bag" Lastuse
          20010802 SourceValue "50100" Filtered False Suppressed False Sign
          False IsKeyOrphanage False IsTruncated False Blanks False

86        CatMake "50101" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Carry-Bags" Label "GO Ski Gear Bag" Lastuse
          20010802 SourceValue "50101" Filtered False Suppressed False Sign
          False IsKeyOrphanage False IsTruncated False Blanks False

87        CatMake "50102" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Carry-Bags" Label "GO Duffel Bag" Lastuse
          20010802 SourceValue "50102" Filtered False Suppressed False Sign
          False IsKeyOrphanage False IsTruncated False Blanks False

88        CatMake "Sport Wear" Dimension "Products" Drill "By Product Line"
          Levels "Product Type" Parent "GO Sport Line" Lastuse 20010802
          SourceValue "Sport Wear" Filtered False Suppressed False Sign False
          IsKeyOrphanage False IsTruncated False Blanks False

89        CatMake "50201" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Sport Wear" Label "GO Headband" Lastuse
          20010802 SourceValue "50201" Filtered False Suppressed False Sign
          False IsKeyOrphanage False IsTruncated False Blanks False

90        CatMake "50201" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Sport Wear" Label "GO Headband" Lastuse
          20010802 SourceValue "50201" Filtered False Suppressed False Sign
          False IsKeyOrphanage False IsTruncated False Blanks False

91        CatMake "50202" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Sport Wear" Label "GO Wristband" Lastuse
          19980825 SourceValue "50202" Filtered False Suppressed False Sign
          False IsKeyOrphanage False IsTruncated False Blanks False

92        CatMake "50203" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Sport Wear" Label "GO Water Bottle" Lastuse
          19980825 SourceValue "50203" Filtered False Suppressed False Sign
          False IsKeyOrphanage False IsTruncated False Blanks False

93        CatMake "Outdoor Products" Dimension "Products" Drill "By Product
          Line" Levels "Product Line" Parent "By Product Line" Lastuse
          20010802 SourceValue "Outdoor Products" Filtered False Suppressed
          False Sign False IsKeyOrphanage False IsTruncated False Blanks
          False
```

```
94          CatMake "Back Packs" Dimension "Products" Drill "By Product Line"
            Levels "Product Type" Parent "Outdoor Products" Lastuse 20010802
            SourceValue "Back Packs" Filtered False Suppressed False Sign False
            IsKeyOrphanage False IsTruncated False Blanks False

95          CatMake "40300" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Back Packs" Label "Day Tripper" Lastuse
            20010802 SourceValue "40300" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False

96          CatMake "40301" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Back Packs" Label "Pack n' Hike" Lastuse
            20010802 SourceValue "40301" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False

97          CatMake "40302" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Back Packs" Label "GO Small Waist Pack"
            Lastuse 20010802 SourceValue "40302" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

98          CatMake "40303" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Back Packs" Label "GO Large Waist Pack"
            Lastuse 20010802 SourceValue "40303" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

99          CatMake "Cooking Equipment" Dimension "Products" Drill "By Product
            Line" Levels "Product Type" Parent "Outdoor Products" Lastuse
            20010802 SourceValue "Cooking Equipment" Filtered False Suppressed
            False Sign False IsKeyOrphanage False IsTruncated False Blanks
            False

100         CatMake "40400" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Cooking Equipment" Label "Dover-1" Lastuse
            20010802 SourceValue "40400" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False

101         CatMake "40401" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Cooking Equipment" Label "Dover-2" Lastuse
            20010802 SourceValue "40401" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False

102         CatMake "40402" Dimension "Products" Drill "By Product Line" Levels
            "Product Id" Parent "Cooking Equipment" Label "GO Cookset" Lastuse
            20010802 SourceValue "40402" Filtered False Suppressed False Sign
            False IsKeyOrphanage False IsTruncated False Blanks False
```

```
103       CatMake "40403" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Cooking Equipment" Label "GO Camp Kettle"
          Lastuse 20010802 SourceValue "40403" Filtered False Suppressed
          False Sign False IsKeyOrphanage False IsTruncated False Blanks
          False

104       CatMake "Sleeping Bags" Dimension "Products" Drill "By Product
          Line" Levels "Product Type" Parent "Outdoor Products" Lastuse
          20010802 SourceValue "Sleeping Bags" Filtered False Suppressed
          False Sign False IsKeyOrphanage False IsTruncated False Blanks
          False

105       CatMake "40200" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Sleeping Bags" Label "MoonBeam" Lastuse
          20010802 SourceValue "40200" Filtered False Suppressed False Sign
          False IsKeyOrphanage False IsTruncated False Blanks False

106       CatMake "40201" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Sleeping Bags" Label "MoonGlow" Lastuse
          20010802 SourceValue "40201" Filtered False Suppressed False Sign
          False IsKeyOrphanage False IsTruncated False Blanks False

107       CatMake "40202" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Sleeping Bags" Label "MoonLite" Lastuse
          20010802 SourceValue "40202" Filtered False Suppressed False Sign
          False IsKeyOrphanage False IsTruncated False Blanks False

108       CatMake "Tents" Dimension "Products" Drill "By Product Line" Levels
          "Product Type" Parent "Outdoor Products" Lastuse 20010802
          SourceValue "Tents" Filtered False Suppressed False Sign False
          IsKeyOrphanage False IsTruncated False Blanks False

109       CatMake "40100" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Tents" Label "Star Lite" Lastuse 20010802
          SourceValue "40100" Filtered False Suppressed False Sign False
          IsKeyOrphanage False IsTruncated False Blanks False

110       CatMake "40101" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Tents" Label "Star Gazer-2" Lastuse 20010802
          SourceValue "40101" Filtered False Suppressed False Sign False
          IsKeyOrphanage False IsTruncated False Blanks False

111       CatMake "40102" Dimension "Products" Drill "By Product Line" Levels
          "Product Id" Parent "Tents" Label "Star Gazer-3" Lastuse 20010802
          SourceValue "40102" Filtered False Suppressed False Sign False
          IsKeyOrphanage False IsTruncated False Blanks False
```

```
112     CatMake "40103" Dimension "Products" Drill "By Product Line" Levels
        "Product Id" Parent "Tents" Label "StarDome" Lastuse 20010802
        SourceValue "40103" Current Filtered False Suppressed False Sign
        False IsKeyOrphanage False IsTruncated False Blanks False

113     CatMake "40104" Dimension "Products" Drill "By Product Line" Levels
        "Product Id" Parent "Tents" Label "Micro Lite" Lastuse 20010802
        SourceValue "40104" Filtered False Suppressed False Sign False
        IsKeyOrphanage False IsTruncated False Blanks False

114     ViewMake "All Categories" Dimension "Products" Type All
        ViewUserClass ""

115     ViewMake "Omit Dimension" Dimension "Products" Type Omit
        ViewUserClass ""

116     MeasureMake "Revenue" Storage Default OutPutScale 0 Decimals 0
        ReverseSign False IsCurrency False DrillThrough False EndList
        Associations "Revenue" AssociationType Type_Query AssociationRole
        Role_Source AssociationReferenced "Revenue"

117     MeasureMake "Product Cost" Storage Default OutPutScale 0 Decimals
        0 ReverseSign False IsCurrency False DrillThrough False EndList
        Associations "Cost" AssociationType Type_Query AssociationRole
        Role_Source AssociationReferenced "Cost"

118     MeasureMake "Product Plan" Storage Default OutPutScale 0 Decimals
        0 ReverseSign False IsCurrency False DrillThrough False EndList
        Associations "Planned Sales" AssociationType Type_Query
        AssociationRole Role_Source AssociationReferenced "Planned sales"

119     SignonMake "I70_DBASE_NTV_PP70_SAMPLE"
        FullDb
        "7C3155E28283DDFEF647A376504802295E3F3A5FACBCD1616F8770A97C3530151BD5644"
        "09010B9D440F8CC047E5E62D7968B87633C7BC6DA56D9539D5FFB47D8A1A5E4EE9790A0"
        "A00CDC111C1CF518BA09F9DAD8971B53E3D1DCA8A822962F223FA52F4CCE534946A57A5"
        "23FF8FA4643644A41E7F18519B37C7E8C881B2A80ADAFAB3A3053D4D728B8BAA51F49B0"
        "C4CCB26314FF28DB56ED4BF7357503BAFF0EB65F0713691B4FC64ED26372563E2042978"
        "2A618F81AF7359208330F265CAD582A19D9200E78660EC690A4EF3186AFEE7A3E7429A8"
        "702C51C693EF7EA7F3D673E4B8C569D51E254781B0C6A3CAAA6ED4CD5F94AC450E79526"
        "1C53078EFF0ECC" DbType "DB" DatabaseSubType BDE PromptForPassword
        False Manual False SignonMake "I50_DBASE_NTV_PP60_SAMPLE"
        PromptForPassword False Manual False
```

```
120       CubeMake "The Great Outdoors Co. Sales" MdcFile "c:\Outdoors.mdc"
          Status OK CubeCreation On Optimize Categories Compress False
          DatabaseInfo "Local;" IncrementalUpdate False ServerCube False
          CubeStamp 996769127 CubeCycle 2 BlockParentTotals False Caching
          False UseAlternateFileName False DrillThrough False EndList
          DimensionView "Years" "All Categories" DimensionView "Products"
          "All Categories" MeasureInclude "Revenue" Yes MeasureInclude
          "Product Cost" Yes MeasureInclude "Product Plan" Yes

121       AllocationAdd Measure "Revenue" Type Default

122       AllocationAdd Measure "Product Cost" Type Default

123       AllocationAdd Measure "Product Plan" Type Default

124       AllocationAdd Dimension "Products" Drill "By Product Line" Levels
          "Product Line" Measure "Product Plan" Type Allocate
          AllocationMeasure "Revenue"

125       AllocationAdd Dimension "Years" Drill "By Time" Levels "Year"
          Measure "Product Plan" Type Allocate AllocationMeasure "Revenue"
```

Sample Scripts

You can modify the following sample scripts to perform some common MDL operations.

Change Data Source Types

This script changes a source file. It specifies the new file type (Excel) as well as the new file (Products.xls), and saves the model.

```
OpenPy "model.pyi"
DataSourceUpdate "Products (CSV)" SourceType ExcelCrossTab
Source "Products.xls"
SavePy "model.pyi"
```

Change Data Source

This script removes all categories in the Locations dimension, and changes the source for the Locations dimension to Locations.mdb, a Microsoft Access file. It then generates categories and creates the .mdc file. It does not save the changes to the model.

```
OpenMDL "c:\Outdoors.mdl"
CleanHouse Dimension "Locations" 19991231
DataSourceUpdate "Locations (CSV)" SourceType Access
Source "c:\Locations.mdb" DataRange "Locations"
PopulateFromQueries "Locations (CSV)"
CreateFiles
```

Update Selected Cubes

This script deletes the measure Product Cost from two cubes and then updates the .mdc files for those cubes. It does not save the changes to the model.

```
OpenMDL "c:\Great Outdoors (Admin).mdl"
CubeUpdate "Americas" MeasureInclude "Product Cost" No
CubeUpdate "Europe" MeasureInclude "Product Cost" No
CreateFromCubes "Americas" "Europe"
```

Update All Cubes

This script deletes all the categories in the Channels dimension, changes the data source for Channels to Channels.dbf, and generates categories from the new source. It then creates the .mdc file. It does not save the changes to the model.

```
OpenMDL "c:\Outdoors.mdl"
CleanHouse Dimension "Channels" 19991231
DataSourceUpdate "Channels (dBase 4.0)" Source "c:\Channels.dbf"

PopulateFromQueries "Channels (dBase 4.0)"
CreateFiles
```

Add a Description

Every Transformer object can have a description associated with it up to 4,095 characters in length. However, in MDL the description must be divided into a series of smaller 256 character blocks. Descriptions must be enclosed in quotation marks.

This example provides a description for the dimension Products and saves the model.

```
OpenMDL "c:\Outdoors.mdl"
DimUpdate "Products" DimInfo "This is an example of a
'description' containing quotes."
SaveMDL "c:\Outdoors.mdl"
```

Add Several Descriptions

This script adds or changes a description for several objects and saves the model.

```
OpenMDL "c:\Outdoors.mdl"
DimUpdate "Products" DimInfo "Updating the description
of a dimension."
ColumnUpdate "Product Line" ColumnInfo "Updating the
description of a column."
LevelUpdate "Product Line" LevelInfo "Updating the description
of a level."
SaveMDL "c:\Outdoors.mdl"
```

Change a Cube Output Location

This script changes the output location of the cube National to c:\mdcdir\national.mdc, and saves the model.

```
OpenPy "model.pyh"
```

```
CubeUpdate "national" MDCFile "c:\mdcdir\national.mdc"
SavePy "model.pyh"
```

Change Several Cube Output Locations

This script changes the output location of several cubes and generates the .mdc files in the new location. It does not save the changes in the model.

```
OpenMDL "c:\Great Outdoors (Admin).mdl"
CubeUpdate "The Great Outdoors Co. Sales" MDCFile "c:\Great
Outdoors 7.x"
CubeUpdate "All Regions" MDCFile "c:\All Regions"
CubeUpdate "Americas" MDCFile "c:\Americas"
CubeUpdate "Far East" MDCFile "c:\Far East"
CubeUpdate "Europe" MDCFile "c:\Europe"
CreateFiles
```

Change the PowerCube Output Type

This script changes the file type of a PowerCube and saves the model.

```
OpenPy "model.pyi"
SignonMake "name" PromptForPassword False UserId "user"
Password "my password"
PowerCubeUpdate "national" DatabaseInfo "Local;;" Database
"name"
SavePy "model.pyi"
```

Change Optimization of Cubes

This script changes the Optimization setting of several cubes to Categories and generates the .mdc files. It does not save the changes.

```
OpenMDL "c:\Great Outdoors (Admin).mdl"
CubeUpdate "The Great Outdoors Co. Sales" Optimize Categories
CubeUpdate "All Regions" Optimize Categories
CubeUpdate "Americas" Optimize Categories
CubeUpdate "Far East" Optimize Categories
CubeUpdate "Europe" Optimize Categories
CreateFiles
```

Automate CleanHouse

The MDL verb CleanHouse is equivalent to Clean House in the Tools menu in the user interface. CleanHouse followed by a date in format YYYYMMDD removes all categories with a Last Use date less than the specified date. To remove all categories in the model, specify a date in the future.

This script removes all categories in the Products dimension that have a Last Use date prior to November 1, 1998, and saves the model.

```
OpenPy "model.pyh"
CleanHouse Dimension "Product" 19981101
SaveMDL "model.mdl"
```

Convert Model File Formats

This script converts a .def file to .mdl and .py? (in this example, .pyh) formats.

```
OpenDef "\customer\defdat\01xbudgt.def"

SaveMDL "\customer\defdat\01xbudgt.mdl"

SavePy "\customer\defdat\01xbudgt.pyh"
```

Create Cubes Based on Dimension Views

This script creates four dimension views, creates four PowerCubes based on the dimension views, and then creates the .mdc files. Each cube relates to one region, except for All Regions which has a drill-through to all regions. The option ViewUserClass "" in each ViewMake statement specifies that it is a dimension view and not a user class view.

```
OpenMDL "c:\Outdoors.mdl"

ViewMake "Region Summary" Dimension "Locations" ViewUserClass
"" LevelSummary Dimension "Locations" Drill "By Region" Levels "Region"
Apex "Region"

ViewMake "North America" Dimension "Locations" ViewUserClass
"" Apex "Region" Summary "Europe" Summary "Far East"

ViewMake "Far East" Dimension "Locations" ViewUserClass
"" Apex "Region" Summary "Europe" Summary "North America"

ViewMake "Europe" Dimension "Locations" ViewUserClass
"" Apex "Region" Summary "Far East" Summary "North America"

CubeMake "All Regions" MdcFile "c:\All Regions.mdc" DrillThrough
True "c:\North America.mdc" "" "c:\europe.mdc" "" "c:\far east.mdc"
"" EndList DimensionView "Locations" "Region Summary"

CubeMake "North America" MdcFile "c:\North America.mdc"
 DimensionView "Locations" "North America"

CubeMake "Far East" MdcFile "c:\Far East.mdc"  DimensionView
"Locations" "Far East"

CubeMake "Europe" MdcFile "c:\Europe.mdc"  DimensionView
"Locations" "Europe"

CreateFiles
```

Update Conversion Rates

This script adds or changes the currency rates for German marks and French francs for November and December, 1997.

```
OpenMDL "C:\Great Outdoors (Admin).mdl"

CurrencyUpdate "German Marks" Levels "Month" CurrencyRateList

EffectiveDate "19971101-19971130" ConversionRate 1.666


EffectiveDate "19971201-19971231" ConversionRate 1.777
EndList

CurrencyUpdate "French Francs" Levels "Month" CurrencyRateList

EffectiveDate "19971101-19971130" ConversionRate 1.666


EffectiveDate "19971201-19971231" ConversionRate 1.777
EndList
```

Turn Off Incremental Update

This script turns off incremental update for one cube and generates the .mdc file for that cube.

```
OpenMDL "C:\Great Outdoors (Admin).mdl"
CubeUpdate "Europe" IncrementalUpdate False
CreateFromCubes "Europe"
```

Chapter 4: Syntax Conventions

When using MDL to define or change Transformer models you must apply the correct syntax.

Syntax and Naming Conventions

This book uses the following syntax and naming conventions:

- MDL verbs and keywords appear in **bold**. Verbs and keywords are not case-sensitive, but otherwise must be typed as is. For example, if the syntax includes **CatMake**, you can type CatMake, catmake, or CATMAKE.
- *Italicized* words are placeholders for things you must supply. If more than one placeholder can be supplied, the italicized word is plural, such as objCats in **StartList** *objCats* **EndList**.
- Italicized words starting with *obj* are placeholders to identify an object. The identification must be an object name, object identifier, or both. For example, *objCat* is a placeholder for a category and might appear in the MDL file as `135 "Dishwashers"`, which identifies a category with object identifier 135 and object name Dishwashers.
- Square brackets [ ] indicate optional items. If only one of two or more optional items may be chosen, they are separated by a pipe (|). For example, [**red** | **green**] means that you can specify red, green, or neither. The syntax [**red**] [**green**] means that you can use either, neither, or both red and green.
- Braces { } indicate a set of choices, separated by a pipe, from which you must choose one. For example, {**red**|**green**} means that you must specify either red or green, but not both.
- When the syntax is too long to fit on one line, it wraps at a space between characters.

Data Type Conventions

This book uses the following data type conventions.

Data type	Description
bignum	number less than 32 bits in length (4,294,967,296)
number	number less than 16 bits in length (65,536)
uns	unsigned number
string	character string, enclosed in single or double quotes, up to 256 characters in length

Example 1

This example shows that to use the verb CatMake, you must type CatMake and the identification of a category, but all other parameters are optional:

CatMake *objCat* [**Dimension** *objDim*] [**Drill** *objDrillCat*] [**Levels** *objLevel*] [**Parent** *objLevel*] [*catopts*] [**ParentList** *objLevels* **EndList**]

Example 2

Here is a more complex example of syntax:

AllocationAdd [**Dimension** *objDim*] [**Drill** *objDrillCat*] [**Levels** *objLevel* |**Category** *objRootCat*|**Category** *objCat*] **Measure** *objMeasure* **Type** *type* [**AllocationMeasure** *objMeasure*]

In this case you must type the keyword AllocationAdd and, optionally, Dimension (followed by the identification of a dimension) and Drill (followed by the identification of a drill category). Then you must type one or none of the following arguments:

- Levels (followed by the identification of a level)
- Category (followed by the identification of a root category)
- Category (followed by the identification of a regular category)

You must then type Measure followed by the identification of a measure, and Type followed by a type. Note that *type* does not have the prefix *obj*. This indicates that it is not a Transformer object.

Finally, you can optionally include AllocationMeasure and the identification of a measure.

Syntax Requirements

You can have up to 256 characters on one line in an MDL file. You can include carriage returns between keywords to reduce the length of a line.

To create a remark or temporarily remove a line from an MDL file, type two slash marks ("//") immediately before the text you want to remove or create as a remark.

Chapter 5: MDL Verbs

This chapter provides an alphabetical listing of all MDL verbs. Each entry includes

- where you can find the equivalent functionality in the user interface (UI)
- syntax
- notes
- examples

About Verbs

Most objects in MDL have four verb types associated with them: Add, Delete, Make, and Update. For example, CatAdd, CatDelete, CatMake, and CatUpdate.

The Make form includes the functionality of both the Add and Update verbs.

- Add verbs add the object to the model, and an error is issued if the object already exists. In this chapter, examples for Add verbs show you how to use MDL to create an object, and specify only those settings that are necessary. You can also use the Make form to do this. For example, you can use CatMake instead of CatAdd.
- Make verbs add the object if it doesn't exist, and update the object if it does exist. Make verbs include the functionality of both the Add and Update verbs. In this chapter, examples for Make verbs are the object definitions that are generated by Transformer when you save a model as .mdl.
- Update verbs update existing objects, and an error is issued if the object does not exist. You can also use the Make form to do this. For example, you can use CatMake instead of CatUpdate.

For more information about the functionality of each verb, see the Transformer online Help.

These are all the types of Transformer objects, and the verbs associated with them:

Transformer objects	Associated verbs
Model	NewModel
Data Source	DataSourceAdd, DataSourceDelete, QueryListUpdate, DataSourceMake, DataSourceUpdate
Column	ColumnAdd, ColumnDelete, ColumnListUpdate, Column-Make, ColumnUpdate
Dimension	DimAdd, DimDelete, DimMake, DimUpdate, Dimension-ListUpdate

Transformer objects	Associated verbs
Subdimension	SubDimRootMake, SubDimRootUpdate
Root Category	RootCatMake, RootCatUpdate
Drill Category	DrillCatMake
Level	LevelAdd, LevelDelete, LevelMake, LevelMoveAfter, LevelMoveBefore, LevelNewDrill, LevelUpdate, FilterLevel
Category	CatAdd, CatDelete, CatJoin, CatMake, CatMorph, CatMoveLevel, CatMoveVertical, CatUpdate, FilterCat
Special Category	SpecialCatAdd, SpecialCatDelete, SpecialCatMake, SpecialCatUpdate
View	ApexCat, ViewAdd, ViewDelete, ViewListUpdate, ViewMake, ViewUpdate
Measure	MeasureAdd, MeasureDelete, MeasureMake, MeasureListUpdate, MeasureUpdate, AllocationAdd
PowerCube	CubeAdd, CubeDelete, CubeMake, CubeUpdate, PowerCubeDelete, PowerCubeListUpdate, PowerCubeUserListUpdate
PowerCube Group	CubeGroupAdd, CubeGroupDelete, CubeGroupMake, CubeGroupUpdate, PowerCubeDelete, PowerCubeListUpdate
PowerCube Group Cube	CubeGroupCubeAdd, CubeGroupCubeDelete, CubeGroupCubeListUpdate, CubeGroupCubeMake, CubeGroupCubeUpdate
Signon	SignonAdd, SignonDelete, SignonListUpdate, SignonMake, SignonUpdate
Userclass	UserClassAdd, UserClassDelete, UserClassListUpdate, UserClassMake, UserClassUpdate, PowerCubeUserListUpdate
Dimension Calculation Definition	DimCalcDefAdd, DimCalcDefDelete, DimCalcDefMake, DimCalcDefUpdate

Transformer objects	Associated verbs
Currency Table	CurrencyTableAdd, CurrencyTableDelete, CurrencyTableMake, CurrencyTableUpdate
Currency	CurrencyAdd, CurrencyDelete, CurrencyMake, CurrencyUpdate
Association	AssociationAdd, AssociationDelete, AssociationMake, AssociationUpdate, CurrencyTableAdd, CurrencyTableMake, CurrencyTableUpdate, DimensionAdd, DimensionMake, DimensionUpdate, LevelAdd, LevelMake, LevelUpdate, MeasureAdd, MeasureMake, MeasureUpdate

There are also action verbs which manipulate objects:

Action verbs	
AutoDesign	OpenMDL
CatUpdateAll	OpenPY
CleanHouse	PopulateFromQueries
CreateColumns	PopulateModel
CreateFiles	ReportPartitions
CreateFromCubes	SaveMDL
CreateFromQueries	SavePY
EventEnd	SendServerModel
EventStart	SyncArchitectSource
MDCCheckServer	SummarizeCat
MDCClear	SummarizeLevel
ModelEnsureCompleteness	UpdateForwardReference
NewModel	UpdatePowerCubes
OpenDef	

The syntax conventions used in this chapter are described in "Syntax Conventions" (p. 53).

AllocationAdd

Allocates summary data to a level, dimension, or category.

UI Equivalent

Level, Dimension, or Category property sheet, Allocation tab.

Syntax

AllocationAdd [**Dimension** *objDim*] [**Drill** *objDrillCat*] [**Levels** *objLevel*|**Category** *objRootCat*|**Category** *objCat*] **Measure** *objMeasure* **Type** *type* [**AllocationMeasure** *objMeasure*]

Argument	Description
Dimension *objDim*	Is specified, if necessary, to uniquely locate the level, root category, or category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely locate the level, root category, or category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). objDrillCat can be the object name, object identifier, or both.
Levels *objLevel*	Indicates that the allocation should be done to a level, and specifies the level. *objLevel* can be the object name, object identifier, or both. This argument is optional.
Category *objRootCat*	Indicates that the allocation should be done to a dimension, and specifies the root category of the dimension. *objRootCat* can be the object name, object identifier, or both. This argument is optional.
Category *objCat*	Indicates that the allocation should be done to a category, and specifies the category. *objCat* can be the object name, object identifier, or both. This argument is optional.
Measure *objMeasure*	Specifies the measure that is being allocated. The measure can be identified by object name, object identifier, or both. This argument is mandatory.

Argument	Description
Type *type*	Specifies the type of allocation. *type* can be **Default**, **None**, **Constant**, or **Allocate**. The default is **Constant** if the allocation is to a dimension; for allocations to levels or categories it is **None**. This argument is mandatory.
AllocationMeasure *objMeasure*	Specifies the measure used as the basis for the allocation proportions. *objMeasure* can be the object name or object identifier. This argument is optional.

Note

Only one of the level, root category or category can be specified.

Every measure in a model must have an AllocationAdd statement that contains the name of the measure and the option Type Default. Transformer creates this statement if the measure is created in the user interface or if you use the verb ModelEnsureCompleteness. Otherwise you must manually create the statement when you create a measure.

If a measure is allocated, then a second AllocationAdd statement defines the allocation.

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

The syntax for AllocationAdd is explained in an example in the chapter Syntax Conventions.

Example

This is an example of the AllocationAdd statement that Transformer requires for every measure:

```
AllocationAdd Measure "Revenue" Type Default
```

A second AllocationAdd statement exists for allocated measures. This example allocates forecast data in the dimension Regions (identified by its root category, Regions) according to the distribution of revenue:

```
AllocationAdd Category "Regions" Measure "Forecast" Type
Allocate AllocationMeasure "Revenue"
```

ApexCat

Makes the selected category the highest category in a view. In PowerCubes that are created using the view, users will see only the apex category and its descendants.

UI Equivalent

In the left pane of the category viewer, for a selected view and a category, Apex (Diagram menu).

Syntax

ApexCat *objView* [**Dimension** *objDim*] {**Category** *objCat*|**Category** objRootCat |**Category** *objDrillCat*}

Argument	Description
ApexCat *objView*	Specifies the dimension view or user class view *objView* in which the category is to be apexed. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the view. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Category *objCat*	Specifies a regular category as the apex. *objCat* can be the object name, object identifier, or both. One of the regular, root, or drill categories must be specified.
Category *objRootCat*	Specifies the root category as the apex, which has the effect of removing the apex. *objRootCat* can be the object name, object identifier, or both. One of the regular, root, or drill categories must be specified.
Category *objDrillCat*	Specifies a drill category as the apex. *objDrillCat* can be the object name, object identifier, or both. One of the regular, root, or drill categories must be specified.

Note

You cannot apex special categories.

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example apexes the Outdoor Products category in the dimension view View1:

```
ApexCat "View1" Dimension "Products" Category "Outdoor
Products"
```

ArchitectDataSourceAdd

Creates an Architect data source in your model.

UI Equivalent

Using the New Model wizard to create a model based on an Architect data source. Alternatively, adding an Architect model as a data source.

Syntax

ArchitectDataSourceAdd *objArch apparchopts*

Argument	Description
ArchitectDataSourceAdd *objArch*	Specifies the name of the Architect data source *objArch* which you want to add into your model. *objArch* can be the object name, object identifier, or both.
apparchopts	Specifies the type and details of the data source *objArch*. For the complete list of options, see "apparchopts" (p. 163).

Example

```
ArchitectDataSourceAdd 117 "GO Model" ArchitectModelName
"GO model" Package "GO Sales Package"
```

ArchitectDataSourceMake

Creates or modifies an Architect data source in your model.

UI Equivalent

Using the New Model wizard to create a model based on an Architect data source. Alternatively, adding an Architect model as a data source. Modifying an Architect data source in your model.

Syntax

ArchitectDataSourceMake *objArch apparchopts*

Argument	Description
ArchitectDataSourceMake *objArch*	Specifies the name of the Architect data source *objArch* which you want to add or modify in your model. *objArch* can be the object name, object identifier, or both.
apparchopts	Specifies the type and details of the data source *objArch*. For the complete list of options, see "apparchopts" (p. 163).

Example

```
ArchitectDataSourceMake 117 "GO Model" ArchitectModelName
"GO model" Package "GO Sales Package"
```

ArchitectDataSourceUpdate

Changes an Architect data source in your model.

UI Equivalent

Modifying an Architect data source in your model.

Syntax

ArchitectDataSourceUpdate *objArch apparchopts*

Argument	Description
ArchitectDataSourceUpdate *objArch*	Specifies the name of the Architect data source *objArch* which you want to modify in your model. *objArch* can be the object name, object identifier, or both.
apparchopts	Specifies the type and details of the data source *objArch*. For the complete list of options, see "apparchopts" (p. 163).

Example

```
ArchitectDataSourceUpdate 117 "GO Model version
2"
```

AssociationAdd

Adds an association to a measure, level, dimension, category, or currency conversion table.

UI Equivalent

Adding an association in a measure, level, dimension, category, or currency conversion table property sheet.

Syntax

AssociationAdd *objAssoc assocopts*

Argument	Description
AssociationAdd *objAssoc*	Specifies the name of the association *objAssoc* which you want to create and associate with an existing object. *objAssoc* can be the object name, object identifier, or both.
assocopts	Specifies the type and details of the association *objAssoc*. For the complete list of options, see "assocopts" (p. 176).

Example

```
LevelMake 2979 "Branch Code" Drill 2969
.
.
.
AssociationAdd 6391 "Branch Code" AssociationLevel 2979
AssociationType Type_Query AssociationRole Role_Source AssociationReferenced
"Branch Code"
```

AssociationDelete

Deletes an association from a measure, level, dimension, category, or currency conversion table.

UI Equivalent

Deleting an association from a measure, level, dimension, category, or currency conversion table property sheet.

Syntax

AssociationDelete *objAssoc*

Argument	Description
AssociationDelete *objAssoc*	Specifies the name of the association *objAssoc* which you want to remove from an existing object. *objAssoc* can be the object name, object identifier, or both.

Example

```
LevelMake 2979 "Branch Code" Drill 2969
.
.
.
AssociationDelete 6391 "Branch Code"
```

AssociationMake

Creates an association or updates an existing one.

UI Equivalent

Creating or updating an association on a measure, level, dimension, category, or currency conversion table property sheet.

Syntax

AssociationMake *objassoc assocopts*

Argument	Description
AssociationMake *objAssoc*	Specifies the name of the association *objAssoc* which you want to create or update and associate with an existing object. *objAssoc* can be the object name, object identifier, or both.
assocopts	Specifies the type and details of the association *objAssoc*. For the complete list of options, see "assocopts" (p. 176).

Example

```
LevelMake 2979 "Branch Code" Drill 2969
.
.
.
AssociationMake 6391 "Branch Code" AssociationLevel 2979
AssociationType Type_Query AssociationRole Role_Source AssociationReferenced
"Branch Code"
```

AssociationUpdate

Updates an existing association.

UI Equivalent

Changing an association.

Syntax

AssociationUpdate *objassoc assocopts*

Argument	Description
AssociationUpdate *objAssoc*	Specifies the name of an existing association *objAssoc* that you want to update. *objAssoc* can be the object name, object identifier, or both.
assocopts	Specifies the type and details of the association *objAssoc*. For the complete list of options, see "assocopts" (p. 176).

Example

```
AssociationUpdate 6391 "Branch Code" AssociationLevel
2979 AssociationType Type_Query AssociationRole Role_Source AssociationReferenced
"Branch Code"
```

AutoDesign

Generates a preliminary model of dimensions and measures based on relationships that Transformer detects in the data source.

UI Equivalent

Run AutoDesign, New Model dialog box; or AutoDesign (Tools menu); or Run AutoDesign When Creating a New Model check box (Preferences, Autodesign tab).

Syntax

AutoDesign

Note

Autodesign requires that at least one data source and one column exist in the model.

CatAdd

Creates a category.

UI Equivalent

Clicking the right side of an existing category and dragging to the right to add a category.

Syntax

CatAdd *objCat* [**Dimension** *objDim*] [**Drill** *objDrillCat*] **Levels** *objLevel* **Parent** *objCat* [*catopts*] [**ParentList** *objCats* **EndList**]

Argument	Description
CatAdd *objCat*	Creates the category *objCat*. *objCat* must be the object name and can include the object identifier.
Dimension *objDim*	Specifies a dimension for the category. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Specifies a drill category for the category. objDrillCat can be the object name, object identifier, or both.
Levels *objLevel*	Specifies a level for the category. *objLevel* can be the object name or object identifier. This argument is mandatory.
Parent *objCat*	Specifies a parent category. *objCat* can be the object name or object identifier. This argument is mandatory.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "catopts" (p. 177).
ParentList *objCats*	Specifies parents for the category if the category is at a convergence level. *objCats* is a list of one or more parent categories, each identified by object name or object identifier.

Note

For more information about creating categories, see "CatMake" (p. 67).

Example

This example creates the category Back Packs:

```
CatAdd "Back Packs" Dimension "Products" Drill "By Product
Line" Levels "Product Type" Parent "Outdoor Products" SourceValue
"Back Packs" IsKeyOrphanage False IsTruncated False Blanks False
```

CatDelete

Deletes a category.

UI Equivalent

Delete Item (Edit menu), when a category is selected.

Syntax

CatDelete *objCat* [**Dimension** *objDim*]

Argument	Description
CatDelete *objCat*	Deletes the category *objCat*. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.

Note

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example deletes the category Germany.

```
CatDelete "Germany" Dimension "Locations"
```

CatJoin

Joins a category to the specified parent category.

UI Equivalent

None. This is done automatically.

Syntax

CatJoin *objCat* [**Dimension** *objDim*] **Parent** *objCat*

Argument	Description
CatJoin *objCat*	Specifies the category *objCat* that is to be joined. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Parent *objCat*	Specifies the parent that is to be joined. *objCat* can be the object name or object identifier. This argument is mandatory.

Note

CatJoin is required when using alternate drill down paths. It links a category in the convergence level with the parent in the alternate path.

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example joins the category 1014 to the parent Canada.

```
CatJoin "1014" Dimension "Channels" Parent "Canada"
```

CatMake

Creates a category or updates an existing one.

UI Equivalent

Modifying its property sheet, if the category exists. Clicking the right side of an existing category and dragging it to the right, for a new category.

Syntax

CatMake *objCat* [**Dimension** *objDim*] [**Drill** *objDrillCat*] [**Levels** *objLevel*] [**Parent** *objLevel*] [*catopts*] [**ParentList** *objLevels* **EndList**]

Argument	Description
CatMake *objCat*	Creates the category *objCat* or modifies it if it exists. *objCat* can be the object name, object identifier, or both. Include the object name if the category does not exist.

Argument	Description
Dimension *objDim*	When creating a category, this specifies its dimension. When updating a category, this is used, if necessary, to uniquely identify it. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	When creating a category, this specifies its drill category. This argument is not necessary when updating a category. *objDrillCat* can be the object name, object identifier, or both.
Levels *objLevel*	When creating a category, this specifies its level. This argument is not necessary when updating a category. *objLevel* can be the object name or object identifier.
Parent *objCat*	When creating a category, this specifies its parent category. This argument is not necessary when updating a category. *objCat* can be the object name, or object identifier.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "catopts" (p. 177). If the category exists, previously set options are retained unless you change them with this command.
ParentList*objCats*	When creating a category that is at a convergence level, this specifies its parents. When updating a category, this can be used to add one or more new parents. *objCats* is a list of one or more parent categories, each identified by object name or object identifier.

Note

Category names cannot contain a tilde (~) or at sign (@).

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example defines the category Back Packs.

```
CatMake "Back Packs" Dimension "Products" Drill "By Product
Line" Levels "Product Type" Parent "Outdoor Products" Lastuse 19971202
SourceValue "Back Packs" Filtered False Suppressed False Sign False
IsKeyOrphanage False IsTruncated False Blanks False
```

CatMorph

Changes the specified category's type.

UI Equivalent

None.

Syntax

CatMorph *objCat* [**Dimension** *objDim*] **CatType** *type*

Argument	Description
CatMorph *objCat*	Specifies the category *objCat* that is to be changed. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
CatType *type*	Specifies the category type. The type can be **Leaf**, **Special**, **Parent**, **Drill**, or **Root**. The type is mandatory.

Note

CatMorph exists so that MDL is a complete representation of a Transformer model when a category is changed. For example, when a leaf category acquires a child it metamorphoses (morphs for short) into a parent. This is an internal operation and when the model is saved as an .mdl file, the CatMorph statement does not appear in the file.

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

CatMoveVertical

Moves a category to a new position within a level.

UI Equivalent

Dragging and dropping a category in the category viewer from one location to another within a level.

Syntax

CatMoveVertical *objCat* [**Dimension** *objDim*] **Parent** *objLevel* **Sibling** *objCat*

Argument	Description
CatMoveVertical *objCat*	Specifies the category that is to be moved. *objCat* can be the category's object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Parent *objLevel*	Specifies the parent level of the category's new position. The parent must be specified even if it remains the same. *objLevel* can be the level's object name or object identifier.
Sibling *objCat*	Specifies the sibling category that becomes the next category below the moved category. *objCat* can be the category's object name or object identifier. This argument is mandatory.

Note

This does not change the category's object identifier. It affects the visual display of a category in the dimension viewer and its position in the .mdc file, which affects how it is viewed in PowerPlay or Analysis Studio.

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example moves the category Sunblock to the position above the category Tents.

```
CatMoveVertical "Sunblock" Dimension "Products" Parent
"Outdoor Products" Sibling "Tents"
```

CatUpdate

Updates an existing category.

UI Equivalent

Modifying the property sheet of a category.

Syntax

CatUpdate *objCat* [**Dimension** *objDim*] [*catopts*] [**ParentList** *objCats* **EndList**]

Argument	Description
CatUpdate *objCat*	Specifies the category to update. *objCat* can be the category's object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
catopts	Optional parameters that describe the category in greater detail. For the complete list of options, see "catopts" (p. 177). Previously set options are retained unless you change them with this command.
ParentList *objCats*	Specifies parents of the category. This can be used to add new parent(s) to the category. *objCats* is one or more parent categories, each identified by object name, or object identifier.

Note

For more information about updating categories, see "CatMake" (p. 67).

CatUpdateAll

Changes the Last Use value of all categories in the model to the current date.

UI Equivalent

None. The Last Used field is in the General tab of the property sheet for each category.

Syntax

CatUpdateAll

CleanHouse

Deletes categories with a Last Use date that is earlier than the date specified.

UI Equivalent

Clean House (Tools menu).

Syntax

CleanHouse [**Dimension** *objDim*] *date*

Argument	Description
Dimension *objDim*	Specifies a dimension within which the appropriate categories should be deleted. *objDim* can be the object name, object identifier, or both. If the dimension is not specified, all dimensions are affected.
date	Specifies the date, in format YYYYMMDD. To delete all categories, specify a future date.

Note

Each time a category is used, its Last Use value is set to the current system date. LastUse is a category option and can be found in CatMake statements in MDL or in the category's property sheet in the user interface.

CleanHouse can be issued only from the client.

Example

This example removes all categories in the Products dimension with a Last Use date less than 19980115 (January 15, 1998).

```
CleanHouse Dimension "Products" 19980115
```

ColumnAdd

Adds a column to a data source.

UI Equivalent

Insert Column (Edit menu).

Syntax

ColumnAdd *objCol* [**DataSource** *objDataSource*] **Origin** *origin* **Offset** *offset* [*colopts*]

Argument	Description
ColumnAdd *objCol*	Creates the column *objCol*. *objCol* must be the column's object name, and can include the object identifier.
DataSource *objDataSource*	Places the column within a data source. *objDataSource* can be the data source's object name, object identifier, or both.
Origin *origin*	Specifies the origin. *origin* must be one of **Source**, **Generated**, **Calculated**, or **Manual**. This argument is mandatory.
Offset *offset*	Specifies the amount by which the column is offset. *offset* can be 0 or greater. This argument is mandatory.

Argument	Description
colopts	Parameters that describe the column in greater detail. All are optional except **Column***objCol*, which is mandatory. Some options are set by default if you don't set them manually. For the complete list of options, see "colopts" (p. 188). For details about **Column**, see "Column" (p. 193).

Note

For more information about creating columns, see "ColumnMake" (p. 74).

The **DataSource** keyword replaced the **Query** keyword in Transformer 6.6 and subsequent versions.

Example

This example adds the Branch Code column to the model.

```
ColumnAdd "Branch Code" DataSource "All Staff Count (Excel)"
Origin Source Offset 1 Column "Branch Code"
```

ColumnDelete

Deletes a column.

UI Equivalent

Delete Item (Edit menu), when a column is selected.

Syntax

ColumnDelete *objCol* [**DataSource** *objDataSource*]

Argument	Description
ColumnDelete *objCol*	Specifies the column that is to be deleted. *objCol* must be the column's object name, and can include its object identifier.
DataSource *objDataSource*	Is specified, if necessary, to uniquely identify the column. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the dimension's object name, object identifier, or both.

Note

The **DataSource** keyword replaced the **Query** keyword in Transformer 6.6 and subsequent versions.

Example

This example deletes the column Branch Code.

```
ColumnDelete "Branch Code" DataSource "Locations (CSV)"
```

ColumnListUpdate

Re-orders the columns in a data source.

UI Equivalent

Dragging and dropping columns.

Syntax

ColumnListUpdate [**DataSource** *objDataSource*] **StartList** *objCols* **EndList**

Argument	Description
DataSource *objDataSource*	Specifies the data source to be re-ordered. *objDataSource* can be the data source's object name, object identifier, or both.
StartList *objCols*	Lists the columns in the desired order. If not all columns are listed, the listed ones move to the top of the list. The columns can be identified by object name or object identifier.

Note

The **DataSource** keyword replaced the **Query** keyword in Transformer 6.6 and subsequent versions.

Example

This example re-orders the columns in the data source All Staff Count (Excel) so that Branch Code is first, followed by Time and Staff Count.

```
ColumnListUpdate DataSource "All Staff Count (Excel)"
StartList "Branch Code" "Time" "Staff Count" EndList
```

ColumnMake

Creates a column or updates an existing one.

UI Equivalent

Modifying its property sheet, if the column exists. Insert Column (Edit menu), for a new column.

Syntax

ColumnMake *objCol* [**DataSource** *objDataSource*] **Origin** *origin* **Offset** *offset* [*colopts*]

Argument	Description
ColumnMake *objCol*	Creates the column *objCol* or modifies it if it exists. *objCol* can be the column's object name, object identifier, or both. Include the object name if the column does not exist.
DataSource *objDataSource*	When creating a column, this specifies the data source where it is to be placed. When updating a column this is used, if necessary, to uniquely identify it. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDataSource* can be the object name, object identifier, or both.
Origin *origin*	Specifies the origin. *origin* must be one of **Source**, **Generated**, **Calculated**, or **Manual**. This argument is mandatory.
Offset *offset*	Specifies the amount to which the column is offset. *offset* can be 0 or greater. This argument is mandatory.
colopts	Parameters that describe the column in greater detail. All are optional except **Column***objCol*, which is mandatory. Some options are set by default if you don't set them manually. For the complete list of options, see "colopts" (p. 188). For details about **Column**, see "Column" (p. 193). If the column exists, previously set options are retained unless you change them with this command.

Note

Column names cannot contain an at sign (@).

The **DataSource** keyword replaced the **Query** keyword in Transformer 6.6. and subsequent versions.

Example

This example defines the column Branch Code.

```
ColumnMake "Branch Code" DataSource "All Staff Count
(Excel)" Origin Source Offset 1 Column "Branch Code" Storage Default
Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off
```

ColumnUpdate

Updates an existing column.

UI Equivalent

Modifying the property sheet for a column.

Syntax

ColumnUpdate *objCol* [**DataSource** *objDataSource*] [*colopts*]

Argument	Description
ColumnUpdate *objCol*	Specifies the column to update. *objCol* can be the object name, object identifier, or both.
DataSource *objDataSource*	Is specified, if necessary, to uniquely identify the column. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDataSource* can be the object name, object identifier, or both.
colopts	Optional parameters that describe the column in greater detail. For the complete list of options, see "colopts" (p. 188). Previously set options are retained unless you change them with this command.

Note

For more information about updating columns, see "ColumnMake" (p. 74).

The **DataSource** keyword replaced the **Query** keyword in Transformer 6.6 and subsequent versions.

Example

This example changes the date format of the Time column to Year-Month, and changes the level of degree to Month.

```
ColumnUpdate "Time" DataSource "Salesrep Plan (Excel)"
Format YM DateLevel Month
```

CreateColumns

Creates columns for data sources.

UI Equivalent

None. Columns are created automatically in the user interface.

Syntax

CreateColumns *objQueries*

Argument	Description
CreateColumns *objQueries*	*objQueries* are one or more data sources for which columns are to be created. The list must be made up of object names or object identifiers.

Note

CreateColumns does not work when the associated data source contains the setting Columns True.

Example

This example creates columns for the data source Products (CSV).

```
CreateColumns "Products (CSV)"
```

CreateFiles

Creates an .mdc file for each PowerCube in the model.

UI Equivalent

Create PowerCube button in the toolbar; or Create PowerCubes (Run menu); or in Transformer on UNIX, Create Server PowerCubes (Server menu).

Syntax

CreateFiles [*platformtype*]

Argument	Description
platformtype	The platform type can be **OnClient** or **OnServer**.

Note

For other ways to create and update .mdc files, see "CreateFromCubes" (p. 77), "CreateFromQueries" (p. 78), and "UpdatePowerCubes" (p. 152).

Example

This example creates .mdc files for every cube in the model.

```
CreateFiles
```

CreateFromCubes

Creates an .mdc file for each specified PowerCube.

UI Equivalent

Create Selected PowerCubes (Run menu), when one or more PowerCubes are selected.

Syntax

CreateFromCubes [*platformtype*] [*objCubes*]

Argument	Description
platformtype	The platform type can be **OnClient** or **OnServer**. This setting is optional.

Argument	Description
objCubes	Specifies one or more PowerCubes for which .mdc files are to be created. PowerCubes can be identified by object name or object identifier.

Example

This example creates an .mdc file for the PowerCube cube1.

```
CreateFromCubes "cube1"
```

Note

For other ways to create and update .mdc files, see "CreateFiles" (p. 77), "CreateFromQueries" (p. 78), and "UpdatePowerCubes" (p. 152).

CreateFromQueries

Creates an .mdc file for each PowerCube in the model by retrieving data from only specific data sources.

UI Equivalent

None.

Syntax

CreateFromQueries [*platformtype*] [*objQueries*]

Argument	Description
platformtype	The platform type can be **OnClient** or **OnServer**. This setting is optional.
objQueries	Specifies one or more data sources that are used to create .mdc files for the PowerCubes. Data sources can be identified by object name or object identifier.

Note

If no data sources are specified, nothing happens.

If one or more of the specified data sources are not set for PowerCube Creation in the General tab of the Data Source property sheet, an error is issued. This occurs with structural data sources because they do not contain transanctional data.

For other ways to create and update .mdc files, see "CreateFromCubes" (p. 77), "CreateFiles" (p. 77), and "UpdatePowerCubes" (p. 152).

Example

This example creates .mdc files from the All Staff Count (Excel) data source.

```
CreateFromQueries "All Staff Count(Excel)"
```

CubeAdd

Creates a PowerCube. CubeAdd creates the PowerCube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 77).

UI Equivalent

Insert Item (Edit menu), when the PowerCubes pane is selected.

Syntax

CubeAdd *objCube* [*powercubeopts*] [**DimensionView** *objDimobjView*] [**MeasureInclude** *objMeasure meaopt*]

Argument	Description
CubeAdd *objCube*	Creates the PowerCube *objCube*. *objCube* must be the object name and can include the object identifier.
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "powercube-opts" (p. 224).
DimensionView *objDimobjView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`
MeasureInclude *objMeasuremeaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of **Yes**, **Indirect**, or **No**. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Note

For more information about creating PowerCubes, see "CubeMake" (p. 89).

Example

This example creates The Great Outdoors Co. Sales PowerCube object.

```
CubeAdd "The Great Outdoors Co. Sales" MdcFile "c:\outdoors.mdc"
```

CubeDelete

Deletes a PowerCube. CubeDelete deletes the PowerCube object in the Transformer model. It does not delete the .mdc file.

UI Equivalent

Delete Item (Edit menu), when a cube is selected.

Syntax

CubeDelete *objCube*

Argument	Description
CubeDelete *objCube*	Specifies the PowerCube to be deleted. *objCube* can be the object name, object identifier, or both.

Note

CubeDelete deletes only the PowerCube object in Transformer.

Example

This example deletes the PowerCube called The Great Outdoors Co. Sales.

```
CubeDelete "The Great Outdoors Co. Sales"
```

CubeGroupAdd

Creates a PowerCubeGroup. CubeGroupAdd creates the PowerCube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 77).

UI Equivalent

Insert Item (Edit menu), when the PowerCubes pane is selected and providing a dimension and level in the Cube Group tab.

Syntax

CubeGroupAdd *objCubeGroup* [*powercubeopts*] [**DimensionView** *objDim objView*] [**MeasureInclude** *objMeasure meaopt*]

Argument	Description
CubeGroupAdd *objCubeGroup*	Specifies the cube group that is to be created. *objCubeGroup* must be the cube group's object name, and can also be the object identifier.
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "powercubeopts" (p. 224).
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of **Yes**, **Indirect** or **No**. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Note

If you do not include the options SegmenterLevel and SegmenterDimension, which are options in powercubeopts, this creates a PowerCube and not a PowerCubeGroupCube. For more information, see "SegmenterLevel" and "SegmenterDimension" (p. 231).

For more information about updating cube groups, see "CubeGroupMake" (p. 86).

Example

This example creates the cube group cubegroup1. Cubes within this group are created from the Products dimension and Product Line level.

```
CubeGroupAdd "cubegroup1" MdcFile "c:\temp\outdoors1.mdc"
SegmenterDimension "Products" SegmenterLevel "Product Line"
```

CubeGroupCubeAdd

Creates a PowerCube for a PowerCubeGroup. CubeGroupCubeAdd creates the PowerCube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 77).

UI Equivalent

Insert Item (Edit menu), when the PowerCubes pane is selected and providing a dimension and level in the Cube Group tab.

Syntax

CubeGroupCubeAdd *objCubeGroupCube* **CubeGroup** *objCubeGroup* [**Dimension** *objDim*] **Code** *objCat* [*powercubeopts*]

Argument	Description
CubeGroupCubeAdd *objCubeGroup*	Creates the PowerCube group cube *objCubeGroup-Cube.objCubeGroupCube* must be the object name and can include the object identifier.
CubeGroup *objCubeGroup*	Specifies a cube group for the PowerCube. *objCubeGroup* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Code *objCat*	Specifies the category on which the PowerCube is based. *objCat* can be the object name, object identifier, or both. This argument is mandatory.
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "powercube-opts" (p. 224).

Note

In the user interface, a PowerCubeGroupCube is created for every category under the drill category in the dimension (unless some have been filtered, cloaked, or suppressed). In MDL, a separate statement must be included for every category under the drill category.

For more information about creating cubes in cube groups, see "CubeGroupCubeMake" (p. 84).

Example

This example creates the Outdoor Products cube within the cube group cubegroup1, based on the Outdoor Products category.

```
CubeGroupCubeAdd "Outdoor Products" CubeGroup "cubegroup1"
Code "Outdoor Products"
```

CubeGroupCubeDelete

Deletes a PowerCube from a PowerCubeGroupCube. CubeGroupCubeDelete deletes the PowerCube object in the Transformer model. It does not delete the .mdc file.

UI Equivalent

Delete Item (Edit menu), when a PowerCube is selected from a cube group.

This can be done only after a dimension view is created that cloaks, suppresses or filters the category at the apex of the cube.

Syntax

CubeGroupCubeDelete *objCubeGroupCube* [**CubeGroup** *objCubeGroup*]

Argument	Description
CubeGroupCubeDelete *objCubeGroupCube*	Deletes the cube group *objCubeGroupCube*. *objCubeGroupCube* can be the object name, object identifier, or both.
CubeGroup *objCubeGroup*	Specifies the cube group in which the PowerCube is located. *objCubeGroup* can be the object name, object identifier, or both.

Note

A PowerCubeGroupCube can be deleted only after it is suppressed in the dimension view that underlies the cube group. This means that you must first create a dimension view that cloaks, suppresses, or filters the category that defines the PowerCubeGroupCube.

Example

This example deletes the Environmental Line cube from cubegroup1. Before the deletion, a dimension view is created to suppress the category.

```
ViewMake "Enviro. Line" Dimension "Products" ViewUserClass
0 Apex
"Product Line" Suppressed "Environmental Line"
CubeGroupCubeDelete "Environmental Line" CubeGroup "cubegroup1"
```

CubeGroupCubeListUpdate

Re-orders the PowerCubes in a PowerCubeGroup.

UI Equivalent

Dragging and dropping PowerCubes within a cube group.

Syntax

CubeGroupCubeListUpdate CubeGroup *objCubeGroup* [**StartList** *objCubeGroups* **EndList**]

Argument	Description
CubeGroupCubeListUpdate CubeGroup *objCubeGroup*	Specifies the cube group. *objCubeGroup* can be the object name, object identifier, or both.
StartList *objCubeGroupCubes*	Specifies the cube group cubes in the desired order. Each cube can be identified by object name or object identifier. If no cubes are specified, nothing happens. If some of the cubes are specified, they move to the top of the list.

Example

This example places the cubes GO Sport Line and Outdoor Products at the top of the list of cubes in cubegroup1.

```
CubeGroupCubeListUpdate CubeGroup "cubegroup1" StartList
"GO Sport Line" "Outdoor Products" EndList
```

CubeGroupCubeMake

Creates a PowerCube in a PowerCubeGroup or updates it if it exists. CubeGroupCubeMake defines the PowerCube group cube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 77).

UI Equivalent

Modifying its property sheet, if the cube group cube exists. Insert Item (Edit menu), when the PowerCubes pane is selected and providing a dimension and level in the Cube Group tab, for a new cube group cube.

Syntax

CubeGroupCubeMake *objCubeGroupCube* [**CubeGroup** *objCubeGroup*] [**Dimension** *objDim*] **Code** *objCat* [*powercubeopts*]

Argument	Description
CubeGroupCubeMake *objCubeGroupCube*	Creates the PowerCube group cube *objCubeGroupCube* or modifies it if it exists. *objCubeGroupCube* can be the object name, object identifier, or both. Include the object name if the cube group cube does not exist.
CubeGroup *objCubeGroup*	When creating a cube in a cube group, this specifies the cube group. *objCubeGroup* can be the object name, object identifier, or both.

Argument	Description
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Code *objCat*	Specifies the category on which the cube is based. *objCat* can be the object name, object identifier, or both.
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "powercube-opts" (p. 224).

Note

The cube group must exist before creating a PowerCube for the group.

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example defines the PowerCube group cube Outdoor Products.

```
CubeGroupCubeMake 6343 "Outdoor Products" CubeGroup "cubegroup1"
Code 4807 Status OK CubeCreation Default Optimize Default Compress
False IncrementalUpdate False ServerCube False CubeStamp 886518687
CubeCycle 24 DrillThrough False EndList
```

CubeGroupCubeUpdate

Updates an existing PowerCube in a PowerCubeGroup. CubeGroupCubeUpdate modifies the PowerCube object in the Transformer model. It does not modify existing .mdc or files.

UI Equivalent

Modifying the property sheet for a cube group cube.

Syntax

CubeGroupCubeUpdate *objCubeGroupCube* [**CubeGroup** *objCubeGroup*] [*powercubeopts*]

Argument	Description
CubeGroupCubeUpdate *objCube-GroupCube*	Specifies the cube group cube to update. *objCubeGroup-Cube* can be the object name, object identifier, or both.
CubeGroup *objCubeGroup*	Specifies the cube group. *objCubeGroup* can be the object name, object identifier, or both.

Argument	Description
powercubeopts	Optional parameters that describe the cube in greater detail. For the complete list of options, see "powercube-opts" (p. 224). Previously set options are retained unless you change them with this command.

Note

For more information about updating cubes in cube groups, see "CubeGroupCubeMake" (p. 84).

Example

This example modifies the cube Outdoor Products from cubegroup1 by enabling drill-through.

```
CubeGroupCubeUpdate "Outdoor Products" CubeGroup "cubegroup1"
DrillThrough True EndList
```

CubeGroupDelete

Deletes a PowerCubeGroup.CubeGroupDelete deletes the PowerCube object in the Transformer model. It does not delete the .mdc file.

UI Equivalent

Delete Item (Edit menu), when a cube group is selected.

Syntax

CubeGroupDelete [*objCubeGroup*]

Argument	Description
CubeGroupDelete *objCubeGroup*	Specifies the cube group that is to be deleted. *objCubeGroup* can be the object name, object identifier, or both.

Example

This example deletes the PowerCube cubegroup1.

```
CubeGroupDelete "cubegroup1"
```

CubeGroupMake

Creates a PowerCubeGroup or updates an existing one. CubeGroupMake defines the PowerCube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 77).

UI Equivalent

Modifying its property sheet, if the cube group exists. Insert Item (Edit menu), when the PowerCubes pane is selected and providing a dimension and level in the Cube Group tab, for a new cube group.

Syntax

CubeGroupMake *objCubeGroup* [*powercubeopts*] [**DimensionView** *objDim objView*] [**MeasureInclude** *objMeasure meaopt*]

Argument	Description
CubeGroupMake *objCubeGroup*	Creates the cube group *objCubeGroup* or modifies it if it exists. *objCubeGroup* can be the object name, object identifier, or both. Include the object name if the category does not exist.
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "powercubeopts" (p. 224). If the cube exists, previously set options are retained unless you change them with this command.
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of **Yes**, **Indirect**, or **No**. Repeat this argument to specify multiple measures, as in `MeasureInclude "Cost" Yes MeasureInclude "Revenue" No`

Note

If you do not include the options SegmenterLevel and SegmenterDimension, which are options in powercubeopts, this will create a PowerCube and not a PowerCubeGroupCube. For more information, see "SegmenterLevel" and "SegmenterDimension" (p. 231).

Example

This example defines the cube group cubegroup1. The cubes within this group are created from the Products dimension, Product Line level and focus type Product Type.

```
CubeGroupMake "cubegroup1" MdcFile "c:\outdoors1.mdc"
CubeCreation On Optimize Default Compress False DatabaseInfo "Local;;"
IncrementalUpdate False ServerCube False DrillThrough False EndList
SegmenterDimension "Products" SegmenterLevel "Product Line" DetailLevel
"Product Type" DimensionView "Years" "All Categories" DimensionView
"Products" "All Categories" DimensionView "Locations" "All Categories"
DimensionView "Channels" "All Categories" DimensionView "Margin
Ranges" "All Categories" MeasureInclude "Revenue" Yes MeasureInclude
```

```
"Product Cost" Yes MeasureInclude "Product Plan" Yes MeasureInclude
"Expense Plan" Yes MeasureInclude "Quantity Sold" Yes MeasureInclude
"Profit Margin %" Yes MeasureInclude "Revenue/Employee" Yes MeasureInclude
"SalesRep Plan" Yes MeasureInclude "Staff Count" Yes MeasureInclude
"SalesRep Count" Yes
```

CubeGroupUpdate

Updates an existing PowerCubeGroup. CubeGroupUpdate modifies the PowerCube object in the Transformer model. It does not affect existing .mdc files.

UI Equivalent

Modifying the property sheet for a cube group.

Syntax

CubeGroupUpdate *objCubeGroup* [*powercubeopts*] [**DimensionView** *objDim objView*] [**MeasureInclude** *objMeasure meaopt*]

Argument	Description
CubeGroupUpdate *objCubeGroup*	Specifies the PowerCube group to update. *objCubeGroup* can be the object name, object identifier, or both.
powercubeopts	Optional parameters that describe the cube in greater detail. For the complete list of options, see "powercube-opts" (p. 224). Previously set options are retained unless you change them with this command.
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of **Yes**, **Indirect**, or **No**. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Note

For more information about updating cube groups, see "CubeGroupMake" (p. 86).

Example

This example modifies Cubegroup1 by changing the dimension view from Locations to Omit Dimension and by excluding the measure Product Cost from the view.

```
CubeGroupUpdate "Cubegroup1" DimensionView "Locations"
"Omit Dimension" MeasureInclude "Product Cost" Indirect
```

CubeMake

Creates a PowerCube or updates an existing one. CubeMake defines the PowerCube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 77).

UI Equivalent

Modifying its property sheet, if the cube exists. When the PowerCubes pane is selected, Insert Item (Edit menu), for a new PowerCube.

Syntax

CubeMake *objCube* [*powercubeopts*] [**DimensionView** *objDim objView*] [**MeasureInclude** *objMeasure meaopt*]

Argument	Description
CubeMake *objCube*	Creates the PowerCube *objCube* or modifies it if it exists. *objCube* can be the object name, object identifier, or both. Include the object name if the category does not exist.
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "powercubeopts" (p. 224). If the cube exists, previously set options are retained unless you change them with this command.
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. meaopt must be one of **Yes**, **Indirect**, or **No**. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Example

This example defines the PowerCube The Great Outdoors Co. Sales.

```
CubeMake "The Great Outdoors Co. Sales" MdcFile "c:\outdoors.mdc"
Status OK CubeCreation On Optimize Categories Compress False DatabaseInfo
"Local;;" IncrementalUpdate False ServerCube False CubeStamp 886096730
CubeCycle 6 DrillThrough False EndList DimensionView "Years" "All
Categories" DimensionView "Products" "All Categories" DimensionView
"Locations" "All Categories" DimensionView "Channels" "All Categories"
DimensionView "Margin Ranges" "All Categories" MeasureInclude "Revenue"
Yes MeasureInclude "Product Cost" Yes MeasureInclude "Product Plan"
Yes MeasureInclude "Expense Plan" Yes MeasureInclude "Quantity Sold"
Yes MeasureInclude "Profit Margin %" Yes MeasureInclude "Revenue/Employee"
Yes MeasureInclude "SalesRep Plan" Yes MeasureInclude "Staff Count"
Yes MeasureInclude "SalesRep Count" Yes
```

CubeUpdate

Updates an existing PowerCube. CubeUpdate modifies the PowerCube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 77).

UI Equivalent

Modifying the property sheet of a PowerCube.

Syntax

CubeUpdate *objCube* [*powercubeopts*] [**DimensionView** *objDim objView*] [**MeasureInclude** *objMeasure meaopt*]

Argument	Description
CubeUpdate *objCube*	Specifies the PowerCube to update. *objCube* can be the object name, object identifier, or both.
powercubeopts	Optional parameters that describe the cube in greater detail. For the complete list of options, see "powercube-opts" (p. 224). If the cube exists, previously set options are retained unless you change them with this command.
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`

Argument	Description
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of **Yes**, **Indirect**, or **No**. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Note

For more information about updating PowerCubes, see "CubeMake" (p. 89).

Example

This example modifies the PowerCube The Great Outdoors Co. Sales by enabling drill-through, changing the dimension view to Omit Dimension and excluding the revenue measure.

```
CubeUpdate "The Great Outdoors Co. Sales" MdcFile "c:\outdoors.mdc" DrillThrough
True EndList DimensionView "Products" "Omit Dimension" MeasureInclude
"Revenue" Indirect
```

CurrencyAdd

Adds a currency to the Currency Table.

UI Equivalent

Right-clicking a currency in the Currency Table property sheet and choosing Add New Currency.

Syntax

CurrencyAdd *objCurrency* [**Dimension** *objDim*] [**Drill** *objDrillCat*] **Levels** *objLevel* [*currencyrecordopts*] **CurrencyRateList EffectiveDate** {*objCat*|0} **ConversionRate** *rate* [**RateIsBySource** {**True**|**False**}] [**CurrencyTableType** *tabletype*] **EndList**

Argument	Description
CurrencyAdd *objCurrency*	Adds the currency *objCurrency* to the list of currencies in the Currency Table. *objCurrency* must be the object name and can include the object identifier.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level *objLevel*. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.

Argument	Description
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the date level *objLevel*. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDrillCat* can be the drill category object name, object identifier, or both.
Levels *objLevel*	Specifies the level of the date category *objCat*. This indicates the granularity of the data (daily, monthly, etc.). *objLevel* can be the object name or object identifier. The level is mandatory.
currencyrecordopts	Optional parameters that describe the currency in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "currencyrecordopts" (p. 197).
CurrencyRateList	Indicates the start of conversion rates.
EffectiveDate {*objCat*|0}	Specifies conversion rates for the currency. This argument is repeated for each time period that has a conversion rate. Each conversion rate specification starts with the keyword **EffectiveDate**. *objCat* indicates the date for each conversion rate; it is the date category and can be specified by object name, object identifier, or both. Zero (0) indicates that the currency has joined the EMU.
ConversionRate *rate*	*rate* is the exchange rate for that date. It can include decimals; for example, 3.42556.
RateIsBySource {**True**|**False**}	**RateIsBySource** indicates whether the conversion rate is sourced from an external data source or was manually entered. The default is **False**.
CurrencyTableType *tabletype*	**CurrencyTableType** specifies the table type that applies to this time period's currency record. *tabletype* can be one of **BaseTable**, **EuroTable**, or **OtherTable**. The default is **BaseTable** unless **EffectiveDate** is followed by zero (0), in which case the default is **EuroTable**.

Note

The **RateIsBySource** keyword replaced the **RateIsByQuery** keyword in Transformer 6.6 and subsequent versions.

Example

This example adds a new currency to the Currency Table, and specifies conversion rates for three months.

```
CurrencyAdd "C$" Dimension "Years" Levels "Month" CountryCode
"Can" CurrencyRateList EffectiveDate "199601" ConversionRate 1.2
RateIsBySource False EffectiveDate "199602" ConversionRate 1.75
RateIsBySource False EffectiveDate "199603" ConversionRate 1.55
RateIsBySource False EndList
```

CurrencyDelete

Deletes a currency from the Currency Table.

UI Equivalent

Removing a currency from the Currency Table.

Syntax

CurrencyDelete *objCurrency*

Argument	Description
CurrencyDelete *objCurrency*	Deletes the Currency Table *objCurrency*. *objCurrency* can be the object name, object identifier, or both.

Example

This example deletes the Canadian Dollars currency from the Currency Table.

```
CurrencyDelete "C$"
```

CurrencyMake

Adds a currency to the Currency Table or updates an existing one.

UI Equivalent

Modifying the property sheet, if the currency exists. Right-clicking a currency in the Currency Table property sheet and choosing Add New Currency, for a new currency.

Syntax

CurrencyMake *objCurrency* [**Dimension** *objDim*] [**Drill** *objDrillCat*] **Levels** *objLevel* [*currencyrecordopts*] **CurrencyRateList EffectiveDate** {*objCat*|0} **ConversionRate** *rate* [**RateIsBySource** {**True**|**False**}] [**CurrencyTableType** *tabletype*] **EndList**

Argument	Description
CurrencyMake *objCurrency*	Adds the currency *objCurrency* to the list of currencies in the Currency Table or modifies it if it exists. *objCurrency* can be the object name, object identifier, or both. Include the object name if the currency does not exist.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level *objLevel*. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the date level *objLevel*. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDrillCat* can be the drill category object name, object identifier, or both.
Levels *objLevel*	Specifies the level of the date category *objCat*. This indicates the granularity of the data (daily, monthly, and so on). *objLevel* can be the object name or object identifier. The level is mandatory.
currencyrecordopts	Optional parameters that describe the currency in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "currencyrecordopts" (p. 197). If the currency exists, previously set options are retained unless you change them with this command.
CurrencyRateList	Indicates the start of conversion rates.
EffectiveDate {*objCat*|0}	Specifies conversion rates for the currency. This argument is repeated for each time period that has a conversion rate. Each conversion rate specification starts with the keyword **EffectiveDate**. *objCat* indicates the date for each conversion rate; it is the date category and can be specified by object name, object identifier, or both. Zero (0) indicates that the currency has joined the EMU.
ConversionRate *rate*	*rate* is the exchange rate for that date. It can include decimals; for example, 3.42556.

Argument	Description
RateIsBySource {**True**|**False**}	**RateIsBySource** indicates whether each rate is sourced from the external data source or was manually entered. The default is **False**.
CurrencyTableType *tabletype*	**CurrencyTableType** specifies the table type that applies to this time period's currency record. *tabletype* can be one of **BaseTable**, **EuroTable**, or **OtherTable**. The default is **BaseTable** unless **EffectiveDate** is followed by zero (0), in which case the default is **EuroTable**.

Note

The **RateIsBySource** keyword replaced the **RateIsByQuery** keyword in Transformer 6.6 and subsequent versions.

A Currency Table must exist before creating currencies. To create a Currency Table, see "CurrencyTableMake" (p. 97).

Currency conversions will only be applied to measures if the measure definition contains the option IsCurrency True. For more information, see "IsCurrency" (p. 220).

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example defines a Canadian currency for the Currency Table. It provides conversion rates for 1996 only, so the rates for 1997 are set to the default, 1.00000.

```
CurrencyMake "Canadian Dollars" Dimension
"Years" Drill "By Time" Levels "Month" CountryCode "CAN" CurrencyCountryLabel
"Canada" CurrencyFormatOverride False CurrencySymbol "$" CurrencyDecimals
2 CurrencyRateList EffectiveDate "19960101-19960131" ConversionRate
1.36545  RateIsBySource False CurrencyTableType BaseTable EffectiveDate
"19960201-19960229"
ConversionRate 1.37524  RateIsBySource False CurrencyTableType BaseTable
EffectiveDate "19960301-19960331" ConversionRate 1.36476  RateIsBySource
False CurrencyTableType BaseTable EffectiveDate "19960401-19960430"
ConversionRate 1.35955  RateIsBySource False CurrencyTableType BaseTable
EffectiveDate "19960501-19960531" ConversionRate 1.36957  RateIsBySource
False CurrencyTableType BaseTable EffectiveDate "19960601-19960630"
ConversionRate 1.367  RateIsBySource False CurrencyTableType BaseTable
EffectiveDate "19960701-19960731" ConversionRate 1.36826  RateIsBySource
False CurrencyTableType BaseTable EffectiveDate "19960801-19960831"
ConversionRate 1.37095  RateIsBySource False CurrencyTableType BaseTable
EffectiveDate "19960901-19960930" ConversionRate 1.369  RateIsBySource
False CurrencyTableType BaseTable EffectiveDate "19961001-19961031"
ConversionRate 1.34955  RateIsBySource False CurrencyTableType BaseTable
EffectiveDate "19961101-19961130" ConversionRate 1.3375  RateIsBySource
False CurrencyTableType BaseTable EffectiveDate "19961201-19961231"
ConversionRate 1.3635  RateIsBySource False CurrencyTableType BaseTable
```

CurrencyTableAdd

Creates the Currency Table.

UI Equivalent

Currency Table (File menu).

Syntax

CurrencyTableAdd [*objCurrencyTable*] [*currencytableopts*] **Associations** *assocopts*

Argument	Description
CurrencyTableAdd *objCurrencyTable*	Creates the Currency Table *objCurrencyTable*. *objCurrencyTable* can be the object name, object identifier, or both. Include the name for clarity in your model.
currencytableopts	Optional parameters that specify columns in an external data source. For the complete list of options, see "currencytableopts" (p. 199). If the currency exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Required parameters that specify associations for the currency table. For the complete list of options, see "assocopts" (p. 176).

Note

If you specify the option CurrencyTableType, the object name of the Currency Table is optional. For more information, see "CurrencyTableType" (p. 201).

For more information about creating Currency Tables, see "CurrencyTableMake" (p. 97).

Example

This example creates the base Currency Table.

```
CurrencyTableAdd 869 "CurrencyBaseTable" CurrencyTableType
BaseTable

Associations 1315 "Country Code" AssociationType Type_Query
AssociationRole Role_CountryCode

AssociationReferenced "Country Code"

Associations 1317 "Date" AssociationType Type_Query AssociationRole
Role_Date

AssociationReferenced "Date"

Associations 1319 "Currency" AssociationType Type_Query
AssociationRole Role_Label

AssociationReferenced "Currency"

Associations 1321 "Conversion Rate" AssociationType Type_Query AssociationRole
Role_Rate
```

```
AssociationReferenced "Conversion Rate"
```

CurrencyTableDelete

Deletes an existing Currency Table.

UI Equivalent

Remove Currency Table (File menu).

Syntax

CurrencyTableDelete *objCurrencyTable*

Argument	Description
CurrencyTableDelete *objCurrencyTable*	Deletes the Currency Table *objCurrencyTable*. *objCurrencyTable* can be the object name, object identifier, or both.

Example

This example deletes the base Currency Table.

```
CurrencyTableDelete "CurrencyBaseTable"
```

CurrencyTableMake

Creates a Currency Table or updates an existing one.

UI Equivalent

Modifying the property sheet of a Currency Table.

Syntax

CurrencyTableAdd [*objCurrencyTable*] [*currencytableopts*] **Associations** *assocopts*

Argument	Description
CurrencyTableMake *objCurrencyTable*	Creates the Currency Table *objCurrencyTable* or modifies it if it exists. *objCurrencyTable* can be the object name, object identifier, or both. Include the object name if the Currency Table does not exist.
currencytableopts	Optional parameters that specify columns in an external data source. For the complete list of options, see "currencytableopts" (p. 199). If the currency exists, previously set options are retained unless you change them with this command.

Argument	Description
Associations *assocopts*	Optional parameters that specify associations for the currency table. For the complete list of options, see "assocopts" (p. 176).

Note

In the user interface, the Currency Table has a single property sheet, even when Euro triangulation is in effect. In MDL, the base Currency Table and Euro table appear as two separate objects. Each has its own CurrencyTableMake statement.

The default object names for the two types of Currency Table are Base Table and Euro Table. You can specify a different object name, but in the user interface the default will always appear.

If you specify the option CurrencyTableType , the object name of the Currency Table is optional. For more information, see "CurrencyTableType" (p. 201).

Example

This example defines a Currency Table.

```
CurrencyTableMake 869 "CurrencyBaseTable" CurrencyTableType
BaseTable

Associations 1315 "Country Code" AssociationType Type_Query
AssociationRole Role_CountryCode

AssociationReferenced "Country Code"

Associations 1317 "Date" AssociationType Type_Query AssociationRole
Role_Date

AssociationReferenced "Date"

Associations 1319 "Currency" AssociationType Type_Query
AssociationRole Role_Label

AssociationReferenced "Currency"

Associations 1321 "Conversion Rate" AssociationType Type_Query AssociationRole
Role_Rate

AssociationReferenced "Conversion Rate"
```

CurrencyTableUpdate

Updates an existing Currency Table.

UI Equivalent

Modifying the property sheet of the Currency Table.

Syntax

CurrencyTableUpdate [*objCurrencyTable*] [*currencyopts*] **Associations** *assocopts*

Argument	Description
CurrencyTableUpdate *objCurrencyTable*	Updates the Currency Table *objCurrencyTable*. *objCurrencyTable* can be the object name, object identifier, or both.
currencytableopts	Optional parameters that specify columns in an external data source. For the complete list of options, see "currencytableopts" (p. 199). If the currency exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Optional parameters that specify associations for the currency table. For the complete list of options, see "assocopts" (p. 176).

Note

If you specify the option CurrencyTableType, the object name of the Currency Table is optional. For more information, see "CurrencyTableType" (p. 201).

For more information about updating Currency Tables, see "CurrencyTableMake" (p. 97).

Example

This example modifies the Currency Table by changing its data source, which requires specifying different columns.

```
CurrencyTableMake 869 "Currency Base Table" CurrencyTableType
BaseTable

Associations 1315 "Country Code" AssociationType Type_Query
AssociationRole Role_CountryCode

AssociationReferenced "Country Code"

Associations 1317 "Date" AssociationType Type_Query AssociationRole
Role_Date

AssociationReferenced "Date"

Associations 1319 "Currency" AssociationType Type_Query
AssociationRole Role_Label

AssociationReferenced "Currency"

Associations 1321 "Conversion Rate" AssociationType Type_Query AssociationRole
Role_Rate

AssociationReferenced "Conversion Rate"
```

CurrencyUpdate

Updates an existing currency in a Currency Table.

UI Equivalent

Modifying the currency's property sheet.

Syntax

CurrencyUpdate *objCurrency* [**Dimension** *objDim*] [**Drill** *objDrillCat*] **Levels** *objLevel* [*currencyrecordopts*] [**CurrencyRateList EffectiveDate** {*objCat*|0} **ConversionRate** *rate* [**RateIsBySource** {**True**|**False**}] [**CurrencyTableType** *tabletype*] **EndList**

Argument	Description
CurrencyUpdate *objCurrency*	Modifies the currency *objCurrency*. *objCurrency* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level *objLevel*. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the date level *objLevel*. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDrillCat* can be the drill category object name, object identifier, or both.
Levels *objLevel*	Specifies the level of the date category *objCat*. This indicates the granularity of the data (daily, monthly, and so on). *objLevel* can be the object name or object identifier. The level is mandatory.
currencyrecordopts	Optional parameters that describe the currency in greater detail. For the complete list of options, see "currencyrecordopts" (p. 197). Previously set options are retained unless you change them with this command.
CurrencyRateList	Indicates the start of conversion rates.
EffectiveDate {*objCat*|0}	Specifies conversion rates for the currency. This argument is repeated for each time period that has a conversion rate. Each conversion rate specification starts with the keyword **EffectiveDate**. *objCat* indicates the date for each conversion rate; it is the date category and can be specified by object name, object identifier, or both. Zero (0) indicates that the currency has joined the EMU.
ConversionRate *rate*	*rate* is the exchange rate for that date. It can include decimals; for example, 3.42556.

Argument	Description
[**RateIsBySource** {**True**|**False**}]	**RateIsBySource** indicates whether each rate is sourced from the external data source or was manually entered. The default is **False**.
CurrencyTableType *tabletype*	**CurrencyTableType** specifies the table type that applies to this time period's currency record. *tabletype* can be one of **BaseTable**, **EuroTable**, or **OtherTable**. The default is **BaseTable** unless **EffectiveDate** is followed by zero (0), in which case the default is **EuroTable**.

Note

The **RateIsBySource** keyword replaced the **RateIsByQuery** keyword in Transformer 6.6 and subsequent versions.

For more information about updating currencies, see "CurrencyMake" (p. 93).

Example

This example modifies the conversion rate for Canadian Dollars for December 1997.

```
CurrencyUpdate  "Canadian Dollars" CurrencyRateList EffectiveDate
"19971201-19971231"
ConversionRate 1.55 EndList
```

DataSourceAdd

Creates a data source.

UI Equivalent

Insert Item (Edit menu), when a data source is selected.

Syntax

DataSourceAdd *objDataSource* [*appqueryopts*] [**OrgName** *objCol* **Origin** *origin* **Offset** *offset*] [*colopts*]

Argument	Description
DataSourceAdd *objDataSource*	Creates the data source *objDataSource*. *objDataSource* must be the object name and can include the object identifier.
appqueryopts	Optional parameters that describe the data source in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "appqueryopts" (p. 165).

Argument	Description
OrgName *objCol* **Origin** *origin* **Offset** *offset*	Specifies the original name of the column, the origin and the offset. origin must be one of **Source**, **Generated**, **Calculated**, or **Manual**. *offset* can be 0 or greater. This allows you to take the structured definition of the column and append it to the data source definition. For more information on structured format, see "Structured MDL".
colopts	Optional parameters that describe the columns in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "colopts" (p. 188).

Note

For more information about creating data sources, see "DataSourceMake" (p. 103).

The **DataSourceAdd** keyword replaced the **QueryAdd** keyword in Transformer 6.6 and subsequent versions.

Example

This example adds the data source Products (CSV) to the model.

```
DataSourceAdd "Products (CSV)" Source "c:\prodinfo.csv"
SourceType FlatFileColNames
```

DataSourceDelete

Deletes a data source.

UI Equivalent

Delete Item (Edit menu), when a data source is selected.

Syntax

DataSourceDelete *objDataSource*

Argument	Description
DataSourceDelete *objDataSource*	Deletes the data source *objDataSource*. *objDataSource* can be the object name, object identifier, or both.

Note

The **DataSourceDelete** keyword replaced the **QueryDelete** keyword in Transformer 6.6 and subsequent versions.

Example

This example deletes the All Staff Count (Excel) data source.

```
DataSourceDelete "All Staff Count (Excel)"
```

DataSourceMake

Creates a data source or updates an existing one.

UI Equivalent

Modifying the property sheet, if the data source exists. When a data source is selected, Insert Item (Edit menu), for a new data source.

Syntax

DataSourceMake *objDataSource* [*appqueryopts*] [**OrgName** *objCol* **Origin** *origin* **Offset** *offset*] [*colopts*]

Argument	Description
DataSourceMake *objDataSource*	Creates the data source *objDataSource* or modifies it if it exists. *objDataSource* can be the object name, object identifier, or both. Include the object name if the category does not exist.
appqueryopts	Optional parameters that describe the data source in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "appqueryopts" (p. 165). If the data source exists, previously set options are retained unless you change them with this command.
OrgName *objCol* **Origin** *origin* **Offset** *offset*	Specifies the original name of the column, the origin, and the offset. *origin* must be one of **Source**, **Generated**, **Calculated**, or **Manual**. *offset* can be 0 or greater. This allows you to take the structured definition of the column and append it to the data source definition. For more information on structured format, see "Structured MDL" (p. 241).
colopts	Optional parameters that can describe the columns in greater detail. For the complete list of options, see "colopts" (p. 188).

Note

The **DataSourceMake** keyword replaced the **QueryMake** keyword in Transformer 6.6 and subsequent versions.

Example

This example defines the data source All Staff Count.

```
DataSourceMake 103 "Products (CSV)" Separator
"," SourceType FlatFile_ColNames CharacterSet Multibyte DecimalSep
"." Thousandsep "," Columns False Timing PopYesCreateDefault Source
"prodinfo.csv"
```

DataSourceUpdate

Updates an existing data source.

UI Equivalent

Modifying the property sheet for a data source.

Syntax

DataSourceUpdate *objDataSource* [*appqueryopts*]

Argument	Description
DataSourceUpdate *objDataSource*	Specifies the data source to update. *objDataSource* can be the object name, object identifier, or both.
appqueryopts	Optional parameters that describe the data source in greater detail. For the complete list of options, see "appquery-opts" (p. 165). Previously set options are retained unless you change them with this command.

Note

For another way to update data sources, see "DataSourceMake" (p. 103).

The **DataSourceUpdate** keyword replaced the **QueryUpdate** keyword in Transformer 6.6 and subsequent versions.

Example

This example changes the source file for the data source All Staff Count (Excel) to bigstaff.asc, and changes the file format to ASCII.

```
DataSourceUpdate "All Staff Count (Excel)" Source "c:
<installation_directory>
\bigstaff.asc" SourceType FlatFile_ColNames
```

DeletionListUpdate

Deletes objects from a model.

UI Equivalent

Deleting an object from a model using incremental update, or deleting an object in the client version of a model, then synchronizing it with the server version.

Syntax

DeletionListUpdate *deletionsopts* **Endlist**

Argument	Description
DeletionListUpdate *deletionsopts*	Deletes an object from the model by referring to its type and object identifier. *deletionsopts* are required parameters that describe the object to be deleted. For the complete list of options, see "deletionsopts" (p. 201).

Note

We strongly recommend that you do not change this syntax in generated MDL, as this could lead to unexpected results in your model.

Example

This example deletes the category with object identifier 1001.

```
DeletionListUpdate Category 1001 Endlist
```

DimAdd

Creates a dimension.

UI Equivalent

Insert Item (Edit menu), when a dimension is selected.

Syntax

DimAdd *objDim* [*dimopts*] **Associations** *assocopts*

Argument	Description
DimAdd *objDim*	Creates the dimension *objDim*. *objDim* must be the object name and can include the object identifier.
dimopts	Optional parameters that describe the dimension in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "dimopts" (p. 203).
Associations *assocopts*	Parameters that specify associations for the dimension. For the complete list of options, see "assocopts" (p. 176).

Note

For more information about creating dimensions, see "DimMake" (p. 111).

Example

This example creates the dimension Years.

```
DimAdd 231 "Years" DimType Date EarliestDate 19000101
LatestDate 99991231

ManualPeriods False DaysInWeek 127 NewCatsLock False
ExcludeAutoPartitioning False

Associations 1249 "Time Category Code" AssociationType
Type_Query AssociationRole Role_Source

AssociationReferenced "Time Category Code"
```

DimCalcDefAdd

Creates a dimension calculation definition.

UI Equivalent

Add (Calculation tab of the dimension property sheet).

Syntax

DimCalcDefAdd *objDimCalcDef* **Dimension** *objDim* [**Calc** *expropts*] [**GroupCalculateCategory** {**True**|**False**}] [**Set** *string* **StartList** *objCats* **EndList**]

Argument	Description
DimCalcDefAdd *objDimCalcDef*	Creates the dimension calculation definition *objDimCalcDef*. *objDimCalcDef* must be the object name and can also be the object identifier.
Dimension *objDim*	Specifies the dimension where the calculated category is to be located. *objDim* can the object name, object identifier, or both.
Calc *expropts*	Specifies the calculation. expropts are a mixture of calculation keywords and objects. The calculation keywords that can be used are **Average**, **Change**, **Percent-Growth**, **Max**, **Min**, and **Share**. The objects are specified as object name followed by an at sign (@), the object type, and, optionally, an at sign (@) and the object identifier. The object type can be Category, Level, or Drill. For example, `"Go Water Bottle@Category@4805"`

Argument	Description
GroupCalculateCategory {**True**|**False**}	Is an option specifying that calculated categories be grouped together in the dimension viewer. The equivalent in the user interface is the Group Calculated Categories Together check box, which appears in the Category Calculation Definition dialog box when you create a category definition. The default is **False**.
Set *string* **StartList** *objCats*	Creates one or more sets and specifies the categories in each set. *string* is the set name: Set 1, Set 2, and so on. *objCats* can be the object name or the object identifier. This argument is repeated for each set, as in `Set "Set 1" StartList 4795 4797 EndList Set "Set 2" StartList 4799 EndList`

Note

For more information about creating dimension calculation definitions, see "DimCalcDefMake" (p. 108).

Example

This example adds a share calculation in the Productions dimension. The statement creates two objects: the dimension calculation definition and an associated category.

```
DimCalcDefAdd 'share ( "Set 1" , "GO Sport Line" )' Dimension
2947 Calc share ( "Set 1@Set" , "GO Sport Line@Category@4789" )
 Set "Set 1" StartList 4793 4797 4801 4803 EndList
```

Note: This example, unlike most in this book, uses object identifiers.

DimCalcDefDelete

Deletes a dimension calculation definition.

UI Equivalent

Remove (Calculation tab of the property sheet for a dimension).

Syntax

DimCalcDefDelete *objDimCalcDef* [**Dimension** *objDim*]

Argument	Description
DimCalcDefDelete *objDimCalcDef*	Deletes the dimension calculation definition *objDimCalcDef*. *objDimCalcDef* can be the object name, object identifier, or both.

Argument	Description
Dimension *objDim*	Is specified, if necessary, to uniquely identify *objDimCalcDef*. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.

Example

This example deletes the dimension calculation definition that has the object identifier 27195.

```
DimCalcDefDelete 27195
```

Note: This example, unlike most in this book, uses object identifiers.

DimCalcDefMake

Creates or updates a dimension calculation definition.

UI Equivalent

Add or Modify (Calculation tab of the property sheet for a dimension).

Syntax

DimCalcDefMake *objDimCalcDef* [**Dimension** *objDim*] [**GroupCalculateCategory** {**True**|**False**}] [**Calc** *expropts*] [**Set** *string* **StartList** *objCats* **EndList**]

Argument	Description
DimCalcDefMake *objDimCalcDef*	Creates the dimension calculation definition *objDimCalcDef* or modifies it if it exists. Include the object name if the definition does not exist.
Dimension *objDim*	Is specified, if necessary, to uniquely identify *objDimCalcDef*. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
GroupCalculateCategory {**True**|**False**}	Is an option specifying that calculated categories be grouped together in the dimension viewer. The equivalent in the user interface is the Group Calculated Categories Together check box, which appears in the Category Calculation Definition dialog box when you create a category definition. The default is **False**.

Argument	Description
Calc *expropts*	Specifies the calculation. *expropts are* a mixture of calculation keywords and objects. The calculation keywords that can be used are **Average**, **Change**, **Percent-Growth**, **Max**, **Min**, and **Share**. The objects are specified as object name followed by an at sign (@), the object type, and, optionally, an at sign (@) and the object identifier. The object type can be Category, Level, or Drill. For example, `"Go Water Bottle@Category@4805"`
Set *string* **StartList** *objCats* **Endlist**	Creates one or more sets and specifies the categories in each set. *string* is the set name: Set 1, Set 2, and so on. *objCats* can be the object name or the object identifier. This argument is repeated for each set, as in `Set "Set 1" StartList 4795 4797 EndList Set "Set 2" StartList 4799 EndList`

Note

There are two parts to a dimension calculation definition in MDL: a **DimCalcDefMake** or **DimCalcDefAdd** statement defining the calculation, and a **CatMake** or **CatAdd** statement defining the category. The link between the two is an option in the **CatMake** or **CatAdd** statement that references the **DimCalcDefMake** or **DimCalcDefAdd** statement. For more information, see the option "CalcDef" (p. 179).

Calculated categories are defined as regular categories (not special categories).

Example

```
This example defines the dimension calculation
definition Share ("Set 1", "GO Sport Line").

DimCalcDefMake 27195 'share ( "Set 1" , "GO Sport Line"
)' Dimension 2947 GroupCalculateCategory True Calc share ( "Set
1@Set" , "GO Sport Line@Category@4789" )  Set "Set 1" StartList
4793 4797 4801 4803 EndList
```

Note: This example, unlike most in this book, uses object identifiers.

DimCalcDefUpdate

Modifies a dimension calculation definition.

UI Equivalent

Modify (Calculation tab of the property sheet for a dimension).

Syntax

DimCalcDefUpdate *objDimCalcDef* [**Dimension** *objDim*] [**GroupCalculateCategory** {**True**|**False**}] [**Calc** *expropts*] [**Set** *string* **StartList** *objCats* **EndList**]

Argument	Description
DimCalcDefUpdate *objDimCalcDef*	Modifies the dimension calculation definition *objDimCalcDef*. *objDimCalcDef* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify *objDimCalcDef*. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
GroupCalculateCategory {**True**|**False**}	Is an option specifying that calculated categories be grouped together in the dimension viewer. The equivalent in the user interface is the Group Calculated Categories Together check box, which appears in the Category Calculation Definition dialog box when you create a category definition. The default is **False**.
Calc *expropts*	Specifies the calculation. expropts are a mixture of calculation keywords and objects. The calculation keywords that can be used are **Average**, **Change**, **Percent-Growth**, **Max**, **Min**, and **Share**. The objects are specified as object name followed by an at sign (@), the object type, and, optionally, an at sign (@) and the object identifier. The object type can be Category, Level, or Drill. For example, `"Go Water Bottle@Category@4805"`
Set *string* **StartList** *objCats* **EndList**	Creates one or more sets and specifies the categories in each set. *string* is the set name: Set 1, Set 2, and so on. *objCats* can be the object name or the object identifier. This argument is repeated for each set, as in `Set "Set 1" StartList 4795 4797 EndList Set "Set 2" StartList 4799 EndList`

Note

For more information about updating dimension calculation definitions, see "DimCalcDefMake" (p. 108).

Example

This example modifies a dimension calculation definition Share ("Set 1", "GO Sport Line") by adding the category with object identifier into Set 1.

```
DimCalcDefUpdate 27195 'share ( "Set 1" , "GO Sport Line"
)' Dimension 2947 Calc share ( "Set 1@Set" , "GO Sport Line@Category@4789"
)  Set "Set 1" StartList 4793 4795 4797 4801 4803 EndList
```

Note: This example, unlike most in this book, uses object identifiers.

DimDelete

Deletes a dimension.

UI Equivalent

Delete Item (Edit menu), when a dimension is selected.

Syntax

DimDelete *objDim*

Argument	Description
DimDelete *objDim*	Deletes the dimension *objDim*. *objDim* can be the object name, object identifier, or both.

Example

This example deletes the dimension Products.

```
DimDelete "Products"
```

DimensionListUpdate

Re-orders the list of dimensions.

UI Equivalent

Dragging and dropping dimensions.

Syntax

DimensionListUpdate *objDims* **EndList**

Argument	Description
DimensionListUpdate *objDims*	Specifies the dimensions in the desired order. If not all dimensions are specified, the specified ones move to the top of the list. The list of dimensions can be object names or object identifiers.

Example

This example re-orders the list of dimensions so that Products and Years are the first dimensions.

```
DimensionListUpdate "Products" "Years" EndList
```

DimMake

Creates a dimension or updates an existing one.

UI Equivalent

Modifying the property sheet, if the dimension exists. Insert Item (Edit menu), when a dimension is selected, for a new dimension.

Syntax

DimMake *objDim* [*dimopts*] **Associations** *assocopts*

Argument	Description
DimMake *objDim*	Creates the dimension *objDim* or modifies it if it exists. *objDim* can be the object name, object identifier, or both. Include the object name if the category does not exist.
dimopts	Optional parameters that describe the dimension in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "dimopts" (p. 203). If the dimension exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Parameters that specify associations for the dimension. For the complete list of options, see "assocopts" (p. 176).

Note

Dimensions require a root category, a drill category and two default dimension views. These are created automatically when a dimension is created in the user interface, but they are not automatically created when a dimension is created in MDL unless you use the verb ModelEnsureCompleteness.

Example

This example defines the dimension Products.

```
DimMake "Products" DimType Regular NewCatsLock False
DimInfo "Products Carried by the Great Outdoors Company"
```

DimUpdate

Updates an existing dimension.

UI Equivalent

Modifying the property sheet for a dimension.

Syntax

DimUpdate *objDim* [*dimopts*] [**Associations** *assocopts]*

Argument	Description
DimUpdate *objDim*	Specifies the dimension to update. *objDim* can be the object name, object identifier, or both.
dimopts	Optional parameters that describe the dimension in greater detail. For the complete list of options, see "dimopts" (p. 203). Previously set options are retained unless you change them with this command.
Associations *assocopts*	Optional parameters that specify associations for the dimension. For the complete list of options, see "assocopts" (p. 176).

Note

For more information about updating dimensions, see "DimMake" (p. 111).

Example

This example changes the earliest and latest dates allowed in the Years dimension.

```
DimUpdate "Years" EarliestDate 19500101 LatestDate 99900101
```

DrillCatMake

Creates or updates a drill category.

UI Equivalent

Modifying the property sheet, if the drill category exists. None, if the drill category does not exist. Drill categories are created automatically.

Syntax

DrillCatMake *objDrillCat* [**Dimension** *objDim*] [**Root** *objRootCat*] [**JoiningLevel** *objLevel*] [*catopts*]

Argument	Description
DrillCatMake *objDrillCat*	Creates the drill category *objDrillCat* or changes it if it exists. *objDrillCat* can be the object name, object identifier, or both. Include the object name if the category does not exist.
Dimension *objDim*	When creating a drill category, this specifies its dimension. When updating a drill category this is used, if necessary, to uniquely identify it. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.

Argument	Description
Root *objRootCat*	When creating a drill category, this specifies its root category. *objRootCat* can be the object name, object identifier, or both.
JoiningLevel *objLevel*	When creating a drill category, this specifies the convergence level. *objLevel* can be the object name or object identifier.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "catopts" (p. 177). If the category exists, previously set options are retained unless you change them with this command.

Example

This example defines the drill category By Product Line.

```
DrillCatMake "By Product Line" Dimension "Products" Root
"Product Line" Inclusion Suppress Filtered False Suppressed True
PrimaryDrill True YearBegins 19960101 PartialWeek Split ExtraWeek
None WeekBegins Sunday
```

EventEnd

Signals the end of an update event that was started by EventStart . The status of the PowerCubes involved in the event reverts to Free.

UI Equivalent

None.

Syntax

EventEnd [*platformtype*]

Argument	Description
platformtype	The platform type can be **OnClient** or **OnServer**. This setting is optional.

Example

This example ends the current update event on the client.

```
EventEnd OnClient
```

EventStart

Signals the start of an update event. The status of the PowerCubes involved in the event is changed from Free to UpdateInProgress, and the event name is written to a log file.

UI Equivalent

None.

Syntax

EventStart *EventName* [*platformtype*]

Argument	Description
EventName	The event name can be up to 256 characters in length, and may not contain a carriage return.
platformtype	The platform type can be **OnClient** or **OnServer**. The platform type is optional.

Note

The log file is a file that is created whenever you run a Transformer model. It has the same root name as the Transformer model but with a .log extension.

Example

This example starts an event called Batch Update on the client.

```
EventStart "Batch Update" OnClient
```

FilterCat

Applies the filter property to a category.

UI Equivalent

Filter (Diagram menu).

Syntax

FilterCat *objView* [**Dimension** *objDim*] **Category** *objCat*

Argument	Description
FilterCat *objView*	Specifies the dimension view within which the category is to be filtered. *objView* can be the object name, object identifier, or both.

Argument	Description
Dimension *objDim*	Is specified, if necessary, to uniquely identify the view. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the dimension's object name, object identifier, or both.
Category *objCat*	Specifies the category that is to be filtered. *objCat* can be the object name, object identifier, or both.

Note

Filtering an already filtered category will remove the filter. Filtering a parent will reduce the filtering of its children.

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example filters the Outdoor Products category for the dimension view View1.

```
FilterCat "View1" Dimension "Products" Category "Outdoor
Products"
```

FilterDelete

Deletes a filter from a query in an Architect model.

UI Equivalent

Deleting a Filter in the Filter dialog box, accessed from SQL Data Source tab, Data Source property sheet for a query contained in an Architect Model.

Syntax

FilterDelete *filterID* [**ArchitectDataSource** *ArchID*]

Argument	Description
FilterDelete *filterID*	Specifies the filter to be deleted. *filterID* can be the object name, object identifier, or both.
ArchitectDataSource *ArchID*	Is specified to uniquely identify the Architect data source for which the filter is being defined. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *ArchID* can be the Architect data source's object name, object identifier, or both.

Note

FilterDelete is only available for Architect data sources.

Example

This example deletes a filter from an Architect data source.

```
FilterDelete 5391 ArchitectDataSource 655
```

FilterLevel

Applies the filter property to a level.

UI Equivalent

Filter (Diagram menu), when a level is selected.

Syntax

FilterLevel *objView* [**Dimension** *objDim*] [**Drill** *objDrillCat*] [**Levels** *objLevel*]

Argument	Description
FilterLevel *objView*	Specifies the dimension view within which the level is to be filtered. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the view. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the dimension's object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the view. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDrillCat* can be the dimension's object name, object identifier, or both.
Levels *objLevel*	Specifies the level that is to be filtered. *objLevel* can be the object name, object identifier, or both.

Example

This example filters the Products level for the View1 dimension view.

```
FilterLevel "View1" Dimension "Products" Drill "By Product
Line" Levels "Product Type"
```

FilterMake

Applies a filter to a query in an Architect model.

UI Equivalent

Creating or modifying a Filter in the Filter dialog box, accessed from SQL Data Source tab, Data Source property sheet for a query contained in an Architect Model.

Syntax

FilterMake *filterID* [**ArchitectDataSource** *ArchID*] [*filteropts*]

Argument	Description
FilterMake *filterID*	Specifies the filter to be created. *filterID* can be the object name, object identifier, or both.
ArchitectDataSource *ArchID*	Is specified to uniquely identify the Architect data source for which the filter is being defined. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *ArchID* can be the Architect data source's object name, object identifier, or both.
filteropts	Optional parameters that describe the filter in greater detail. For the complete list of options, see "filteropts" (p. 206).

Note

FilterMake is only available for Architect data sources.

Example

This example filters the Architect data source to select only managers.

```
FilterMake 5391 "Managers_Only" ArchitectDataSource 655
"OrderMethod" ExpMark filter[Managers in SalesStaff]  ExpMarkEnd
FilterDescription "Selects Managers from the SalesStaff table."
```

FilterName

Associates a filter to an Architect DataSource.

UI Equivalent

None

Syntax

FilterName *filterID* [**ArchitectDataSource**] [*ArchAttribute*]

Argument	Description
FilterName *filterID*	Specifies the filter to be associated with the Architect DataSource. *filterID* can be the object name, object identifier, or both.

Argument	Description
ArchitectDataSource	Is specified to uniquely identify the Architect data source for which the filter is being defined. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21).
ArchAttribute	Specifies the Architect Attribute that is being filtered.

Note

FilterName is only available for Architect data sources.

FilterUpdate

Updates a filter in a query in an Architect model.

UI Equivalent

Modifying a Filter in the Filter dialog box, accessed from SQL Data Source tab, Data Source property sheet for a query contained in an Architect Model.

Syntax

FilterUpdate *filterID* [**ArchitectDataSource** *ArchID*] [*filteropts*]

Argument	Description
FilterUpdate *filterID*	Specifies the filter to be modified. *filterID* can be the object name, object identifier, or both.
ArchitectDataSource *ArchID*	Is specified, if required, to uniquely identify the Architect data source for which the filter is being defined. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *ArchID* can be the Architect data source's object name, object identifier, or both.
filteropts	Optional parameters that describe the filter in greater detail. For the complete list of options, see "filteropts" (p. 206).

Note

FilterUpdate is only available for Architect data sources.

Example

This example changes the description of the filter with object identifier 5391.

```
FilterUpdate 5391 FilterDescription "Selects Managers
from Europe from the SalesStaff table."
```

LevelAdd

Creates a level.

UI Equivalent

Insert Item (Edit menu), when a level is selected.

Syntax

LevelAdd *objLevel* [**Dimension** *objDim*] [**Drill** *objDrillCat*] [**Parent** *objLevel*] [*levelopts*] [**DrillList** *objDrillCats* **EndList**] **Associations** *assocopts*

Argument	Description
LevelAdd *objLevel*	Creates the level *objLevel*. *objLevel* must be the object name and can include the object identifier.
Dimension *objDim*	Specifies a dimension for the level. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Specifies a drill category for the level. *objDrillCat* can be the object name, object identifier, or both.
Parent *objLevel*	Specifies a parent for the level. *objLevel* can be the parent's object name, or object identifier.
levelopts	Optional parameters that describe the level in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "levelopts" (p. 207).
DrillList *objDrillCats*	Specifies the multiple drill categories *objDrillCats*. *objDrillCats* is one or more drill categories, each identified by object name or object identifier. (In Transformer version 5.0, only object identifiers can be used.)
Associations *assocopts*	Parameters that specify associations for the level. For the complete list of options, see "assocopts" (p. 176).

Note

For more information about creating levels, see "LevelMake" (p. 121).

Example

This example creates the level Year.

```
LevelAdd 237 "Year" Blanks "( blank )" Inclusion Generate
DateFunction Year
```

```
Generate Need RefreshLabel False RefreshDescription False
RefreshShortName False

NewCatsLock False CatLabFormat "YYYY" Timerank 10 UniqueCategories
True

UniqueMove False Associations 1251 "Time" AssociationType
Type_Query AssociationRole Role_Source

AssociationReferenced "Time"

Associations 1253 "Time" AssociationContext 235 AssociationType
Type_Query

AssociationRole Role_OrderBy AssociationReferenced "Time"
SortOrder Default

SortAs Ascending
```

LevelDelete

Deletes a level.

UI Equivalent

Delete Item (Edit menu), when a level is selected.

Syntax

LevelDelete *objLevel* [**Dimension** *objDim*] [**Drill** *objDrill*]

Argument	Description
LevelDelete *objLevel*	Deletes the level *objLevel*. *objLevel* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the level. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name or object identifier.

Example

This example deletes the level Product Line.

```
LevelDelete "Product Line" Dimension "Products" Drill
"By Product Line"
```

LevelMake

Creates a level or updates an existing one.

UI Equivalent

Modifying the property sheet, if the level exists. Insert Item (Edit menu), when a level is selected, for a new level.

Syntax

LevelMake *objLevel* [**Dimension** *objDim*] [**Drill** *objDrillCat*] [**Parent** *objLevel*] [*levelopts*] [**DrillList** *objDrillCats* **EndList**] **Associations** *assocopts*

Argument	Description
LevelMake *objLevel*	Creates the level *objLevel* or modifies it if it exists. *objLevel* can be the object name, object identifier, or both. Include the object name if the category does not exist.
Dimension *objDim*	When creating a level, this specifies its dimension. When updating a level this is used, if necessary, to uniquely identify it. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name or object identifier.
Drill *objDrillCat*	When creating a level, this specifies its drill category. When updating a level this is used, if necessary, to uniquely identify it. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDrillCat* can be the object name, object identifier, or both.
Parent *objLevel*	When creating a level, this specifies its parent. When updating a level this is used, if necessary, to uniquely identify it. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objLevel* can be the object name or object identifier.
levelopts	Optional parameters that describe the level in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "level-opts" (p. 207). If the category exists, previously set options are retained unless you change them with this command.
DrillList *objDrillCats*	Specifies the multiple drill categories *objDrillCats*. *objDrillCats* is one or more drill categories, each identified by object name or object identifier.
Associations *assocopts*	Parameters that specify associations for the level. For the complete list of options, see "assocopts" (p. 176).

Note

Level names cannot contain an at sign (@).

Example

This example defines the level Year.

```
LevelMake 237 "Year" Blanks "( blank )" Inclusion Generate
DateFunction Year

Generate Need RefreshLabel False RefreshDescription False
RefreshShortName False

NewCatsLock False CatLabFormat "YYYY" Timerank 10 UniqueCategories
True

UniqueMove False Associations 1251 "Time" AssociationType
Type_Query AssociationRole Role_Source

AssociationReferenced "Time"

Associations 1253 "Time" AssociationContext 235 AssociationType
Type_Query

AssociationRole Role_OrderBy AssociationReferenced "Time"
SortOrder Default

SortAs Ascending
```

LevelMoveAfter

Switches the positions of two levels, putting the specified level after the child level.

UI Equivalent

Dragging and dropping a level.

Syntax

LevelMoveAfter *objLevel* [**Dimension** *objDim*] [**Drill** *objDrill*] **Child** *objLevel*

Argument	Description
LevelMoveAfter *objLevel*	Specifies the level that is to be placed after. *objLevel* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the level. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDrillCat* can be the object name, object identifier, or both.

Argument	Description
Child *objLevel*	Specifies the level that is to be placed before. *objLevel* can be the object name or object identifier.

Note

This does not change the level's object identifier or position in the MDL file. It affects the visual display of a level in the user interface and its position in the .mdc file, which affects how it is viewed in PowerPlay or Analysis Studio.

Example

This example moves the level Product Type after the level Product Id.

```
LevelMoveAfter "Product Type"  Drill "By Product Line"
Child "Product Id"
```

LevelMoveBefore

Switches the positions of two levels, putting the specified level before the child level.

UI Equivalent

Dragging and dropping a level.

Syntax

LevelMoveBefore *objLevel* [**Dimension** *objDim*] [**Drill** *objDrill*] **Child** *objLevel*

Argument	Description
LevelMoveBefore *objLevel*	Specifies the level that is to be placed before. *objLevel* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the level. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDrillCat* can be the object name, object identifier, or both.
Child *objLevel*	Specifies the level that is to be placed after. *objLevel* can be the object name or object identifier. The child is mandatory.

Note

This does not change the level's object identifier or position in the MDL file. It affects the visual display of a level in the user interface and its position in the .mdc file, which affects how it is viewed in PowerPlay or Analysis Studio.

Example

This example moves the level Product Line before the level Product Id.

```
LevelMoveBefore "Product Line" Drill "By Product Line"
Child "Product Id"
```

LevelNewDrill

Creates a new drill path in a level.

UI Equivalent

Dragging and dropping levels and columns.

Syntax

LevelNewDrill *objLevel* [**Dimension** *objDim*] [**Drill** *objDrillCat*] **Drill** *objDrillCat* **Child** *objLevel*

Argument	Description
LevelNewDrill *objLevel*	Specifies the level that is to have a new drill path. *objLevel* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Specifies the old drill category. *objDrillCat* can be the object name, object identifier, or both.
Drill *objDrillCat*	Specifies the new drill category. *objDrillCat* can be the object name, object identifier, or both.
Child *objLevel*	Specifies the joining level. *objLevel* can be the object name or object identifier.

LevelUpdate

Updates an existing level.

UI Equivalent

Modifying the property sheet for a level.

Syntax

LevelUpdate *objLevel* [**Dimension** *objDim*] [**Drill** *objDrillCat*] [*levelopts*]
[**DrillList** *objDrillCats* **EndList**] [**Associations** *assocopts*]

Argument	Description
LevelUpdate *objLevel*	Specifies the level to update. *objLevel* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the level. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDrillCat* can be the object name, object identifier, or both.
levelopts	Optional parameters that describe the level in greater detail. For the complete list of options, see "levelopts" (p. 207). Previously set options are retained unless you change them with this command.
DrillList *objDrillCats*	Specifies the multiple drill categories *objDrillCats*. *objDrillCats* is one or more drill categories, each identified by object name or object identifier.
Associations *assocopts*	Optional parameters that specify associations for the level. For the complete list of options, see "assocopts" (p. 176).

Note

For more information about updating levels, see "LevelMake" (p. 121).

Example

This example changes the inclusion property of level Product Type to Suppress. Because the object name does not uniquely identify the level, the dimension and drill category are specified.

```
LevelUpdate "Product Type" Dimension "Products" Drill
"By Product Line" Inclusion Suppress
```

MDCCheckServer

Issues a request to Transformer on UNIX to check the integrity and status of all PowerCubes. This is part of the internal client-server communication.

UI Equivalent

None.

Syntax

MDCCheckServer

MDCClear

Clears the status flags from a PowerCube.

UI Equivalent

None. However the effect can be seen in the PowerCube property sheet, Processing tab, Status.

Syntax

MDCClear *objCube*

Argument	Description
MDCClear *objCube*	Specifies the PowerCube that is to have its status flags cleared. *objCube* can be the object name, object identifier, or both.

Example

This example changes the status for The Great Outdoors Co. Sales PowerCube from OK to New.

```
MDCClear "The Great Outdoors Co. Sales"
```

MeasureAdd

Adds a measure. If the object exists, an error message is issued.

UI Equivalent

Insert Item (Edit menu), when a measure is selected.

Syntax

MeasureAdd *objMeasure* [*meaopts*] **Associations** *assocopts*

Argument	Description
MeasureAdd *objMeasure*	Creates the measure *objMeasure*. *objMeasure* must be the object name and can include the object identifier.

Argument	Description
meaopts	Optional parameters that describe the measure in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "meaopts" (p. 214).
Associations *assocopts*	Parameters that specify associations for the measure. For the complete list of options, see "assocopts" (p. 176).

Note

Every measure requires an **AllocationAdd** statement that contains the measure name and the option **TypeDefault**. Transformer creates this statement automatically when a measure is added in the user interface or when you use the verb **ModelEnsureCompleteness**. Otherwise, you must create the statement. For more information, see "AllocationAdd" (p. 58).

For more information about creating measures, see "MeasureMake" (p. 129).

Example

This example adds to the model the Staff Count measure from the Staff Count column.

```
MeasureAdd 863 "Staff Count" Missing N/A TimeStateRollup
Average Storage Float64

OutPutScale 0 Decimals 0 ReverseSign False IsCurrency
False WeightId 899

DrillThrough False EndList Associations 1309 "Staff Count"
AssociationType Type_Query

AssociationRole Role_Source AssociationReferenced "Staff
Count"
```

MeasureDelete

Deletes a measure.

UI Equivalent

Delete Item (Edit menu), when a measure is selected.

Syntax

MeasureDelete *objMeasure*

Argument	Description
MeasureDelete *objMeasure*	Deletes the category *objMeasure*. *objMeasure* can be the object name, object identifier, or both.

Example

This example deletes the measure Profit Margin %.

```
MeasureDelete "Profit Margin %"
```

MeasureListUpdate

Re-orders a list of measures.

UI Equivalent

Dragging and dropping measures.

Syntax

MeasureListUpdate *objMeasures* **EndList**

Argument	Description
MeasureListUpdate *objMeasures*	Lists the measures in the desired order. If not all measures in the model are listed, those listed are moved to the top of the list. *objMeasures* can be the object names or object identifiers.

Example

This example changes the order of the list of measures in the Measures pane in the user interface, placing Product Cost and Revenue at the top of the list.

```
MeasureListUpdate "Product Cost" "Revenue" EndList
```

MeasureMake

Creates a measure or updates an existing one.

UI Equivalent

Modifying the property sheet, if the measure exists. Insert Item (Edit menu), when a measure is selected, for a new measure.

Syntax

MeasureMake *objMeasure* [*meaopts*] **Associations** *assocopts*

Argument	Description
MeasureMake *objMeasure*	Creates the measure objMeasure or modifies it if it exists. objMeasure can be the object name, object identifier, or both. Include the object name if the category does not exist.

Argument	Description
meaopts	Optional parameters that describe the measure in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "meaopts" (p. 214). If the category exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Parameters that specify associations for the measure. For the complete list of options, see "assocopts" (p. 176).

Note

Every measure requires an **AllocationAdd** statement that contains the measure name and the option **Type Default**. Transformer creates this statement automatically when a measure is added in the user interface or when you use the verb **ModelEnsureCompleteness**. Otherwise, you must create the statement. For more information, see "AllocationAdd" (p. 58).

Measure names cannot contain an at sign (@).

Example

The following statements were generated by Transformer when a model, created in the user interface, was saved as .mdl.

This example creates or updates the StaffCount measure.

```
MeasureMake 863 "Staff Count" Missing N/A TimeStateRollup
Average Storage Float64

OutPutScale 0 Decimals 0 ReverseSign False IsCurrency
False WeightId 899

DrillThrough False EndList Associations 1309 "Staff Count"
AssociationType Type_Query

AssociationRole Role_Source AssociationReferenced "Staff
Count"
```

This example creates or updates the calculated measure Profit Margin %.

```
MeasureMake "Profit Margin %" Calc ( "Revenue@259" -
"Product Cost@261" ) / "Revenue@259"  Missing N/A Timing After_Rollup
Storage Float64 Scale 0 Decimals 1 Sign False Format "0%~1" MeasureInfo
"Profit as a percentage. The calculation is  (Revenue - Product
Cost)" " / Revenue." DrillThrough False EndList
```

This example creates or updates a Revenue measure that allows drill-through to an Impromptu report.

```
MeasureMake  "Revenue" Association "Revenue" Storage
Default Scale 0 Decimals 0 Sign False Format "#,##0~0" MeasureInfo
"Gross revenue from product sales." DrillThrough True "c:\dt_cust.imr"
"" EndList
```

MeasureUpdate

Updates an existing measure.

UI Equivalent

Modifying the property sheet for a measure.

Syntax

MeasureUpdate *objMeasure* [*meaopts*] [**Associations** *assocopts]*

Argument	Description
MeasureUpdate *objMeasure*	Specifies the measure to update. *objMeasure* can be the object name, object identifier, or both.
meaopts	Optional parameters that describe the measure in greater detail. For the complete list of options, see "meaopts" (p. 214). If the category exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Optional parameters that specify associations for the measure. For the complete list of options, see "assocopts" (p. 176).

Note

For more information about updating measures, see "MeasureMake" (p. 129).

Example

This example changes the timing of the Profit Margin % measure.

```
MeasureUpdate "Profit Margin %" Timing Before_Rollup
```

ModelEnsureCompleteness

Scans all dimensions and cubes and creates necessary default settings and objects, including root categories, drill categories and default dimension views.

UI Equivalent

None. These functions are done automatically.

Syntax

ModelEnsureCompleteness

Note

By using ModelEnsureCompleteness, it is not necessary to create Dimension views or link them to cubes and user classes. Add this verb to the end of a user-defined MDL model to ensure that the model is valid.

NewModel

Creates, names, describes, and opens a Transformer model.

UI Equivalent

New (File menu).

Syntax

NewModel *objModel* [**ServerModelPath** *filename*] [*appqueryopts*]

Argument	Description
NewModel *objModel*	Specifies the model that is to be opened. *objModel* must be the model's name.
ServerModelPath *filename*	Specifies the file name, including path if desired. This argument is optional.
appqueryopts	Optional parameters that describe the model in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "appquery-opts" (p. 165).

Note

The model path is saved in the user interface in the Server tab of the Model Properties dialog box. If no path is specified, Transformer saves model files on the server in the current working directory.

Running **NewModel** closes open files without saving them. If you want to save a file, you can use **SaveMDL** or **SavePY** before **NewModel**.

The **ServerModelPath** keyword replaced the **AppServerPath** keyword in Transformer 6.6 and subsequent versions.

Example

This example creates a model called PowerPlay Sample, and provides a description.

```
NewModel "PowerPlay Sample" AppInfo "This model is used
in many discussions and examples in the PowerPlay online documentation."
```

OpenDef

Opens .def, .gen, and .dat files.

UI Equivalent

Files of type: PowerPlay definition files (*.def & *.gen) (Open dialog box).

Syntax

OpenDef *name*

Argument	Description
OpenDef *filename*	Specifies the file name, including path if desired.

Note

OpenDef does not work with files on the server.

Example

This example opens the file file.def.

```
OpenDef "file.def"
```

OpenMDL

Opens the specified .mdl file.

UI Equivalent

Files of type: Model files (*.py? and *.mdl) (Open dialog box).

Syntax

OpenMDL *filename*

Argument	Description
OpenMDL *filename*	Specifies the file name, including path if desired, for the current model. The file name should have the extension .mdl.

Note

Execution of the script continues after the file is opened. Only one model at a time can be open. Any model that was open before **OpenMDL** is closed without being saved. To save the model, you can issue the **SaveMDL** or **SavePY** command before **OpenMDL**.

Example

This example opens the model outdoors.mdl, located in the root directory of the c: drive.

```
OpenMDL "c:\outdoors.mdl"
```

OpenPY

Opens the specified .py? file. The question mark in the extension .py? is replaced by the character that is used in your release of Transformer.

UI Equivalent

Files of type: Model files (*.py? and *.mdl) (Open dialog box).

Syntax

OpenPY *filename*

Argument	Description
OpenPY *filename*	Specifies the file name, including path if desired. The file name should have the extension .py?, where ? is dependent on your version of Transformer.

Note

Execution of the script continues after the file is opened. Only one file at a time can be open. Any model that was open before **OpenPY** is closed without being saved. To save the model, you can issue the **SaveMDL** or **SavePY** command before **OpenPY**.

Example

This example opens Model.py? (in this example, .pyh) and saves it as Model.mdl.

```
OpenPY "C:\Model.pyh"
SaveMDL "C:\Model.mdl"
```

PopulateFromQueries

Populates the model with categories from the specified data sources.

UI Equivalent

Right-clicking a data source and choosing Generate Categories.

Syntax

PopulateFromQueries [*platformtype*] *objQueries*

Argument	Description
platformtype	The platform type can be **OnClient** or **OnServer**. This setting is optional.
objQueries	Specifies one or more data sources that are to be used. *objQueries* can be the object identifiers or object names.

Example

This example populates the model on the client using only the All Staff Count data source.

```
PopulateFromQueries OnClient "All Staff Count (Excel)"
```

PopulateModel

Populates the model with categories using all data sources in the model.

UI Equivalent

Generate Categories (Run menu).

Syntax

PopulateModel [*platformtype*]

Argument	Description
platformtype	The platform type can be **OnClient** or **OnServer**. This setting is optional.

Example

This example populates the model on the client using all data sources.

```
PopulateModel OnClient
```

PowerCubeDelete

Deletes the specified PowerCube or PowerCube group. This deletes the PowerCube objects in the model but does not affect existing .mdc files.

UI Equivalent

Delete Item (Edit menu), when a PowerCube or PowerCube group is selected.

Syntax

PowerCubeDelete {*objCube*|*objCubeGroup*}

Argument	Description
PowerCubeDelete {*objCube*|*objCubeGroup*}	Deletes the PowerCube *objCube* or the PowerCube group *objCubeGroup*. *objCube* and *objCubeGroup* can be the object name, object identifier, or both.

Example

This example deletes the PowerCube The Great Outdoors Co. Sales.

```
PowerCubeDelete "The Great Outdoors Co. Sales"
```

PowerCubeListUpdate

Re-orders the list of PowerCubes and PowerCube groups.

UI Equivalent

Dragging and dropping PowerCubes and PowerCube groups.

Syntax

PowerCubeListUpdate [*objCubes*] [*objCubeGroups*] **EndList**

Argument	Description
objCubes	Lists the PowerCubes in the desired order. If not all cubes are specified, those specified move to the top of the list. The cubes can be identified by object name or object identifier.
objCubeGroups	Lists the cube groups in the desired order. If not all groups are specified, those specified move to the top of the list. The cube groups can be identified by object name or object identifier.

Example

This example changes the order of a PowerCube and a cube group.

```
PowerCubeListUpdate "cubegroup1" "The Great Outdoors
Co. Sales" EndList
```

PowerCubeUserListUpdate

Specifies the order when creating user classes associated with a PowerCube.

UI Equivalent

None.

Syntax

PowerCubeUserListUpdate Cube *objCube* **StartList** *objUserClasses* **EndList**

Argument	Description
Cube *objCube*	Specifies the PowerCube that is to be re-ordered. *objCube* can be the PowerCube's object name, object identifier, or both.
StartList *objUserClasses*	Lists the user classes in the desired order. If not all users are specified, those specified move to the top of the list. The users can be identified by object name or object identifier.

ReportPartitions

Writes a partition status report to the log file for a specified PowerCube.

UI Equivalent

None.

Syntax

ReportPartitions *objCube*

Argument	Description
ReportPartitions *objCube*	Specifies the PowerCube that is to have a partition status report written to its log file.

Example

This example writes a partition report to the log file for The Great Outdoors Co. Sales PowerCube.

```
ReportPartitions "The Great Outdoors Co. Sales"
```

RootCatMake

Creates a root category or updates an existing one.

UI Equivalent

Modifying the property sheet, if the root category exists. None, if the root category does not exist. Root categories are created automatically.

Syntax

RootCatMake *objRootCat* [**Dimension** *objDim*] [*catopts*]

Argument	Description
RootCatMake *objRootCat*	Creates the category *objRootCat* or changes it if it exists. *objRootCat* can be the object name, object identifier, or both. Include the object name if the category does not exist.
Dimension *objDim*	Specifies the dimension of a new root category or, if necessary, uniquely identifies an existing root category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.

Argument	Description
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "catopts" (p. 177). If the category exists, previously set options are retained unless you change them with this command.

Example

The example defines the root category Product Line.

```
RootCatMake "Product Line" Dimension "Products" Inclusion
Generate Lastuse 19971202 Filtered False Suppressed False Sign False
IsKeyOrphanage False IsTruncated False Blanks False
```

RootCatUpdate

Updates an existing root category.

UI Equivalent

Modifying the property sheet for a root category.

Syntax

RootCatUpdate *objRootCat* **Dimension** [*objDim*] [*catopts*]

Argument	Description
RootCatUpdate *objRootCat*	Specifies the root category to update. *objRootCat* can be the object name, object identifier, or both.
Dimension *objDim*	Required. Is specified to uniquely identify the root category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
catopts	Optional parameters that describe the category in greater detail. For the complete list of options, see "catopts" (p. 177). Previously set options are retained unless you change them with this command.

Note

For more information about updating root categories, see "RootCatMake" (p. 137).

Example

This example adds the description "Products carried by the Great Outdoors Company" to the Products root category.

```
RootCatUpdate "Products" Dimension "Products" Catinfo
"Products carried by the Great Outdoors Company."
```

SaveMDL

Saves the current model in MDL format.

UI Equivalent

Saving a model with an .mdl extension.

Syntax

SaveMDL *filename*

Argument	Description
SaveMDL *filename*	Specifies the file name, including path if desired, for the current model. The file name should have the extension .mdl.

Example

This example opens a .py? file (in this example, .pyh) and saves it as an .mdl file.

```
OpenPY "C:\model.pyh"
SaveMDL "C:\model.mdl"
```

SavePY

Saves the current model in .py? format.

UI Equivalent

Saving a model with a .py? extension.

Syntax

SavePY*filename*

Argument	Description
SavePY *filename*	Specifies the file name, including path if desired, for the current model. The file name should have the extension .py?, where ? is replaced by the character that is used in your release of Transformer.

Example

This example saves the open model in PY format in the root directory of the C drive, and calls it model.py? (in this example, .pyi).

```
SavePY "C:\model.pyi"
...
```

SendServerModel

Sends a model from the server to the client.

UI Equivalent

Restore Client Model from Server (Server/Maintenance menu).

Syntax

SendServerModel

SignonAdd

Creates a Transformer signon object.

UI Equivalent

Insert Item (Edit menu), when the Signons pane is selected.

Syntax

SignonAdd *objSignon* [*signonopts*]

Argument	Description
SignonAdd *objSignon*	Creates the signon *objSignon*. *objSignon* must be the object name, and can include the object identifier.
signonopts	Optional parameters that describe the signon object in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "signonopts" (p. 233).

Note

For more information about creating signons, see "SignonMake" (p. 141).

Example

This example creates a signon with a user id and password so cube generation can proceed without prompting. The next time the MDL file is interpreted, this password information will be encrypted.

```
SignonAdd "I40_DBASE_NTV_PP52_SAMPLE" PromptForPassword
False Password "passtransf" UserID "transf1"
```

SignonDelete

Deletes a signon from the model.

UI Equivalent

Delete Item (Edit menu), when a signon is selected.

Syntax

SignonDelete *objSignon*

Argument	Description
SignonDelete*objSignon*	Specifies the signon that is to be deleted. *objSignon* can be the object name, object identifier, or both.

Example

This example deletes the signon "I40_DBASE_NTV_PP52_SAMPLE"

```
SignonDelete "I40_DBASE_NTV_PP52_SAMPLE"
```

SignonListUpdate

Re-orders the list of signon objects in the model.

UI Equivalent

Dragging and dropping signons.

Syntax

SignonListUpdate *objSignons* **EndList**

Argument	Description
SignonListUpdate *objSignons*	Lists the signon objects in the desired order. If not all signons are specified, the specified ones are moved to the top of the list. *objSignons* can be the object names or object identifiers.

Example

This example reorders the list of signons.

```
SignonListUpdate "I40_DBASE_NTV_PP52_SAMPLE" "database2_sample"
EndList
```

SignonMake

Creates or updates a signon object.

UI Equivalent

Modifying the property sheet, if the signon exists. Insert Item (Edit menu), when the signon pane is selected, for a new signon.

Syntax

SignonMake *objSignon* [*signonopts*]

Argument	Description
SignonMake *objSignon*	Creates the object *objSignon* or modifies it if it exists. *objSignon* can be the object name, object identifier, or both. Include the object name if the signon does not exist.
signonopts	Optional parameters that describe the database connection in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "signonopts" (p. 233). If the connection exists, previously set options are retained unless you change them with this command.

Note

If you set a password and user id for a signon in the General tab of the Signon property sheet and then save the model as MDL, the information will be saved to the .mdl file if Transformer is set to generate verb MDL. If Transformer generates Structured MDL the password is not saved and then PromptForPassword is set to True.

For security purposes, passwords and user IDs are saved in encrypted format.

The options FullDB and EncryptedPW require encrypted strings that can only be produced by Transformer. The options Database and Password are identical except that they do not require encrypted strings and so can be manually entered. For more information, see the entries in "signonopts" (p. 233).

Example

This example defines a signon with the object name I40_DBASE_NTV_PP52_SAMPLE. The alphanumeric strings, which were truncated for this example, are encryptions for security purposes. This statement is generated by Transformer when a model that is created in the user interface is saved as an .mdl file.

```
SignonMake  "I40_DBASE_NTV_PP52_SAMPLE" FullDb
"026EBCB2FCBA15649EA5165BBC2F073BB3C06B5952F321BC01F8BD5BBC2F073BB3C06B5952F321BC01F8BD5B"
DbType "DB" PromptForPassword False EncryptedPW
"81FC8F43B84BA2E2CBE68500010A7EFCC1B199DDACC833554"
"B10431F71D2C6B5071ECB96DDA6282CD2EF5EA64C55A7EB735A8D0895956AE"
"5EC7FF4886B74D169CEB5D"...
```

SignonUpdate

Updates an existing signon object.

In Transformer for UNIX (*rsserver*), which does not use Access Manager to determine the proper datasource connection parameters, you must use this MDL verb and an encrypted definition of the connection, to update the signon information.

UI Equivalent

Modifying the property sheet for a signon.

Syntax

SignonUpdate *objSignon* [*signonopts*]

Argument	Description
SignonUpdate *objSignon*	Specifies the signon to update. *objSignon* can be the object name, object identifier, or both.
signonopts	Optional parameters that describe the database connection in greater detail. For the complete list of options, see "signonopts" (p. 233). Previously set options are retained unless you change them with this command.

Note

For more information about updating signons, see "SignonMake" (p. 141).

Example

This example tells Transformer to not prompt for a password when an .mdc file is created, and creates a password and user ID to be used instead. The next time that the MDL file is interpreted, this password information will be encrypted in the SignonMake statement.

```
SignonUpdate "I40_DBASE_NTV_PP52_SAMPLE" PromptForPassword
False Password "passtransf" UserID "transf1"
```

SourceListUpdate

Re-orders the list of data sources in the model.

UI Equivalent

Dragging and dropping data sources.

Syntax

SourceListUpdate *objDataSources* **EndList**

Argument	Description
SourceListUpdate *objDataSources*	Lists the data sources in the desired order. If not all data sources are specified, the specified ones are moved to the top of the list. The data sources can be identified by object name or object identifier.

Note

The **SourceListUpdate** keyword replaced the **QueryListUpdate** keyword in Transformer 6.6 and subsequent versions.

Example

This example re-orders the list of data sources so that Locations, Products, and Main are the first three in the list.

```
SourceListUpdate "Locations (CSV)" "Products (CSV)" "MAIN
(IQD)" EndList
```

SpecialCatAdd

Creates a special category.

UI Equivalent

Clicking the right side of a root category or special category and dragging the connection to the right.

Syntax

SpecialCatAdd *objSpecialCat* [**Dimension** *objDim*] **Parent** *objCat* [*catopts*]

Argument	Description
SpecialCatAdd *objSpecialCat*	Creates the special category *objSpecialCat*. *objSpecialCat* must be the object name and can include the object identifier.
Dimension *objDim*	Specifies a dimension *objDim* for the special category. *objDim* can be the object name, object identifier, or both.
Parent *objCat*	Specifies the parent category *objCat*. *objCat* can be the object name or object identifier.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "catopts" (p. 177).

Note

For more information about creating special categories, see "SpecialCatMake" (p. 145).

Example

This example creates the special category Current Month.

```
SpecialCatAdd "Current Month" Parent "Time"
```

SpecialCatDelete

Deletes a special category.

UI Equivalent

Delete Item (Edit menu), when a special category is selected.

Syntax

SpecialCatDelete *objSpecialCat* [**Dimension** *objDim*]

Argument	Description
SpecialCatDelete *objSpecialCat*	Deletes the special category *objSpecialCat*. *objSpecialCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.

Note

If a special category has children, the children must be deleted before the special category or an error message is issued. This is different from the user interface, where deleting a special category automatically deletes its children.

Example

This example deletes the special category Current Month.

```
SpecialCatDelete "Current Month"
```

SpecialCatMake

Creates a special category or updates an existing one.

UI Equivalent

Modifying the property sheet, if the special category exists. Clicking the right side of a root category or special category and dragging the connection to the right, for a new special category.

Syntax

SpecialCatMake *objSpecialCat* [**Dimension** *objDim*] [**Parent** *objLevel*] [*catopts*]

Argument	Description
SpecialCatMake *objSpecialCat*	Creates the category *objSpecialCat* or modifies it if it exists. *objSpecialCat* can be the object name, object identifier, or both. Include the object name if the special category does not exist.
Dimension *objDim*	When creating a special category, this specifies the dimension in which the category is to be placed. When updating a category, this is used, if necessary, to uniquely identify the category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Parent *objLevel*	When creating a category, this specifies its parent and is mandatory. This argument is not needed when updating a category. *objLevel* can be object name or object identifier.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "catopts" (p. 177). If the category exists, previously set options are retained unless you change them with this command.

Note

Calculated categories are defined as regular categories in MDL. For more information, see "DimCalcDefMake" (p. 108).

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example defines the special category Current Month.

```
SpecialCatMake "Current Month" Parent "Time" Lastuse
19970425 Rollup True TimeAggregate Single RunningPeriods 0 TargetOffset
0 TargetLevel "Month" ContextOffset 0 DateDrill 5237 SplitWeek False
Primary 5897 Filtered False Suppressed False Sign False IsKeyOrphanage
False IsTruncated False Blanks False
```

SpecialCatUpdate

Updates an existing special category.

UI Equivalent

Modifying the property sheet for a special category.

Syntax

SpecialCatUpdate *objSpecialCat* [**Dimension** *objDim*] [*catopts*]

Argument	Description
SpecialCatUpdate *objSpecialCat*	Updates the special category *objSpecialCat*. *objSpecialCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the special category. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
catopts	Optional parameters that describe the category in greater detail. For the complete list of options, see "catopts" (p. 177). Previously set options are retained unless you change them with this command.

Note

For more information about updating special categories, see "SpecialCatMake" (p. 145).

Example

This example turns off the RollUp attribute for the special category Current Month.

```
SpecialCatUpdate "Current Month" RollUp False
```

SubDimRootMake

Creates a subdimension or updates the category at the root of an existing one.

UI Equivalent

Modifying the property sheet of the category that is at the root of the subdimension, if the subdimension exists. Create/Delete Subdimension (Diagram menu), when a category is selected, for a new subdimension.

Syntax

SubDimRootMake *objCat* [**Dimension** *objDim*] [**Parent** *objLevel*] [**Drill** *objDrillCat*] [**Levels** *objLevel*] [*catopts*]

Argument	Description
SubDimRootMake *objCat*	Specifies the category that is to be at the root of the subdimension. *objCat* can be the object name, object identifier, or both.

Argument	Description
Dimension *objDim*	When creating a subdimension or updating the root category of a subdimension, this is used, if necessary, to uniquely identify it. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDim* can be the object name, object identifier, or both.
Parent *objLevel*	When creating a subdimension, this specifies the parent and must be included. *objLevel* can be object name or object identifier.
Drill *objDrillCat*	When creating a subdimension, this specifies the drill category and is optional. *objDrillCat* can be the drill category object name, object identifier, or both.
Levels *objLevel*	When creating a subdimension, this specifies its level and must be included. *objLevel* can be the object name or object identifier.
catopts	Optional parameters that describe objCat in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "catopts" (p. 177). If the category exists, previously set options are retained unless you change them with this command.

Note

The category acting as the root category of the subdimension is a regular category, and has the attributes of a regular category.

If you create a subdimension in the user interface and save the model as .mdl to generate MDL statements for the subdimension, a new **SubDimRootMake** statement replaces the **CatMake** statement for the category; a new **DrillCatMake** and **LevelMake** statement are created and placed immediately after the **SubDimRootMake** statement; and all of the categories in the subdimension remain but have new parents, drills and levels.

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example defines the root category 1996 for the subdimension Years.

```
SubDimRootMake "1996" Dimension "Years" Parent "By Time"
Drill "By Time" Levels "Year" Label "1996"
```

SubDimRootUpdate

Updates the category that is at the root of a subdimension.

UI Equivalent

Modifying the property sheet of the category that is at the root of the subdimension.

Syntax

SubDimRootUpdate *objCat* **Dimension** *objDim* [*catopts*]

Argument	Description
SubDimRootUpdate *objCat*	Updates the category that is at the root of the subdimension. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	Specifies the dimension in which the subdimension is located. *objDim* can be the object name, object identifier, or both. This argument is mandatory.
catopts	Optional parameters that describe the category in greater detail. For the complete list of options, see "catopts" (p. 177). Previously set options are retained unless you change them with this command.

Note

For more information about subdimensions, see "SubDimRootMake" (p. 147).

Example

This example modifies the category Sport Wear, which is at the root of a subdimension, by adding the description Sport Wear.

```
SubDimRootUpdate "Sport Wear" Dimension "Products" Parent
"GO Sport Line" Drill "By Product Line"  Levels "Product Type"
CatInfo " Sport Wear"
```

SummarizeCat

Summarizes the data for a specified category's descendants in any PowerCube that is created using the specified dimension view.

UI Equivalent

Summarize (Diagram menu), when a category is selected.

Syntax

SummarizeCat *objView* [**Dimension** *objDim*] **Category** *objCat*

Argument	Description
SummarizeCat *objView*	Specifies the dimension view. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Must be specified if the view is not referenced by object identifier. *objDim* can be the object name, object identifier, or both.
Category *objCat*	Specifies the category to be summarized. *objCat* can be the object name, object identifier, or both.

Note

Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Category Names" (p. 20).

Example

This example summarizes the Belgium category in the Europe dimension view.

```
SummarizeCat "Europe" Dimension "Locations" Category
"Belgium"
```

SummarizeLevel

Summarizes a level in a view.

UI Equivalent

Summarize (Diagram menu), when a level is selected.

Syntax

SummarizeLevel *objView* [**Dimension** *objDim*] [**Drill** *objDrill*] **Levels** *objLevel*

Argument	Description
SummarizeLevel *objView*	Specifies the view. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Must be specified if the view is not referenced by object identifier. *objDim* can be the object name, object identifier, or both.
Drill *objDrill*	Is specified, if necessary, to uniquely identify the view. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21). *objDrill* can be the object name, object identifier, or both.

Argument	Description
Levels *objLevel*	Specifies the level to be summarized. *objLevel* can be the object name, object identifier, or both.

Example

This example summarizes the Branch Code level in the Europe dimension view.

```
SummarizeLevel "Europe" Dimension "Locations" Drill "By
Region" Levels "Branch Code"
```

SyncArchitectSource

Keeps the SQL for an Architect data source current with any settings you may have changed in your Transformer model.

UI Equivalent

None.

Syntax

SyncArchitectSource [*objArchDataSource*]

Argument	Description
SyncArchitectSource *objArchDataSource*	*objArchDataSource* can be the object name, object identifier, or both of the Architect data source for which you want to update the SQL. You don't have to specify the data source if it is the only one in your model.

UpdateForwardReference

Resolves forward referencing problems which may occur when you change a model by using a verb in the model definition file. For more information, see "Change Models with MDL" (p. 14).

UI Equivalent

None.

Syntax

UpdateFowardReference

Note

Processing times may be slower when you use UpdateForwardReference.

UpdatePowerCubes

Allows you to update the metadata in the .mdc file(s) for one or more existing PowerCubes without updating the data in the .mdc file.

UI Equivalent

Update PowerCubes or Update Selected PowerCubes (Run menu).

Syntax

UpdatePowerCubes [*objCube*] [*updatepowercubeopts*]

Argument	Description
UpdatePowerCubes *objCube*	Updates the PowerCube *objCube*. *objCube* can be the object name, object identifier, or both. If no PowerCube is specified, all are updated.
updatepowercubeopts	Optional parameters that describe the PowerCube in greater detail. These options are: **DrillThrough** {**True**|**False**} **Objects** {**True**|**False**} **UserClasses** {**True**|**False**} **CurrencyConversion** {**True**|**False**}

UserClassAdd

Add a user class to the User Classes list.

UI Equivalent

No equivalent. User Classes appear in the User Classes list after you specify a namespace in the Access Manager Configuration box in the Authentication tab of the Model Properties dialog box (File menu.)

Syntax

UserClassAdd *objUserClass* [**ID** *AccessManagerID*] [**DimensionView** *objDim objView*] [**MeasureInclude** *objMeasures meaopt*]

Argument	Description
UserClassAdd *objUserClass*	Creates the user class *objUserClass*. *objUserClass* must be the object name and can include the object identifier.

Argument	Description

ID *AccessManagerID*	Specifies the Access Manager ID for the user class. This ID is created by Access Manager and may change whenever Access Manager is modified. You can repeat this argument and insert a list of Access Manager IDs, as in:
	`Id 1172426465`
	`Id 1172426467`

DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in
	`DimensionView "Years"`
	`"All Categories" DimensionView "Products" "All Categories"`

MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. meaopt must be one of **Yes**, **Indirect** or **No**. Repeat this argument to specify multiple measures, as in
	`MeasureInclude`
	`159 Yes MeasureInclude 175 No`

Note

For more information about creating user classes, see "UserClassMake" (p. 154).

UserClassDelete

Deletes a user class from the User Classes list.

UI Equivalent

No equivalent. User Classes appear in the User Classes list after you specify a namespace in the Access Manager Configuration box in the Authentication tab of the Model Properties dialog box (File menu.)

Syntax

UserClassDelete *objUserClass*

Argument	Description
UserClassDelete *objUserClass*	Deletes the user class *objUserClass*. *objUserClass* can be the object name, object identifier, or both.

Example

This example deletes the user class John Smith.

```
UserClassDelete "John Smith"
```

UserClassListUpdate

Defines the structure of a user class.

UI Equivalent

None. This information is defined in Access Manager.

Syntax

UserClassListUpdate *objUserClass* **StartList** [*objUserClass*] **EndList**

Argument	Description
UserClassListUpdate *objUserClass*	Specifies the user class for which a structure is being defined. *objUserClass* can be the object name, object identifier, or both.
StartList [*objUserClass*]	Specifies one or more user classes that are members of the user class specified above. *objUserClass* can be the object name or object identifier.

Note

For more information about user classes, see "UserClassMake" (p. 154).

Each user class must have a UserClassListUpdate statement. If the user class has members, its direct descendants are listed in its UserClassListUpdate statement. If the user class has no sub-classes, the list is blank.

Example

This example defines the structure of the user class Managers. Managers has two members, John Smith and Jenny Tyne.

```
UserClassListUpdate "Managers" StartList "John Smith" "Jenny Tyne" EndList
```

This example defines the structure of the user class John Smith. It has no members, and so the list is empty.

```
UserClassListUpdate "John Smith" StartList EndList
```

UserClassMake

Creates a user class or updates an existing one in the User Classes list.

UI Equivalent

No equivalent. User Classes appear in the User Classes list after you specify a namespace in the Access Manager Configuration box in the Authentication tab of the Model Properties dialog box (File menu.)

Syntax

UserClassMake *objUserClass* [**ID** *AccessManagerID*] [**DimensionView** *objDim objView*] [**Measure-Include** *objMeasures meaopt*]

Argument	Description
UserClassMake *objUserClass*	Creates the user class *objUserClass* or modifies it if it exists. *objUserClass* can be the object name, object identifier, or both. Include the object name if the user class does not exist.

ID *AccessManagerID*	Specifies the Access Manager ID for the user class. This ID is created by Access Manager and may change whenever Access Manager is modified. You can repeat this argument and insert a list of Access Manager IDs, as in:
	`Id 1172426465`
	`Id 1172426467`

DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in
	`DimensionView "Years"`
	`"All Categories" DimensionView "Products" "All Categories"`

MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of **Yes**, **Indirect**, or **No**. Repeat this argument to specify multiple measures, as in
	`MeasureInclude`
	`159 Yes MeasureInclude 175 No`

Note

In Transformer, each user class can be identified in three ways: by object identifier, object name, and Access Manager ID. Since the Access Manager ID can be changed by Access Manager, it is safer to use the object name or object identifier.

The ID attribute is an encrypted value from Access Manager. If this value is zero for any user class, Transformer will attempt to connect to Access Manager to get the correct value.

Example

This example defines the Managers user class. The structure of Managers is defined by its UserClassListUpdate statement.

```
UserClassMake "Managers" Id 3492794712 DimensionView
"Years" "All Categories" DimensionView "Products" "All Categories"
DimensionView "Locations" "All Categories" DimensionView "Channels"
"All Categories" DimensionView "Margin Ranges" "All Categories"
MeasureInclude "Revenue" Yes MeasureInclude "Product Cost" Yes MeasureInclude
"Product Plan" Yes MeasureInclude "Expense Plan" Yes MeasureInclude
"Quantity Sold" Yes MeasureInclude "Profit Margin %" Yes MeasureInclude
"Revenue/Employee" Yes MeasureInclude "SalesRep Plan" Yes MeasureInclude
"Staff Count" Yes MeasureInclude "SalesRep Count" Yes
```

UserClassUpdate

Updates an existing user class object in the User Classes list.

UI Equivalent

No equivalent. User Classes appear in the User Classes list after you specify a namespace in the Access Manager Configuration box in the Authentication tab of the Model Properties dialog box (File menu.) .

Syntax

UserClassUpdate *objUserClass* [**ID** *AccessManagerID*] [**DimensionView** *objDim objView*] [**MeasureInclude** *objMeasure meaopt*]

Argument	Description
UserClassUpdate *objUserClass*	Specifies the user class to update. *objUserClass* can be the object name, object identifier, or both.
ID *AccessManagerID*	Specifies the Access Manager ID for the user class. This ID is created by Access Manager and may change whenever Access Manager is modified. You can repeat this argument and insert a list of Access Manager IDs, as in `Id 1172426465 Id 1172426467`
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`

Argument	Description
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of **Yes**, **Indirect**, or **No**. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Note

For more information about updating user classes, see "UserClassMake" (p. 154).

Example

This example adds a custom dimension view and excludes a measure for the user class John Smith.

```
UserClassUpdate "John Smith" DimensionView "Locations"
"NAmerica" MeasureInclude "Revenue" No
```

ViewAdd

Creates a dimension view. If the object exists, an error message is issued.

UI Equivalent

Add New View in the menu that is accessed by right-clicking a dimension view.

Syntax

ViewAdd *objView* [**Dimension** *objDim*] [**Type** *viewtype*] [*viewopts*]

Argument	Description
ViewAdd *objView*	Creates the dimension view *objView*. *objView* must be the object name and can include the object identifier.
Dimension *objDim*	Is specified to locate the dimension view within a dimension. *objDim* can be the object name, object identifier, or both.
Type *viewtype*	Specifies the type of view. *viewtype* can be **All**, **Omit** or **Custom**. If not specified, the default is Custom.
viewopts	Optional parameters that describe the view in greater detail. Some options are set by default if you don't set them manually. For the complete list of options, see "viewopts" (p. 235).

Example

This example creates the dimension view Europe for the Locations dimension and applies an apex.

```
ViewAdd "Europe" Dimension "Locations" ViewUserClass
"" Apex "Europe"
```

ViewDelete

Deletes a dimension view.

UI Equivalent

Delete Item in the menu that is accessed by right-clicking a dimension view.

Syntax

ViewDelete *objView* [**Dimension** *objDim*]

Argument	Description
ViewDelete *objView*	Deletes the dimension view *objView*. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Must be specified if the view is not referenced by object identifier. *objDim* can be the object name, object identifier, or both.

Note

You cannot delete a default view.

Example

This example deletes the dimension view Europe.

```
ViewDelete "Europe" Dimension "Locations"
```

ViewListUpdate

Re-orders a dimension view list.

UI Equivalent

Dragging and dropping dimension views.

Syntax

ViewListUpdate [**Dimension** *objDim*] **StartList** *objViews* **EndList**

Argument	Description
Dimension *objDim*	Must be specified if the views are not referenced by object identifier. *objDim* can be the object name, object identifier, or both.
StartList *objViews*	Lists the dimension views in the desired order. The dimension views can be identified by object name or object identifier.

Note

You cannot move default views. The default views All Categories and Omit Dimensions are always first and second in the list.

Example

This example moves the views Far East and Europe to the top of the user-created views in the dimension view list. (Default views are always first.)

```
ViewListUpdate Dimension "Locations" StartList "Far East"
"Europe" EndList
```

ViewMake

Creates a dimension, user class view or updates an existing one.

UI Equivalent

Options (Diagram menu), when a view is selected, if the view exists. Add New View in the menu that is accessed by right-clicking a dimension view.

Syntax

ViewMake *objView* [**Dimension** *objDim*] [**Type** *viewtype*] [*viewopts*]

Argument	Description
ViewMake *objView*	Creates the dimension view *objView* or modifies it if it exists. *objView* can be the object name, object identifier, or both. Include the object name if the dimension view does not exist.
Dimension *objDim*	When creating a dimension view, this specifies the dimension in which it is placed. When updating a view, this must be specified if the view is not referenced by object identifier. *objDim* can be the object name, object identifier, or both.
Type *viewtype*	Specifies the type of view. *viewtype* can be **All**, **Omit**, or **Custom**. If the view does not exist, the default is Custom.

Argument	Description
viewopts	Optional parameters that describe the view in greater detail. Some options are set by default if you don't set them manually.For the complete list of options, see "view-opts" (p. 235). If the view exists, previously set options are retained unless you change them with this command.

Example

This example defines the dimension view Europe, which restricts the view to data pertaining to Europe.

```
ViewMake "Europe" Dimension "Locations" ViewUserClass
"" Apex "Europe"
```

This example shows the two default views for the Dimension Products. Transformer creates these default views for each dimension in the model.

```
ViewMake "All Categories" Dimension "Products" Type All
ViewUserClass ""
```

```
ViewMake "Omit Dimension" Dimension "Products" Type Omit
ViewUserClass ""
```

ViewUpdate

Updates an existing dimension, user class view.

UI Equivalent

Options (Diagram menu), when a view is selected.

Syntax

ViewUpdate *objView* [**Dimension** *objDim*] [**Type** *viewtype*] [*viewopts*]

Argument	Description
ViewUpdate *objView*	Specifies the view to update. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Must be specified if the view is not referenced by object identifier. *objDim* can be the object name, object identifier, or both.
Type *viewtype*	Specifies the type of view. viewtype can be **All**, **Omit**, or **Custom**. If not specified, the existing view type is retained.

Argument	Description
viewopts	Optional parameters that describe the view in greater detail. For the complete list of options, see "viewopts" (p. 235). Previously set options are retained unless you change them with this command.

Note

You cannot update default views.

Example

This example applies a summarize operation to the Belgium category in the Europe dimension view.

```
ViewUpdate "Europe" Dimension "Locations" Summary "Belgium"
```

For more information about updating views, see "ViewMake" (p. 159).

Chapter 6: MDL Options

This chapter describes the options that you can use to define objects in MDL.

The options are grouped as lists. Each list applies to one or more objects. The syntax conventions used in this chapter are described in "Syntax Conventions" (p. 53).

Most options correspond to a setting in a property sheet, and this UI equivalency is given for most options.

For more information about any of these options, see the What's This? Help in the relevant property sheet.

apparchopts

Applies to: ArchitectDataSourceAdd, ArchitectDataSourceMake, ArchitectDataSourceUpdate.

ArchitectDataSource

UI Equivalent:	Specifying the identity of an Architect model to check for packages
Syntax:	**ArchitectDataSource** *objArch*
Where:	*objArch* can be the object name, object identifier, or both.
Example:	`ArchitectDataSource Model1`

ArchitectModelName

UI Equivalent:	New Model and New Data Source wizards, Architect Model box. Also, setting the Name of the Architect model on the Architect Model property sheet.
Syntax:	**ArchitectModelName** *string*
Where:	*string* is the name of the Architect model that you want to use.
Example:	`ArchitectModelName "GreatOutdoors"`

ArchitectMultiProcess

UI Equivalent:	Architect Model property sheet, selecting Enable Multi-processing on the Timing tab

Syntax:	**ArchitectMultiProcess**
Example:	`ArchitectMultiProcess`

ArchitectSetCurrentPeriod

UI Equivalent:	Architect Model property sheet, selecting Sets the Current Period on the Timing tab
Syntax:	**ArchitectSetCurrentPeriod**
Example:	`ArchitectSetCurrentPeriod`

ArchitectServer

UI Equivalent:	Architect Model property sheet, selecting Server in the Data Source Location area (General tab)
Syntax:	**ArchitectServer**
Example:	`ArchitectServer`

ArchitectSpeed

UI Equivalent:	Architect Model property sheet, selecting Maximize Data Access Speed in the Uniqueness Verification area (Timing tab)
Syntax:	**ArchitectSpeed**
Example:	`ArchitectSpeed`

ArchitectTiming

UI Equivalent:	Architect Data Source property sheet, Timing tab, Timing section
Syntax:	**ArchitectTiming** *datasourcetiming*

Where:	*datasourcetiming* is one of **PopNoCreateNo**, **PopYesCreateNo**, **PopNoCreateDefault**, **PopYesCreateDefault**, **PopNoCreateYes**, **PopYesCreateYes** The components of these codes relate to the property sheet settings as follows: **PopYes:** Generate Categories is selected **PopNo:** Generate Categories is not selected **CreateDefault:** PowerCube Creation box is selected and Default option button is selected **CreateYes:** PowerCube Creation box is selected and Create the PowerCubes option button is selected **CreateNo:** PowerCube Creation box is selected and Generate Categories Only option button is selected
Example:	`ArchitectTiming PopYesCreateNo`

Package

UI Equivalent:	New Model and New Data Source wizards, Package Box
Syntax:	**Package** *string*
Where:	*string* is the name of the Architect package that you want to use.
Example:	`Package "Default"`

RemoteProcessing

UI Equivalent:	Architect Model property sheet, selecting Remote in the Processing area (General tab)
Syntax:	**RemoteProcessing**
Example:	`RemoteProcessing`

appqueryopts

Applies to: DataSourceAdd, DataSourceMake, DataSourceUpdate, and NewModel

AccessManager

UI Equivalent:	Model property sheet, Authentication tab, Include Access Manager Auto-Accesses in the Model check box
Syntax:	**AccessManager** {**True**|**False**}
Example:	`AccessManager True`

AccessManagerLogin

UI Equivalent:	Access Manager Login, Access Manager dialog box (accessed from Add User Classes button in Model property sheet, Authentication tab)
Syntax:	**AccessManagerLogin** *string*

AccessManagerNamespace

UI Equivalent:	Model property sheet, Authentication tab, Access Manager Configuration box
Syntax:	**AccessManagerNamespace** *string*

AccessManagerPassword

UI Equivalent:	Access Manager Password, Access Manager dialog box (accessed from Add User Classes button in Model property sheet, Authentication tab)
Syntax:	**AccessManagerPassword** *string*

AccessManagerUserClass

UI Equivalent:	Model property sheet, Authentication tab, User Class box
Syntax:	**AccessManagerUserClass** *string*

AppInfo

UI Equivalent:	Model property sheet, General tab, Description
Syntax:	**AppInfo***string* [*string*...]

Where:	Each string may be up to 256 characters and the total description may be up to 4,095 characters.
Example:	`AppInfo "This model created by DBX for HHY."`

CharacterSet

UI Equivalent:	Data Source property sheet, Source tab, Character Set box
Syntax:	**CharacterSet** *charset*
Where:	*charset* is one of **Default**, **ANSI**, **OEM** or **Multibyte**.
Example:	`CharacterSet Multibyte`

ClientStamp

UI Equivalent:	None
Syntax:	**ClientStamp** *bignum*
Example:	`ClientStamp 827358629`
Note:	For internal use only. This option relates to synchronization and should not be manually set.

Columns

UI Equivalent:	None
Syntax:	**Columns** {**True**|**False**}
Where:	**False** indicates that the data is fixed length.
Example:	`Columns True`
Note:	This option relates to the structure of the source data and should not be manually set.

DataRange

UI Equivalent:	Data Source property sheet, Source tab, Table or Range drop-down list

Syntax:	**DataRange** *string*
Example:	`DataRange "reptotal_rng"`
Note:	This option applies only to queries of the following data types: Lotus 1-2-3 Database, Excel Database.

DataSourceIsArchitect

UI Equivalent:	Data source is based on an Architect package
Syntax:	**DataSourceIsArchitect**
Example:	`DataSourceIsArchitect`

DecimalSep

UI Equivalent:	Data Source property sheet, General tab, Decimal Separator box
Syntax:	**DecimalSep** *string*
Where:	*string* is the decimal separator.
Example:	`DecimalSep "."`

EnableMultiProcess

UI Equivalent:	Data Source property sheet, General tab, Enable Multi-Processing check box
Syntax:	**EnableMultiProcess** {**True**|**False**}
Example:	`EnableMultiProcess False`
Note:	This option only applies to IQD and flat file data sources.

ImrName

UI Equivalent:	None
Syntax:	**ImrName** *string*
Where:	*string* is a file name and optional path.

Example:	`ImrName "C:\ProgramFiles\COGNOS\PowerPlay 7.4\ Samples\PowerCube and Reports\bsc_msrs.imr"`
Note:	This option is required for drill-through to an Impromptu report via an IQD data source.

Isolation

UI Equivalent:	Data Source property sheet, Source tab, Isolation level drop-down list
Syntax:	**Isolation** *number*
Where:	*number* is a number from 0 to 6 that corresponds to the drop-down list as follows: **0:** Default **1:** Read Uncommitted **2:** Read Committed **3:** Cursor Stability **4:** Cursor Stability **5:** Phantom Protection **6:** Serializable
Example:	`Isolation 2`
Note:	This option applies only to IQD data sources.

ModelStamp

UI Equivalent:	None
Syntax:	**ModelStamp** *bignum*
Example:	`ModelStamp 827358629`
Note:	This option is for internal use only.

PreSummarized

UI Equivalent:	Data Source property sheet, General tab, Contains Externally Rolled Up Measure Values check box

Syntax:	**PreSummarized** {**True**|**False**}
Example:	`PreSummarized False`

RootUser

UI Equivalent:	Model property sheet, Authentication tab, Add User Classes box
Syntax:	**RootUser** *bignum*

ServerConnection

UI Equivalent:	Model property sheet, Server tab, Connection box
Syntax:	**ServerConnection** *string*
Example:	`ServerConnection "\\Sotr0022\BI"`

Separator

UI Equivalent:	Data Source property sheet, Source tab, Field Delimiter
Syntax:	**Separator** *string*
Example:	`Separator ","`
Note:	This option applies only to delimited-field text and delimited-field text with column titles data sources.

ServerSource

UI Equivalent:	Data Source property sheet, Source tab, Data Source Location options list
Syntax:	**ServerSource** {**True**|**False**}
Where:	**True** is Server and **False** is Local.
Example:	`ServerSource True`

ServerStamp

UI Equivalent:	None
Syntax:	**ServerStamp** *bignum*
Note:	For internal use only.

SetCurrent

UI Equivalent:	Data Source property sheet, General tab, Sets the Current Period check box
Syntax:	**SetCurrent** {**True**|**False**}
Example:	`SetCurrent False`

Source

UI Equivalent:	Data Source property sheet, Source tab, Local Data File
Syntax:	**Source** *string*
Where:	*string* is a file name and optional path.
Example:	`Source "c:installation_directory>\powercubes and reports\prodinfo.csv"`

SourceInfo

UI Equivalent:	Data Source property sheet, Description tab
Syntax:	**SourceInfo** *string* [*string*...]
Where:	Each string may be up to 256 characters, and the total description may be up to 4,095 characters.
Example:	`SourceInfo "This data source created by DBX."`

SourceServerPath

UI Equivalent:	Data source property sheet, Source tab, Server Data File box
Syntax:	SourceServerPath *string*

Where:	*string* is the server data file name and optional path.
Note:	This option applies only to data sources located on a server, indicated by the setting ServerSource True.

SourceSignonList

UI Equivalent:	Signons in the Signons list
Syntax:	SourceSignonList *signonlist* **EndList**
Where:	*signonlist* is the list of signons for a data source. Signons can be the object name, object identifier, or both.

Example:	`DataSourceMake 103 "GO Market Research" Separator " " SourceType DataSource`
	`DecimalSep " " Thousandsep " " Columns True Timing PopYesCreateDefault`
	`Source ".\samples\powerplay\cubes and reports\go market research.iqd"`
	`...`
	`SourceSignonList 105 EndList`

Note:	This option applies only to data sources requiring signons.

SourceType

UI Equivalent:	Data Source property sheet, Source tab, Source Type
Syntax:	**SourceType** *sourcetype*
Where:	*sourcetype* is one of **Architect**, **FlatFileColNames**, **FlatFile**, **DataSource**, **Access**, **AccessQuery**, **DBase**, **Paradox**, **Lotus123Crosstab**, **ExcelCrosstab**, **Lotus123Database**, **ExcelDatabase**, **PowerHousePortable**, **FoxPro**, **Clipper**, **FixedASCII**, **FixedASCIINoCrlf**, **Dictionary**.
Example:	`SourceType ExcelDatabase`
Note:	**DataSource** is the keyword for an IQD data source. Architect is not supported in all versions.

Speed

UI Equivalent:	Data Source property sheet, General tab, Uniqueness Verification
Syntax:	**Speed** {**True**|**False**}
Where:	**True** is Maximize Data Access Speed and **False** is Verify Category Uniqueness.
Example:	`Speed False`

SQL

UI Equivalent:	None
Syntax:	**SQL** '*string* [*string*...]'
Where:	The apostrophe (') is the delimitation character for any SQL statement. It must appear at the beginning and end of every line of SQL. The maximum number of characters on any line is 256.
Example:	This is the SQL string for the Main (IQD) data source:
Note:	This option cannot be changed by the user. Any changes must be made in the source program.

```
SQL 'select T1."ORDER_DT" as c1,' 'T2."PROD_NO" as
c2, T1."REP_NO" as c3,' 'T1."CUST_NO" as c4,
T2."QTY" as c5,' '(T2."QTY" * T2."PRI' 'CE") as
c6,' '(T2."QTY" * T3."PROD_COST") as c7,' '(CASE
' 'WHEN ((((T2."QTY" * T2."PRICE") -' '(T2."QTY"
* T3."PROD_COST")) / (T2."QTY" * ' 'T2."PRICE"))
<= 0.19) ' 'THEN ("'Under 20%')' 'WHEN ((((T2."QTY"
* T2."PRICE") - ' '(T2."QTY" * T3."PROD_COST")) /
' '(T2."QTY" * T2."PRICE")) BETWEEN 0.2 AND '
'0.65) ' 'THEN ("'20% - 65%')' 'WHEN ((((T2."QTY"
* T2."PRICE") - ' '(T2."QTY" * T3."PROD_COST")) '
'/ (T2."QTY" * T2."PRICE")) >= 0.66) ' 'THEN ('
"'Over 65%') ' 'ELSE ('ERROR') ' 'END) as c8 from
"ORDER" T1,' '("PRODUCT" T3 left outer join
"ORDRDETL" T2 ' 'on T2."PROD_NO" = T3."PROD_NO")
where ' '(T2."ORDER_NO" = T1."ORDER_NO") ' 'order
by c1 asc,c3 asc,c4 asc,c2 asc'
```

Stamp

UI Equivalent:	None
Syntax:	**Stamp** *number*
Example:	`Stamp 890332424`
Note:	This option is for internal use only.

SynchroCycle

UI Equivalent:	None
Syntax:	**SynchroCycle** *bignum*
Note:	This option is for internal use only.

Synchronize

UI Equivalent:	None
Syntax:	**Synchronize** {**True**|**False**}
Example:	`Synchronize True`
Note:	This option is for internal use only.

SynchroStamp

UI Equivalent:	None
Syntax:	**SynchroStamp** *bignum*
Note:	This option is for internal use only.

ThousandSep

UI Equivalent:	Data Source property sheet, General tab, 1000 separator
Syntax:	**ThousandSep** *string*
Example:	`ThousandSep ","`

Timing

UI Equivalent:	Data Source property sheet, General tab, Timing section
Syntax:	**Timing** *datasourcetiming*
Where:	*datasourcetiming* is one of **PopNoCreateNo**, **PopYesCreateNo**, **PopNoCreateDefault**, **PopYesCreateDefault**, **PopNoCreateYes**, **PopYesCreateYes** The components of these codes relate to the property sheet settings as follows: **PopYes:** Generate Categories is selected **PopNo:** Generate Categories is not selected **CreateDefault:** PowerCube Creation box is selected and Default option button is selected **CreateYes:** PowerCube Creation box is selected and Create the PowerCubes option button is selected **CreateNo:** PowerCube Creation box is selected and Generate Categories Only option button is selected
Example:	`Timing PopYesCreateNo`

UpdateCycle

UI Equivalent:	None
Syntax:	**UpdateCycle** *bignum*
Note:	This option is for internal use only.

UserClasses

UI Equivalent:	Model property sheet, Authentication tab, Add user classes
Syntax:	**UserClasses** {**True**|**False**}
Example:	`UserClasses False`

Version

UI Equivalent:	Version in the About Transformer dialog box, accessed from the Help menu

Syntax:	**Version** *string*
Example:	`Version "7.4 1012"`
Note:	This option should not be set manually.

assocopts

Applies to: AssociationAdd, AssociationMake, AssociationUpdate, CurrencyTableAdd, CurrencyTable Make, CurrencyTableUpdate, DimensionAdd, DimensionMake, DimensionUpdate, LevelAdd, LevelMake, LevelUpdate, MeasureAdd, MeasureMake, MeasureUpdate

AssociationContext

UI Equivalent:	The object that you are making an association for.
Syntax:	**AssociationContext** *objcontext*
Where:	*objcontext* can be the object name or object identifier or both
Example:	`AssociationContext 2979 "Branch Code"`

AssociationReferenced

UI Equivalent:	Object property sheet, Source tab, clicking the ellipsis at the end of a row, choosing the Name of the column or attribute to associate.
Syntax:	**AssociationReferenced** *objref*
Where:	*objref* is the name of the association
Example:	`AssociationReferenced "Branch Code"`

AssociationRole

UI Equivalent:	Object property sheet, Source tab, choosing a role for an association.
Syntax:	**AssociationRole** *objrole*

Where:	*objrole* is one of **Role_Catcode**, **Role_Description**, **Role_Drillthrough**, **Role_Label**, **Role_Orderby**, **Role_Source**, or **Role_Tag**. For Currency tables, *objrole* can be one of **Role_CountryCode**, **Role_Date**, **Role_Label**, or **Role_Rate**.
Example:	`AssociationRole Role_Label`

AssociationType

UI Equivalent:	Object property sheet, Source tab, clicking the ellipsis at the end of a row, clicking More, and choosing the Type of the column or attribute to associate.
Syntax:	**AssociationType** *objtype*
Where:	*objtype* is one of **Type_Architect**, **Type_PowerCube**, or **Type_Query**.
Example:	`AssociationType Type_Architect`
Note:	The **Type_Query** option is used for IQD data sources.

catopts

Applies to: CatAdd, CatMake, CatUpdate, DrillCatMake, RootCatMake, RootCatUpdate, SpecialCatAdd, SpecialCatMake, SpecialCatUpdate, SubDimRootMake, SubDimRootUpdate

Blanks

UI Equivalent:	None
Syntax:	**Blanks** {**True**|**False**}
Where:	True means the category is blank.
Example:	`Blanks False`
Note:	Applies to regular categories only.

Calc

UI Equivalent:	<Dimension> Category Calculation Definition dialog box, accessed from the <Dimension> property sheet, Calculation tab
Syntax:	**Calc** *calculation*
Where:	*calculation* is a mixture of *objects* and *keywords*. objects are object name followed by the at sign (@), the object type and, optionally, the object identifier. The object type can be Category, Level or Drill. For example, "GO Water Bottle@Category@4805"

Example:	Example 1:
	`Calc Percent-Growth ( "Prior YTD" , "YTD" )`
	Example 2:
	`Calc Share ("GO Water Bottle@Category@4805, "Sport Wear@Category@4799")`

Note:	*keywords* are given in the following table:

Calculation keyword	Type	Description
/	Number	Divides
-	Date or Number	Minus
+	Character, Date, or Number	Plus
^	Number	Returns a number raised to the power of a second number
Accumulate	Number	Computes the running total
AccPOB	Number	Computes the running total as a percentage of the total
Average	Number	Averages
Max	Number	Maximum instance
Min	Number	Minimum instance

Calculation keyword	Type	Description
*	Number	Multiplies
Percent-Growth	Number	Computes the percentage change parameter 2 is of parameter 1
Share		Computes the share that parameter 2 is of parameter 1
Change		Computes the difference between two parameters (parameter 2 minus parameter 1)

CalcDef

UI Equivalent:	None. This references the dimension calculation definition that creates the calculation for the category, if any. It creates the link between the DimCalcDef object and the category. For more information, see "DimCalcDefDelete" (p. 107).
Syntax:	**CalcDef** *objDimCalcDef*
Where:	*objDimCalcDef* can be the object name or object identifier
Example:	`CalcDef 6375`
Note:	The option CalcDef can forward-reference the DimCalcDef object. This means that a CatMake statement that references a calculation can appear in the MDL file before the calculation object is defined in a DimCalcDefMake statement.

CatInfo

UI Equivalent:	Category property sheet, Description tab
Syntax:	**CatInfo** *string* [*string*...]
Where:	Each string may be up to 256 characters and the total description may be up to 4,095 characters.
Example:	`CatInfo "Exceptional money makers! Why are these entities doing so well? Examine and see if you can apply strategies to entities of lower ranges."`

Note:	CatInfo applies to regular, special, drill and root categories.

ContextLevel

UI Equivalent:	Special Category property sheet, Relative Time tab, Context Period box
Syntax:	**ContextLevel** *string*
Where:	*string* is context-sensitive
Example:	`ContextLevel "Quarter"`

ContextOffset

UI Equivalent:	Special Category property sheet, Relative Time tab, Context Offset box
Syntax:	**ContextOffset** *number*
Example:	`ContextOffset -1`

Current

UI Equivalent:	None
Syntax:	**Current**
Example:	`Current`
Note:	This option sets a date category to the current period.

Date

UI Equivalent:	None
Syntax:	**Date** *date*
Where:	*date* is the start of the date period.
Example:	`Date 19990101`
Note:	This option must be set for date categories.

DateDrill

UI Equivalent:	Special category property sheet, Relative Time tab, Drill Category
Syntax:	**DateDrill** *objDrillCat*
Where:	*objDrillCat* is the object name or object identifier.
Example:	`DateDrill "By Time"`

ExtraWeek

UI Equivalent:	Drill Category property sheet, Time tab, Add an Extra Week box
Syntax:	**ExtraWeek** *extraweek*
Where:	*extraweek* is one of **None**, **7only**, **6orMore**, **5orMore**, **4orMore**.
Example:	`ExtraWeek 4orMore`

Filtered

UI Equivalent:	Exclude from the Diagram menu when a category is selected
Syntax:	**Filtered** {**True**|**False**}
Example:	`Filtered True`

Format

UI Equivalent:	Property sheet for a calculated category, Format tab, Format box and Decimal Places box.
Syntax:	**Format** *string*
Where:	*string* is the format, followed by a tilde (~) and the number of decimal places.
Example:	This example sets the format to #,##0 and sets the decimal places to 2: `Format "#,##0~2"`

Inclusion

UI Equivalent:	Category property sheet, General tab, Inclusion box
Syntax:	**Inclusion** *inclusion*
Where:	*inclusion* is one of **Default**, **Generate**, **Suppress**, **Retain**, or **Filtered**. These MDL codes relate to settings in the user interface as follows: No setting for Inclusion: Default (when needed) **Default:** Default (when needed) **Generate:** Always Include **Suppress:** Suppress (Diagram menu) **Retain:** Include When Needed, or Suppress Blank Categories **Filtered:** Diagram menu, Exclude
Example:	`Inclusion Suppress`
Note:	Applies to regular, special, and drill categories.

IsKeyOrphanage

UI Equivalent:	Category property sheet, General tab, Category is Orphanage check box
Syntax:	**IsKeyOrphanage** {**True**|**False**}
Example:	`IsKeyOrphanage False`

IsTruncated

UI Equivalent:	None. This is an internal flag.
Syntax:	**IsTruncated** {**True**|**False**}
Example:	`IsTruncated True`

Label

UI Equivalent:	Category property sheet, Category Label box
Syntax:	**Label** *stringwithnewline*

Example:	`Label "Day Tripper"`

LastUse

UI Equivalent:	Category property sheet, General tab, Last Used On box
Syntax:	**LastUse** *date*
Where:	*date* is in the format YYYYMMDD.
Example:	`Lastuse 19980324`

NewPartition

UI Equivalent:	Category property sheet, General tab, Partition Number box
Syntax:	**NewPartition** *uns*
Example:	`NewPartition 1`
Note:	Applies to regular categories only

HideValue

Syntax:	**HideValue True**
Example:	`RootCatUpdate "Products" Dimension "Products" Inclusion Generate ... HideValue True`
Note:	Used in Transformer Version 7.3 and above to hide meaningless values at the root or any other category level in a scenario dimension. We recommend that this option be used with `DimDefaultCategory` to specify a new default category (lower level) where the cube will open.

Orphanage

UI Equivalent:	None
Syntax:	**Orphanage**
Note:	**Orphanage** appears as an option when the category is an orphanage. This option is for internal use only.

PartialWeek

UI Equivalent:	Drill Category property sheet, Time tab, Partial Weeks box
Syntax:	**PartialWeek** *partialweek*
Where:	*partialweek* is one of **First**, **Last**, **Largest**, **Split**, **SplitIfGreater**, or **None**. These MDL codes relate to settings in the user interface as follows.
Example:	`PartialWeek SplitIfGreater`

MDL code	UI equivalent
First	First Period
Last	Last Period
Largest	Largest Period
Split	Always Split
SplitIfGreater	Split > 1 Day
None	

Primary

UI Equivalent:	This option lists the children attached to a special category.
Syntax:	**Primary** *objCat*
Where:	*objCat* can be the object name or object identifier.
Example:	`Primary "19971201-19971231"`

PrimaryDrill

UI Equivalent:	Drill Category property sheet, General tab, Primary Drill Category box
Syntax:	**PrimaryDrill** {**True**|**False**}
Example:	`PrimaryDrill True`

Rollup

UI Equivalent:	Special Category property sheet, General tab, Category Rollup box.
Syntax:	**RollUp** {**True**|**False**}
Example:	`Rollup True`

RunningPeriods

UI Equivalent:	Special Category property sheet, Relative Time tab, Number of Periods box
Syntax:	**RunningPeriods***uns*
Example:	`RunningPeriods 0`

Share

UI Equivalent:	Category property sheet, General tab, Share Category box
Syntax:	**Share** *objCat*
Where:	*objCat* can be the object name or object identifier.
Example:	`Share "Back Packs"`

ShortName

UI Equivalent:	Category property sheet, General tab, Short Name box
Syntax:	**ShortName** *string*
Example:	`ShortName "ENV"`
Note:	Applies to regular, special, drill, and root categories.

Sign

UI Equivalent:	Category property sheet, General tab, Reverse the Sign check box
Syntax:	**Sign** {**True**|**False**}

Where:	**True** means that the sign is reversed.
Example:	`Sign False`
Note:	Applies to regular and special categories.

SourceValue

UI Equivalent:	Category property sheet, General tab, Source Value box
Syntax:	**SourceValue** *stringwithnewline*
Example:	`SourceValue "Environmental Line"`
Note:	Applies to regular categories only.

SplitWeek

UI Equivalent:	None
Syntax:	**SplitWeek** {**True**|**False**}
Example:	`SplitWeek False`
Note:	This option is for internal use only.

Suppressed

UI Equivalent:	Diagram menu, Suppress
Syntax:	**Suppressed** {**True**|**False**}
Example:	`Suppressed True`

TargetLevel

UI Equivalent:	Special Category property sheet, Relative Time tab, Target Period box
Syntax:	**TargetLevel** *string*
Where:	*string* is context-sensitive.
Example:	`TargetLevel "Month"`

TargetOffset

UI Equivalent:	Special Category property sheet, Relative Time tab, Target Offset box
Syntax:	**TargetOffset** *number*
Example:	`TargetOffset -1`

TimeAggregate

UI Equivalent:	Special category property sheet, Relative Time tab, Basic Approach box
Syntax:	**TimeAggregate** [*timeaggregate*]
Where:	*timeaggregate* is one of **None**, **Single**, **ToDate**, **ToDate_Grp**, **Running**, or **Running_Grp**.
Example:	`TimeAggregate ToDate`

ToDateLevel

UI Equivalent:	Special Category property sheet, Relative Time tab, To Date Period
Syntax:	**ToDateLevel** *objLevel*
Where:	*objLevel* is the object name.
Example:	`ToDateLevel "Quarter"`

WeekBegins

UI Equivalent:	Drill Category property sheet, Time tab, Week Begins On box
Syntax:	**WeekBegins** *day*
Where:	*day* is one of **Sunday**, **Monday**, **Tuesday**, **Wednesday**, **Thursday**, **Friday**, or **Saturday**.
Example:	`WeekBegins Sunday`

YearBegins

UI Equivalent:	Drill Category property sheet, Time tab, Year Begins box
Syntax:	**YearBegins** *date*
Where:	*date* is in the format YYYYMMDD
Example:	`YearBegins 19990401`

colopts

Applies to: ColumnAdd, ColumnMake, ColumnUpdate, DataSourceAdd, DataSourceMake, DataSourceUpdate, SourceAttributeAdd, SourceAttributeMake, SourceAttributeUpdate

Calc

UI Equivalent:	Calculation dialog box, which is accessed by clicking the Calculation button on the General tab of the Column property sheet
Syntax:	**Calc** *keywords objects*
Where:	*keywords* are given in the following table. *objects* are object name followed by the at sign (@) and the object identifier; for example, "Revenue@259"
Example:	`Calc "Order Qty" * "Price"`

Calculation keyword	Type	Description
/	Number	Divided by
=	Boolean	Equals
>	Boolean	Greater than
>=	Boolean	Greater than or equal to
<	Boolean	Less than
<=	Boolean	Less than or equal to
(		Left parenthesis

Calculation keyword	Type	Description
-	Date or Number	Minus
*	Number	Multiplied by
+	Character, Date, or Number	Adds numbers or concatenates strings.
^	Number	Returns a number raised to the power of a second number
)		Right parenthesis
Absolute	Number	Converts numbers to their unsigned value
Add-Days	Date	Returns the date resulting from adding a number of days to a date
Add-Months	Date	Returns the date resulting from adding a number of months to a date
Add-Years	Date	Returns the date resulting from adding a number of years to a date
Age	Date	Age
Ceiling	Number	Returns a number rounded to the next highest integer
Char-Length	Number	Returns the number of characters in a string
Date-To-Days-From-1900	Number	Returns the number of days from January 1, 1900 inclusive
Day	Number	Returns the day of the month (1-31) from a date

Calculation keyword	Type	Description
Days-From-1900-To-Date-Time	Date	Returns the date obtained from converting a number of days from January 1, 1900 inclusive to a date
Days-To-End-Of-Month	Date	Returns the number of days to the last day of the month represented by a date
Else	Boolean	Else
First-Of-Month	Date	Returns the first day of a month from a date or datetime
First-Word	Character	Returns the first word of a string
Floor	Number	Returns a number rounded to the next lowest integer
If	Boolean	If
Integer-Divide	Number	Returns the integer obtained by truncating the result of an integer divided by a second integer
IsNull	Character, Date, or Number	Is null
Last-Of-Month	Date	Returns the last day of a month from a date or datetime
Left	Character	Returns a specific number of characters, starting at the left of the string
Lower	Character	Converts uppercase characters to lowercase
Mod	Number	Returns the remainder (modulus) of an integer divided by a second integer

Calculation keyword	Type	Description
Month	Number	Returns the month number from a date, datetime or interval
Months-Between	Number	Returns the integer number of months from a date to a second date
Not	Boolean	Not
Number-To-String	Character	Returns the string representation of a number rounded to the next integer
Or	Boolean	Or
Pack	Character	Returns a string with preceding and trailing spaces removed, and a single space
PHDate-To-Date	Date	Returns the date obtained from converting the PowerHouse date and adding the int century
Position	Number	Returns the starting position of a string in a second string
Reverse	Character	Returns a string with the order of the characters reversed
Right	Character	Returns a specific number of characters, starting at the right of the string
Round-Down	Number	Returns a number rounded down
Round-Near	Number	Returns a number rounded to the nearest value
Round-Up	Number	Returns a number rounded up

Calculation keyword	Type	Description
Round-Zero	Number	Returns a number rounded towards zero
Spread	Character	Returns a string with a space added between each of the original characters
SqRt	Number	Returns the square root of a number
String-To-Number	Number	Converts a string to a number
Substitute	Character	Replaces a substitution character (^) with a string where it is used in another string
SubString	Character	Returns a substring from a string
Then	Boolean	Then
Today	Date	Returns the current date according to the clock in your computer
Trim-Leading	Character	Returns a string with leading spaces removed
Trim-Trailing	Character	Returns a string with trailing spaces removed
Upper	Character	Converts lowercase characters to uppercase
Year	Number	Returns the year from the date
Years-Between	Number	Returns the number of years from a date to another date

Class

UI Equivalent:	Column property sheet, General tab, Data Class

Syntax:	**Class** [*dataclass*]
Where:	*dataclass* is one of the following:
Example:	`Class Quantity`

dataclass	User interface
Default	Unspecified
Description	Text
Date	Date
Quantity	Numeric
Ignore	Ignore
Member	Array Member
No setting	Unspecified

Column

UI Equivalent:	Column property sheet, General tab, Original name
Syntax:	**Column** *objCol*
Where:	*objCol* is the object name.
Example:	`Column "Product Id"`
Note:	If you alter a column name, the object name may differ between MDL and the user interface. MDL takes the object name from the Original Name box in the column property sheet. The user interface takes the object name from the Column Name box. The Column must be specified.

ColumnInfo

UI Equivalent:	Column property sheet, Description tab
Syntax:	**ColumnInfo** *string* [*string*...]

Where:	Each string may be up to 256 characters and the total description may be up to 4,095 characters.
Example:	`ColumnInfo "Gross revenue from products sales"`

Dateconstant

UI Equivalent:	None
Syntax:	**Dateconstant** '*date*'
Example:	`months-between (dateconstant '1999-05-20', today)`
Note:	This option must precede any date constant in a calculation.

DateLevel

UI Equivalent:	Column property sheet, General tab, Degree of Detail (appears only when the Contains Externally Rolled Up Measures check box is selected in the Data Source property sheet)
Syntax:	**DateLevel** *datelevel*
Where:	*datelevel* is one of **None**, **Year**, **Quarter**, **Month**, **Week**, or **Day**.
Example:	`DateLevel Day`

Decimals

UI Equivalent:	None
Syntax:	**Decimals** *uns*
Example:	`Decimals 2`
Note:	This option indicates the number of decimal spaces in the source data and should be set to the actual value, or can be defined in the Measure property sheet, Format tab, Decimal Places box.

Detail

UI Equivalent:	Column property sheet, Time tab, Degree of Detail box

Syntax:	**Detail** *string*
Where:	The codes are context-sensitive and the valid list in each instance appears in the drop-down list.
Example:	`Detail "Unspecified"`

Format

UI Equivalent:	Column property sheet, Time tab, Date Input Format
Syntax:	**Format** [*dateformat*]
Where:	*dateformat* is one of **PreDefined**, **YMD**, **DMY**, **MDY**, **YM**, **MY**, **Y**, **Q**, **M**, **W**, or **D**.
Example:	`Format YMD`

InputScale

UI Equivalent:	Column property sheet, General tab, Input Scale check box (appears only when Contains Externally Rolled Up Measures box in the Data Source property sheet is not selected)
Syntax:	**InputScale** *number*
Where:	*number* is not greater than 16.
Example:	`InputScale 1`

Offset

UI Equivalent:	Column property sheet, General tab, Position box
Syntax:	**Offset***uns*
Example:	`Offset 2`

Origin

UI Equivalent:	None
Syntax:	**Origin** {**Source**|**Generated**|**Calculated**|**Manual**}

Where:	**Generated** is the default. **Source** specifies an IQD data source. **Calculated** specifies a calculated column. **Manual** specifies a newly- created column.
Example:	`Origin Generated`
Note:	This option should not be manully set.

Scale

UI Equivalent:	None
Syntax:	**Scale** *uns*
Example:	`Scale 0`
Note:	This option, which specifies the output scale, is read from the data source or specified in the measure. For more information, see "Scale" (p. 222).

Size

UI Equivalent:	Column property sheet, General tab, Size box
Syntax:	**Size** *uns*
Example:	`Size 1`
Note:	Size only applies when the associated data source's SourceType is set to one of: **PowerHousePortable**, **FixedASCII**, or **FixedASCIINoCrlf**.

Storage

UI Equivalent:	None
Syntax:	**Storage** [*storagetype*]
Where:	*storagetype* is one of **Default**, **Int8**, **Int16**, **Int32**, **Float64**, **Text**, **Character**, or **LongText**.
Example:	`Storage Int32`

Note:	This option relates directly to the source data. It should be set to the actual storage type of the date.

TimeArray

UI Equivalent:	Column property sheet, Array tab, Array Type box
Syntax:	**TimeArray** *timearray*
Where:	*timearray* is one of **Off**, **Quarter**, **Month**, **Month13**, **Quarter6**, **Month18**, **Quarter8**, or **Month24**.
Example:	`TimeArray Quarter`

TimeArrayCol

UI Equivalent:	Column property sheet, Array tab, Date Column box
Syntax:	**TimeArrayCol** *string*
Example:	`TimeArrayCol "Time"`

TimeArrayMonth

UI Equivalent:	Column property sheet, Array tab, Start Month box
Syntax:	**TimeArrayMonth** *uns*
Example:	`TimeArrayMonth 2`

currencyrecordopts

Applies to: CurrencyAdd, CurrencyMake, CurrencyUpdate

BaseCurrency

UI Equivalent:	Currency appears as <Base Default>
Syntax:	**BaseCurrency**
Where:	`BaseCurrency`

Note:	This refers to the default currency used by the model. In the currency property sheet, the name <Base Default> can be modified.

CountryCode

UI Equivalent:	Currency property sheet, Country Code box
Syntax:	**CountryCode** *code*
Where:	*code* is a unque identifier for a country.
Example:	`CountryCode "Can"`

CurrencyCountryLabel

UI Equivalent:	Currency property sheet, Currency Label box
Syntax:	**CurrencyCountryLabel** *label*
Example:	`CurrencyCountryLabel "Canada"`

CurrencyDecimals

UI Equivalent:	Currency property sheet, Decimals box
Syntax:	**CurrencyDecimals** *decimals*
Where:	*decimals* can be from 0 to 9. The default is 2.
Example:	`CurrencyDecimals 5`

CurrencyFormatOverride

UI Equivalent:	Currency property sheet, Override the Country Code check box
Syntax:	**CurrencyFormatOverride** {**True**|**False**}
Example:	`CurrencyFormatOverride True`

CurrencyIsEMU

UI Equivalent:	Currency appears in the Euro Table section of the Currency Table property sheet, which appears when the Use Euro Triangulation check box is selected
Syntax:	**CurrencyIsEmu** {**True**|**False**}
Example:	`CurrencyIsEmu True`

CurrencySymbol

UI Equivalent:	Currency property sheet, Currency Symbol box
Syntax:	**CurrencySymbol** *symbol*
Where:	The default is *.
Example:	`CurrencySymbol "$"`

EmuEntryDate

UI Equivalent:	Currency property sheet, Entry Date in the EMU box
Syntax:	**EmuEntryDate** *date*
Where:	*date* is in YYYYMMDD format
Example:	`EmuEntryDate 19990101`

EuroBaseCurrency

UI Equivalent:	Currency appears as <Base euro> (EUR) in the Euro Table section of the Currency Table property sheet
Syntax:	**EuroBaseCurrency**
Where:	`EuroBaseCurrency`

currencytableopts

Applies to: CurrencyTableAdd, CurrencyTableDelete, CurrencyTableMake, CurrencyTableUpdate

CurrencyCountryCodeColumn

UI Equivalent:	Currency Table property sheet, Country Code Column box
Syntax:	**CurrencyCountryCodeColumn** *objCol*
Where:	*objCol* is the object name. You must specify **CurrencyCountryCodeColumn**, **CurrencyLabelColumn**, or both.
Example:	`CurrencyCountryCodeColumn "Country_Code"`

CurrencyDateColumn

UI Equivalent:	Currency Table property sheet, Date Column box
Syntax:	**CurrencyDateColumn** *objCol*
Where:	*objCol* is the object name.
Example:	This argument is mandatory.
Note:	`CurrencyDateColumn "Currency Date"`

CurrencyLabelColumn

UI Equivalent:	Currency Table property sheet, Currency Label Column box
Syntax:	**CurrencyLabelColumn** *objCol*
Where:	*objCol* is the object name. You must specify **CurrencyCountryCodeColumn**, **CurrencyLabelColumn**, or both.
Example:	`CurrencyLabelColumn "Currency_Label"`

CurrencyRateColumn

UI Equivalent:	Currency Table property sheet, Conversion Rate Column box
Syntax:	**CurrencyRateColumn** *objCol*
Where:	*objCol* is the object name.

Example:	`CurrencyRateColumn "Conversion_Rate"`
Note:	This argument is mandatory.

CurrencyTableType

UI Equivalent:	Currency Table property sheet, Base Table, or Euro Table
Syntax:	**CurrencyTableType** *type*
Where:	*type* is **BaseTable**, **EuroTable**, or **OtherTable**
Example:	`CurrencyTableType BaseTable`

deletionsopts

Applies to: DeletionListUpdate

CalcDef

UI Equivalent:	None. This references the dimension calculation definition that creates the calculation for the category, if any.
Syntax:	**CalcDef** *objDimCalcDef*
Where:	*objDimCalcDef* is the object identifier
Example:	`CalcDef 6375`

Column

UI Equivalent:	Column
Syntax:	**Column** *objCol*
Where:	*objCol* is the object identifier
Example:	`Column 1092`

Cube

UI Equivalent:	PowerCube
Syntax:	**Cube** *objCub*

Where:	*objCub* is the object identifier.
Example:	`Cube 1601`

DataSource

UI Equivalent:	Data source
Syntax:	**DataSource** *objDataSource*
Where:	*objDataSource* is the object identifier.
Example:	`DataSource 4213`

Dimension

UI Equivalent:	Dimension
Syntax:	**Dimension** *objDim*
Where:	*objDim* is the object identifier.
Example:	`Dimension 1844`

Levels

UI Equivalent:	Level
Syntax:	**Levels** *objLev*
Where:	*objLev* is the object identifier.
Example:	`Levels 1385`

Measure

UI Equivalent:	Measure
Syntax:	**Measure** *objMeasure*
Where:	*objMeasure* is the object identifier.
Example:	`Measure 1500`

Signon

UI Equivalent:	Signon
Syntax:	**Signon** *objSignon*
Where:	*objSignon* is the object identifier.
Example:	`Signon 9876`

UserClass

UI Equivalent:	User Class
Syntax:	**UserClass** *objUserClass*
Where:	*objUserClass* is the object identifier.
Example:	`UserClass 1906`

View

UI Equivalent:	View
Syntax:	**View** *objView*
Where:	*objView* is the object identifier.
Example:	`View 1092`

dimopts

Applies to: DimAdd, DimMake, DimUpdate

Association

UI Equivalent:	Dimension property sheet, General tab, External Rollup Column (only appears when the Contains Externally Rolled Up Measures check box is selected in the associated data source's property sheet (General tab); or associated column for time dimensions
Syntax:	**Association** *objCol*

Where:	*objCol* is the object name.
Example:	`Association "Time Category Code"`

DaysInWeek

UI Equivalent:	Dimension property sheet, Time tab, Days in Week box
Syntax:	**DaysInWeek** *bignum*
Where:	*bignum* is based on a bit representation of the days in the week, as follows: **1:** Sunday **2:** Monday **4:** Tuesday **8:** Wednesday **16:** Thursday **32:** Friday **64:** Saturday To specify groups of days, add the codes. For example, Monday to Friday is represented by **62** (**2+4+8+16+32**). All seven days are represented by **127**. The default is **127**.
Example:	`DaysInWeek 62`

DimDefaultCategory

Syntax:	**DimDefaultCategory <ID>**
Example:	`DimUpdate Dimension 185 "Products" DimType Regular NewCatsLock False ExcludeAutoPartitioning False` DimDefaultCategory `293`
Note:	Option available in Transformer Version 7.3 and above to specify a new default category (lower level) where a scenario dimension will open.

DimInfo

UI Equivalent:	Dimension property sheet, Description tab

Syntax:	**DimInfo** *string* [*string*...]
Where:	Each string may be up to 256 characters and the total description may be up to 4,095 characters.
Example:	`DimInfo "Products carried by the Great Outdoors Company."`

DimType

UI Equivalent:	Dimension property sheet, General tab, Dimension type
Syntax:	**DimType** {**Regular**|**Date**|**NonStandard**}
Where:	**Regular** is Regular and **Date** is Time. **NonStandard** applies when the dimension is a time dimension, PreSummarized True is set for the associated data source, and Association is blank (or not set). The default is Date.
Example:	`DimType Date`

EarliestDate

UI Equivalent:	Dimension property sheet, Time tab, Earliest date
Syntax:	**EarliestDate** *date*
Where:	*date* is in the format YYYYMMDD. The default is 19000101.
Example:	`EarliestDate 19000101`

ExcludeAutoPartitioning

UI Equivalent:	Dimension property sheet, General tab, Exclude the Dimension from Auto-partitioning check box
Syntax:	**ExcludeAutoPartitioning** {**True**|**False**}
Where:	The default is **False**.
Example:	`ExcludeAutoPartitioning True`

LatestDate

UI Equivalent:	Dimension property sheet, Time tab, Latest date
Syntax:	**LatestDate** *date*
Where:	*date* is in the format YYYYMMDD. The default is 99991231.
Example:	`LatestDate 99991231`

ManualPeriods

UI Equivalent:	Dimension property sheet, Time tab, Automatically Set Current Time Period
Syntax:	**ManualPeriods** {**True**|**False**}
Where:	**False** means that Automatically Set Current Time Period is selected; **True** means that it is not selected. The default is False
Example:	`ManualPeriods False`

NewCatsLock

UI Equivalent:	Dimension property sheet, General tab, Prohibit Automatic Creation of New Categories check box
Syntax:	**NewCatsLock** {**True**|**False**}
Where:	The default is **False**.
Example:	`NewCatsLock False`

filteropts

Applies to: FilterMake, FilterUpdate

ArchitectAttribute

UI Equivalent:	This option is for internal use only.
Syntax:	**ArchitectAttribute**

Example:	`FilterName 661 "Sales Managers" ArchitectDataSource 593 "Measure Query 1" ArchitectAttribute "151100:6"`

ExpMark

UI Equivalent:	Creating an Expression in the Expression box in the Filter dialog box, accessed from SQL Data Source tab, Data Source property sheet for a data source contained in an Architect Model.
Syntax:	**ExpMark** *SQLstring* **ExpMarkEnd**
Example:	`ExpMark filter[Managers in SalesStaff] ExpMarkEnd`

FilterDescription

UI Equivalent:	Creating a Description in the Description box in the Filter dialog box, accessed from SQL Data Source tab, Data Source property sheet for a query contained in an Architect Model.
Syntax:	**Description** *string* [*string*...]
Where:	Each string may be up to 256 characters and the total description may be up to 4,095 characters.
Example:	`Description "Filters out managers located in North America."`

levelopts

Applies to: LevelAdd, LevelMake, LevelUpdate

Association

UI Equivalent:	Level property sheet, General tab, Blank Substitution box
Syntax:	**Blanks** *string*
Example:	`Blanks "N/A"`

Blanks

UI Equivalent:	Level property sheet, General tab, Blank Substitution box

Syntax:	**Blanks** *string*
Example:	`Blanks "N/A"`

CategoryCode

UI Equivalent:	Level property sheet, Source tab, Category Code Column box
Syntax:	**CategoryCode** *objCol*
Where:	*objCol* is the object name.
Example:	`CategoryCode "Product Line"`

CatLabFormat

UI Equivalent:	Level property sheet, Time tab, Date Format box
Syntax:	**CatLabFormat** *string*
Where:	*string* is the desired format.
Example:	The following format produces the label **1998 Q 3**: `CatLabFormat 'YYYY "Q" Q'`

DateFunction

UI Equivalent:	Level property sheet, Time tab, Date Function box
Syntax:	**DateFunction** *datefunction*
Where:	*datefunction* is one of **None**, **Year**, **LunarYear**, **Quarter**, **LunarQuarter**, **Month**, **LunarMonth**, **LunarMonth445**, **LunarMonth454**, **LunarMonth544**, **Week**, **Day**
Example:	`DateFunction Year`

Description

UI Equivalent:	Level property sheet, Source tab, Description Column box
Syntax:	**Description** *objCol*

Where:	*objCol* is the object name of a column.
Example:	`Description "Product types"`

Filtered

UI Equivalent:	Diagram menu, Exclude (when a level in a view is selected)
Syntax:	**Filtered** {**True**|**False**}
Example:	`Filtered True`

Format

UI Equivalent:	Level property sheet, Time tab, Date Format box
Syntax:	**Format** *string*
Example:	`Format "YMD"`

Generate

UI Equivalent:	Level property sheet, Time tab, Generate All Categories in the Period check box
Syntax:	**Generate** [*generatedate*]
Where:	*generatedate* is one of **Default**, **All**, or **Need**. When the check box is selected, the setting is **All**. When the check box is not selected, the setting is **Need**.
Example:	`Generate All`

Inclusion

UI Equivalent:	Level property sheet, General tab, Inclusion box
Syntax:	**Inclusion** *inclusion*

Where:	*inclusion* is one of **Default**, **Generate**, **Suppress**, **Retain**, or **Filtered**. The MDL codes map to the user interface as follows. All the codes are in the Inclusion menu except for Suppress and Filtered, which are in the Diagram menu. No setting: Default (when needed) **Default:** Default (when needed) **Generate:** Always Include **Suppress:** Diagram menu, Suppress **Retain:** Include When Needed, or Suppress Blank Categories **Filtered:** Diagram menu, Exclude
Example:	`Inclusion Suppress`

LevelInfo

UI Equivalent:	Level property sheet, Description tab
Syntax:	**LevelInfo** *string* [*string*...]
Where:	Each string may be up to 256 characters and the total description may be up to 4,095 characters.
Example:	`LevelInfo "The type of reseller can shed some light on the sales numbers."`

Label

UI Equivalent:	Level property sheet, Source tab, Label Column box
Syntax:	**Label** *objCol*
Where:	*objCol* is the object name.
Example:	`Label "Customer Name"`

NewCatsLock

UI Equivalent:	Level property sheet, General tab, Prohibit Automatic Creation of New Categories check box
Syntax:	**NewCatsLock** {**True**|**False**}

Example:	`NewCatsLock False`

OrderBy

UI Equivalent:	Category property sheet, Order tab, Order Value box
Syntax:	**OrderBy Drill** *objDrillCat* **Value** *string*
Where:	*objDrillCat* can be the object name, object identifier, or both. *string* is what appears in the Order Value box.
Example:	`OrderBy Drill 6881 Value "199712"`

SortOrder

UI Equivalent:	Level property sheet, Order By tab
Syntax:	**SortOrder Drill** *objDrillCat* **Column** *objCol* **SortOrder** *storagetype* **SortAs** {**Ascending**|**Descending**}
Where:	*objDrillCat* and *objCol* can both be identified by Object Name, Object Identifier, or both; and *storagetype* is one of **Default**, **Int8**, **Int16**, **Int32**, **Float64**, **Text**, **Character**, or **Longtext**.
Example:	`SortOrder Character`
Note:	In versions prior to Series 7, called Orderby. If alpha-numeric, use **Character**; otherwise, use **Float64**.

SortAs

UI Equivalent:	Level property sheet, Order By tab
Syntax:	**SortAs Drill** *objDrillCat* **Column** *objCol* **SortOrder** *storagetype* **SortAs** {**Ascending**|**Descending**}
Where:	*objDrillCat* and *objCol* can both be identified by Object Name, Object Identifier, or both; and *storagetype* is one of **Default**, **Int8**, **Int16**, **Int32**, **Float64**, **Text**, **Character**, or **Longtext**.
Example:	`SortAs Ascending`
Note:	If alpha-numeric, use **Character**; otherwise, use **Float64**.

Partition

UI Equivalent:	Level property sheet, General tab, Partition number box
Syntax:	**Partition** *uns*
Where:	*uns* is a number from 0 to 15.
Example:	`Partition 3`

RefreshLabel

UI Equivalent:	Level property sheet, Source tab, Refresh check box next to Label
Syntax:	**RefreshLabel** {**True**|**False**}
Example:	`RefreshLabel True`

RefreshDescription

UI Equivalent:	Level property sheet, Source tab, Refresh check box next to Description
Syntax:	**RefreshDescription** {**True**|**False**}
Example:	`RefreshDescription False`

RefreshShortName

UI Equivalent:	Level property sheet, Source tab, Refresh check box next to Short Name
Syntax:	**RefreshShortName** {**True**|**False**}
Example:	RefreshShortName False

Share

UI Equivalent:	Level property sheet, General tab, Share Object box
Syntax:	**Share** *objLevel*

Where:	*objLevel* can be the object name or object identifier. *objLevel* must be an ancestor in the same dimension.
Example:	`Share "Product Line"`

ShortName

UI Equivalent:	Level property sheet, Source tab, Short Name Column drop-down list
Syntax:	**ShortName** *objCol*
Where:	*objCol* is the object name.
Example:	`ShortName "CUST_SHRTN"`

Source

UI Equivalent:	Level property sheet, Source tab, Source Column drop-down list
Syntax:	**Source** *objCol*
Where:	*objCol* is the object name.
Example:	`Source "Customer No"`

Suppressed

UI Equivalent:	Diagram menu, Suppress (when a level in a view is selected)
Syntax:	**Suppressed** {**True**|**False**}
Example:	`Suppressed True`

TimeRank

UI Equivalent:	Level property sheet, Time tab, Time Level Ranking box
Syntax:	**TimeRank** *uns*
Example:	`TimeRank 10`

UniqueCategories

UI Equivalent:	Level property sheet, Source tab, Unique check box
Syntax:	**UniqueCategories** {**True**|**False**}
Example:	`UniqueCategories True`

UniqueMove

UI Equivalent:	Level property sheet, Source tab, Unique check box and Move check box next to Source Column drop-down list
Syntax:	**UniqueMove** {**True**|**False**}
Example:	`UniqueMove True`

meaopts

Applies to: MeasureAdd, MeasureMake, MeasureUpdate

ActivityMeasure

UI Equivalent:	Measure property sheet, Rollup tab, Activity Measure (available only when Category Count is selected on the Type tab)
Syntax:	**ActivityMeasure** *objMeasure*
Where:	*objMeasure* is the object name. The default (if you don't specify an ActivityMeasure) is that if any measure exists for a category, then a count is returned.

Example:	The following is an example of a category count that checks for any measure that exists for a category:
	`MeasureMake 159 "Revenue" Levels 139 Rollup CategoryCount Storage Default Scale 0 Decimals 0 Sign False IsCurrency False DrillThrough False EndList`
	The following is an example of a category count that checks a specific measure using ActivityMeasure:
	`Measure 161 "Cost" Levels 137 ActivityMeasure "Revenue" Rollup CategoryCount Storage Default Scale 0 Decimals 0 Sign False IsCurrency False DrillThrough False EndList`

Note:	Three options create a Category Count measure: **ActivityMeasure**, **Dimension**, and **Rollup**. For more information, see "Dimension" (p. 202) and "Rollup " (p. 185).

Allocation

UI Equivalent:	None
Syntax:	**Allocation** *allocation* {**AllocationMeasure** *objMeasure*}
Where:	*allocation* is one of **Default**, **None**, **Constant** or **Allocate**. *objMeasure* specifies the measure and is the object name or object identifier.

Association

UI Equivalent:	Measure property sheet, Type tab, Column
Syntax:	**Association** *objCol*
Where:	*objCol* is the column name.
Example:	`Association "Cost"`

Calc

UI Equivalent:	Measure property sheet, Type tab, Calculated
Syntax:	**Calc** *keywords objects*
Where:	*keywords* are given in the following table. Each *object* is an object name followed by the at sign (@) and the object identifier; for example, "Revenue@259". If object identifiers are suppressed, *objects* are the object names only.

Example:	`Calc "Revenue@259" / "SalesRep Count @3037"`
	If-then-else example:
	`Measure 173 "Revenue" calc IF (ISNULL ("Cost@177"+"Quantity@181")) THEN ("Cost@177") ELSE (("Quantity@181")`

Calculationkeyword	Type	Description
ISNULL		Zero
IF		conditional if expression
THEN		conditional then expression
ELSE		conditional else expression
AND		Both
OR		Either
NULL		Does not exist
!=		
=		Equal to
<		Less than
<=		Less than or equal to
>		Greater than
>=		Greater than or equal to
/	Number	Divided by
-	Date or Number	Minus
*	Number	Multiplied by
+	Character, Date, or Number	Plus
^	Number	Returns a number raised to the power of a second number
Absolute	Number	Converts numbers to their unsigned value
Average	Number	Creates an arithmetic mean
Max	Number	Maximum instance

Calculationkeyword	Type	Description
Min	Number	Minimum instance
Percent	Number	Computes percent

Decimals

UI Equivalent:	Measure property sheet, General tab, Precision box
Syntax:	**Decimals** *uns*
Example:	`Decimals 2`

Dimension

UI Equivalent:	Measure property sheet, Type tab, Dimension and Level boxes (available only when Category Count is selected)
Syntax:	**Dimension** *objDim* **Drill** *objDrillCat* **Levels** *objLevel*
Example:	`Dimension "Products" Levels "Product ID"`
Note:	There are three options that create a Category Count measure: **ActivityMeasure**, **Dimension**, and **Rollup**. For more information, see "ActivityMeasure " (p. 214) and "Rollup " (p. 185).

DrillThrough

UI Equivalent:	Measure property sheet, Drill Through tab, Allow Drill Through for this Measure check box and Custom Reports box
Syntax:	**DrillThrough** {**True**|**False**} [*string*] [*string*...] **EndList**
Where:	*string* is a report name with optional path.

Example:	In this example, DrillThrough is not selected:
	`DrillThrough False EndList`
	In this example, DrillThrough is selected and an .imr report is specified:
	`DrillThrough True "c:<installation_directory>\PowerCube and Reports\bsc_msrs.imr" EndList`

DuplicateRollup

UI Equivalent:	Measure property sheet, Rollup tab, Duplicates Rollup box
Syntax:	**DuplicateRollup** [*duplicatesrollup*]
Where:	*duplicatesrollup* is one of **Default**, **Sum**, **Minimum**, **Maximum**, **Average**, **First**, or **Last**.
Example:	`DuplicateRollup Sum`

DuplicateWeight

UI Equivalent:	Measure property sheet, Rollup tab, Duplicate Weight box
Syntax:	**DuplicateWeight** *objMeasure*
Where:	*objMeasure* is the object name.
Example:	`DuplicateWeight "Revenue"`
Note:	DuplicateWeight applies only when DuplicateRollup is set to Average.

Exclude

In Transformer Version 7.3 and above, you can use this option and the excluded object's unique ID and/or name to restrict drill-through targets to a specified level. By restricting the scope, you ensure that users navigate directly to the relevant portion of the cube, before beginning their drill-through operation.

After deciding on the measure or cube on which to restrict drill-through by level, you must specify the dimension and any levels for which you want to restrict drill-through, and the drill-through target to which the restriction applies.

You must use MDL if you want to exclude levels in a sub-dimension, because these are not displayed on the Dimension Map user interface.

If you use this option, your new script will be fully compatible with your existing MDL scripts. These older scripts do not have any exclusions. They are therefore deemed to have an empty exclusion list, and do not need to conclude with the normally mandatory `EndExclude` keyword.

Syntax:	The syntax for the exclude level option is as follows: **Exclude** [**Dimension***objDim*] [**Drill** *objDrill*] **Levels** *objLevel*. The dimension and drill objects are only mandatory if the level is identified only by name.

Example:	`DrillThrough True ...`
	`"C:\path_to_cube.mdc" "Description text"`
	`Exclude Dimension "Line" Exclude Levels 219 "State"`
	`Exclude Dimension "Market" Levels "Market"`
	`EndExclude`
	`"C:\path_to_cube2.mdc" "Description text2"`
	`EndList`

Note:	*objDim*, *objLevel*, and *objDrill* can be the unique object identifiers, names, or both. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21).

Format

UI Equivalent:	Measure property sheet, Format tab
Syntax:	**Format** *string*
Where:	*string* is the format, followed by a tilde (~) and the number of decimal places.
Example:	This example sets the format to #,##0 and sets the decimal places to 2: `Format "#,##0~2"`

IgnoreMissingValue

In Transformer Version 7.4 and above, you can set this option to `TRUE` to ignore null and missing (NA) values when performing certain kinds of time-state rollup (Average and Weighted Average). If you do not specify the option when you create, make, or update a measure, the default setting (include these values) is applied.

Note: You cannot set the `IgnoreMissingValue` flag for time state rollups of type None, First Period, Last Period, or Current Period. Null and missing values are always included when aggregating or calculating these kinds of rollup, or when used in versions of Transformer earlier than Series 7 Version 4. Missing (null) data values are always excluded from Min and Max calculations for rollups, whether they are set by Transformer to display as "0" or "n/a" (the NA setting).

UI Equivalent:	Measure property sheet, Rollup tab, Ignore missing value check box (only enabled for rollup types that support this feature)
Syntax:	`TimeStateRollup` [*type*]
Where:	Rollup *type* is one of Average or Weighted Average

Example: Structured MDL for a Weighted Average Rollup	`Measure 173 "Cost"` TimeStateRollup Average TimeStateWeight "Outlet" IgnoreMissingValue True `Storage Default`
	`OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False IsFolder False`
	`WeightId 243 DrillThrough False EndList Associations 175 "Revenue" AssociationType Type_Query`
	`AssociationRole Role_Source AssociationReferenced "Revenue"`

Example: Verb MDL for Average Rollup type	`MeasureCreate 173 "Profit"` TimeStateRollup Average IgnoreMissingValue True `Storage Default`
	`OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False IsFolder False`
	`WeightId 243 DrillThrough False EndList Associations 175 "Revenue" AssociationType Type_Query`
	`AssociationRole Role_Source AssociationReferenced "Revenue"`
	`MeasureMake 173 "Profit"` TimeStateRollup Average IgnoreMissingValue True
	`MeasureUpdate 173 "Profit"` TimeStateRollup Average IgnoreMissingValue True

Note:	When this optional keyword and its Boolean value are not added to your MDL syntax, its value is considered to be False (the default prior to version 7.4). For information about specifying measures, see "MeasureMake" (p. 129).

IsCurrency

UI Equivalent:	Measure property sheet, General tab, Allow Currency Conversion check box
Syntax:	**IsCurrency** {**True**|**False**}
Example:	`IsCurrency True`

IsFolder

UI Equivalent:	Measure property sheet, General tab, Measure Folder check box
Syntax:	**IsFolder** {**True**|**False**}
Example:	`IsFolder True`

Label

UI Equivalent:	Measure property sheet, General tab, Measure label box
Syntax:	**Label** *string*
Example:	`Label "Qty"`

MeasureInfo

UI Equivalent:	Measure property sheet, Description tab
Syntax:	**MeasureInfo** *string* [*string*...]
Where:	Each string may be up to 256 characters and the total description may be up to 4,095 characters.
Example:	`MeasureInfo "Cost to purchase products"`

Missing

UI Equivalent:	Measure property sheet, General tab, Missing Value box
Syntax:	**Missing** [*missingvalue*]
Where:	*missingvalue* is one of **Default**, **Zero**, or **NA**.
Example:	`Missing Zero`

Rollup

UI Equivalent:	Measure property sheet, Rollup tab, Regular Rollup box
Syntax:	**Rollup** [*rollup*]

Where:	*rollup* is one of **CategoryCount**, **Default**, **Sum**, **Minimum**, **Maximum**, **Average**, **Count**, **CountAll**, **Any**, **None**, or **External**.
Example:	`Rollup Average`
Note:	For more information about **CategoryCount**, see "ActivityMeasure " (p. 214) and "Dimension" (p. 202).

Parent

UI Equivalent:	Right-click the measures pane, and select Insert a Measure Folder
Syntax:	**Parent** *objMeasure*
Where:	*objMeasure* is the unique identifier of the parent.
Example:	`Parent 4719`

Scale

UI Equivalent:	Measure property sheet, General tab, Output Scale
Syntax:	**Scale** *uns*
Example:	`Scale 0`

ShortName

UI Equivalent:	Measure property sheet, General tab, Short Name
Syntax:	**ShortName** *string*
Example:	`ShortName "QtySold"`

Sign

UI Equivalent:	Measure property sheet, General tab, Reverse the sign check box
Syntax:	**Sign** {**True**|**False**}
Example:	`Sign False`

Storage

UI Equivalent:	Measure property sheet, General tab, Storage Type box
Syntax:	**StorageType** [*storagetype*]
Where:	*storagetype* is one of **Default**, **Int8**, **Int16**, **Int32**, **Float64**, **Text**, **Character**, or **LongText**.
Example:	`Storage Default`

TimeStateRollup

UI Equivalent:	Measure property sheet, Rollup tab, Time State Rollup box
Syntax:	**TimeStateRollup** [*timerollup*]
Where:	*timerollup* is one of **Default**, **Minimum**, **Maximum**, **First**, **Last**, **Current**, **Average**, or **Summary**.
Example:	`TimeStateRollup Minimum`

TimeStateWeight

UI Equivalent:	Measure property sheet, Rollup tab, Time State Weight box
Syntax:	**TimeStateWeight** *objMeasure*
Where:	objMeasure is the object name.
Example:	`TimeStateWeight "Revenue"`
Note:	TimeStateWeight applies only when TimeStateRollup is set to Average.

Timing

UI Equivalent:	Measure property sheet, Rollup tab, Regular Timing box
Syntax:	**Timing** [*measuretiming*]
Where:	*measuretiming* is one of **Default**, **Before_Rollup**, or **After_Rollup**.
Example:	`Timing After_Rollup`

Weight

UI Equivalent:	Measure property sheet, Rollup tab, Regular Weight box
Syntax:	**Weight** *objMeasure*
Where:	*objMeasure* is the object name.
Example:	`Weight "Revenue"`
Note:	This option is identical to WeightID except that the object name is used. Weight and WeightID apply only when Rollup is set to Average.

WeightID

UI Equivalent:	Measure property sheet, Rollup tab, Regular Weight box
Syntax:	**WeightID** *objMeasure*
Where:	*objMeasure* is the object identifier.
Example:	`Weight 259`
Note:	This option is identical to Weight except that the object identifier is used. WeightID and Weight apply only when Rollup is set to Average.

powercubeopts

Applies to: CubeAdd, CubeMake, CubeUpdate, CubeGroupAdd, CubeGroupMake, CubeGroupUpdate, CubeGroupCubeAdd, CubeGroupCubeMake, CubeGroupCubeUpdate

BlockParentTotals

UI Equivalent:	PowerCube property sheet, Processing tab, Block totals for parents with excluded children check box
Syntax:	**BlockParentTotals** {**True**|**False**}
Example:	`BlockParentTotals True`
Note:	Optional. The default is false.

Caching

UI Equivalent:	PowerCube property sheet, Processing tab, Enable Crosstab Caching check box
Syntax:	**Caching** {**True**|**False**}
Example:	`Caching True`
Note:	True sets a flag in the .mdc file that performs Crosstab caching.

Compress

UI Equivalent:	PowerCube property sheet, Processing tab, Store This PowerCube Compressed check box
Syntax:	**Compress** {**True**|**False**}
Example:	`Compress False`
Note:	This option is not available in all versions of Transformer.

Consolidate

UI Equivalent:	PowerCube property sheet, General tab, Consolidate box
Syntax:	**Consolidate** *consolidate*
Where:	*consolidate* is one of **Default**, **Yes**, **No**, or **PreSorted**.
Example:	`Consolidate Yes`

ConsolidatedRecords

UI Equivalent:	PowerCube property sheet, Auto-Partition tab, Estimated Number of Consolidated Records box
Syntax:	**ConsolidatedRecords** *bignum*
Example:	`ConsolidatedRecords 10,000,000`
Note:	The default is 10,000,000.

CubeCreation

UI Equivalent:	PowerCube property sheet, Processing tab, Cube Creation Enabled/Disabled buttons
Syntax:	**CubeCreation** *makecube*
Where:	*makecube* is one of **Default**, **Implicit**, **On**, or **Off**.
Example:	`CubeCreation On`

CubeCycle

UI Equivalent:	None
Syntax:	**CubeCycle** *bignum*
Example:	`CubeCycle 604`
Note:	This option is for internal use only.

CubeStamp

UI Equivalent:	None
Syntax:	**CubeStamp** *bignum*
Example:	`CubeStamp 861985194`
Note:	This option is for internal use only.

CubeUpdateLock

UI Equivalent:	None
Syntax:	**CubeUpdateLock** {**None**|**All**}
Example:	`CubeUpdateLock None`
Note:	This option is for internal use only. Do not modify this setting

Database

UI Equivalent:	PowerCube property sheet, Output tab, Signon box

Syntax:	**DataBase** *objSignon*
Where:	**objSignon** can be the object name, object identifier, or both.
Example:	`Database 6317`
Note:	This option applies only to certain database types.

DatabaseInfo

UI Equivalent:	PowerCube property sheet, Output tab, Database type and next two settings below it. (The settings change depending on the database type that is chosen.)
Syntax:	**DatabaseInfo** *string*
Where:	The database type and two settings are separated by semi-colons.
Example:	`DatabaseInfo "Local;;"`

DataSource

UI Equivalent:	PowerCube property sheet, General tab, Source file box
Syntax:	**DataSource** *objDataSource* **AlternateSource** *filename* [*altdata-sourceopts*]
Where:	*objDataSource* is the object name. *altdatasourceopts* is one or more of the following: **Isolation** *number* **DmsSignon** *objSignon* **IMRName** *string* **SourceSignonList** *signon* [*signons...*] **EndList** **SQL** *string* **Stamp** *number*

Example:	`DataSourceMake 103 "Report1" Separator " "`
	`SourceType DataSource DecimalSep " "`
	`Thousandsep " " Columns True Timing`
	`PopYesCreateDefault Source "C:\Report1.iqd"`
	`SQL 'select T1."PROD_TYPE" as c1,'`
	`'T1."PROD_LINE" as c2, T1."PRODUCT"'`
	`'as c3, T1."PROD_COST" as c4,'`
	`'T2."ORDER_DT" as c5 from "ORDRDETL" T3,'`
	`'"PRODUCT" T1,"ORDER" T2'`
	`'where (T3."PROD_NO" = T1."PROD_NO")'`
	`'and (T2."ORDER_NO" = T3."ORDER_NO") and'`
	`'(T1."PROD_TYPE" = "Outdoor Products) "`
	`Isolation 0 SourceSignonList 105 1105 EndList`
	`ImrName "C:\Report1.imr"`
	`Stamp 930750524 Speed False SetCurrent True`
	`ServerSource False Presummarized False`
	`EnableMultiProcess False`

Note:	The data source file that you specify with the **AlternateSource** option must have the same type and organization of columns as the original source file.

DetailLevel

UI Equivalent:	PowerCube property sheet, Cube Group tab, Lowest Detail of Categories in the Level box
Syntax:	**DetailLevel** *objLevel*
Where:	*objLevel* can be the object name.
Example:	`DetailLevel "Product Type"`

DrillThrough

UI Equivalent:	PowerCube property sheet, Drill Through tab, Allow Drill-through for this PowerCube box, and Custom Reports and Report Description
Syntax:	**DrillThrough** [**True**|**False**] *reportnames* **EndList**

Where:	*reportnames* are from the Custom Reports and Report Description boxes.
Example:	`DrillThrough True "c:installation_directory>\PowerCube and Reports\all regions.mdc" EndList`

Exclude

In Transformer Version 7.3 and above, you can use this option and the excluded object's unique ID and/or name to restrict drill-through targets to a specified level. By restricting the scope, you ensure that users navigate directly to the relevant portion of the cube, before beginning their drill-through operation.

After deciding on the measure or cube on which to restrict drill-through by level, you must specify the dimension and any levels for which you want to restrict drill-through, and the drill-through target to which the restriction applies.

You must use MDL if you want to exclude levels in a sub-dimension, because these are not displayed on the Dimension Map user interface.

If you use this option, your new script will be fully compatible with your existing MDL scripts. These older scripts do not have any exclusions. They are therefore deemed to have an empty exclusion list, and do not need to conclude with the normally mandatory `EndExclude` keyword.

Syntax:	The syntax for the exclude level option is as follows: **Exclude** [**Dimension***objDim*] [**Drill** *objDrill*] **Levels** *objLevel*. The dimension and drill objects are only mandatory if the level is identified only by name.

Example:	`DrillThrough True ...`
	`"C:\path_to_cube.mdc" "Description text"`
	`Exclude Dimension "Line" Exclude Levels 219 "State" Exclude Dimension "Market" Levels "Market" EndExclude`
	`"C:\path_to_cube2.mdc" "Description text2"`
	`EndList`

Note:	*objDim*, *objLevel*, and *objDrill* can be the unique object identifiers, names, or both. For more information about uniquely identifying objects, see "Locate Objects Uniquely" (p. 21).

IncrementalUpdate

UI Equivalent:	PowerCube property sheet, Processing tab, This Cube Is Incrementally Updated check box

Syntax:	**IncrementalUpdate** {**True**|**False**}
Example:	`IncrementalUpdate True`
Note:	By default, an object identifier will be assaigned automatically when a cube is incrementally updated.

Information

UI Equivalent:	PowerCube property sheet, Description tab
Syntax:	**Information** *string* [*string*...]
Where:	Each string may be up to 256 characters and the total description may be up to 4,095 characters.
Example:	`Information "PowerPlay 7.4 online documentation PowerCube example."`

MDCFile

UI Equivalent:	PowerCube property sheet, Output tab, PowerCube File Name box.
Syntax:	**MDCFile** *string*
Example:	`MDCFile "c:\Temp\outdoors.mdc"`

MeasureName

UI Equivalent:	PowerCube property sheet, General tab, Measure Name box
Syntax:	**MeasureName** *string*
Example:	`MeasureName "Basic Measures"`

Optimize

UI Equivalent:	PowerCube property sheet, Processing tab, Optimization box
Syntax:	**Optimize** *cubeoptimize*
Where:	*cubeoptimize* is one of **Default**, **Categories**, **DataPasses**, **DirectCreate**, **PriorMarking**, or **AutoPartition**.

Example:	`Optimize Categories`

PartitionSize

UI Equivalent:	PowerCube property sheet, Auto-Partition tab, Desired Partition Size
Syntax:	**PartitionSize** *bignum*
Example:	`PartitionSize 500,000`

PassesNumber

UI Equivalent:	PowerCube property sheet, Auto-Partition tab, Maximum Number of Passes box
Syntax:	**PassesNumber** *bignum*
Example:	`PassesNumber 5`
Note:	The default is 5.

Password

UI Equivalent:	PowerCube property sheet, General tab, Password
Syntax:	**Password** *string*
Where:	The password is not encrypted in the MDL file but is saved exactly as typed.
Example:	Password "my_pass"

SegmenterDimension

UI Equivalent:	PowerCube property sheet, Cube Group tab, Dimension box
Syntax:	**SegmenterDimension** *objDim*
Where:	*objDim* is the object name.
Example:	`SegmenterDimension "Products"`

SegmenterLevel

UI Equivalent:	PowerCube property sheet, Cube Group tab, Level box
Syntax:	**SegmenterLevel** *objLevel*
Where:	*objLevel* is the object name.
Example:	`SegmenterLevel "Product Line"`

ServerCube

UI Equivalent:	PowerCube property sheet, Processing tab, Processed option buttons
Syntax:	**ServerCube** {**True**|**False**}
Where:	**True** is equivalent to selecting the On the Server option button.
Example:	`ServerCube True`

Status

UI Equivalent:	PowerCube property sheet, Processing tab, Status box
Syntax:	**Status** [*cubestatus*]
Where:	*cubestatus* is one of **New**, **OK**, **Inactive**, **Warnings**, **Invalid**, **Failed**, **Busy**, or **Missing**.
Example:	`Status Inactive`

SummaryLevel

UI Equivalent:	PowerCube property sheet, Cube Group tab, Summarize All External Categories in the Level box
Syntax:	**SummaryLevel** *string*
Example:	`SummaryLevel "Product Type"`

UseAlternateFilename

UI Equivalent:	PowerCube property sheet, Output tab, Use Temporary File Name if the original file is locked check box
Syntax:	**UseAlternateFilename** {**True**|**False**}
Example:	`UseAlternateFilename True`

TimeBasedPartitionedCube

UI Equivalent:	PowerCube property sheet, Cube Group tab, Enable Time-based Partitioning check box
Syntax:	**TimeBasedPartitionedCube** {**True**|**False**}
Example:	`TimeBasedPartitionedCube True`

signonopts

Applies to: SignonAdd, SignonMake, SignonUpdate; not used with all versions of Transformer.

Database

UI Equivalent:	NonePowerCube property sheet, Output tab, Database Type box
Syntax:	Database string
Where:	*string* is the unencrypted connection information.

DBType

UI Equivalent:	None
Syntax:	**DBType** *string*
Example:	`DBType "DB"`

EncryptedPW

UI Equivalent:	None
Syntax:	**EncryptedPW** *string*

Where:	*string* is the password in encrypted format.
Example:	EncryptedPW "81D0B91A71F9F4DC70009A68358948 5097C2FA287D76C7BFAE402CC33760691C41BBC14" "611677ACC316DF82E77F699B5534C76D0 CAA0F64E76CD3DFD5B40B4E962C753517082E1" ...

FullDB

UI Equivalent:	None
Syntax:	**FullDB** *string*
Where:	*string* is the encrypted database information
Example:	FullDB 326EBCA30A64199C33C5C3BDDB476D14DDF E3B90367B5AF16A54A876C11A99FE3CBC025" "34B71350982D9C3FBEEA81E1B7 BA648C0FB685D329DC60A5714126CFD284848B6A95842" ...

Password

UI Equivalent:	PowerCube property sheet, General tab, Password
Syntax:	**Password** *string*
Where:	*string* is the password in unencrypted format.
Example:	`Password "my_pass"`

PromptForPassword

UI Equivalent:	Signon property sheet, General tab, Prompt for Password check box
Syntax:	**PromptForPassword** {**True**|**False**}
Example:	`PromptForPassword True`

SignonInfo

UI Equivalent:	Signon property sheet, Description tab

Syntax:	**SignonInfo** *description* [*description*...]
Where:	Each string may be up to 256 characters and the total description may be up to 4,095 characters.
Example:	`SignonInfo "The logical database name that contains the source data for MAIN (IQD) data source."`

UserID

UI Equivalent:	Signon property sheet, General tab, User ID box
Syntax:	**UserID** *string*
Example:	`UserID "corpadm"`

viewopts

Applies to: ViewAdd, ViewMake, ViewUpdate

Apex

UI Equivalent:	Diagram menu, Apex (when a category in a view is selected)
Syntax:	**Apex** *objCat*
Where:	*objCat* specifies the category and is the object name, object identifier, or both.
Example:	`Apex "Europe"`

Cloak

UI Equivalent:	Diagram menu, Cloak (when a category in a view is selected)
Syntax:	**Cloak** *objCat*
Where:	objCat specifies the category and is the object name, object identifier, or both.
Example:	`Cloak "Far East"`

Filter

UI Equivalent:	Diagram menu, Exclude (when a category in a view is selected)
Syntax:	**Filter** *objCat*
Where:	*objCat* specifies the category and is the object name, object identifier, or both.
Example:	`Filter "Europe"`

LevelCloak

UI Equivalent:	Diagram menu, Cloak (when a level in a view is selected)
Syntax:	**LevelCloak** *objLevel* [**Drill** *objDrillCat*]
Where:	*objLevel* specifies the level and is the object name, object identifier or both. *objDrillCat* specifies the drill category and is the object name, object identifier, or both.
Example:	`LevelCloak "Customer Type"`

LevelFilter

UI Equivalent:	Diagram menu, Exclude (when a level in a view is selected)
Syntax:	**LevelFilter** *objLevel* [**Drill** *objDrillCat*]
Where:	*objLevel* specifies the level and is the object name, object identifier or both. *objDrillCat* specifies the drill category and is the object name, object identifier, or both.
Example:	`LevelFilter "Customer Type"`

LevelSummary

UI Equivalent:	Diagram menu, Summary (when a level in a view is selected)
Syntax:	**LevelSummary** *objLevel* [**Drill** *objDrillCat*]
Where:	*objLevel* specifies the level and is the object name, object identifier, or both. *objDrillCat* specifies the drill category and is the object name, object identifier, or both.

Example:	`LevelSummary "Region"`

LevelSuppressed

UI Equivalent:	Diagram menu, Suppressed (when a level in a view is selected)
Syntax:	**LevelSuppressed** *objLevel*
Where:	**objLevel** specifies the level and is the object name, object identifier or both.
Example:	`LevelSuppressed "Region"`

Name

UI Equivalent:	View names appear in the left hand side of the Category Viewer
Syntax:	**Name** *objView*
Where:	*objView* is the object name.
Example:	`Name "Europe"`

NotCloak

UI Equivalent:	Diagram menu, Cloak (cleared)
Syntax:	**NotCloak** *objCat*
Where:	*objCat* specifies the category and is the object name, object identifier, or both.
Example:	`NotCloak "Far East"`

NotFilter

UI Equivalent:	Diagram menu, Exclude (cleared)
Syntax:	**NotFilter** *objCat*
Where:	*objCat* specifies the category and is the object name, object identifier, or both.
Example:	`NotFilter "Customer Type"`

NotSummary

UI Equivalent:	Diagram menu, Summary (cleared)
Syntax:	**NotSummary** *objCat*
Where:	*objCat* specifies the category and is the object name, object identifier, or both
Example:	`NotSummary "Region"`

NotSuppressed

UI Equivalent:	Diagram menu, Suppressed (cleared)
Syntax:	**NotSuppressed** *objCat*
Where:	*objCat* specifies the category and is the object name, object identifier, or both.
Example:	`NotSuppressed "Region"`

Summary

UI Equivalent:	Diagram menu, Summarize (when a category in a view is selected)
Syntax:	**Summary** *objCat*
Where:	*objCat* specifies the category and is the object name, object identifier, or both.
Example:	`Summary "Europe"`

Suppressed

UI Equivalent:	Diagram menu, Suppressed (when a category in a view is selected)
Syntax:	**Suppressed** *objCat*
Where:	*objCat* specifies the category and is the object name, object identifier, or both.
Example:	`Suppressed "Far East"`

ViewUserClass

UI Equivalent:	None
Syntax:	**ViewUserClass** {*objUserClass*|*none*}
Where:	*objUserClass* is the user class object name, object identifier or both. *none* specifies that the view is not a user class view but is a dimension view. *none* is zero (0) if you are using object identifiers, or empty quotes ("") if you are using only object names.
Example:	`ViewUserClass ""`

Appendix A: Structured MDL

This appendix includes

There are two forms of MDL: verb MDL and structured MDL. Each form can provide a complete and independent definition of a Transformer model. Although the purpose of this book is to provide a reference to verb MDL, this chapter briefly describes structured MDL. Structured MDL keywords are defined in relation to their verb counterparts.

History of MDL

MDL was originally created to store Transformer models so that they could be opened in future or upgraded versions of Transformer. The original format of MDL contains statements that define the model in a fixed order, or structure. That is structured MDL.

Later, MDL verbs were created to enhance the functionality of MDL and verb MDL became an independent, parallel format. To maintain upward compatibility between versions of Transformer, structured MDL remains and continues to be the default, even though verb MDL has effectively replaced it. If you are planning to use MDL you should change the .ini setting to generate verb MDL.

For more information about Trnsfrmr.ini settings, see "Change Models with MDL" (p. 14).

Comparison of Structured and Verb Keywords

The following table lists Transformer objects. Beside each object is the structured keyword that is used in MDL models to define the object, and the verbs that define and manipulate the object.

There are additional verbs not included in this list. For a complete list of verbs, see "MDL Verbs" (p. 55).

Transformer object	Structured keyword	Verbs
Model	Name	NewModel
Data Source	DataSource	DataSourceAdd, DataSourceDelete, SourceListUpdate, DataSourceMake, DataSourceUpdate

Transformer object	Structured keyword	Verbs
Column	OrgName	ColumnAdd, ColumnDelete, ColumnListUpdate, ColumnMake, ColumnUpdate
Dimension	Dimension	DimAdd, DimDelete, DimMake, DimUpdate, DimensionListUpdate
Subdimension	SubDimCat	SubDimRootMake, SubDimRootUpdate
Root Category	Root	RootCatMake, RootCatUpdate
Drill Category	Drill	DrillCatMake
Level	Levels	LevelAdd, LevelDelete, LevelMake, LevelMoveAfter, LevelMoveBefore, LevelNewDrill, LevelUpdate, FilterLevel
Category	Category	CatAdd, CatDelete, CatJoin, CatMake, CatMorph, CatMoveLevel, CatMoveVertical, CatUpdate, FilterCat
Special Category	SpecialCategory	SpecialCatAdd, SpecialCatDelete, SpecialCatMake, SpecialCatUpdate
View	ViewName, DimensionView	ApexCat, ViewAdd, ViewDelete, ViewListUpdate, ViewMake, ViewUpdate
Measure	Measure	MeasureAdd, MeasureDelete, MeasureMake, MeasureListUpdate, MeasureUpdate
PowerCube	Cube	CubeAdd, CubeDelete, CubeMake, CubeUpdate, PowerCubeDelete, PowerCubeListUpdate, PowerCubeUserListUpdate
PowerCube Group	CubeGroup	CubeGroupAdd, CubeGroupDelete, CubeGroupMake, CubeGroupUpdate, PowerCubeDelete, PowerCubeListUpdate
PowerCube Group Cube	CubeGroupCube	CubeGroupCubeAdd, CubeGroupCubeDelete, CubeGroupCubeListUpdate, CubeGroupCubeMake, CubeGroupCubeUpdate

Transformer object	Structured keyword	Verbs
Signon	Signon	SignonAdd, SignonDelete, SignonListUpdate, SignonMake, SignonUpdate
User class	UserClass	UserClassAdd, UserClassDelete, UserClassListUpdate, UserClassMake, UserClassUpdate, PowerCubeUserListUpdate
Dimension Calculation Definition	DimCalcDef	DimCalcDefAdd, DimCalcDefDelete, DimCalcDefMake, DimCalcDefUpdate
Currency Table	CurrencyTable	CurrencyTableAdd, CurrencyTableDelete, CurrencyTableMake, CurrencyTableUpdate
Currency	CurrencyRecord	CurrencyAdd, CurrencyDelete, CurrencyMake, CurrencyUpdate, CurrencyTableAdd

Using Structured MDL

Structured MDL requires that object definitions appear in a specific order, or structure. For example, after a data source definition, all the columns in the data source must be defined. Then the associated dimensions are defined, followed by the root category, drill category, levels and categories.

This structure provides information about the location of each object. For example, to uniquely identify a category in verb MDL, you may need to specify its parents, level, drill category and dimension. In structured MDL, all this information is supplied by the placement of the category definition in the structured MDL file.

Verb statements can be used independently of structured statements to completely define a Transformer model, or they can be used in conjunction with structured statements, as long as the verb statements are at the end of the model. Once a verb statement is used, structured statements cannot follow.

Example

Following is an example of a simple model in structured MDL format:

```
Name "New Model" AutoAccess False Synchronize False SynchroCycle
0 SynchroStamp 0

UpdateCycle 1 ModelStamp 944340933 ClientStamp 944340933
ServerStamp 0

Generation 0 RootUser 0 Version "7.x - Not For Resale"
AccessManager False

AccessManagerUserClass False

DataSource 103 "national" Separator "," SourceType FlatFile_ColNames CharacterSet
Ansi
```

```
DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault

Source "c:\national.asc"

Speed False SetCurrent True ServerSource False Presummarized
False EnableMultiProcess False

OrgName 105 "Date" Origin Generated Offset 0 Column "Date"
Storage Default

Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off

OrgName 107 "Line" Origin Generated Offset 1 Column "Line"
Storage Default

Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off

OrgName 109 "Brand" Origin Generated Offset 2 Column
"Brand" Storage Default

Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off

OrgName 111 "Item Name" Origin Generated Offset 3 Column
"Item Name" Storage Default

Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off

OrgName 113 "Item" Origin Generated Offset 4 Column "Item"
Storage Default

Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off

OrgName 115 "State" Origin Generated Offset 5 Column
"State" Storage Default

Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off

OrgName 117 "Outlet" Origin Generated Offset 6 Column
"Outlet" Storage Default

Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off

OrgName 121 "Revenue" Origin Generated Offset 8 Column
"Revenue" Storage Default

Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off

Dimension 209 "State" DimType Regular NewCatsLock False ExcludeAutoPartitioning
False

Categories

Root 211 "State" Inclusion Generate Lastuse 19991204
Filtered False Suppressed False

Sign False IsKeyOrphanage False IsTruncated False Blanks
False

Drill 213 "By State" Inclusion Suppress Filtered False
Suppressed True

PrimaryDrill True

Levels 219 "State" Blanks "( blank )" DateFunction None
Generate Need RefreshLabel False

RefreshDescription False RefreshShortName False NewCatsLock
False Timerank 0

UniqueCategories False UniqueMove False Associations
221 "State" AssociationType Type_Query

AssociationRole Role_Source AssociationReferenced "State"

Levels 223 "Outlet" Blanks "( blank )" DateFunction None
Generate Need

RefreshLabel False RefreshDescription False RefreshShortName
False NewCatsLock False

Timerank 0 UniqueCategories False UniqueMove False Associations
225 "Outlet"
```

```
AssociationType Type_Query AssociationRole Role_Source
AssociationReferenced "Outlet"

MapDrills

MapDrill 213

ViewName 215 "All Categories" Type All ViewUserClass
0

ViewName 217 "Omit Dimension" Type Omit ViewUserClass
0

Measure 173 "Revenue" Storage Default OutPutScale 0 Decimals
0 ReverseSign False

IsCurrency False DrillThrough False EndList Associations
175 "Revenue"

AssociationType Type_Query AssociationRole Role_Source
AssociationReferenced "Revenue"

AllocationAdd Measure 173 Type Default
```

Index

D

N

S

W

Y