IBM Cognos Transformer
Version 10.2.1

Developer Guide

IBM
Product Information

This document applies to IBM Cognos Business Intelligence Version 10.2.1 and may also apply to subsequent releases.

Licensed Materials - Property of IBM
© Copyright IBM Corporation 2007, 2013.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
# Contents

**Introduction** ........................................... xiii

**Chapter 1. What’s New** ........................................ 1
New features in version 10.2.1 ........................................ 1
New features in version 10.2.0 ........................................ 1
New option for defining hierarchies in a time dimension .............. 1

**Chapter 2. Model Definition Language (MDL)** ................. 3
.psy and .mdl formats ............................................ 3
User interface and MDL representations of a model object ............... 4
MDL usage ............................................. 5
Example - Changing a model object using MDL ......................... 6
Cognos Transformer and verb MDL ................................... 7

**Chapter 3. Cognos Transformer objects** ....................... 9
Object identifiers ........................................ 9
Object names ............................................ 10
  Uniqueness of object names ........................................ 11
  Category names ............................................ 11
  Column names ............................................ 12
  View names ............................................ 12
Locating objects uniquely ........................................ 13
  Uniquely locate existing objects .................................... 13
  Uniquely locate new objects ..................................... 14
Setting global preferences .......................................... 14
  Specify a preferred category sorting option ......................... 14

**Chapter 4. Creating a complete MDL model** ................. 15
Example - MDL model using IBM Cognos data sources in verb MDL ....... 15
  Create the data sources in Cognos Transformer ....................... 15
Example - MDL model using an IQD data source in verb MDL ............ 16
  Step 1: Create the data sources in Cognos Transformer ............... 16
  Step 2: Create the time dimension .................................. 17
  Step 3: Create the products dimension ................................ 18
  Step 4: Create key product measures .................................. 19
  Step 5: Create allocations ....................................... 19
  Step 6: Create the signon ........................................ 19
  Step 7: Populate the model and create the PowerCube ................. 20
  Step 8: Create the model by combining the scripts ................. 20
Checking the model generated by Cognos Transformer .................... 21
Example - MDL model using an IBM Cognos data source in structured MDL . . . 34
Example - MDL model using an IBM Cognos data source in verb MDL ....... 45
Sample scripts ........................................... 56
  Change data source types sample .................................. 57
  Change data source sample ....................................... 57
  Update selected cubes sample .................................... 57
  Update all cubes sample ....................................... 57
  Description samples ........................................... 57
  Add several descriptions sample ................................... 57
  Change a cube output location sample ................................ 58
  Change several cube output locations sample ......................... 58
  Change the PowerCube output type sample ................................ 58
  Change optimization setting sample ................................ 58
  Automate CleanHouse sample ..................................... 58
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeMake</td>
<td>99</td>
</tr>
<tr>
<td>CubeUpdate</td>
<td>100</td>
</tr>
<tr>
<td>CurrencyAdd</td>
<td>101</td>
</tr>
<tr>
<td>CurrencyDelete</td>
<td>102</td>
</tr>
<tr>
<td>CurrencyMake</td>
<td>103</td>
</tr>
<tr>
<td>CurrencyTableAdd</td>
<td>105</td>
</tr>
<tr>
<td>CurrencyTableDelete</td>
<td>105</td>
</tr>
<tr>
<td>CurrencyTableMake</td>
<td>106</td>
</tr>
<tr>
<td>CurrencyTableUpdate</td>
<td>107</td>
</tr>
<tr>
<td>CurrencyUpdate</td>
<td>108</td>
</tr>
<tr>
<td>CustomViewAdd</td>
<td>109</td>
</tr>
<tr>
<td>CustomViewChildListUpdate</td>
<td>110</td>
</tr>
<tr>
<td>CustomViewDelete</td>
<td>110</td>
</tr>
<tr>
<td>CustomViewMake</td>
<td>110</td>
</tr>
<tr>
<td>CustomViewUpdate</td>
<td>111</td>
</tr>
<tr>
<td>DataSourceAdd</td>
<td>112</td>
</tr>
<tr>
<td>DataSourceDelete</td>
<td>113</td>
</tr>
<tr>
<td>DataSourceMake</td>
<td>114</td>
</tr>
<tr>
<td>DataSourceUpdate</td>
<td>115</td>
</tr>
<tr>
<td>DeletionListUpdate</td>
<td>115</td>
</tr>
<tr>
<td>DimAdd</td>
<td>116</td>
</tr>
<tr>
<td>DimCalcDefAdd</td>
<td>116</td>
</tr>
<tr>
<td>DimCalcDefDelete</td>
<td>118</td>
</tr>
<tr>
<td>DimCalcDefMake</td>
<td>118</td>
</tr>
<tr>
<td>DimCalcDefUpdate</td>
<td>119</td>
</tr>
<tr>
<td>DimDelete</td>
<td>121</td>
</tr>
<tr>
<td>DimDimensionListUpdate</td>
<td>121</td>
</tr>
<tr>
<td>DimMake</td>
<td>121</td>
</tr>
<tr>
<td>DimUpdate</td>
<td>122</td>
</tr>
<tr>
<td>DrillCatMake</td>
<td>123</td>
</tr>
<tr>
<td>EventEnd</td>
<td>124</td>
</tr>
<tr>
<td>EventStart</td>
<td>124</td>
</tr>
<tr>
<td>FilterCat</td>
<td>124</td>
</tr>
<tr>
<td>LevelAdd</td>
<td>125</td>
</tr>
<tr>
<td>LevelDelete</td>
<td>126</td>
</tr>
<tr>
<td>LevelMake</td>
<td>127</td>
</tr>
<tr>
<td>LevelMoveAfter</td>
<td>128</td>
</tr>
<tr>
<td>LevelMoveBefore</td>
<td>128</td>
</tr>
<tr>
<td>LevelNewDrill</td>
<td>129</td>
</tr>
<tr>
<td>LevelUpdate</td>
<td>130</td>
</tr>
<tr>
<td>MDCClear</td>
<td>130</td>
</tr>
<tr>
<td>MeasureAdd</td>
<td>131</td>
</tr>
<tr>
<td>MeasureDelete</td>
<td>132</td>
</tr>
<tr>
<td>MeasureListUpdate</td>
<td>132</td>
</tr>
<tr>
<td>MeasureMake</td>
<td>132</td>
</tr>
<tr>
<td>MeasureUpdate</td>
<td>134</td>
</tr>
<tr>
<td>ModelEnsureCompleteness</td>
<td>134</td>
</tr>
<tr>
<td>NewModel</td>
<td>134</td>
</tr>
<tr>
<td>OpenDef</td>
<td>135</td>
</tr>
<tr>
<td>OpenMDL</td>
<td>135</td>
</tr>
<tr>
<td>OpenPY</td>
<td>136</td>
</tr>
<tr>
<td>PopulateFromQueries</td>
<td>137</td>
</tr>
<tr>
<td>PopulateModel</td>
<td>137</td>
</tr>
<tr>
<td>PowerCubeCustomViewListUpdate</td>
<td>137</td>
</tr>
<tr>
<td>PowerCubeDelete</td>
<td>138</td>
</tr>
<tr>
<td>PowerCubeListUpdate</td>
<td>138</td>
</tr>
<tr>
<td>PromptAdd</td>
<td>139</td>
</tr>
<tr>
<td>PromptDelete</td>
<td>139</td>
</tr>
<tr>
<td>PromptMake</td>
<td>139</td>
</tr>
<tr>
<td>PromptUpdate</td>
<td>140</td>
</tr>
<tr>
<td>ReportPartitions</td>
<td>140</td>
</tr>
</tbody>
</table>
Chapter 7. MDL options .................................... 161
appqueryopts ........................................ 161
AppInfo ............................................ 161
CharacterSet ......................................... 161
CognosSource ........................................ 162
Columns ............................................ 162
DataRange ........................................... 162
DecimalSep ........................................... 162
ImrName ............................................. 163
Isolation ............................................. 163
ModelStamp .......................................... 163
ModelNonRollupHierarchies ......................... 164
PackageReportSource ................................ 164
PackageTimeStamp .................................... 164
PreSummarized ....................................... 165
SegmenterPrompt ..................................... 165
SegmenterPromptEnabled .............................. 165
Separator ............................................ 165
SetCurrent ........................................... 165
Source ............................................... 166
SourceInfo ............................................ 166
SourcePath .......................................... 166
SourceSignonList .................................... 166
SourceType .......................................... 166
Speed ............................................... 166
SQL ................................................... 167
Stamp ................................................. 168
StreamExtractSize .................................... 168
Decimals ................................... 188
Detail ..................................... 188
Format .................................... 188
InputScale .................................. 188
Offset ..................................... 189
Origin ..................................... 189
Scale ...................................... 189
Size ....................................... 189
Storage .................................... 190
TimeArray .................................. 190
TimeArrayCol ............................... 190
TimeArrayMonth ............................ 190
currencyrecordopts ................................ 190
BaseCurrency ............................... 190
CountryCode ............................... 191
CurrencyCountryLabel ....................... 191
CurrencyDecimals ........................... 191
CurrencyFormatOverride ..................... 192
CurrencyIsEMU ............................. 192
CurrencySymbol ............................ 192
EmuEntryDate ............................. 192
EuroBaseCurrency .......................... 192
currencytableopts .................................. 193
CurrencyCountryCodeColumn .................. 193
CurrencyDateColumn ........................ 193
CurrencyLabelColumn ....................... 193
CurrencyRateColumn ....................... 194
CurrencyTableType .......................... 194
deletionopts .................................. 194
CalcDef .................................... 194
Column ..................................... 194
Cube ....................................... 195
DataSource .................................. 195
Dimension ................................... 195
Levels ...................................... 195
Measure ..................................... 195
Signon ...................................... 196
View ........................................ 196
dimopts ....................................... 196
Association .................................. 196
DaysInWeek .................................. 196
DimDefaultCategory ......................... 197
DimInfo ..................................... 197
DimType ..................................... 197
EarliestDate .................................. 198
ExcludeAutoPartitioning .................... 198
LatestDate ................................... 198
ManualPeriods .............................. 198
NewCatsLock .................................. 199
filteropts ..................................... 199
CognosPackageFilterRef ..................... 199
DATASOURCE .................................. 199
CognosPackageFilterDelete .................. 199
CognosPackageFilterUpdate .................. 199
ExpMark ..................................... 200
FilterDescription ........................... 200
levelopts ..................................... 200
Association .................................. 200
Blanks ....................................... 200
CategoryCode ............................... 201
CatLabLabFormat ............................. 201
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeUpdateLock</td>
<td>218</td>
</tr>
<tr>
<td>Database</td>
<td>219</td>
</tr>
<tr>
<td>DatabaseInfo</td>
<td>219</td>
</tr>
<tr>
<td>DataSource</td>
<td>219</td>
</tr>
<tr>
<td>DeployLocations</td>
<td>220</td>
</tr>
<tr>
<td>DeployType</td>
<td>220</td>
</tr>
<tr>
<td>DetailLevel</td>
<td>220</td>
</tr>
<tr>
<td>DrillThrough</td>
<td>220</td>
</tr>
<tr>
<td>Exclude</td>
<td>221</td>
</tr>
<tr>
<td>IncrementalUpdate</td>
<td>221</td>
</tr>
<tr>
<td>Information</td>
<td>221</td>
</tr>
<tr>
<td>MDCFile</td>
<td>222</td>
</tr>
<tr>
<td>MeasureName</td>
<td>222</td>
</tr>
<tr>
<td>Optimize</td>
<td>222</td>
</tr>
<tr>
<td>PartitionSize</td>
<td>222</td>
</tr>
<tr>
<td>PassesNumber</td>
<td>223</td>
</tr>
<tr>
<td>Password</td>
<td>223</td>
</tr>
<tr>
<td>PublishAllowAccessToSuppressionOptions</td>
<td>223</td>
</tr>
<tr>
<td>PublishAllowMultiEdgeSuppression</td>
<td>223</td>
</tr>
<tr>
<td>PublishAllowNullSuppression</td>
<td>223</td>
</tr>
<tr>
<td>SegmenterDimension</td>
<td>224</td>
</tr>
<tr>
<td>SegmenterLevel</td>
<td>224</td>
</tr>
<tr>
<td>ServerCube</td>
<td>224</td>
</tr>
<tr>
<td>Status</td>
<td>224</td>
</tr>
<tr>
<td>SummaryLevel</td>
<td>225</td>
</tr>
<tr>
<td>UseAlternateFilename</td>
<td>225</td>
</tr>
<tr>
<td>TimeBasedPartitionedCube</td>
<td>225</td>
</tr>
<tr>
<td>promptopts</td>
<td>225</td>
</tr>
<tr>
<td>PromptType</td>
<td>225</td>
</tr>
<tr>
<td>PromptValue</td>
<td>226</td>
</tr>
<tr>
<td>signonopts</td>
<td>226</td>
</tr>
<tr>
<td>PromptForPassword</td>
<td>226</td>
</tr>
<tr>
<td>UserId</td>
<td>226</td>
</tr>
<tr>
<td>password</td>
<td>226</td>
</tr>
<tr>
<td>AutoLogon</td>
<td>227</td>
</tr>
<tr>
<td>SignonNamespace</td>
<td>227</td>
</tr>
<tr>
<td>SignonType</td>
<td>227</td>
</tr>
<tr>
<td>SignonInfo</td>
<td>227</td>
</tr>
<tr>
<td>viewopts</td>
<td>227</td>
</tr>
<tr>
<td>Apex</td>
<td>228</td>
</tr>
<tr>
<td>Cloak</td>
<td>228</td>
</tr>
<tr>
<td>Filter</td>
<td>228</td>
</tr>
<tr>
<td>LevelCloak</td>
<td>228</td>
</tr>
<tr>
<td>LevelFilter</td>
<td>229</td>
</tr>
<tr>
<td>LevelSummary</td>
<td>229</td>
</tr>
<tr>
<td>LevelSuppressed</td>
<td>229</td>
</tr>
<tr>
<td>Name</td>
<td>230</td>
</tr>
<tr>
<td>NotCloak</td>
<td>230</td>
</tr>
<tr>
<td>NotFilter</td>
<td>230</td>
</tr>
<tr>
<td>NotSummary</td>
<td>230</td>
</tr>
<tr>
<td>NotSuppressed</td>
<td>231</td>
</tr>
<tr>
<td>Summary</td>
<td>231</td>
</tr>
<tr>
<td>Suppressed</td>
<td>231</td>
</tr>
</tbody>
</table>

**Chapter 8. Structured MDL** .......................... 233

History of MDL ............................................ 233
Comparison of structured and verb keywords ................. 233
Using structured MDL ...................................... 235
  Example ................................................. 235

**Notices** .................................................. 241
**Introduction**

This information is intended for use with IBM® Cognos® Transformer, the OLAP modeling component delivered with IBM Cognos Business Intelligence.

You can use the *IBM Cognos Transformer Developer Guide* as a reference when writing Model Definition Language (MDL) scripts for IBM Cognos Transformer.

**Audience**

This book is intended for experienced Cognos Transformer users who have an advanced understanding of Cognos Transformer concepts, functionality, and terminology. It provides all the information you need to understand MDL representation of Cognos Transformer models, but it does not explain the underlying Cognos Transformer features. Where possible, an equivalent in the user interface is given, and you should refer to the Cognos Transformer documentation for more information.

**Finding information**

To find IBM Cognos product documentation on the web, including all translated documentation, access one of the [IBM Cognos Information Centers](http://pic.dhe.ibm.com/infocenter/cogic/v1r0m0/index.jsp). Release Notes are published directly to Information Centers, and include links to the latest technotes and APARs.

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

**Accessibility features**

IBM Cognos Transformer does not currently support accessibility features that help users with a physical disability, such as restricted mobility or limited vision, to use this product.

IBM Cognos HTML documentation has accessibility features. PDF documents are supplemental and, as such, include no added accessibility features.

**Forward-looking statements**

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of features or functionality remain at the sole discretion of IBM.

**Samples disclaimer**

The Sample Outdoors Company, Great Outdoors Company, GO Sales, any variation of the Sample Outdoors or Great Outdoors names, 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. Other sample files may contain fictional data manually or
machine generated, factual data compiled from academic or public sources, or data
used with permission of the copyright holder, for use as sample data to develop
sample applications. Product names referenced may be the trademarks of their
respective owners. Unauthorized duplication is prohibited.
Chapter 1. What's New

This chapter contains a list of new, changed, and removed features for this release. It also contains a cumulative list of similar information for previous releases. Knowing this information will help you plan your upgrade and application deployment strategies and the training requirements for your users.

For information about upgrading, see the IBM Cognos Business Intelligence Installation and Configuration Guide.

For an overview of new features for this release, see the IBM Cognos Business Intelligence New Features Guide.

What's New information for past releases, including versions 8.3 and 8.4, is available by accessing documentation within the IBM Cognos Business Intelligence 10.1.1 (http://pic.dhe.ibm.com/infocenter/cbi/v10r1m1/index.jsp?) or IBM Cognos Business Intelligence 10.1.0 (http://pic.dhe.ibm.com/infocenter/cbi/v10r1m0/index.jsp?) information center.

To review an up-to-date list of environments supported by IBM Cognos products, such as operating systems, patches, browsers, Web servers, directory servers, database servers, and application servers, visit the IBM Cognos Customer Center (http://www.ibm.com/software/data/cognos/customercenter).

New features in version 10.2.1

There are no new features in version 10.2.1.

New features in version 10.2.0

Listed below are features that are new in version 10.2.0.

New option for defining hierarchies in a time dimension

Starting with this version of IBM Cognos Transformer, all non-rollup children of the time dimension root category are defined as separate hierarchies in a PowerCube.

In PowerCubes created with previous versions of Transformer, when there is only one non-rollup child of the root category in a time dimension, that category is not considered to be a base of a hierarchy. Instead, each of its children defines a hierarchy.

In MDL, the new rule for defining hierarchies in a time dimension is represented by the ModelNonRollupHierarchies option associated with the NewModel verb. For more information, see “ModelNonRollupHierarchies” on page 164.
Chapter 2. Model Definition Language (MDL)

MDL is the proprietary model definition language for Cognos Transformer. You use MDL to create, manipulate, update, and store models without accessing the user interface. You can set preferences globally and also automate Cognos Transformer functions for greater efficiency.

Tip: You may want to modify the cogtr.xml file, installed in the installation_location/configuration directory, to work more effectively in your particular production environment. For example, you can choose to work in structured MDL rather than verb format.

MDL represents Cognos Transformer models in text format (.mdl) 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 Cognos Transformer. Note that it is equivalent to the .mdl file, with the exception of any adjustments made on the user interface and, in some instances, passwords. For more information about Cognos Transformer file formats, see “.py? and .mdl formats.”

MDL model statements 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 its object name, identifying a specific data source object
  - SourceType ExcelDatabase is the option that specifies the type of data source

You can use MDL in two ways:
  - To create an MDL model that completely defines a Cognos Transformer model
  - To create an MDL script that manipulates or updates an existing Cognos Transformer model

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

For examples of MDL scripts, see “Sample scripts” on page 56.

.py? and .mdl formats

You can save Cognos Transformer models in two formats: .py? and .mdl. You can only open models saved as .py? files in the Cognos Transformer user interface. However, you can open models saved as .mdl files in either the user interface or a text editor.

Each .mdl file is a plain-text representation of the model, and may be compatible with other versions of Cognos Transformer than the one used to create it. However, it loads more slowly than a .py?-format file because it recreates all of the Cognos Transformer objects each time it loads.
You can open models saved as .mdl files on any supported UNIX, Linux, or Windows operating system, making this the preferred format to use in a mixed production environment.

The .py? format is a binary representation of the model. Although not compatible between versions of Cognos Transformer, it loads faster than the .mdl-format equivalent, even though it is larger, because the Cognos Transformer objects are not recreated when the model reloads.

As you edit a model, the size of the associated .py? file increases because information about the operations performed during model editing is retained in the file. Cognos Transformer uses this information when performing incremental updates and other subsequent cube operations. Lengthy descriptions of the appended operations can cause fragmentation of the binary file. Prevent this by periodically saving your .py? models as .mdl files.

**User interface and MDL representations of a model object**

Based on an older version of the Great Outdoors sample, this illustration shows the Products dimension property sheet as it displays on the Cognos Transformer user interface.

To replicate this view, open the model on the user interface and, on the Titles tab of the Preferences dialog box, select the Object identifier display option. The code for the dimension now displays in the property sheet title; in our example, Products is identified by code 385.
To see the corresponding MDL definition, save the model as an .mdl file, and reopen it in a text editor. You can then use the Find utility with the identifier (385) to locate the MDL statement that defines that dimension.

Tip: View your text format .mdl files in an editor that enables line-by-line numbering because Cognos Transformer error messages refer to the line on which an error occurred. However, you can use any text editor to edit MDL, as long as you save the file in text format.

On the user interface, the dimension type is Regular and the Prohibit automatic creation of new categories option is not selected. These choices are represented in the MDL statement by the options DimType Regular and NewCatsLock False as follows:

```
Dimension 385 "Products" DimType Regular NewCatsLock False ExcludeAutoPartitioning False DimDefaultCategory 0
```

Note: For easier readability, line breaks are inserted in lines over 70 characters in length. This does not affect the functionality.

**MDL usage**

Although most users find it easier to create and maintain their Cognos Transformer models and cubes from the user interface, there are situations where an experienced Cognos Transformer modeler will benefit from using MDL.
The following are some of the ways you can use MDL, beginning with basic operations and progressing to more advanced applications:

- You can save your Cognos Transformer model as an .mdl file by selecting **Save As** from the **File** menu, changing the extension of the file to .mdl, and then saving the file. You can then use the .mdl file to open the model in subsequent versions of Cognos Transformer, move models between various production servers and platforms, or archive Cognos Transformer models.

Models stored as .mdl files are compact, and because they can be opened in subsequent versions of Cognos Transformer, you need not worry about updating them.

You can also use an .mdl file to improve the performance of Cognos Transformer models.

Saving a .py? file in .mdl format integrates all of the changes made since the last MDL save into a clean model. It also prevents possible corruption of the model.

- You can open an .mdl file in any text editor to perform global edits, such as changing all directory paths from C:\ to D:\ or changing all instances of "Years" to "Yrs".

- You can write scripts in MDL to manipulate Cognos Transformer models.

For example, you can store repetitive tasks in an MDL batch file that can be run any time.

- You can use a text editor to work directly in .mdl files with Cognos Transformer models.

For example, you can change a model on a UNIX server, create models as needed, and store parts or all of the resulting models in separate files.

- You can create custom Cognos Transformer applications.

Because MDL is an interpreted language, you can process .mdl files in any of the following ways:

- Open the .mdl file on the Cognos Transformer user interface.
- Run the .mdl file from the Windows command line using cogtr.exe.
- Access the .mdl file on a UNIX server, using the command line option -m.

### Example - Changing a model object using MDL

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

The three supported methods are listed below:

- Update the definition with a script.
- Change the MDL definition directly.
- Update the definition within the model.

For the sample model Great Outdoors.mdl, suppose you want to change the source file for Products (CSV) from Prodinfo.csv to P0001.csv. The original definition of the data source is as follows:

```
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 data source file to P0001.csv. This script 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"

Alternatively, you can change the object definition in the model. In this case, you can delete the word Prodinfo in the DataSourceMake statement, and then 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 this DataSourceUpdate statement to the end of the model definition file:

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

The next time you open the model in Cognos Transformer, the change will be implemented. The original DataSourceMake statement is not altered, but your addition remains in effect as long as the DataSourceUpdate statement remains attached to the model definition.

If you update an object within the model, Cognos 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, as follows:

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

For information about command line options that you can use when running scripts, see the Cognos Transformer User Guide. For sample MDL scripts, see "Sample scripts" on page 56.

### Cognos Transformer and verb MDL

There are two types of Model Definition Language: verb MDL and structured MDL. Both types can be used to completely define a Cognos Transformer model. However, by default, Cognos Transformer generates models in structured format, the original version of MDL. This helps to ensure compatibility between models created in different versions of Cognos Transformer.

Use verb MDL when you want to make changes to an existing model, but first you must add the following line to the cogtr.xml file:

```
<Preference Name="VerbOutput" Value="1"/>
```
Setting this parameter only affects the format of model definitions subsequently generated and saved by Cognos Transformer. Regardless of this setting, you can still use both verb and structured format on any output file.

For more information about using structured MDL, see Chapter 8, “Structured MDL,” on page 233.
Chapter 3. Cognos Transformer objects

This section describes the objects required to build models in Cognos Transformer and explains how to specify unique object identifiers, names, and locations in Model Definition Language (MDL).

An object is a definition of a particular component of the model. For example, the Locations data source object contains all the information in Cognos Transformer about the data source called Locations.

On the user interface, the Locations object is defined in the Data Source property sheet for Locations. In MDL, the same information is found in the MDL statement that defines the Locations data source object. The property sheet settings are represented in the MDL statement as options.

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

- Data Source
- IBM Cognos Source (an IBM Cognos package)
- Column
- Dimension
- Level
- Root Category
- Drill Category
- Regular Category
- Special Category
- Measure
- PowerCube
- PowerCubeGroup
- PowerCubeGroupCube
- Subdimension
- Signon
- View
- Custom View
- Currency Table
- Currency
- Dimension Calculation Definition

All Cognos Transformer objects have property sheets on the Cognos Transformer user interface. Only objects have property sheets.

Object identifiers

You can track every Cognos Transformer object on the user interface and in MDL by its object identifier or its object name.
To show both object identifiers and object names on the Cognos Transformer user interface, ensure that both the Object Identifier and Object Name check boxes are selected on the Titles tab of the Preferences dialog box.

To exclude the object identifier from the user interface, clear the Object Identifier check box. You can also prevent the object identifier from appearing in the MDL file by clearing the Save Object Unique Identifier values in MDL check box on the General tab of the Preferences property sheet.

In addition to the requirement that the object identifier be unique within each model, Cognos Transformer object identifiers have several other defining characteristics, as follows:

- When auto-generated by Cognos Transformer (Windows), odd numbers identify all objects. This is a carry-over from Cognos Transformer version 7.x, which supported synchronized client-server operations, and assigned only even numbers to the equivalent server objects.
- Object identifiers are greater than 100 and less than 4,294,967,295.
- Once assigned, an object identifier cannot be changed by the modeler or the application.

As the modeler, you can specify a particular identifier and name when you create an object. For example, you can use the MDL statement DataSourceMake 103 "Natsmall" to uniquely identify the data source object by the identifier 103 and the name Natsmall.

However, because manually assigned identifiers can be difficult to maintain, and cannot be used with incrementally updated cubes, allow Cognos Transformer to automatically generate the object identifiers. A different odd number is associated with each object, and an object name is assigned if it can be determined from the source data.

Tip: If you copy and paste parts of your model from different MDL files, change the ObjecyIdOutput setting in the cogtr.xml file to zero, to prevent object identifiers from appearing in the MDL files. Otherwise, you cannot guarantee that the object identifiers will remain unique after they are merged.

### Object names

If you create a Cognos Transformer object manually using MDL, you must supply a name for it.

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

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"
```

An object name must have the following structure:

- The name must be enclosed by single or double quotes.
- The name cannot contain a carriage return.
- The name cannot be more than 256 characters in length.
## 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:

<table>
<thead>
<tr>
<th>Object names</th>
<th>Restrictions on uniqueness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category names</td>
<td>Must be within a dimension; applies to regular, root, drill, and special categories; see “Category names.”</td>
</tr>
<tr>
<td>Column names</td>
<td>Must be within a data source; see “Column names” on page 12.</td>
</tr>
<tr>
<td>Currency names</td>
<td>Must be within a model</td>
</tr>
<tr>
<td>Currency table names</td>
<td>Must be within a model</td>
</tr>
<tr>
<td>Data Source names, including package and report names</td>
<td>Must be within a model</td>
</tr>
<tr>
<td>DimCalcDef names</td>
<td>Must be within a dimension</td>
</tr>
<tr>
<td>Dimension names</td>
<td>Must be within a model</td>
</tr>
<tr>
<td>Level names</td>
<td>Must be within dimensions, if used in an advanced subset</td>
</tr>
<tr>
<td>Measure names</td>
<td>Must be within a model</td>
</tr>
<tr>
<td>PowerCube names</td>
<td>Must be within a model; cannot be the same as a PowerCubeGroup name</td>
</tr>
<tr>
<td>PowerCubeGroup names</td>
<td>Must be within a model; cannot be the same as a PowerCube name</td>
</tr>
<tr>
<td>PowerCubeGroupCube names</td>
<td>Must be within a group</td>
</tr>
<tr>
<td>Signon names (IQD sources)</td>
<td>Must be within a model</td>
</tr>
<tr>
<td>Subdimension names</td>
<td>Must be within a dimension</td>
</tr>
<tr>
<td>View names</td>
<td>Must be within a dimension; see “View names” on page 12.</td>
</tr>
</tbody>
</table>

### Category names

Categories are identified by a category label and a category code, rather than by a single name. On the Cognos Transformer user interface, category names are derived from category labels. However, in MDL, category names are derived from category codes.

Category codes are not the same as object identifiers. To show both on the Cognos Transformer user interface, you must select the Object Identifier and Category Code check boxes on the Titles tab of the Preferences dialog box.
For example, in this **Category** property sheet, both the object identifier **6609** and the category code **104** display in the title bar for the category named **Extreme Outdoors**. The name is derived from the category label and cannot contain either a tilde ("~") or an at sign (@).

In the corresponding MDL statement, the object identifier is the same: **6809**. However, the object name **104** is enclosed in double quotation marks and is based on the category code.

```
Category 6809 "104"
Parent 6425 Levels 1067 OrderBy Drill 1057 Value "104" Label "Extreme Outdoors"
Lastuse 20060920 SourceValue "104" Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False
```

### Column names

Column names are identical in MDL and on the Cognos Transformer user interface, unless they are manually changed. Column names cannot contain an at sign (@).

### View names

In structured MDL, views are represented by the following syntax:

```
ViewName id "Name"
```
where \( i d \) is the unique IBM Cognos Transformer identifier and "Name" is the name of the view.

### Locating objects uniquely

If an object cannot be uniquely identified by its name or identifier, you must specify its relationship within a hierarchy of objects, such as its ancestors or level in a drill-down path.

When you use both an object identifier and an object name to identify an object in MDL, the object identifier determines whether the object exists. If there is a conflict between the object name and object identifier, the object identifier is used to locate the object, and the name is then changed accordingly.

If the object does not already exist or is not uniquely identified by its object name or object identifier, then you must supply the additional information required to locate it, in the syntax for each verb.

For example, suppose you want to refer to a column that exists in your data source. If your model does not provide a unique object name or object identifier for that column, you must uniquely identify it by specifying the data source in which it exists.

Similarly, if you add a new category to a dimension, in the context of a drill-down path, a level, and a parent category, you must specify all of those objects to locate it precisely and uniquely.

On the other hand, when you refer to an existing category, even if you do not uniquely identify it by its object identifier, you only need to specify the dimension name. This is sufficient because category names are unique within dimensions, and dimension names are unique within models.

The following rules govern the 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 Cognos 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 contextual information to locate it in the model precisely.

#### Uniquely locate existing objects

Use the following table to uniquely locate an existing object not already identified by its object identifier:

<table>
<thead>
<tr>
<th>Object</th>
<th>Objects that must be referenced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currency, Dimension, Measure, Model, Data Source (query), IBM Cognos Source (package or report), PowerCube, PowerCube Group, Signon</td>
<td>None</td>
</tr>
<tr>
<td>Category</td>
<td>Dimension</td>
</tr>
<tr>
<td>Column</td>
<td>Data source</td>
</tr>
<tr>
<td>DimCalcDef</td>
<td>Dimension</td>
</tr>
<tr>
<td>Object</td>
<td>Objects that must be referenced</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Drill Category</td>
<td>Dimension, Root Category</td>
</tr>
<tr>
<td>Level</td>
<td>Dimension, Drill Category</td>
</tr>
<tr>
<td>PowerCube Group Cube</td>
<td>PowerCube Group, Dimension, Category</td>
</tr>
<tr>
<td>Root Category</td>
<td>Dimension</td>
</tr>
<tr>
<td>Subdimension</td>
<td>Dimension, Drill Category, Level, Parent Category</td>
</tr>
<tr>
<td>Special Category</td>
<td>Dimension, Parent Category</td>
</tr>
<tr>
<td>View</td>
<td>Dimension</td>
</tr>
</tbody>
</table>

**Uniquely locate new objects**

Use the following table to uniquely locate a new object:

<table>
<thead>
<tr>
<th>Object</th>
<th>Objects that must be referenced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currency, Dimension, Measure, Model, Data Source (query), IBM Cognos Source (package or report), PowerCube, PowerCube Group, Signon</td>
<td>None</td>
</tr>
<tr>
<td>Category</td>
<td>Dimension, Drill Category, Level, Parent Category</td>
</tr>
<tr>
<td>Column</td>
<td>Data source</td>
</tr>
<tr>
<td>DimCalcDef</td>
<td>Dimension</td>
</tr>
<tr>
<td>Drill Category</td>
<td>Dimension, Root Category</td>
</tr>
<tr>
<td>Level</td>
<td>Dimension, Drill Category, Parent Category</td>
</tr>
<tr>
<td>PowerCube Group Cube</td>
<td>PowerCube Group, Dimension, Category</td>
</tr>
<tr>
<td>Root Category</td>
<td>Dimension</td>
</tr>
<tr>
<td>Subdimension</td>
<td>Dimension, Drill Category, Level, Parent Category</td>
</tr>
<tr>
<td>Special Category</td>
<td>Dimension, Parent Category</td>
</tr>
<tr>
<td>View</td>
<td>Dimension</td>
</tr>
</tbody>
</table>

**Setting 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.

**Specify a preferred category sorting option**

To globally set a sorting option for all categories in the model, specify one of the following ModelCategoryOrderDefault options in the first few header lines of your MDL script:

- OrderYes
- OrderNo
- OrderUsePreference

For more information about each option, see the Cognos Transformer User Guide.
Chapter 4. Creating a complete MDL model

The following examples illustrate how to create complete models using Model Definition Language (MDL). Even if you never attempt this task yourself, by reviewing these examples you will learn the structure and interdependencies of Cognos Transformer MDL models.

Example - MDL model using IBM Cognos data sources in verb MDL

This example creates two data sources to be used in the model. The example shows the data source definitions, not the complete model.

Note: The Add form illustrates the verb that you use to define an object, to differentiate the syntax from that which Cognos Transformer uses when defining objects (the Make form). However, either form is acceptable. Also, you will note that each statement starts with the verb followed by a lengthy series of options.

When you create the data sources, you define the time and products dimensions and the measures used in the model.

Create the data sources in Cognos Transformer

This example is based on an IBM Cognos package with one query and an IBM Cognos report with one query.

```
CognosPackageAdd 103 "GO Sales and Retailers" SourceType Package
SourcePath "/content/package[@name='GOSales and Retailers']" PackageTimeStamp
"/content/package[@name='GOSales and Retailers']/model[@name='model']"

CognosPackageAdd 115 "GSR_rpt_1" SourceType Report SourcePath
"/content/package[@name='GO Sales and Retailers']/report[@name='GSR_rpt_1']" PackageTimeStamp
"/content/package[@name='GOSales and Retailers']/report[@name='GSR_rpt_1']"

DataSourceAdd 105 "GO Sales and Retailers~1" Separator "," SourceType CognosSourceQuery CharacterSetDefault DecimalSep "." Thousandsep "," Columns True Timing PopYes CreateDefaultPackageReportSource 103 "GO Sales and Retailers" AutoSummary True SetCurrent True ServerSource False Speed False Presummarized False

ColumnAdd 107 "[gosales_goretailers].[Orders].[Ordernumber]"
DataSource 105 Origin Source Offset 0 Column "Order number" Storage Float64 Scale 0 Size 4 Decimals 0 InputScale 0 TimeArrayOff Rollup CountAll

ColumnAdd 109 "[gosales_goretailers].[Orders].[Retailername]"
DataSource 105 Origin Source Offset 1 Column "Retailer name" Storage Text Scale 0 Size 102 Decimals 0 Class Description InputScale 0 TimeArray Off

ColumnAdd 111 "[gosales_goretailers].[Orders].[Orderyear]"
DataSource 105 Origin Source Offset 2 Column "Order year" Storage Float64 Scale 0 Size 4 Decimals 0 InputScale 0 TimeArrayOff Rollup CountAll

ColumnAdd 113 "[gosales_goretailers].[Orders].[Ordermonth]"
DataSource 105 Origin Source Offset 3 Column "Order month" Storage Float64 Scale 0 Size 4 Decimals 0 InputScale 0 TimeArrayOff Rollup CountAll

DataSourceAdd 117 "GSR_rpt_1~1" Separator "," SourceType CognosSourceQuery CharacterSet Default DecimalSep "," Thousandsep "," Columns True Timing PopYes CreateDefaultPackageReportSource 115 "GSR_rpt_1" AutoSummary False SetCurrent True ServerSource False Speed False Presummarized False
```
Example - MDL model using an IQD data source in verb MDL

In this example, separate files are created for each part of the model. The files are then merged by means of an MDL script, using the appropriate commands and their mandatory options to create the final model. This example does not include dimension views or custom views.

Note: The Add form illustrates the verb that you use to define an object, to differentiate the syntax from that which Cognos Transformer uses when defining objects (the Make form). However, either form is acceptable. Also, you will note that each statement starts with the verb, followed by a lengthy series of options. For readability, you may wish to insert carriage returns between the keywords.

This is the order in which you would create the separate files for a typical complete model based on an IQD data source, using verb MDL:

• Create the data sources in Cognos Transformer.
• Create the Time dimension.
• Create the Products dimension.
• Create key product measures.
• Create allocations.
• Create the signon.
• Populate the model and create the PowerCube.
• Create the model by combining the scripts.

Step 1: Create the data sources in Cognos Transformer

This example is based on an Impromptu® Query Definition (IQD) data source. As a result, the initial DataSourceAdd and ColumnAdd MDL statements are copied directly from the Impromptu SQL string that is automatically generated when Cognos Transformer connects to the IQD data source.

Que_Col.mdl file

The first file adds information to the model from the relevant columns of the IQD data source.

```
DataSource "Products (CSV)" Separator "," SourceTypeFlatFile_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)"
```
Step 2: Create the time dimension

When you create any dimension using MDL, you must manually create both the root category and drill category for that dimension.

The verbs `RootCatMake` and `DrillCatMake` are used in this example because there is no `Add` verb form for these objects. Also, you must set the inclusion property to `Suppress` for the drill category so that it does not display in the reporting components.

Before you begin

If you do not manually create the two default dimension views required for each dimension (All Categories and Omit Dimension), you must use the `ModelEnsureCompleteness` script to have Cognos Transformer create them automatically.
The definition of each level contains the specification for its dimension, drill
category, and parent. Together, these define the hierarchical structure for the
dimension.

When you create a query in Cognos Transformer based on a Framework Manager
package that contains hierarchical time-related categories, Cognos Transformer
interprets the time-related categories as a regular dimension and not as a time
dimension. As a result, the time dimension in your PowerCube will not contain
any relative time categories. Ensure that you import all of the data needed to
define your time dimension, and then create the date levels and categories.

**Dim_Years.mdl file**
The second file creates a standard Year-Quarter-Month time dimension for the
model.

```
DimAdd "Years" DimType Date
RootCatMake "Time" Dimension "Years" Root "Time" Inclusion
Suppress Level Add "Year" Dimension "Years" Drill "By Time" Parent
"Source "Time" Date Function Year Unique Categories True OrderBy
Drill "By Time" Column "Time" SortOrder Default SortAs Ascending
LevelAdd "Quarter" Dimension "Years" Drill "By Time" Parent
"Year" Source "Time" Date Function Quarter CatLabFormat 'YYYY
"Q" UniqueCategories True OrderBy Drill "By Time" Column "Time" SortOrder
Default SortAs Ascending
```

**Step 3: Create the products dimension**
The next step is to specify the details necessary to properly structure the primary
dimension, Products.

**Dim_Products.mdl**
The third file pulls information needed for the Products dimension from the
imported data.

```
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 key product measures

Next, you need separate script files to add the three measures associated with the Products dimension.

(Me_Prodcost.mdl)
This file creates the Product Cost measure, which is taken from the Cost column.
MeasureAdd "Product Cost"
Associations "Cost" AssociationType Type_Query
AssociationRole Role_Source AssociationReferenced "Cost"

(Me_Prodplan.mdl)
This file creates the Product Plan measure, which is taken from the Planned Sales column.
MeasureAdd "Product Plan"
Associations "Planned Sales" AssociationType Type_Query
AssociationRole Role_Source AssociationReferenced "Planned Sales"

(Me_Revenue.mdl)
This script creates the Revenue measure, which is taken from the Revenue column.
MeasureAdd "Revenue"
Associations "Revenue" AssociationType Type_Query
AssociationRole Role_Source AssociationReferenced "Revenue"

Step 5: Create allocations

After you add the required measures, you may want to specify how they will be allocated across the various dimensions in your model. Only do this if you do not use the ModelEnsureCompleteness script to generate the default measure allocations.

(Alloc_Me.mdl)
This script allocates Product Plan summary data to both the Product Line and Year levels of their respective dimensions proportionate to the product Revenue distribution.
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 6: Create the signon

These scripts create IBM Cognos signons and data source signons.

(CognosSignon.mdl)
This sample script creates a Cognos signon object and disables automatic signon access.
SignonAdd "name" UserId "user name" password "password"
AutoLogon False SignonNamespace "namespace id" SignonType "Cognos"

**DataSourceSignon.mdl**
This sample script creates a data source signon object and disables interactive prompting for a password.

SignonAdd "signon name" PromptForPassword True
UserId "user name"
password <password> SignonType "DataSource"

Data sources defined in IBM Cognos BI can have multiple connections, each with multiple signons. You can use MDL to create, delete, and update connection and signon information for ambiguous connections and signons.

You may want to use a script to create or later update the object required to provide automatic signon access.

**Step 7: Populate the model and create the PowerCube**
This is a three-stage process. You should create separate script files for each process.

**Populate.mdl**
This script generates categories for the model.

PopulateModel

**Cube.mdl**
This script adds the PowerCube object to the model, specifying where the .mdc file is located.

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

**CreateCube.mdl**
This script generates the PowerCube in the form of an .mdc file.
CreateFiles

**Step 8: Create the model by combining the scripts**
As a final step, prepare a script that, when run, combines all the previous scripts to create a complete Cognos Transformer model and generates an .mdc file based on that model.

**NewModel.mdl**
This script runs all the previously created scripts in sequence. Ensure model completeness before you generate your final .mdl file based on the model.

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:\Alloc_Me.mdl"
OpenMDL "C:\CognosSignon.mdl"
OpenMDL "C:\DataSourceSignon.mdl"
Check the model generated by Cognos Transformer

When you check the FinalModel.mdl file that results from running your NewModel.mdl script, you will note that the model definition is much longer than it was when you started. This is because it contains all the default options that Cognos Transformer automatically adds to each object definition. In addition,

- The verb PopulateModel adds categories, all of which are defined
- The verb ModelEnsureCompleteness adds the two default views

Line numbers are included to make the file easier to read. The cogtr.xml setting ObjectIdOutput=0 is used, hiding the object identifiers by default, so each object is identified by its name alone.

FinalModel.mdl

Below is a line-by-line rendering of the sample mdl file.

<table>
<thead>
<tr>
<th>No.</th>
<th>MDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>DataSourceMake &quot;Products (CSV)&quot; Separator &quot;,&quot; SourceType FlatFile_ColNames CharacterSet Ansi DecimalSep &quot;,&quot; Thousandsep &quot;,,&quot; Columns True Timing PopYesCreateDefault Source &quot;c:\prodinfo.csv&quot; EnableMultiProcess False SetCurrent True ServerSource False Speed False Presummarized False</td>
</tr>
<tr>
<td>3</td>
<td>ColumnMake &quot;Product Line&quot; DataSource &quot;Products (CSV)&quot; Origin Generated Offset 0 Column &quot;Product Line&quot; Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off</td>
</tr>
<tr>
<td>4</td>
<td>ColumnMake &quot;Product Type&quot; DataSource &quot;Products (CSV)&quot; Origin Generated Offset 1 Column &quot;Product Type&quot; Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off</td>
</tr>
<tr>
<td>5</td>
<td>ColumnMake &quot;Product Id&quot; DataSource &quot;Products (CSV)&quot; Origin Generated Offset 2 Column &quot;Product Id&quot; Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off</td>
</tr>
<tr>
<td>6</td>
<td>ColumnMake &quot;Product Name&quot; DataSource &quot;Products (CSV)&quot; Origin Generated Offset 3 Column &quot;Product Name&quot; Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off</td>
</tr>
<tr>
<td>7</td>
<td>DataSourceMake &quot;Prod Line Plan&quot; Separator &quot;,&quot; SourceType FlatFile_ColNames CharacterSet Ansi DecimalSep &quot;,&quot; Thousandsep &quot;,,&quot; Columns True Timing PopYesCreateDefault Source &quot;c:\ProdPlan.asc&quot; EnableMultiProcess False SetCurrent True ServerSource False Speed False Presummarized False</td>
</tr>
<tr>
<td>8</td>
<td>ColumnMake &quot;YEAR&quot; DataSource &quot;Prod Line Plan&quot; Origin Generated Offset 0 Column &quot;Time&quot; Storage Default Format Y DateLevel Year Scale 0 Size 1 Decimals 0 Class Date InputScale 0 TimeArray Off</td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>9</td>
<td>ColumnMake &quot;PROD_LINE&quot; DataSource &quot;Prod Line Plan&quot; Origin Generated</td>
</tr>
<tr>
<td></td>
<td>Offset 1 Column &quot;Product Line&quot; Storage Default Scale 0 Size 1 Decimals</td>
</tr>
<tr>
<td></td>
<td>0 InputScale 0 TimeArray Off</td>
</tr>
<tr>
<td>10</td>
<td>ColumnMake &quot;FORECAST&quot; DataSource &quot;Prod Line Plan&quot; Origin Generated</td>
</tr>
<tr>
<td></td>
<td>Offset 2 Column &quot;Planned sales&quot; Storage Default Scale 0 Size 1 Decimals</td>
</tr>
<tr>
<td></td>
<td>0 InputScale 0 TimeArray Off</td>
</tr>
<tr>
<td>11</td>
<td>DataSourceMake &quot;MAIN (IQD)&quot; Separator &quot;,&quot;SourceType DataSource</td>
</tr>
<tr>
<td></td>
<td>CharSet Ansi DecimalSep &quot;,&quot; Thousandsep &quot;,&quot; Columns TrueTiming PopYesCreateDefault</td>
</tr>
</tbody>
</table>
|     | Source "c:<install_directory>\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" * T2."PRICE" as c6,
|     |   (T2."QTY" * T3."PROD_COST") as c7,
|     |   (T2."QTY" * T2."PRICE") - (T2."QTY" * T3."PROD_COST")
|     |   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')
<p>|     |   EndListImrName &quot;C:&lt;install_directory&gt;\bsc_msrs.imr&quot; Stamp 996601938 EnableMultiProcess False SetCurrent TrueServerSource False Speed False Presummarized False |
| 12  | ColumnMake &quot;Order Dt&quot; DataSource &quot;MAIN (IQD)&quot; Origin Source Offset 0 |
|     | Column &quot;Time&quot; Storage Int32 Scale 0 Size 4 Decimals 0 Class Date     |
|     | InputScale 0 TimeArray Off                                          |
| 13  | ColumnMake &quot;od-Prod No&quot; DataSource &quot;MAIN (IQD)&quot; Origin Source Offset 1|
|     | Column &quot;Product Id&quot; Storage Float64 Scale 0 Size 5 Decimals 0 Class |
|     | Quantity InputScale 0 TimeArray Off                                 |
| 14  | ColumnMake &quot;Revenue&quot; DataSource &quot;MAIN (IQD)&quot; Origin Generated Offset 5|
|     | Column &quot;Revenue&quot; Storage Default Scale 0 Size 1 Decimals 0 InputScale|
|     | 0 TimeArray Off                                                     |
| 15  | ColumnMake &quot;Cost&quot; DataSource &quot;MAIN (IQD)&quot; Origin Source Offset 6     |
|     | Column &quot;Cost&quot; Storage Float64 Scale 0 Size 6 Decimals 0 Class       |
|     | Quantity InputScale 0 TimeArray Off                                 |
| 16  | DimMake &quot;Years&quot; DimType NonStandard EarliestDate 19000101 LatestDate |
|     | 99991231 ManualPeriods False DaysInWeek 127 NewCatsLock False       |
|     | ExcludeAutoPartitioning False                                      |</p>
<table>
<thead>
<tr>
<th>No.</th>
<th>MDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>RootCatMake &quot;Time&quot; Dimension &quot;Years&quot; Inclusion Generate Lastuse 20060802 Date 0 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>18</td>
<td>DrillCatMake &quot;By Time&quot; Dimension &quot;Years&quot; Root &quot;Time&quot; Inclusion Suppress Filtered False Suppressed True PrimaryDrill True YearBegins 20060101 PartialWeek Split ExtraWeek None WeekBegins Sunday</td>
</tr>
<tr>
<td>19</td>
<td>LevelMake &quot;Year&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Parent &quot;&quot; Blanks &quot;( blank )&quot; DateFunction Year Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False CatLabFormat &quot;YYYY&quot; Timerank 10 UniqueCategories True UniqueMove False Associations &quot;Time&quot; AssociationContext &quot;By Time&quot; AssociationType Type_Query AssociationRole Role_OrderBy AssociationReferenced &quot;Time&quot; SortOrder Default SortAs Ascending Associations &quot;Time&quot; AssociationType Type_Query AssociationRole Role_Source AssociationReferenced &quot;Time&quot;</td>
</tr>
<tr>
<td>20</td>
<td>LevelMake &quot;Quarter&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Parent &quot;Year&quot; Blanks &quot;( blank )&quot; DateFunction Quarter Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False CatLabFormat &quot;YYYY Q Q'&quot; Timerank 20 UniqueCategories True UniqueMove False Associations &quot;Time&quot; AssociationContext &quot;By Time&quot; AssociationType Type_Query AssociationRole Role_OrderBy AssociationReferenced &quot;Time&quot; SortOrder Default SortAs Ascending Associations &quot;Time&quot; AssociationType Type_Query AssociationRole Role_Source AssociationReferenced &quot;Time&quot;</td>
</tr>
<tr>
<td>21</td>
<td>LevelMake &quot;Month&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Parent &quot;Quarter&quot; Blanks &quot;( blank )&quot; DateFunction Month Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False CatLabFormat &quot;YYYY/MMM&quot; Timerank 30 UniqueCategories True UniqueMove False Associations &quot;Time&quot; AssociationContext &quot;By Time&quot; AssociationType Type_Query AssociationRole Role_OrderBy AssociationReferenced &quot;Time&quot; SortOrder Default SortAs Ascending Associations &quot;Time&quot; AssociationType Type_Query AssociationRole Role_Source AssociationReferenced &quot;Time&quot;</td>
</tr>
<tr>
<td>22</td>
<td>CatMake &quot;20010101-20011231&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Year&quot; Parent &quot;&quot; OrderBy Drill &quot;By Time&quot; Value &quot;2001&quot; Label &quot;2001&quot; Lastuse 20060802 SourceValue &quot;2001&quot; Date 20010101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>23</td>
<td>CatMake &quot;20010101-20010331&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Quarter&quot; Parent &quot;20010101-20011231&quot; OrderBy Drill &quot;By Time&quot; Value &quot;20010101&quot; Label &quot;2001 Q 1&quot; Lastuse 20060802 SourceValue &quot;20010101&quot; Date 20010101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>24</td>
<td>CatMake &quot;20010101-20010331&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20010101-20010331&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200101&quot; Label &quot;2001 Jan&quot; Lastuse 20060802 SourceValue &quot;200101&quot; Date 20010101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>25</td>
<td>CatMake &quot;20010201-20010229&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20010101-20010331&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200102&quot; Label &quot;2001/Feb&quot; Lastuse 20060802 SourceValue &quot;200102&quot; Date 20010201 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>26</td>
<td>CatMake &quot;20010301-20010331&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20010101-20010331&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200103&quot; Label &quot;2001/Mar&quot; Lastuse 20060802 SourceValue &quot;200103&quot; Date 20010301 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>27</td>
<td>CatMake &quot;20010401-20010630&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Quarter&quot; Parent &quot;20010101-20011231&quot; OrderBy Drill &quot;By Time&quot; Value &quot;20010401&quot; Label &quot;2001 Q 2&quot; Lastuse 20060802 SourceValue &quot;20010401&quot; Date 20010401 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>28</td>
<td>CatMake &quot;20010401-20010430&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20010401-20010630&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200104&quot; Label &quot;2001/Apr&quot; Lastuse 20060802 SourceValue &quot;200104&quot; Date 20010401 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>29</td>
<td>CatMake &quot;20010501-20010531&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20010401-20010630&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200105&quot; Label &quot;2001/May&quot; Lastuse 20060802 SourceValue &quot;200105&quot; Date 20010501 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>30</td>
<td>CatMake &quot;20010601-20010630&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20010401-20010630&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200106&quot; Label &quot;2001/Jun&quot; Lastuse 20060802 SourceValue &quot;200106&quot; Date 20010601 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>31</td>
<td>CatMake &quot;20010701-20010930&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Quarter&quot; Parent &quot;20010101-20011231&quot; OrderBy Drill &quot;By Time&quot; Value &quot;20010701&quot; Label &quot;2001 Q 3&quot; Lastuse 20060802 SourceValue &quot;20010701&quot; Date 20010701 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>32</td>
<td>CatMake &quot;20010701-20010731&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20010701-20010930&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200107&quot; Label &quot;2001/Jul&quot; Lastuse 20060802 SourceValue &quot;200107&quot; Date 20010701 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>33</td>
<td>CatMake &quot;20010801-20010831&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20010701-20010930&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200108&quot; Label &quot;2001/Aug&quot; Lastuse 20060802 SourceValue &quot;200108&quot; Date 20010801 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>34</td>
<td>CatMake &quot;20010901-20010930&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels</td>
</tr>
<tr>
<td></td>
<td>&quot;Month&quot; Parent &quot;20010701-20010930&quot; OrderBy Drill &quot;By Time&quot; Value</td>
</tr>
<tr>
<td></td>
<td>&quot;200109&quot; Label &quot;2001/Sep&quot; Lastuse 20060802 SourceValue &quot;200109&quot; Date</td>
</tr>
<tr>
<td></td>
<td>20010901 Filtered False Suppressed False Sign False IsKeyOrphanage</td>
</tr>
<tr>
<td></td>
<td>False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>35</td>
<td>CatMake &quot;20011001-20011231&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels</td>
</tr>
<tr>
<td></td>
<td>&quot;Quarter&quot; Parent &quot;20010101-20011231&quot; OrderBy Drill &quot;By Time&quot; Value</td>
</tr>
<tr>
<td></td>
<td>&quot;200110&quot; Label &quot;2001 Q 4&quot; Lastuse 20060802 SourceValue &quot;20011001&quot; Date</td>
</tr>
<tr>
<td></td>
<td>20011001 Filtered False Suppressed False Sign False IsKeyOrphanage</td>
</tr>
<tr>
<td></td>
<td>False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>36</td>
<td>CatMake &quot;20011101-20011130&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels</td>
</tr>
<tr>
<td></td>
<td>&quot;Month&quot; Parent &quot;20011001-20011130&quot; OrderBy Drill &quot;By Time&quot; Value</td>
</tr>
<tr>
<td></td>
<td>&quot;200111&quot; Label &quot;2001/Nov&quot; Lastuse 20060802 SourceValue &quot;200111&quot; Date</td>
</tr>
<tr>
<td></td>
<td>20011101 Filtered False Suppressed False Sign False IsKeyOrphanage</td>
</tr>
<tr>
<td></td>
<td>False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>37</td>
<td>CatMake &quot;20011201-20011231&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels</td>
</tr>
<tr>
<td></td>
<td>&quot;Month&quot; Parent &quot;20011101-20011231&quot; OrderBy Drill &quot;By Time&quot; Value</td>
</tr>
<tr>
<td></td>
<td>&quot;200112&quot; Label &quot;2001/Dec&quot; Lastuse 20060802 SourceValue &quot;200112&quot; Date</td>
</tr>
<tr>
<td></td>
<td>20011201 Filtered False Suppressed False Sign False IsKeyOrphanage</td>
</tr>
<tr>
<td></td>
<td>False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>38</td>
<td>CatMake &quot;20020101-20020131&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels</td>
</tr>
<tr>
<td></td>
<td>&quot;Month&quot; Parent &quot;20020101-20020131&quot; OrderBy Drill &quot;By Time&quot; Value</td>
</tr>
<tr>
<td></td>
<td>&quot;200201&quot; Label &quot;2002/Jan&quot; Lastuse 20060802 SourceValue &quot;200201&quot; Date</td>
</tr>
<tr>
<td></td>
<td>20020101 Filtered False Suppressed False Sign False IsKeyOrphanage</td>
</tr>
<tr>
<td></td>
<td>False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>39</td>
<td>CatMake &quot;20020201-20020228&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels</td>
</tr>
<tr>
<td></td>
<td>&quot;Month&quot; Parent &quot;20020201-20020228&quot; OrderBy Drill &quot;By Time&quot; Value</td>
</tr>
<tr>
<td></td>
<td>&quot;200202&quot; Label &quot;2002/Feb&quot; Lastuse 20060802 SourceValue &quot;200202&quot; Date</td>
</tr>
<tr>
<td></td>
<td>20020201 Filtered False Suppressed False Sign False IsKeyOrphanage</td>
</tr>
<tr>
<td></td>
<td>False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>43</td>
<td><code>CatMake &quot;20020301-20020331&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20020101-20020331&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200203&quot; Label &quot;2002/Mar&quot; Lastuse 20060802 SourceValue &quot;200203&quot; Date 20020301 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</code></td>
</tr>
<tr>
<td>44</td>
<td><code>CatMake &quot;20020401-20020430&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Quarter&quot; Parent &quot;20020101-20021231&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200204&quot; Label &quot;2002 Q 2&quot; Lastuse 20060802 SourceValue &quot;200204&quot; Date 20020401 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</code></td>
</tr>
<tr>
<td>45</td>
<td><code>CatMake &quot;20020501-20020531&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20020401-20020630&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200205&quot; Label &quot;2002/May&quot; Lastuse 20060802 SourceValue &quot;200205&quot; Date 20020501 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</code></td>
</tr>
<tr>
<td>46</td>
<td><code>CatMake &quot;20020601-20020630&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20020501-20020630&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200206&quot; Label &quot;2002/June&quot; Lastuse 20060802 SourceValue &quot;200206&quot; Date 20020601 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</code></td>
</tr>
<tr>
<td>47</td>
<td><code>CatMake &quot;20020701-20020731&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20020601-20020630&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200207&quot; Label &quot;2002/July&quot; Lastuse 20060802 SourceValue &quot;200207&quot; Date 20020701 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</code></td>
</tr>
<tr>
<td>48</td>
<td><code>CatMake &quot;20020801-20020831&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20020701-20020731&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200208&quot; Label &quot;2002/Aug&quot; Lastuse 20060802 SourceValue &quot;200208&quot; Date 20020801 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</code></td>
</tr>
<tr>
<td>49</td>
<td><code>CatMake &quot;20020901-20020930&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20020801-20020831&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200209&quot; Label &quot;2002/Sept&quot; Lastuse 20060802 SourceValue &quot;200209&quot; Date 20020901 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</code></td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>52</td>
<td>CatMake &quot;20021001-20021231&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Quarter&quot; Parent &quot;20020101-20021231&quot; OrderBy Drill &quot;By Time&quot; Value &quot;20021001&quot; Label &quot;2002 Q 4&quot; Lastuse 20060802 SourceValue &quot;20021001&quot; Date 20021001 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>53</td>
<td>CatMake &quot;20021001-20021031&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20021001-20021231&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200210&quot; Label &quot;2002/Oct&quot; Lastuse 20060802 SourceValue &quot;200210&quot; Date 20021001 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>54</td>
<td>CatMake &quot;20021101-20021130&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20021001-20021231&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200211&quot; Label &quot;2002/Nov&quot; Lastuse 20060802 SourceValue &quot;200211&quot; Date 20021101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>55</td>
<td>CatMake &quot;20021201-20021231&quot; Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Month&quot; Parent &quot;20021001-20021231&quot; OrderBy Drill &quot;By Time&quot; Value &quot;200212&quot; Label &quot;2002/Dec&quot; Lastuse 20060802 SourceValue &quot;200212&quot; Date 20021201 Current Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>56</td>
<td>ViewMake &quot;All Categories&quot; Dimension &quot;Years&quot; Type All</td>
</tr>
<tr>
<td>57</td>
<td>ViewMake &quot;Omit Dimension&quot; Dimension &quot;Years&quot; Type Omit</td>
</tr>
<tr>
<td>58</td>
<td>DimMake &quot;Products&quot; DimType Regular NewCatsLock False ExcludeAutoPartitioning False RootCatMake &quot;Product Line&quot; Dimension &quot;Products&quot; Inclusion Generate Lastuse 20060802 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>59</td>
<td>DrillCatMake &quot;By Product Line&quot; Dimension &quot;Products&quot; Root &quot;Product Line&quot; Inclusion Suppress Filtered False Suppressed True PrimaryDrill True</td>
</tr>
<tr>
<td>60</td>
<td>LevelMake &quot;Product Line&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Parent &quot;&quot; Blanks &quot;( blank )&quot; DateFunction None Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False Timerank 0 UniqueCategories False UniqueMove False Associations &quot;Product Line&quot; AssociationType Type_Query AssociationRole Role_Source AssociationReferenced &quot;Product Line&quot;</td>
</tr>
<tr>
<td>61</td>
<td>LevelMake &quot;Product Type&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Parent &quot;Product Line&quot; Blanks &quot;( blank )&quot; DateFunction None Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False Timerank 0 UniqueCategories False UniqueMove False Associations &quot;Product Type&quot; AssociationType Type_Query AssociationRole Role_Source AssociationReferenced &quot;Product Type&quot;</td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>62</td>
<td>LevelMake &quot;Product Id&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Parent &quot;Product Type&quot; Blanks &quot;( blank )&quot; DateFunction None Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False Timerank 0 UniqueCategories True UniqueMove False Associations &quot;Product Id&quot; AssociationType Type_Query AssociationRole Role_Source AssociationReferenced &quot;Product Id&quot; Associations &quot;Product Name&quot; AssociationType Type_Query AssociationRole Role_Label AssociationReferenced &quot;Product Name&quot;</td>
</tr>
<tr>
<td>63</td>
<td>CatMake &quot;Environmental Line&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Line&quot; Parent &quot;By Product Line&quot; Lastuse 20060802 SourceValue &quot;Environmental Line&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>64</td>
<td>CatMake &quot;Alert Devices&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;Environmental Line&quot; Lastuse 20060802 SourceValue &quot;Alert Devices&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>65</td>
<td>CatMake &quot;60100&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Alert Devices&quot; Label &quot;Pocket U.V. Alerter&quot; Lastuse 20060802 SourceValue &quot;60100&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>66</td>
<td>CatMake &quot;60101&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Alert Devices&quot; Label &quot;Microwave Detector&quot; Lastuse 20060802 SourceValue &quot;60101&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>67</td>
<td>CatMake &quot;60102&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Alert Devices&quot; Label &quot;Pocket Radon Alerter&quot; Lastuse 20060802 SourceValue &quot;60102&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>68</td>
<td>CatMake &quot;Bio-Friendly Soaps&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;Environmental Line&quot; Lastuse 20060802 SourceValue &quot;Bio-Friendly Soaps&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>69</td>
<td>CatMake &quot;60300&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Bio-Friendly Soaps&quot; Label &quot;RiverKind Shampoo&quot; Lastuse 20060802 SourceValue &quot;60300&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>70</td>
<td>CatMake &quot;60301&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Bio-Friendly Soaps&quot; Label &quot;RiverKind Soap&quot; Lastuse 20060802 SourceValue &quot;60301&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>71</td>
<td>CatMake &quot;60302&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Bio-Friendly Soaps&quot; Label &quot;RiverKind Detergent&quot; Lastuse 20060802 SourceValue &quot;60302&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>72</td>
<td>CatMake &quot;Recycled Products&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;Environmental Line&quot; Lastuse 20060802 SourceValue &quot;Recycled Products&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>73</td>
<td>CatMake &quot;60201&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Recycled Products&quot; Label &quot;Enviro-Kit&quot; Lastuse 20060802 SourceValue &quot;60201&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>74</td>
<td>CatMake &quot;60200&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Recycled Products&quot; Label &quot;EnviroSak&quot; Lastuse 20060802 SourceValue &quot;60200&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>75</td>
<td>CatMake &quot;60202&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Recycled Products&quot; Label &quot;Enviro-T&quot; Lastuse 20060802 SourceValue &quot;60202&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>76</td>
<td>CatMake &quot;Sunblock&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;Environmental Line&quot; Lastuse 20060802 SourceValue &quot;Sunblock&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>77</td>
<td>CatMake &quot;60400&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Sunblock&quot; Label &quot;Sun Shelter-8&quot; Lastuse 20060802 SourceValue &quot;60400&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>78</td>
<td>CatMake &quot;60401&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Sunblock&quot; Label &quot;Sun Shelter-15&quot; Lastuse 20060802 SourceValue &quot;60401&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>79</td>
<td>CatMake &quot;60402&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Sunblock&quot; Label &quot;Sun Shelter-30&quot; Lastuse 20060802 SourceValue &quot;60402&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>80</td>
<td>CatMake &quot;Water Purifiers&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;Environmental Line&quot; Lastuse 20060802 SourceValue &quot;Water Purifiers&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>81</td>
<td>CatMake &quot;60500&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Water Purifiers&quot; Label &quot;Pro-Lite Water Filter&quot; Lastuse 20060802 SourceValue &quot;60500&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>82</td>
<td>CatMake &quot;60501&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Water Purifiers&quot; Label &quot;Pocket Water Filter&quot; Lastuse 20060802 SourceValue &quot;60501&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>83</td>
<td>CatMake &quot;GO Sport Line&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Line&quot; Parent &quot;By Product Line&quot; Lastuse 20060802 SourceValue &quot;GO Sport Line&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>84</td>
<td>CatMake &quot;Carry-Bags&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;GO Sport Line&quot; Lastuse 20060802 SourceValue &quot;Carry-Bags&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>85</td>
<td>CatMake &quot;50100&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Carry-Bags&quot; Label &quot;GO Sport Bag&quot; Lastuse 20060802 SourceValue &quot;50100&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>86</td>
<td>CatMake &quot;50101&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Carry-Bags&quot; Label &quot;GO Ski Gear Bag&quot; Lastuse 20060802 SourceValue &quot;50101&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>87</td>
<td>CatMake &quot;50102&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Carry-Bags&quot; Label &quot;GO Duffel Bag&quot; Lastuse 20060802 SourceValue &quot;50102&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>88</td>
<td>CatMake &quot;Sport Wear&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;GO Sport Line&quot; Lastuse 20060802 SourceValue &quot;Sport Wear&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>89</td>
<td>CatMake &quot;50201&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Sport Wear&quot; Label &quot;GO Headband&quot; Lastuse 20060802 SourceValue &quot;50201&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>90</td>
<td>CatMake &quot;50201&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Sport Wear&quot; Label &quot;GO Headband&quot; Lastuse 20060802 SourceValue &quot;50201&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>91</td>
<td>CatMake &quot;50202&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Sport Wear&quot; Label &quot;GO Wristband&quot; Lastuse 20060802 SourceValue &quot;50202&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>92</td>
<td>CatMake &quot;50203&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Sport Wear&quot; Label &quot;GO Water Bottle&quot; Lastuse 20060802 SourceValue &quot;50203&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>93</td>
<td>CatMake &quot;Outdoor Products&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Line&quot; Parent &quot;By Product Line&quot; Lastuse 20060802 SourceValue &quot;Outdoor Products&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>94</td>
<td>CatMake &quot;Back Packs&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;Outdoor Products&quot; Lastuse 20060802 SourceValue &quot;Back Packs&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>95</td>
<td>CatMake &quot;40300&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Back Packs&quot; Label &quot;Day Tripper&quot; Lastuse 20060802 SourceValue &quot;40300&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>96</td>
<td>CatMake &quot;40301&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Back Packs&quot; Label &quot;Pack n' Hike&quot; Lastuse 20060802 SourceValue &quot;40301&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>97</td>
<td>CatMake &quot;40302&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Back Packs&quot; Label &quot;GO Small Waist Pack&quot; Lastuse 20060802 SourceValue &quot;40302&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>98</td>
<td>CatMake &quot;40303&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Back Packs&quot; Label &quot;GO Large Waist Pack&quot; Lastuse 20060802 SourceValue &quot;40303&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>99</td>
<td>CatMake &quot;Cooking Equipment&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;Outdoor Products&quot; Lastuse 20060802 SourceValue &quot;Cooking Equipment&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>100</td>
<td>CatMake &quot;40400&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Cooking Equipment&quot; Label &quot;Dover-1&quot; Lastuse 20060802 SourceValue &quot;40400&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>101</td>
<td>CatMake &quot;40401&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Cooking Equipment&quot; Label &quot;Dover-2&quot; Lastuse 20060802 SourceValue &quot;40401&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>102</td>
<td>CatMake &quot;40402&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Cooking Equipment&quot; Label &quot;GO Cookset&quot; Lastuse 20060802 SourceValue &quot;40402&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>103</td>
<td>CatMake &quot;40403&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Cooking Equipment&quot; Label &quot;GO Camp Kettle&quot; Lastuse 20060802 SourceValue &quot;40403&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>104</td>
<td>CatMake &quot;Sleeping Bags&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;Outdoor Products&quot; Lastuse 20060802 SourceValue &quot;Sleeping Bags&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>105</td>
<td>CatMake &quot;40200&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Sleeping Bags&quot; Label &quot;MoonBeam&quot; Lastuse 20060802 SourceValue &quot;40200&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>106</td>
<td>CatMake &quot;40201&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Sleeping Bags&quot; Label &quot;MoonGlow&quot; Lastuse 20060802 SourceValue &quot;40201&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>107</td>
<td>CatMake &quot;40202&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Sleeping Bags&quot; Label &quot;MoonLite&quot; Lastuse 20060802 SourceValue &quot;40202&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>108</td>
<td>CatMake &quot;Tents&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Type&quot; Parent &quot;Outdoor Products&quot; Lastuse 20060802 SourceValue &quot;Tents&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>109</td>
<td>CatMake &quot;40100&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Tents&quot; Label &quot;Star Lite&quot; Lastuse 20060802 SourceValue &quot;40100&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>110</td>
<td>CatMake &quot;40101&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Tents&quot; Label &quot;Star Gazer-2&quot; Lastuse 20060802 SourceValue &quot;40101&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>111</td>
<td>CatMake &quot;40102&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Tents&quot; Label &quot;Star Gazer-3&quot; Lastuse 20060802 SourceValue &quot;40102&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>112</td>
<td>CatMake &quot;40103&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Tents&quot; Label &quot;StarDome&quot; Lastuse 20060802 SourceValue &quot;40103&quot; Current Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>113</td>
<td>CatMake &quot;40104&quot; Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels &quot;Product Id&quot; Parent &quot;Tents&quot; Label &quot;Micro Lite&quot; Lastuse 20060802 SourceValue &quot;40104&quot; Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>114</td>
<td>ViewMake &quot;All Categories&quot; Dimension &quot;Products&quot; Type All</td>
</tr>
<tr>
<td>115</td>
<td>ViewMake &quot;Omit Dimension&quot; Dimension &quot;Products&quot; Type Omit</td>
</tr>
<tr>
<td>116</td>
<td>MeasureMake &quot;Revenue&quot; Storage Default OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False DrillThrough False EndList Associations &quot;Revenue&quot; AssociationType Type_Query AssociationRole Role_Source AssociationReferenced &quot;Revenue&quot;</td>
</tr>
<tr>
<td>No.</td>
<td>MDL</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>117</td>
<td>MeasureMake &quot;Product Cost&quot; Storage Default OutPutScale 0 Decimals 0</td>
</tr>
<tr>
<td></td>
<td>ReverseSign False IsCurrency False DrillThrough False EndList</td>
</tr>
<tr>
<td></td>
<td>Associations &quot;Cost&quot; AssociationType Type_Query AssociationRole</td>
</tr>
<tr>
<td></td>
<td>Role_Source AssociationReferenced &quot;Cost&quot;</td>
</tr>
<tr>
<td>118</td>
<td>MeasureMake &quot;Product Plan&quot; Storage Default OutPutScale 0 Decimals 0</td>
</tr>
<tr>
<td></td>
<td>ReverseSign False IsCurrency False DrillThrough False EndList</td>
</tr>
<tr>
<td></td>
<td>Associations &quot;Planned Sales&quot; AssociationType Type_Query AssociationRole</td>
</tr>
<tr>
<td></td>
<td>Role_Source AssociationReferenced &quot;Planned sales&quot;</td>
</tr>
<tr>
<td>119</td>
<td>SignonMake &quot;I70_DBASE_NTV_PP70_SAMPLE&quot;FullDb&quot;</td>
</tr>
<tr>
<td></td>
<td>7C3155E282B3D0F6E647A376504802295E3F3A5FACBD1616FB770A97C</td>
</tr>
<tr>
<td></td>
<td>5301565D5456&quot;09010890440FBCC047566E5207968B87633C7BGDA56D9539D5</td>
</tr>
<tr>
<td></td>
<td>FB470BA1A5D4E9E790A0&quot;A00C1C1111CF5188A0F9DAD97B1B53E3D1CA8A2</td>
</tr>
<tr>
<td></td>
<td>396F22365C624CCE534946A5A5&quot;23FFBFAA45A444A4E17F1851937C7E8CB</td>
</tr>
<tr>
<td></td>
<td>B1B2A8B0A8F3A3053D4D728BAA51F49B0&quot;C4CCB26314FF28DB956ED4B73</td>
</tr>
<tr>
<td></td>
<td>575038AF60B65F0713691B4FC64ED26372563E2042978&quot;2A618F81AF7358720</td>
</tr>
<tr>
<td></td>
<td>B330F265CAD82A19D220E7B660EC690A4EF31B6AFEE7A3742A8&quot;702C51C</td>
</tr>
<tr>
<td></td>
<td>693FE7A77F30DE73E4B8C569D51E254781B0C63AA64ED4CD5F94AC450E79526</td>
</tr>
<tr>
<td></td>
<td>&quot;1C3507B3F0ECC&quot;DbType &quot;DB&quot; DatabaseSubType</td>
</tr>
<tr>
<td></td>
<td>BDEPromptForPassword False Manual False</td>
</tr>
<tr>
<td></td>
<td>SignonMake &quot;I50_DBASE_NTV_PP60_SAMPLE&quot;</td>
</tr>
<tr>
<td></td>
<td>PromptForPassword FalseManual False</td>
</tr>
<tr>
<td>120</td>
<td>CubeMake &quot;Great Outdoors Sales&quot; MdcFile &quot;c:\Outdoors.mdc&quot; Status OK</td>
</tr>
<tr>
<td></td>
<td>CubeCreation On Optimize Categories Compress False DatabaseInfo</td>
</tr>
<tr>
<td></td>
<td>&quot;Local;&quot; IncrementalUpdate False ServerCube False CubeStamp 996769127</td>
</tr>
<tr>
<td></td>
<td>CubeCycle 2 BlockParentTotals False Caching False UseAlternateFileName</td>
</tr>
<tr>
<td></td>
<td>False DrillThrough False EndList DimensionView &quot;Years&quot; &quot;All Categories&quot;</td>
</tr>
<tr>
<td></td>
<td>DimensionView &quot;Products&quot; &quot;All Categories&quot; MeasureInclude &quot;Revenue&quot;</td>
</tr>
<tr>
<td></td>
<td>Yes MeasureInclude &quot;Product Cost&quot; Yes MeasureInclude &quot;Product Plan&quot;</td>
</tr>
<tr>
<td></td>
<td>Yes Measure Include &quot;Revenue&quot; Yes Measure Include &quot;Product Cost&quot; Yes</td>
</tr>
<tr>
<td></td>
<td>Measure Include &quot;Product Plan&quot; Yes</td>
</tr>
<tr>
<td>121</td>
<td>AllocationAdd Measure &quot;Revenue&quot; Type Default</td>
</tr>
<tr>
<td>122</td>
<td>AllocationAdd Measure &quot;Product Cost&quot; Type Default</td>
</tr>
<tr>
<td>123</td>
<td>AllocationAdd Measure &quot;Product Plan&quot; Type Default</td>
</tr>
<tr>
<td>124</td>
<td>AllocationAdd Dimension &quot;Products&quot; Drill &quot;By Product Line&quot; Levels</td>
</tr>
<tr>
<td></td>
<td>&quot;Product Line&quot; Measure &quot;Product Plan&quot; Type Allocate AllocationMeasure</td>
</tr>
<tr>
<td></td>
<td>&quot;Revenue&quot;</td>
</tr>
<tr>
<td>125</td>
<td>AllocationAdd Dimension &quot;Years&quot; Drill &quot;By Time&quot; Levels &quot;Year&quot; Measure</td>
</tr>
<tr>
<td></td>
<td>&quot;Product Plan&quot; Type Allocate AllocationMeasure &quot;Revenue&quot;</td>
</tr>
</tbody>
</table>

Chapter 4. Creating a complete MDL model 33
Example - MDL model using an IBM Cognos data source in structured MDL

This is the order in which you would create a typical model using structured MDL:

- Create the data sources in Cognos Transformer.
- Create the dimensions and key performance measures.
- Create the standard dimension views.
- Create dimension views and custom views.
- Add the namespace and security objects.
- Populate the model and create the PowerCube.

The following example shows a model, created using an IBM Cognos data source, in structured MDL format.

<table>
<thead>
<tr>
<th>Structured MDL</th>
<th>Description</th>
</tr>
</thead>
</table>
| **CognosSource** 103  
"GO Sales and Retailers"  
**SourceType** Package  
**SourcePath** 
"/content/package[@name='GO Sales and Retailers']"  
**PackageTimeStamp** 
"/content/package[@name='GO Sales and Retailers']/model[@name='model']" | CognosSource defines an IBM Cognos package or report as the data source in the model.  
"GO Sales and Retailers" is the name of the IBM Cognos package. All packages and queries must have unique names.  
**SourceType** defines the type of IBM Cognos data source. The type can be Package, Report, or CognosSourceQuery. In this example, **SourceType** defines a package data source type.  
**SourcePath** is the path to the IBM Cognos package stored in Content Manager.  
**PackageTimeStamp** is the version of the package last used in the model. |
| **CognosSource** 11543  
"QuantityReport"  
**SourceType** Report  
SourcePath 
"/content/package[@name='GO Sales and Retailers']/report[@name='QuantityReport']"  
**PackageTimeStamp** 
"/content/package[@name='GO Sales and Retailers']/report[@name='QuantityReport']" | CognosSource defines a second IBM Cognos package or report data source in the model.  
In this example, the **SourceType** defines a report data source type. |
### Structured MDL

<table>
<thead>
<tr>
<th>Structured MDL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataSource 105</strong>&lt;br&gt;&quot;GO Sales and Retailers&quot;1&quot;&lt;br&gt;Separator &quot;,&quot;&lt;br&gt;SourceType CognosSourceQuery&lt;br&gt;CharacterSet Default DecimalSep &quot;,&quot; Thousandsep &quot;,&quot; Columns True Timing&lt;br&gt;PopYesCreateDefault&lt;br&gt;PackageReportSource 103&lt;br&gt;&quot;GO Sales and Retailers&quot;&lt;br&gt;AutoSummary True SetCurrent True&lt;br&gt;ServerSource False Speed False&lt;br&gt;Presummarized False</td>
<td><strong>CognosSourceQuery</strong> defines a query defined against a package or report SourceType. <strong>PackageReportSource 103 &quot;GO Sales and Retailers&quot;</strong> is the reference to the package on which this query is defined.</td>
</tr>
<tr>
<td>OrgName 107 &quot;[gosales_goretailers].[Orders].[Order number]&quot; Origin Source Offset 0&lt;br&gt;Column &quot;Order number&quot; Storage Float64&lt;br&gt;Scale 0&lt;br&gt;Size 4 Decimals 0 InputScale 0&lt;br&gt;TimeArray Off&lt;br&gt;Rollup CountAll</td>
<td>&quot;[gosales_goretailers].[Orders].[Order number]&quot; is the reference within the package or report to the query item or measure on which this column is defined. Following are more references to query items or measures on which the columns are defined.</td>
</tr>
<tr>
<td>OrgName 109 &quot;[gosales_goretailers].[Orders].[Retailer name]&quot; Origin Source Offset 1&lt;br&gt;Column &quot;Retailer name&quot; Storage Text&lt;br&gt;Scale 0 Size 102 Decimals 0&lt;br&gt;Class Description InputScale 0&lt;br&gt;TimeArray Off</td>
<td></td>
</tr>
<tr>
<td>OrgName 111 &quot;[gosales_goretailers].[Orders].[Order date]&quot; Origin Source Offset 2 Column &quot;Order date&quot;&lt;br&gt;Storage Int32 Scale 0 Size 12 Decimals 0&lt;br&gt;Class Date InputScale 0 TimeArray Off</td>
<td></td>
</tr>
<tr>
<td>OrgName 113 &quot;[gosales_goretailers].[Orders].[Order method]&quot; Origin Source Offset 3 Column &quot;Order method&quot;&lt;br&gt;Storage Text Scale 0 Size 102 Decimals 0&lt;br&gt;Class Description InputScale 0 TimeArray Off</td>
<td></td>
</tr>
<tr>
<td>OrgName 115 &quot;[gosales_goretailers].[Orders].[Order method code]&quot;&lt;br&gt;Origin Source Offset 4 Column &quot;Order method code&quot; Storage Float64&lt;br&gt;Scale 0 Size 4 Decimals 0&lt;br&gt;InputScale 0 TimeArray Off Rollup CountAll</td>
<td></td>
</tr>
<tr>
<td>OrgName 117 &quot;[gosales_goretailers].[Orders].[Product name]&quot; Origin Source Offset 5 Column &quot;Product name&quot;&lt;br&gt;Storage Text Scale 0 Size 102 Decimals 0&lt;br&gt;Class Description InputScale 0 TimeArray Off</td>
<td></td>
</tr>
<tr>
<td>Structured MDL</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OrgName 119 &quot;[gosales_goretailers].[Orders].[Quantity]&quot; Origin Source Offset 6 Column &quot;Quantity&quot;</td>
<td></td>
</tr>
<tr>
<td>Storage Float64 Scale 0 Size 2 Decimals 0</td>
<td>Filter 11553 &quot;Americas&quot; selects a filter from the data source and imports it into the query.</td>
</tr>
<tr>
<td>Class Quantity InputScale 0 TimeArray Off</td>
<td>FilterRef &quot;[gosales_goretailers].[Americas]&quot; is the reference within the package or report to the filter on which this column is defined.</td>
</tr>
<tr>
<td>Rollup Sum</td>
<td></td>
</tr>
<tr>
<td>Filter 11553 &quot;Americas&quot;</td>
<td></td>
</tr>
<tr>
<td>FilterRef &quot;[gosales_goretailers].[Americas]&quot;</td>
<td></td>
</tr>
<tr>
<td>DataSource 11545 &quot;QuantityReport~1&quot; SourceType CognosSourceQuery CharacterSet Default DecimalSep &quot;,&quot; Thousandsep &quot;,&quot; Columns True Timing PopYesCreateDefault</td>
<td>PackageReportSource 11543 &quot;QuantityReport&quot; is a second query, defined against a second package.</td>
</tr>
<tr>
<td>PackageReportSource 11543 &quot;QuantityReport&quot; AutoSummary False</td>
<td>Following are more references to query items or measures on which the columns are defined.</td>
</tr>
<tr>
<td>SetCurrent True</td>
<td></td>
</tr>
<tr>
<td>ServerSource False</td>
<td></td>
</tr>
<tr>
<td>Speed False</td>
<td></td>
</tr>
<tr>
<td>Presummarized False</td>
<td></td>
</tr>
<tr>
<td>OrgName 11547 &quot;[Report].[Query1.0].[Product number]&quot;</td>
<td></td>
</tr>
<tr>
<td>Origin Source Offset 0 Column &quot;Product number&quot; Storage Float64 Scale 0 Size 1 Decimals 0</td>
<td></td>
</tr>
<tr>
<td>Class Quantity InputScale 0 TimeArray Off</td>
<td></td>
</tr>
<tr>
<td>OrgName 11549 &quot;[Report].[Query1.0].[Production cost]&quot;</td>
<td></td>
</tr>
<tr>
<td>Origin Source Offset 1 Column &quot;Production cost&quot; Storage Float64 Scale 2 Size 1 Decimals 2</td>
<td></td>
</tr>
<tr>
<td>Class Quantity InputScale 0 TimeArray Off</td>
<td></td>
</tr>
<tr>
<td>Dimension 149 &quot;Order date&quot; DimType Date EarliestDate 19010101</td>
<td></td>
</tr>
<tr>
<td>LatestDate 21001231 ManualPeriods False DaysInWeek 127 NewCatsLock False ExcludeAutoPartitioning False DimDefaultCategory 0</td>
<td></td>
</tr>
<tr>
<td>Structured MDL</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Categories Root 153 "Order date"  
Inclusion Generate Lastuse 20070910  
Date 0  
Filtered False  
Suppressed False  
Sign False  
HideValue False  
IsKeyOrphanage False  
IsTruncated False  
Blanks False  
Drill 155 "By Order date" Inclusion  
Suppress Filtered False  
Suppressed True  
PrimaryDrill True  
HideValue False  
YearBegins 20070101  
PartialWeek Split ExtraWeek None  
WeekBegins Sunday  
Levels 161 "Year" Blanks "( blank )"  
Inclusion Generate DateFunction Year  
Generate Need RefreshLabel False  
RefreshDescription False  
RefreshShortName False  
NewCatsLock False  
CatLabFormat "YYYY"  
Timerank 10 | UniqueCategories True indicates that the Product Id level is unique. This is necessary because the level is associated with columns from multiple queries. |
| Associations 163 "Order date"  
AssociationType Type_Query  
AssociationRole Role_Source  
AssociationReferenced "Order date"  
Associations 165 "Order date"  
AssociationContext 155 AssociationType Type_Query AssociationRole Role_OrderBy  
AssociationReferenced "Order date"  
SortOrder Int16 SortAs Ascending | |
<table>
<thead>
<tr>
<th>Structured MDL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Levels 167 &quot;Quarter&quot; Blanks &quot;{ blank)&quot; Inclusion Generate Date Function Quarter</td>
<td></td>
</tr>
<tr>
<td>Generate All Refresh Label False Refresh Description False Refresh Short Name</td>
<td></td>
</tr>
<tr>
<td>False NewCats Lock False CatLab Format 'YYYY &quot;Q&quot; Q' Timerank 20 Unique Categories</td>
<td>True UniqueMove False Associations 169 &quot;Order date&quot; Association Type Type_Query</td>
</tr>
<tr>
<td>AssociationRole Role_Source AssociationReferenced &quot;Order date&quot; Associations</td>
<td>171 &quot;Order date&quot; AssociationContext 155 AssociationType Type_Query AssociationRole</td>
</tr>
<tr>
<td>Category 233 &quot;20040101-20041231&quot; Parent 155 Levels 161 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;2004&quot; Label &quot;2004&quot; Lastuse 20070910 SourceValue &quot;2004&quot; Date 20040101</td>
<td>Filtered False Suppressed False Sign False Hide Value False IsKeyOrphanage</td>
</tr>
<tr>
<td>False IsTruncated False Blanks False Category 235 &quot;20040101-20040331&quot; Parent</td>
<td>233 Levels 167 OrderBy Drill 155 Value &quot;20040101&quot; Label &quot;2004 Q 1&quot; Lastuse</td>
</tr>
<tr>
<td>20070910 SourceValue &quot;20040101&quot; Date 20040101 Filtered False Suppressed False</td>
<td>Sign False Hide Value False IsKeyOrphanage False IsTruncated False Blanks</td>
</tr>
<tr>
<td>False Category 243 &quot;20040401-20040630&quot; Parent 233 Levels 167 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;20040401&quot; Label &quot;2004 Q 2&quot; Lastuse 20070910 SourceValue &quot;20040401&quot; Date</td>
<td>20040401 Filtered False Suppressed False Sign False Hide Value False IsKeyOrphanage</td>
</tr>
<tr>
<td>False IsTruncated False Blanks False Category 251 &quot;20040701-20040930&quot; Parent 233</td>
<td></td>
</tr>
<tr>
<td>Levels 167 OrderBy Drill 155 Value &quot;20040701&quot; Label &quot;2004 Q 3&quot; Lastuse 20070910</td>
<td>SourceValue &quot;20040701&quot; Date 20040701 Filtered False Suppressed False Sign False</td>
</tr>
<tr>
<td>Hide Value False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>Structured MDL</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Category 259 &quot;20041001-20041231&quot;</td>
<td>Parent 233 Levels 167 OrderBy Drill 155 Value &quot;20041001&quot; Label &quot;2004 Q 4&quot; Lastuse 20070910 SourceValue &quot;20041001&quot; Date 20041001 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>Category 581 &quot;20050101-20051231&quot;</td>
<td>Parent 155 Levels 161 OrderBy Drill 155 Value &quot;2005&quot; Label &quot;2005&quot; Lastuse 20070910 SourceValue &quot;2005&quot; Date 20050101 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>Category 591 &quot;20050401-20050630&quot;</td>
<td>Parent 581 Levels 167 OrderBy Drill 155 Value &quot;20050401&quot; Label &quot;2005 Q 2&quot; Lastuse 20070910 SourceValue &quot;20050401&quot; Date 20050401 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>Category 599 &quot;20050701-20050930&quot;</td>
<td>Parent 581 Levels 167 OrderBy Drill 155 Value &quot;20050701&quot; Label &quot;2005 Q 3&quot; Lastuse 20070910 SourceValue &quot;20050701&quot; Date 20050701 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>Category 607 &quot;20051001-20051231&quot;</td>
<td>Parent 581 Levels 167 OrderBy Drill 155 Value &quot;20051001&quot; Label &quot;2005 Q 4&quot; Lastuse 20070910 SourceValue &quot;20051001&quot; Date 20051001 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>Structured MDL</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Category 683 &quot;20060101-20061231&quot; Parent 155 Levels 161 OrderBy Drill 155 Value &quot;2006&quot; Label &quot;2006&quot; Lastuse 20070910 SourceValue &quot;2006&quot; Date 20060101 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>Category 693 &quot;20060401-20060630&quot; Parent 683 Levels 167 OrderBy Drill 155 Value &quot;20060401&quot; Label &quot;2006 Q 2&quot; Lastuse 20070910 SourceValue &quot;20060401&quot; Date 20060401 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>Category 701 &quot;20060701-20060930&quot; Parent 683 Levels 167 OrderBy Drill 155 Value &quot;20060701&quot; Label &quot;2006 Q 3&quot; Lastuse 20070910 SourceValue &quot;20060701&quot; Date 20060701 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>Category 709 &quot;20061001-20061231&quot; Parent 683 Levels 167 OrderBy Drill 155 Value &quot;20061001&quot; Label &quot;2006 Q 4&quot; Lastuse 20070910 SourceValue &quot;20061001&quot; Date 20061001 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>MapDrills MapDrill 155</td>
<td></td>
</tr>
<tr>
<td>ViewName 157 &quot;All Categories&quot; Type All ViewCustomView 0</td>
<td></td>
</tr>
<tr>
<td>ViewName 159 &quot;Omit Dimension&quot; Type Omit ViewCustomView 0</td>
<td></td>
</tr>
<tr>
<td>Structure MDL</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Dimension 195 &quot;Order method&quot;</td>
<td></td>
</tr>
<tr>
<td>DimType Regular NewCatsLock False ExclAutoPartitioning False DimDefaultCategory 0</td>
<td></td>
</tr>
<tr>
<td>Categories Root 197 &quot;Order method&quot;</td>
<td></td>
</tr>
<tr>
<td>Inclusion Generate Lastuse 20070910 Filtered False</td>
<td></td>
</tr>
</tbody>
</table>
| Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False Drill 199 "By Order method" Inclusion Suppress Filtered False Suppressed True PrimaryDrill True HideValue False Levels 205 "Order method" Blanks "( Blank )" DateFunction None Generate None RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False Timerank 0 UniqueCategories False UniqueMove False Associations 207 "Order method code" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Order method code" Associations 209 "Order method" AssociationType Type_Query AssociationRole Role_Label AssociationReferenced "Order method" Category 267 "7" Parent 199 Levels 205 Label "Sales visit" Lastuse 20070910 SourceValue "7" Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False Category 285 "4" Parent 199 Levels 205 Label "E-mail" Lastuse 20070910 SourceValue "4" Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False Category 309 "5" Parent 199 Levels 205 Label "Web" Lastuse 20070910 SourceValue "5" Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False
Structured MDL  |  Description
--- | ---
Category 319 "2" Parent 199 Levels 205  
Label "Telephone" Lastuse 20070910  
SourceValue "2" Filtered False  
Suppressed False Sign False  
HideValue False IsKeyOrphanage False  
IsTruncated False Blanks False

Category 357 "1" Parent 199 Levels 205  
Label "Fax" Lastuse 20070910  
SourceValue "1" Filtered False  
Suppressed False Sign False  
HideValue False IsKeyOrphanage False  
IsTruncated False Blanks False

Category 437 "3" Parent 199 Levels 205  
Label "Mail" Lastuse 20070910  
SourceValue "3" Filtered False  
Suppressed False Sign False  
HideValue False IsKeyOrphanage False  
IsTruncated False Blanks False

Category 475 "8" Parent 199 Levels 205  
Label "Special"  
Lastuse 20070910 SourceValue "8"  
Current Filtered False  
Suppressed False Sign False  
HideValue False IsKeyOrphanage False  
IsTruncated False Blanks False

MapDrills MapDrill 199

ViewName 201 "All Categories"  
Type All ViewCustomView 0

ViewName 203 "Omit Dimension"  
Type Omit ViewCustomView 0

These are the two standard views created for all dimensions.

If you do not manually create the two default dimension views required for each dimension (All Categories and Omit Dimension), you must use the ModelEnsureCompleteness script to have Cognos Transformer create them automatically.

ViewName 11555 "Authors~User View"  
ViewCustomView 11529  
Apex 197 Filter 319

ViewName 11557 "Sub Authors~User View"  
ViewCustomView 11537 Apex 197 Filter 319
<table>
<thead>
<tr>
<th>Structured MDL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure 225 &quot;Quantity&quot; Rollup Sum IgnoreMissingValue False Storage Float64 OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False IsFolder False DrillThrough False EndList Associations 227 &quot;Quantity&quot; AssociationType Type_Query AssociationRole Role_Source AssociationReferenced &quot;Quantity&quot;</td>
<td></td>
</tr>
<tr>
<td>CustomView 11529 &quot;Authors&quot; DimensionView 149 &quot;All Categories&quot; DimensionView 195 &quot;Authors~User View&quot; MeasureInclude 225 Yes</td>
<td>CustomView indicates that a custom view has been defined. &quot;Authors&quot; is the name of the custom view. Custom view names must be unique. DimensionView 149 &quot;All Categories&quot; is a reference to a dimension (149) and the view that this custom view uses in that dimension, including all categories in the dimension. The second dimension, DimensionView 195 &quot;Authors~User View&quot;, is the customized dimension view used for this custom view. MeasureInclude 225 Yes indicates that measure 225 is included in the custom view. MeasureInclude values are Yes or No.</td>
</tr>
<tr>
<td>CustomView 11537 &quot;Sub Authors&quot; DimensionView 149 &quot;All Categories&quot; DimensionView 195 &quot;Sub Authors~User View&quot;</td>
<td>This entry creates a second custom view.</td>
</tr>
<tr>
<td>MeasureInclude 225 No</td>
<td>MeasureInclude 225 No indicates that measure 225 is excluded from this custom view.</td>
</tr>
<tr>
<td>CustomViewChildList 11529 StartList 11537 EndList CustomViewChildList 11537 StartList 11537 EndList</td>
<td>CustomViewChildList 11529 StartList 11537 EndList defines the custom views that are descendents of the custom view &quot;Authors&quot;. References to custom views may be unique identifiers or names.</td>
</tr>
<tr>
<td>SecurityNameSpace 11533 &quot;Cognos&quot; SecurityNameSpaceCAMID 'CAMID(&quot;::&quot;)'</td>
<td>SecurityNameSpace... defines the namespace from which security objects are retrieved. In this example, &quot;Cognos&quot; is the name of the namespace in Content Manager.</td>
</tr>
</tbody>
</table>
Structured MDL | Description
--- | ---
SecurityObject 11531 'CAMID("Authors")' SecurityObjectDisplayName "Authors" SecurityObjectType SecurityType_Role CustomViewList 11529 EndList | SecurityObject is the imported security object from the last defined namespace. 'CAMID("Authors")' is the security control mechanism identifier for the security object from Content Manager. SecurityType_Role indicates that the type of security object from Content Manager is a role. Possible values are SecurityType_Role, SecurityType_Group, and SecurityType_User. CustomViewList 11529 EndList is a list of the custom views to which this security object is assigned. References to custom views may be unique identifiers or names.

SecurityObject 11539 'CAMID("Analysis Users")' SecurityObjectDisplayName "Analysis Users" SecurityObjectType SecurityType_Role CustomViewList 11537 EndList |  

SecurityNameSpace 11563 "GOnamespace" SecurityNameSpaceCAMID 'CAMID("GOnamespace")' | SecurityNameSpace defines a second namespace from which security objects are retrieved. When multiple namespaces are defined in a custom view, they are organized by namespace, followed by the security objects from that namespace.

SecurityObject 11561 'CAMID("GOnamespace:r:authid=2589996611")' SecurityObjectDisplayName "Root User Class" SecurityObjectType SecurityType_Role CustomViewList 11537 EndList |  

---

44 IBM Cognos Transformer Version 10.2.1: Developer Guide
Example - MDL model using an IBM Cognos data source in verb MDL

This is the order in which you would create a typical model using verb MDL:

- Create the data sources in Cognos Transformer.
- Create the dimensions and key performance measures.
- Create the standard dimension views.
- Create dimension views and custom views.
- Add the namespace and security objects.
 Populate the model and create the PowerCube.

The following example shows the same model, created using an IBM Cognos data source, in verb MDL format.

**Note:** The Add form illustrates the verb that you use to define an object, to
differentiate the syntax from that which Cognos Transformer uses when defining
objects (the Make form). However, either form is acceptable. Also, you will note that
each statement starts with the verb, followed by a lengthy series of options. For
readability, you may wish to insert carriage returns between the keywords.

<table>
<thead>
<tr>
<th>Verb MDL</th>
<th>Description</th>
</tr>
</thead>
</table>
| Make 103 "GO Sales and Retailers" SourceType Package SourcePath "/content/package[@name='GOSales and Retailers']" PackageTimeStamp "/content/package[@name='GOSales and Retailers']/model[@name='model']" | Make is the verb used to create a package or report.
"GO Sales and Retailers" is the name of the IBM Cognos package. All packages and queries must have unique names.
SourceType defines the type of IBM Cognos data source. The type can be Package, Report, or CognosSourceQuery.
In this example, SourceType defines a package data source type.
SourcePath is the path to the IBM Cognos package stored in Content Manager.
PackageTimeStamp is the version of the package last used in the model. |
| Make 11543 "QuantityReport" SourceType Report SourcePath "/content/package[/name='GOSales and Retailers']/report[@name='QuantityReport']" PackageTimeStamp "/content/package[@name='GOSales and Retailers']/report[@name='QuantityReport']" | Make defines a second IBM Cognos package or report data source in the model.
In this example, the SourceType defines a report data source type. |
| DataSourceMake 105 "GO Sales and Retailers" Separator ""," SourceType CognosSourceQuery CharacterSet Default DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault PackageReportSource 103 "GOSales and Retailers" AutoSummary True SetCurrent True ServerSource False Speed False Presummarized False | DataSourceMake defines a query defined against a package or report SourceType.
PackageReportSource 103 "GOSales and Retailers" is the reference to the package on which this query is defined. |
<table>
<thead>
<tr>
<th>Verb MDL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnMake 107 &quot;$\text{gosales戈oretailers}. [Orders]. [Order number]&quot;</td>
<td>&quot;$\text{gosales戈oretailers}. [Orders]. [Order number]&quot; is the reference within the package or report to the query item or measure on which this column is defined. Following are more references to query items or measures on which the columns are defined.</td>
</tr>
<tr>
<td>DataSource 105 Origin Source Offset 0 Column &quot;Order number&quot; Storage Float64 Scale 0 Size 4 Decimals 0 InputScale 0 TimeArray Off Rollup CountAll</td>
<td></td>
</tr>
<tr>
<td>ColumnMake 109 &quot;$\text{gosales戈oretailers}. [Orders]. [Retailer name]&quot;</td>
<td></td>
</tr>
<tr>
<td>DataSource 105 Origin Source Offset 1 Column &quot;Retailer name&quot; Storage Text Scale 0 Size 102 Decimals 0 Class Description InputScale 0 TimeArray Off</td>
<td></td>
</tr>
<tr>
<td>ColumnMake 111 &quot;$\text{gosales戈oretailers}. [Orders]. [Order date]&quot;</td>
<td></td>
</tr>
<tr>
<td>DataSource 105 Origin Source Offset 2 Column &quot;Order date&quot; Storage Int32 Scale 0 Size 12 Decimals 0 Class Date InputScale 0 TimeArray Off</td>
<td></td>
</tr>
<tr>
<td>ColumnMake 113 &quot;$\text{gosales戈oretailers}. [Orders]. [Order method]&quot;</td>
<td></td>
</tr>
<tr>
<td>DataSource 105 Origin Source Offset 3 Column &quot;Order method&quot; Storage Text Scale 0 Size 102 Decimals 0 Class Description InputScale 0 TimeArray Off</td>
<td></td>
</tr>
<tr>
<td>ColumnMake 115 &quot;$\text{gosales戈oretailers}. [Orders]. [Order method code]&quot;</td>
<td></td>
</tr>
<tr>
<td>DataSource 105 Origin Source Offset 4 Column &quot;Order method code&quot; Storage Float64 Scale 0 Size 4 Decimals 0 InputScale 0 TimeArray Off Rollup CountAll</td>
<td></td>
</tr>
<tr>
<td>ColumnMake 117 &quot;$\text{gosales戈oretailers}. [Orders]. [Product name]&quot;</td>
<td></td>
</tr>
<tr>
<td>DataSource 105 Origin Source Offset 5 Column &quot;Product name&quot; Storage Text Scale 0 Size 102 Decimals 0 Class Description InputScale 0 TimeArray Off</td>
<td></td>
</tr>
<tr>
<td>ColumnMake 119 &quot;$\text{gosales戈oretailers}. [Orders]. [Quantity]&quot;</td>
<td></td>
</tr>
<tr>
<td>DataSource 105 Origin Source Offset 6 Column &quot;Quantity&quot; Storage Float64 Scale 0 Size 2 Decimals 0 Class Quantity InputScale 0 TimeArray Off Rollup Sum FilterMake 11553 &quot;Americas&quot; FilterRef &quot;$\text{gosales戈oretailers}. [Americas]&quot;</td>
<td></td>
</tr>
<tr>
<td>FilterMake 11553 &quot;Americas&quot; FilterRef &quot;$\text{gosales戈oretailers}. [Americas]&quot;</td>
<td>FilterMake 115533 &quot;Americas&quot; selects a filter from the data source and imports it into the query. FilterRef &quot;$\text{gosales戈oretailers}. [Americas]&quot; is the reference within the package or report to the filter on which this column is defined.</td>
</tr>
</tbody>
</table>
### Verb MDL

<table>
<thead>
<tr>
<th>Verb MDL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DataSourceMake 11545 &quot;QuantityReport&quot;</code></td>
<td><code>DataSourceMake 11545 &quot;QuantityReport&quot;</code> describes the data source for the query.</td>
</tr>
<tr>
<td><code>Separator &quot;,&quot;</code></td>
<td>Separates fields in the data source.</td>
</tr>
<tr>
<td><code>SourceType CognosSourceQuery</code></td>
<td>Query source type.</td>
</tr>
<tr>
<td><code>CharacterSet Default</code></td>
<td>Default character set.</td>
</tr>
<tr>
<td><code>DecimalSep &quot;.&quot;</code></td>
<td>Decimal separator.</td>
</tr>
<tr>
<td><code>ThousandsSep &quot;,&quot;</code></td>
<td>Thousand separator.</td>
</tr>
<tr>
<td><code>Columns True Timing PopYes CreateDefault</code></td>
<td>Indicates the columns are defined.</td>
</tr>
<tr>
<td><code>PackageReportSource 11543&quot;QuantityReport&quot;</code></td>
<td><code>PackageReportSource</code> is a second query, defined against a second package.</td>
</tr>
<tr>
<td><code>AutoSummary False SetCurrent True ServerSource False Speed False Presummarized False</code></td>
<td>Settings for the query.</td>
</tr>
<tr>
<td><code>ColumnMake 11547 &quot;[Report].[Query1.0].[Product number]&quot;</code></td>
<td>Defines a column in the query.</td>
</tr>
<tr>
<td><code>Origin Source Offset 0 Column &quot;Product number&quot; Storage Float64 Scale 0 Size 1 Decimals 0 Class Quantity InputScale 0 TimeArray Off</code></td>
<td>Column details.</td>
</tr>
<tr>
<td><code>ColumnMake 11549 &quot;[Report].[Query1.0].[Production cost]&quot;</code></td>
<td>Defines a column in the query.</td>
</tr>
<tr>
<td><code>Origin Source Offset 1 Column &quot;Production cost&quot; Storage Float64 Scale 2 Size 1 Decimals 2 Class Quantity InputScale 0 TimeArray Off</code></td>
<td>Column details.</td>
</tr>
<tr>
<td><code>DimMake 149 &quot;Order date&quot; DimType Date EarliestDate 19010101 LatestDate 21001231 ManualPeriods False DaysInWeek 127 NewCatsLock False ExcludeAutoPartitioning False DimDefaultCategory 0</code></td>
<td>Defines a dimension.</td>
</tr>
<tr>
<td><code>RootCatMake 153 &quot;Order date&quot; Dimension 149 Inclusion Generate Lastuse 20070910 Date 0 Suppressed False Suppressed False IsKeyOrphanage False IsTruncated False Blanks False</code></td>
<td>Creates a root category.</td>
</tr>
<tr>
<td><code>DrillCatMake 155 &quot;By Order date&quot; Dimension 149 Root 153 Inclusion Suppress Filtered False Suppressed True PrimaryDrill True HideValue False YearBegins 20070101 PartialWeek Split ExtraWeek None WeekBegins Sunday LevelMake 161 &quot;Year&quot; Drill 155 Parent 0 Blanks &quot;( blank )&quot; Inclusion Generate DateFunction Generate Need RefreshLabel False RefreshDescription False</code></td>
<td>Creates a drill category.</td>
</tr>
<tr>
<td><code>RefreshShortName False NewCatsLock False CatLabFormat &quot;YYYY&quot; Timerank 10 UniqueCategories True UniqueMove False</code></td>
<td>Refresh and category settings.</td>
</tr>
<tr>
<td><code>Associations 163 &quot;Order date&quot;</code></td>
<td>Associations with other objects.</td>
</tr>
<tr>
<td><code>AssociationType Type_Query AssociationRole Role_Source AssociationReferenced &quot;Order date&quot; Associations 165 &quot;Order date&quot; AssociationContext 155 AssociationType Type_Query AssociationRole Role_OrderBy AssociationReferenced &quot;Order date&quot; SortOrder Int16 SortAs Ascending</code></td>
<td>Association details.</td>
</tr>
</tbody>
</table>

---

When you create any dimension using MDL, you must manually create both the root category and drill category for that dimension.

The verbs `RootCatMake` and `DrillCatMake` are used in this example because there is no `Add` verb form for these objects. Also, you must set the inclusion property to `Suppress` for the drill category so that it does not display in the reporting components.

`UniqueCategories True` indicates that the Year level is unique. This is necessary because the level is associated with columns from multiple queries.
<table>
<thead>
<tr>
<th>Verb MDL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LevelMake 167 &quot;Quarter&quot; Drill 155</td>
<td></td>
</tr>
<tr>
<td>Parent 161</td>
<td></td>
</tr>
<tr>
<td>Blanks &quot;( blank )&quot; Inclusion</td>
<td></td>
</tr>
<tr>
<td>Generate DateFunction Quarter</td>
<td></td>
</tr>
<tr>
<td>Generate All RefreshLabel False</td>
<td></td>
</tr>
<tr>
<td>RefreshDescription False</td>
<td></td>
</tr>
<tr>
<td>RefreshShortName False</td>
<td></td>
</tr>
<tr>
<td>NewCatsLock False</td>
<td></td>
</tr>
<tr>
<td>CatLabFormat 'YYYY &quot;Q&quot; Q'</td>
<td></td>
</tr>
<tr>
<td>Timerank 20 UniqueCategories True</td>
<td></td>
</tr>
<tr>
<td>UniqueMove False</td>
<td></td>
</tr>
<tr>
<td>Associations 169 &quot;Order date&quot;</td>
<td></td>
</tr>
<tr>
<td>AssociationType Type_Query</td>
<td></td>
</tr>
<tr>
<td>AssociationRole Role_Source</td>
<td></td>
</tr>
<tr>
<td>AssociationReferenced &quot;Order date&quot;</td>
<td></td>
</tr>
<tr>
<td>Associations 171 &quot;Order date&quot;</td>
<td></td>
</tr>
<tr>
<td>AssociationContext 155</td>
<td></td>
</tr>
<tr>
<td>AssociationType Type_Query</td>
<td></td>
</tr>
<tr>
<td>AssociationRole Role_OrderBy</td>
<td></td>
</tr>
<tr>
<td>AssociationReferenced &quot;Order date&quot;</td>
<td></td>
</tr>
<tr>
<td>SortOrder Int16 SortAs Ascending</td>
<td></td>
</tr>
<tr>
<td>CatMake 233 &quot;20040101-20041231&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 149 Drill 155</td>
<td></td>
</tr>
<tr>
<td>Levels 161 Parent 155 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;2004&quot; Label &quot;2004&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;2004&quot;</td>
<td></td>
</tr>
<tr>
<td>Date 20040101 Filtered False Suppressed False Sign False HideValue False</td>
<td></td>
</tr>
<tr>
<td>IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 235 &quot;20040101-20040331&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 149 Drill 155</td>
<td></td>
</tr>
<tr>
<td>Levels 167 Parent 233 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;20040101&quot; Label &quot;2004 Q 1&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;20040101&quot; Date 20040101 Filtered False</td>
<td></td>
</tr>
<tr>
<td>Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 243 &quot;20040401-20040630&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 149 Drill 155</td>
<td></td>
</tr>
<tr>
<td>Levels 167 Parent 233 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;20040401&quot; Label &quot;2004 Q 2&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;20040401&quot; Date 20040401 Filtered False</td>
<td></td>
</tr>
<tr>
<td>Sign False HideValueFalse IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 251 &quot;20040701-20040930&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 149 Drill 155</td>
<td></td>
</tr>
<tr>
<td>Levels 167 Parent 233 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;20040701&quot; Label &quot;2004 Q 3&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;20040701&quot; Date 20040701 Filtered False</td>
<td></td>
</tr>
<tr>
<td>Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>Verb MDL</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>CatMake 259 &quot;20041001-20041231&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 149 Drill 155 Levels 167</td>
<td></td>
</tr>
<tr>
<td>Parent 233 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;20041001&quot; Label &quot;2004 Q 4&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;20041001&quot;</td>
<td></td>
</tr>
<tr>
<td>Date 20041001 Filtered False</td>
<td></td>
</tr>
<tr>
<td>Suppressed False Sign False</td>
<td></td>
</tr>
<tr>
<td>HideValue False IsKeyOrphanage False</td>
<td></td>
</tr>
<tr>
<td>IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 581 &quot;20050101-20051231&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 149 Drill 155 Levels 161</td>
<td></td>
</tr>
<tr>
<td>Parent 155 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;2005&quot; Label &quot;2005&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;2005&quot;</td>
<td></td>
</tr>
<tr>
<td>Date 20050101 Filtered False</td>
<td></td>
</tr>
<tr>
<td>Suppressed False Sign False</td>
<td></td>
</tr>
<tr>
<td>HideValue False IsKeyOrphanage False</td>
<td></td>
</tr>
<tr>
<td>IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 591 &quot;20050401-20050630&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 149 Drill 155 Levels 167</td>
<td></td>
</tr>
<tr>
<td>Parent 581 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;20050401&quot; Label &quot;2005 Q 2&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;20050401&quot;</td>
<td></td>
</tr>
<tr>
<td>Date 20050401 Filtered False</td>
<td></td>
</tr>
<tr>
<td>Suppressed False Sign False</td>
<td></td>
</tr>
<tr>
<td>HideValue False IsKeyOrphanage False</td>
<td></td>
</tr>
<tr>
<td>IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 599 &quot;20050701-20050930&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 149 Drill 155 Levels 167</td>
<td></td>
</tr>
<tr>
<td>Parent 581 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;20050701&quot; Label &quot;2005 Q 3&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;20050701&quot;</td>
<td></td>
</tr>
<tr>
<td>Date 20050701 Filtered False</td>
<td></td>
</tr>
<tr>
<td>Suppressed False Sign False</td>
<td></td>
</tr>
<tr>
<td>HideValue False IsKeyOrphanage False</td>
<td></td>
</tr>
<tr>
<td>IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 607 &quot;20051001-20051231&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 149 Drill 155 Levels 167</td>
<td></td>
</tr>
<tr>
<td>Parent 581 OrderBy Drill 155</td>
<td></td>
</tr>
<tr>
<td>Value &quot;20051001&quot; Label &quot;2005 Q 4&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;20051001&quot;</td>
<td></td>
</tr>
<tr>
<td>Date 20051001 Filtered False</td>
<td></td>
</tr>
<tr>
<td>Suppressed False Sign False</td>
<td></td>
</tr>
<tr>
<td>HideValue False IsKeyOrphanage False</td>
<td></td>
</tr>
<tr>
<td>IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>Verb MDL</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>CatMake 683 &quot;20060101-20061231&quot;</td>
<td>Dimension 149 Drill 155 Levels 161 Parent 155 OrderBy Drill 155 Value &quot;2006&quot; Label &quot;2006&quot; Lastuse 20070910 SourceValue &quot;2006&quot; Date 20060101 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>CatMake 685 &quot;20060101-20060331&quot;</td>
<td>Dimension 149 Drill 155 Levels 167 Parent 683 OrderBy Drill 155 Value &quot;20060101&quot; Label &quot;2006 Q 1&quot; Lastuse 20070910 SourceValue &quot;20060101&quot; Date 20060101 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>CatMake 693 &quot;20060401-20060630&quot;</td>
<td>Dimension 149 Drill 155 Levels 167 Parent 683 OrderBy Drill 155 Value &quot;20060401&quot; Label &quot;2006 Q 2&quot; Lastuse 20070910 SourceValue &quot;20060401&quot; Date 20060401 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>CatMake 701 &quot;20060701-20060930&quot;</td>
<td>Dimension 149 Drill 155 Levels 167 Parent 683 OrderBy Drill 155 Value &quot;20060701&quot; Label &quot;2006 Q 3&quot; Lastuse 20070910 SourceValue &quot;20060701&quot; Date 20060701 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>CatMake 709 &quot;20061001-20061231&quot;</td>
<td>Dimension 149 Drill 155 Levels 167 Parent 683 OrderBy Drill 155 Value &quot;20061001&quot; Label &quot;2006 Q 4&quot; Lastuse 20070910 SourceValue &quot;20061001&quot; Date 20061001 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
</tr>
<tr>
<td>ViewMake 157 &quot;All Categories&quot;</td>
<td>Dimension 149 Type All ViewCustomView 0</td>
</tr>
<tr>
<td>ViewMake 159 &quot;Omit Dimension&quot;</td>
<td>Dimension 149 Type Omit ViewCustomView 0</td>
</tr>
<tr>
<td>DimMake 195 &quot;Order method&quot;</td>
<td>DimType Regular NewCatsLock False ExcludeAutoPartitioning False DimDefaultCategory 0</td>
</tr>
<tr>
<td>Verb MDL</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>RootCatMake 197 &quot;Order method&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 195</td>
<td></td>
</tr>
<tr>
<td>Inclusion Generate Lastuse 20070910</td>
<td></td>
</tr>
<tr>
<td>Filtered False Suppressed False</td>
<td></td>
</tr>
<tr>
<td>Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>DrillCatMake 199 &quot;By Order method&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 195 Root 197 Inclusion Suppress Filtered False Suppressed True PrimaryDrill True HideValue False LevelMake 205 &quot;Order method&quot; Drill 199 Parent 0 Blanks &quot;( Blank )&quot; DateFunction None Generate None RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False Timerank 0 UniqueCategories False UniqueMove False Associations 207 &quot;Order method code&quot; AssociationType Type_Query AssociationRole Role_Source AssociationReferenced &quot;Order method code&quot; Associations 209 &quot;Order method&quot; AssociationType Type_Query AssociationRole Role_Label AssociationReferenced &quot;Order method&quot;</td>
<td></td>
</tr>
<tr>
<td>CatMake 267 &quot;7&quot; Dimension 195 Drill 199 Levels 205 Parent 199 Label &quot;Sales visit&quot; Lastuse 20070910 SourceValue &quot;7&quot; Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 285 &quot;4&quot; Dimension 195 Drill 199 Levels 205 Parent 199 Label &quot;E-mail&quot; Lastuse 20070910 SourceValue &quot;4&quot; Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 309 &quot;5&quot; Dimension 195 Drill 199 Levels 205 Parent 199 Label &quot;Web&quot; Lastuse 20070910 SourceValue &quot;5&quot; Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 319 &quot;2&quot; Dimension 195 Drill 199 Levels 205 Parent 199 Label &quot;Telephone&quot; Lastuse 20070910 SourceValue &quot;2&quot; Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False</td>
<td></td>
</tr>
<tr>
<td>Verb MDL</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>CatMake 357 &quot;1&quot; Dimension 195 Drill 199</td>
<td></td>
</tr>
<tr>
<td>Levels 205 Parent 199 Label &quot;Fax&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;1&quot;</td>
<td></td>
</tr>
<tr>
<td>Filtered False Suppressed False</td>
<td></td>
</tr>
<tr>
<td>Sign False HideValue False</td>
<td></td>
</tr>
<tr>
<td>IsKeyOrphanage False IsTruncated False</td>
<td></td>
</tr>
<tr>
<td>Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 437 &quot;3&quot; Dimension 195 Drill 199</td>
<td></td>
</tr>
<tr>
<td>Levels 205 Parent 199 Label &quot;Mail&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;3&quot;</td>
<td></td>
</tr>
<tr>
<td>Filtered False Suppressed False</td>
<td></td>
</tr>
<tr>
<td>Sign False HideValue False</td>
<td></td>
</tr>
<tr>
<td>IsKeyOrphanage False IsTruncated False</td>
<td></td>
</tr>
<tr>
<td>Blanks False</td>
<td></td>
</tr>
<tr>
<td>CatMake 475 &quot;8&quot; Dimension 195 Drill 199</td>
<td></td>
</tr>
<tr>
<td>Levels 205 Parent 199 Label &quot;Special&quot;</td>
<td></td>
</tr>
<tr>
<td>Lastuse 20070910 SourceValue &quot;8&quot;</td>
<td></td>
</tr>
<tr>
<td>Current Filtered False Suppressed False</td>
<td></td>
</tr>
<tr>
<td>Sign False HideValue False</td>
<td></td>
</tr>
<tr>
<td>IsKeyOrphanage False IsTruncated False</td>
<td></td>
</tr>
<tr>
<td>Blanks False</td>
<td></td>
</tr>
<tr>
<td>ViewMake 201 &quot;All Categories&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 195 Type All ViewCustomView 0</td>
<td></td>
</tr>
<tr>
<td>ViewMake 203 &quot;Omit Dimension&quot;</td>
<td></td>
</tr>
<tr>
<td>Dimension 195 Type Omit</td>
<td></td>
</tr>
<tr>
<td>ViewCustomView 0</td>
<td></td>
</tr>
</tbody>
</table>

These are the two standard views created for all dimensions. If you do not manually create the two default dimension views required for each dimension (All Categories and Omit Dimension), you must use the ModelEnsureCompleteness script to have Cognos Transformer create them automatically.

| ViewMake 11555 "Authors~UserView"          |             |
| Dimension 195 ViewCustomView 11529        |             |
| Apex 197 Filter 319                       |             |

ViewMake 11555 "Authors~User View" creates a customized dimension view. ViewCustomView 11529 is a reference to the view that this custom view uses in the dimension. Apex 197 Filter 319 is a list of the categories that have customization options set, such as apexing or cloaking.
<table>
<thead>
<tr>
<th>Verb MDL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomViewMake 11529 &quot;Authors&quot;</td>
<td>CustomViewMake is the verb used to define a custom view.</td>
</tr>
<tr>
<td>DimensionView 149 &quot;All Categories&quot;</td>
<td>&quot;Authors&quot; is the name of the custom view. Custom view names must be unique.</td>
</tr>
<tr>
<td>DimensionView 195 &quot;Authors~User View&quot;</td>
<td>DimensionView 149 &quot;All Categories&quot; is a reference to a dimension (149) and the view that this custom view uses in that dimension, including all categories in the dimension.</td>
</tr>
<tr>
<td>MeasureInclude 225 Yes</td>
<td>The second dimension, DimensionView 195 &quot;Authors~User View&quot;, is the customized dimension view used for this custom view.</td>
</tr>
<tr>
<td>MeasureInclude 225 No</td>
<td>MeasureInclude 225 Yes indicates that measure 225 is included in the custom view. MeasureInclude values are Yes or No.</td>
</tr>
<tr>
<td>MeasureInclude 225 No</td>
<td>This entry creates a second custom view.</td>
</tr>
<tr>
<td>MeasureInclude 225 No</td>
<td>MeasureInclude 225 No indicates that measure 225 has been excluded from this custom view.</td>
</tr>
<tr>
<td>CustomViewChildListUpdate 11529</td>
<td>CustomViewChildListUpdate is the verb used to define the list of descendent custom views. There is only an Update version of this verb.</td>
</tr>
<tr>
<td>StartList 11537 EndList</td>
<td>CustomViewChildListUpdate 11529 StartList 11537 EndList defines the custom views that are descendents of the custom view &quot;Authors&quot;.</td>
</tr>
<tr>
<td>CustomViewChildListUpdate 11529</td>
<td>References to custom views may be unique identifiers or names.</td>
</tr>
<tr>
<td>SecurityNameSpaceMake 11533 &quot;Cognos&quot;</td>
<td>SecurityNameSpace 11533 &quot;Cognos&quot; SecurityNameSpaceCAMID 'CAMID(&quot;:&quot;)' is the namespace from which security objects are retrieved. In this example, &quot;Cognos&quot; is the name of the namespace in Content Manager.</td>
</tr>
<tr>
<td>Verb MDL</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>SecurityObjectMake 11531 'CAMID(&quot;:Authors&quot;)' SecurityNameSpace 11533 SecurityObjectDisplayName &quot;Authors&quot; SecurityObjectType SecurityType_Role CustomViewList 11529 EndList</td>
<td>SecurityObjectMake is the verb used to define a security object. SecurityObject is the imported security object from the last defined namespace. 'CAMID(&quot;:Authors&quot;)' is the security control mechanism identifier for the security object from Content Manager. SecurityNameSpace 11533 is the namespace that this security object comes from. SecurityType_Role indicates that the type of security object from Content Manager is a role. Possible values are SecurityType_Role, SecurityType_Group and SecurityType_User. CustomViewList 11529 EndList is a list of the custom views to which this security object is assigned. References to custom views may be unique identifiers or names.</td>
</tr>
<tr>
<td>SecurityObjectMake 11539 'CAMID(&quot;:Analysis Users&quot;)' SecurityNameSpace 11533 SecurityObjectDisplayName &quot;Analysis Users&quot; SecurityObjectType SecurityType_Role CustomViewList 11537 EndList</td>
<td></td>
</tr>
<tr>
<td>SecurityNameSpaceMake 11563 &quot;GOnamespace&quot; SecurityNameSpaceCAMID 'CAMID(&quot;GOnamespace&quot;)'</td>
<td>SecurityNameSpaceMake defines a second namespace from which security objects are retrieved. When multiple namespaces are defined in a custom view, they are organized by namespace, followed by the security objects from that namespace.</td>
</tr>
<tr>
<td>SecurityObjectMake 11561 'CAMID(&quot;GOnamespace::authid=2589996611&quot;)' SecurityNameSpace 11563 SecurityObjectDisplayName &quot;Root User Class&quot; SecurityObjectType SecurityType_Role CustomViewList 11537 EndList</td>
<td></td>
</tr>
</tbody>
</table>
This section creates the PowerCube.

**PowerCubeCustomViewListUpdate** is the verb used to define the custom views assigned to the PowerCube. There is only an Update version of this verb.

The list includes only the lowest custom views in the hierarchies. All ancestors are also considered to be assigned to the cube.

After you add the required measures, you may want to specify how they will be allocated across the various dimensions in your model.

### Sample scripts

The following sample scripts are included for use as a starting point. You can modify them to perform some of the more common MDL scripting operations.
Change data source types sample

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

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

Change data source sample

This script removes all categories in the Locations dimension and changes the source for this 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" 20011231
DataSourceUpdate "Locations (CSV)" SourceType Access Source "c:\Locations.mdb" DataRange "Locations"
PopulateFromQueries "Locations (CSV)"
CreateFiles

Update selected cubes sample

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 sample

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" 20011231 DataSourceUpdate
"Channels (dBase 4.0)" Source "c:\Channels.dbf"
PopulateFromQueries "Channels (dBase 4.0)" CreateFiles

Description samples

Every Cognos Transformer object can have a description up to 4,095 characters in length. However, in MDL the description must be divided into a series of smaller 256-character blocks, and all descriptions must be enclosed in quotation marks.

This example provides a description for the Products dimension 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 sample

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

OpenMDL "c:\Outdoors.mdl" DimUpdate
"Products" DimInfo "Updating dimension description" ColumnUpdate "Product Line"
ColumnInfo "Updating column description"
LevelUpdate "Product Line" LevelInfo
"Updating level description" SaveMDL "c:\Outdoors.mdl"
Change a cube output location sample
This script changes the output location of the national cube to c:\mdcdir\national.mdc, and saves the model.
OpenPy "model.pyj" CubeUpdate "national"
MDCFile "c:\mdcdir\national.mdc" SavePy "model.pyj"

Change several cube output locations sample
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 "Great Outdoors Sales" MDCFile "c:\Great Outdoors"
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 sample
This script changes the file type of a PowerCube and saves the model.
OpenPy "model.pyj"
SignonMake "name" PromptForPassword False UserId "user" Password "mypassword"
PowerCubeUpdate "national" DatabaseInfo "Local;;"
" Database "name" SavePy "model.pyj"

Change optimization setting sample
This script changes the optimization setting of several cubes to use the older Categories option in cases where autopartitioning is either not supported or not desired. It then generates the .mdc files without saving the changes.
OpenMDL "c:\Great Outdoors (Admin).mdl"
CubeUpdate "Great Outdoors Sales" Optimize Categories
CubeUpdate "All Regions" Optimize Categories
CubeUpdate "Americas" Optimize Categories
CubeUpdate "Far East" Optimize Categories
CubeUpdate "Europe" Optimize Categories
CreateFiles

Automate CleanHouse sample
This script uses the MDL verb CleanHouse, which is equivalent to selecting the Clean House command on the Tools menu of the user interface, followed by a date in the format YYYYMMDD. This command removes all categories with a last use date less than the specified date.

Tip: To remove all categories in the model, specify a future date.

This script removes all categories in the Product dimension that have a last use date prior to November 1, 2006, and saves the model.
OpenPy "model.pyj"
CleanHouse Dimension "Product" 20061101 SaveMDL "model.mdl"

Convert model file formats sample
This script converts a legacy format (*.def) file into .mdl and then .pyj format.
OpenDef "\customer\defdat\01xbudgt.def"
SaveMDL "\customer\defdat\01xbudgt.mdl"
SavePy "\customer\defdat\01xbudgt.pyj"
Create cubes based on dimension views sample

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.

OpenMDL "c:\Outdoors.mdl"
ViewMake "Region Summary" Dimension "Locations" LevelSummary
Dimension "Locations" Drill "By Region" Levels "Region" Apex "Region"
ViewMake "North America" Dimension "Locations"
 Apex "Region" Summary "Europe" Summary "Far East"
ViewMake "Far East" Dimension "Locations"
Apex "Region" Summary "Europe" Summary "North America"
ViewMake "Europe" Dimension "Locations"
Apex "Region" Summary "Far East" Summary "North America"
CubeMake "All Regions" MdcFile "c:\All Regions.mdc"
DrillThrough True "c:\NorthAmerica.mdc" "c:\europe.mdc" "c:\fareast.mdc" "c:\all regions.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 sample

This script adds or changes the currency rates for German marks and French francs for November and December, 2006; that is, before these countries or regions entered the EMU.

OpenMDL "C:\Great Outdoors (Admin).mdl"
CurrencyUpdate "German Marks" Levels "Month" CurrencyRateList
EffectiveDate "20021101-20061130" ConversionRate 1.666
EffectiveDate "20021201-20061231" ConversionRate 1.777 EndList
CurrencyUpdate "French Francs" Levels "Month" CurrencyRateList
EffectiveDate "20021101-20061130" ConversionRate 1.666
EffectiveDate "20021201-20061231" ConversionRate 1.777 EndList

Disable incremental update sample

This script disables incremental update for a cube and generates the .mdc file for that cube.

OpenMDL "C:\Great Outdoors (Admin).mdl"
CubeUpdate "Europe" IncrementalUpdate False CreateFromCubes "Europe"
Chapter 5. Syntax conventions

The Cognos Transformer Developer Guide uses the following syntax and naming conventions:

- MDL verbs and keywords display in code or **bold** form. 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 the following example:
  ```mdl
  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 or 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

The Cognos Transformer Developer Guide uses the following data type conventions.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bignum</td>
<td>32-bit number between -2,147,483,648 and 2,147,483,647</td>
</tr>
<tr>
<td>number</td>
<td>16-bit number between -32,768 and 32,767</td>
</tr>
<tr>
<td>uns</td>
<td>unsigned number</td>
</tr>
<tr>
<td>string</td>
<td>character string, enclosed in single or double quotes, up to 256 characters in length</td>
</tr>
</tbody>
</table>

### Syntax example

The following example uses the MDL verb `CatMake`.
You must type this verb and then the identification for a category on which it operates, but all other parameters are optional, as indicated by their enclosing square brackets.

```
CatMakeobjCat [Dimension objDim] [Drill objDrillCat] [Levels objLevel] [Parent objLevel] [catopts] [ParentList objLevels] EndList
```

The following more complex example is based on the MDL keyword AllocationAdd. Options include Dimension, which must be followed by the dimension identifier, and Drill, which must be followed by a drill category identifier.

These mandatory parameters may be followed by one or more optional 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 then type Measure followed by the identification of a measure, and Type followed by the measure type. Note that type does not have the prefix obj. This indicates that it is not a Cognos Transformer object.

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

```
AllocationAdd [Dimension objDim] [Drill objDrillCat] [Levels objLevel] [Category objRootCat] [Category objCat] Measure objMeasure Typetype [AllocationMeasure objMeasure]
```

**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 6. MDL verbs

This section of the Developer Guide provides an alphabetical list of all the MDL verbs that you can use with IBM Cognos Transformer.

Before you begin, make sure you review Chapter 5, “Syntax conventions,” on page 61 and the topic that explains when to use object identifiers or category codes: “Locating objects uniquely” on page 13.

Verb types

Most objects in MDL have four verb types associated with them.

The four verb types include Add, Delete, Make, and Update; for example, CatAdd, CatDelete, CatMake, and CatUpdate.

- Add verbs add the object to the model. An error is issued if the object already exists. Examples for this verb type show you how to use it to create an object and specify its required settings.
- Delete verbs remove the object from the model. An error is issued if the object does not exist.
- Make verbs add the object if it does not already exist, and update the object if it does exist. Make verbs combine the functionality of both the Add and Update verbs. Examples for this verb type show the object definitions that are generated by Cognos Transformer when you save a model as an .mdl file.
- Update verbs update existing objects. An error is issued if the object does not exist.

Below is a list of the Cognos Transformer objects and the verbs you can use with each one.

<table>
<thead>
<tr>
<th>Cognos Transformer objects</th>
<th>Associated verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>NewModel</td>
</tr>
<tr>
<td>DataSource</td>
<td>DataSourceAdd,</td>
</tr>
<tr>
<td></td>
<td>DataSourceDelete,</td>
</tr>
<tr>
<td></td>
<td>DataSourceMake,</td>
</tr>
<tr>
<td></td>
<td>DataSourceUpdate,</td>
</tr>
<tr>
<td></td>
<td>SourceListUpdate,</td>
</tr>
<tr>
<td></td>
<td>CognosPackageAdd,</td>
</tr>
<tr>
<td></td>
<td>CognosPackageDelete,</td>
</tr>
<tr>
<td></td>
<td>CognosPackageMake,</td>
</tr>
<tr>
<td></td>
<td>CognosPackageUpdate</td>
</tr>
<tr>
<td>Filter</td>
<td>CognosPackageFilterAdd,</td>
</tr>
<tr>
<td></td>
<td>CognosPackageFilterMake,</td>
</tr>
<tr>
<td></td>
<td>CognosPackageFilterDelete,</td>
</tr>
<tr>
<td></td>
<td>CognosPackageFilterUpdate</td>
</tr>
<tr>
<td>Prompt</td>
<td>PromptAdd, PromptDelete, PromptMake,</td>
</tr>
<tr>
<td></td>
<td>PromptUpdate</td>
</tr>
<tr>
<td>Column</td>
<td>ColumnAdd, ColumnDelete, ColumnMake,</td>
</tr>
<tr>
<td></td>
<td>ColumnUpdate, ColumnListUpdate</td>
</tr>
<tr>
<td>Dimension</td>
<td>DimAdd, DimDelete, DimMake, DimUpdate,</td>
</tr>
<tr>
<td></td>
<td>DimensionListUpdate</td>
</tr>
<tr>
<td>Subdimension</td>
<td>SubDimRootMake, SubDimRootUpdate</td>
</tr>
<tr>
<td>Cognos Transformer objects</td>
<td>Associated verbs</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Root Category</td>
<td>RootCatMake, RootCatUpdate</td>
</tr>
<tr>
<td>Drill Category</td>
<td>DrillCatMake</td>
</tr>
<tr>
<td>Level</td>
<td>LevelAdd, LevelDelete, LevelMake, LevelMoveAfter, LevelMoveBefore, LevelNewDrill, LevelUpdate</td>
</tr>
<tr>
<td>Category</td>
<td>CatAdd, CatDelete, CatJoin, CatMake, CatMorph, CatMoveLevel, CatMoveVertical, CatUpdate, FilterCat</td>
</tr>
<tr>
<td>Special Category</td>
<td>SpecialCatAdd, SpecialCatDelete, SpecialCatMake, SpecialCatUpdate</td>
</tr>
<tr>
<td>View</td>
<td>ApexCat, ViewAdd, ViewDelete, ViewMake, ViewUpdate, ViewListUpdate</td>
</tr>
<tr>
<td>Measure</td>
<td>MeasureAdd, MeasureDelete, MeasureMake, MeasureListUpdate, MeasureUpdate, AllocationAdd</td>
</tr>
<tr>
<td>PowerCube</td>
<td>CubeAdd, CubeDelete, CubeMake, CubeUpdate, PowerCubeDelete, PowerCubeListUpdate, PowerCubeCustomViewListUpdate</td>
</tr>
<tr>
<td>PowerCube Group</td>
<td>CubeGroupAdd, CubeGroupDelete, CubeGroupMake, CubeGroupUpdate, PowerCubeDelete, PowerCubeListUpdate</td>
</tr>
<tr>
<td>PowerCube Group Cube</td>
<td>CubeGroupCubeAdd, CubeGroupCubeDelete, CubeGroupCubeListUpdate, CubeGroupCubeMake, CubeGroupCubeUpdate</td>
</tr>
<tr>
<td>Signon</td>
<td>SignonAdd, SignonListUpdate, SignonMake, SignonUpdate, SignonDelete, CognosPackageDatasourceConnectionAdd, CognosPackageDatasourceConnectionMake, CognosPackageDatasourceConnectionUpdate, CognosPackageDatasourceConnectionDelete</td>
</tr>
<tr>
<td>Dimension Calculation Definition</td>
<td>DimCalcDefAdd, DimCalcDefDelete, DimCalcDefMake, DimCalcDefUpdate</td>
</tr>
<tr>
<td>Currency Table</td>
<td>CurrencyTableAdd, CurrencyTableDelete, CurrencyTableMake, CurrencyTableUpdate</td>
</tr>
<tr>
<td>Currency</td>
<td>CurrencyAdd, CurrencyDelete, CurrencyMake, CurrencyUpdate</td>
</tr>
<tr>
<td>Cognos Transformer objects</td>
<td>Associated verbs</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Association</td>
<td>AssociationAdd, AssociationDelete, AssociationMake, AssociationUpdate, CurrencyTableAdd, CurrencyTableMake, CurrencyTableUpdate, DimensionAdd, DimensionMake, DimensionUpdate, LevelAdd, LevelMake, LevelUpdate, MeasureAdd, MeasureMake, MeasureUpdate</td>
</tr>
</tbody>
</table>

You can also use the following action verbs to manipulate Cognos Transformer objects.

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

## AllocationAdd

The `AllocationAdd` verb allocates summary data to a level, dimension, or category.

Its equivalent on the user interface is the **Allocation** tab on the **Level**, **Dimension**, and **Category** property sheets.

Only one of 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**. Cognos Transformer creates
this statement if the measure is created on the user interface or if the verb ModelEnsureCompleteness is used. Otherwise, you must manually add the statement when you create a measure.

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

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see “Locating objects uniquely” on page 13.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension objDim</td>
<td>Specifies the level, root category, or category. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Drill objDrillCat</td>
<td>Specifies the level, root category, or category. objDrillCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Levels objLevel</td>
<td>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.</td>
</tr>
<tr>
<td>Category objRootCat</td>
<td>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.</td>
</tr>
<tr>
<td>Category objCat</td>
<td>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.</td>
</tr>
<tr>
<td>Measure objMeasure</td>
<td>Specifies the measure that is being allocated. The measure can be identified by object name, object identifier, or both. This argument is mandatory.</td>
</tr>
<tr>
<td>Type type</td>
<td>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.</td>
</tr>
<tr>
<td>AllocationMeasure objMeasure</td>
<td>Specifies the measure used as the basis for the allocation proportions. objMeasure can be the object name or object identifier. This argument is optional.</td>
</tr>
</tbody>
</table>

**Example**

This is an example of the AllocationAdd statement that Cognos 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 Regions dimension (identified by its root category) according to the distribution of the Revenue measure.

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

### ApexCat

The ApexCat verb makes the selected category the highest category in a view. In PowerCubes that are created using this view, users will see only the apex category and its descendants.

To invoke the equivalent command on the user interface, open the left pane of the category viewer for the selected view and category and, from the Diagram menu, click Apex.

You cannot apply this verb to special categories.

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see “Locating objects uniquely” on page 13.

The syntax is as follows:

\[
\text{ApexCat } \text{objView} \{ \text{Dimension } \text{objDim} \} \{ \text{Category } \text{objCat} \} \\
\text{Category } \text{objRootCat} \{ \text{Category } \text{objDrillCat} \}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApexCat \text{objView}</td>
<td>Specifies the dimension view or custom security view \text{objView} in which the category is to be apexed. \text{objView} can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension \text{objDim}</td>
<td>Specifies the view. \text{objDim} can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Category \text{objCat}</td>
<td>Specifies a regular category as the apex. \text{objCat} can be the object name, object identifier, or both. One of the regular, root, or drill categories must be specified.</td>
</tr>
<tr>
<td>Category \text{objRootCat}</td>
<td>Specifies the root category as the apex, which has the effect of removing the apex. \text{objRootCat} can be the object name, object identifier, or both. One of the regular, root, or drill categories must be specified.</td>
</tr>
<tr>
<td>Category \text{objDrillCat}</td>
<td>Specifies a drill category as the apex. \text{objDrillCat} can be the object name, object identifier, or both. One of the regular, root, or drill categories must be specified.</td>
</tr>
</tbody>
</table>

**Example**

This example applies the verb to the Outdoor Products category in the dimension view View1.

ApexCat "View1" Dimension "Products" Category "Outdoor Products"
**AssociationAdd**

The AssociationAdd verb adds an association to a measure, level, dimension, category, or currency conversion table.

Its equivalent is adding an association on the property sheet for a measure, level, dimension, category, or currency conversion table.

The syntax is as follows:

```
AssociationAdd objAssoc assocopts
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssociationAdd</td>
<td>Specifies the name of the association <code>objAssoc</code> that you want to create and associate with an existing object. <code>objAssoc</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>assocopts</td>
<td>Specifies the type and details of the association <code>objAssoc</code>. For the complete list of options, see &quot;assocopts&quot; on page 170.</td>
</tr>
</tbody>
</table>

**Example**

This is an example of an AssociationAdd statement.

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

**AssociationDelete**

The AssociationDelete verb removes the association from a measure, level, dimension, category, or currency conversion table.

Its user interface equivalent is deleting an association from the property sheet for a measure, level, dimension, category, or currency conversion table.

The syntax is as follows:

```
AssociationDelete objAssoc
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssociationDelete</td>
<td>Specifies the name of the association <code>objAssoc</code> whose association you want to remove from an existing object. <code>objAssoc</code> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

This is an example of an AssociationDelete statement.

```
LevelMake 2979 "Branch Code" Drill 2969
AssociationDelete 6391 "Branch Code"
```
**AssociationMake**

The `AssociationMake` verb creates an association or updates an existing one.

Its user interface equivalent is creating or updating an association on the property sheet for a measure, level, dimension, category, or currency conversion table.

The syntax is as follows:

```
AssociationMake objAssoc assocopts
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AssociationMake objAssoc</code></td>
<td>Specifies the name of the association <code>objAssoc</code> that you want to create or update and associate with an existing object. <code>objAssoc</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><code>assocopts</code></td>
<td>Specifies the type and details of the association <code>objAssoc</code>. For the complete list of options, see &quot;assocopts&quot; on page 170.</td>
</tr>
</tbody>
</table>

**Example**

This is an example of an `AssociationMake` statement.

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

**AssociationUpdate**

The `AssociationUpdate` verb updates an existing association.

Its user interface equivalent is updating an association on the property sheet for a measure, level, dimension, category, or currency conversion table.

The syntax is as follows:

```
AssociationUpdate objAssoc assocopts
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AssociationUpdate objAssoc</code></td>
<td>Specifies the name of an existing association <code>objAssoc</code> that you want to update. <code>objAssoc</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><code>assocopts</code></td>
<td>Specifies the type and details of the association <code>objAssoc</code>. For the complete list of options, see &quot;assocopts&quot; on page 170.</td>
</tr>
</tbody>
</table>

**Example**

This is an example of an `AssociationUpdate` statement.
AutoDesign

The AutoDesign verb generates a preliminary model of dimensions and measures based on relationships that Cognos Transformer detects in a non-IBM Cognos data source.

In order to use the AutoDesign verb, your model must contain at least one data source and one column.

On the user interface, you may choose to run AutoDesign from the New Model wizard or the Tools menu. You can also set it to run automatically on model creation from the Preferences dialog box.

The syntax, which does not require any parameters, is as follows:

```
AutoDesign
```

CatAdd

The CatAdd verb creates a category at the specified location in the model.

The equivalent on the user interface is to open the category viewer (diagram), click the right side of an existing category, and drag to the right to add a category at that same level.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CatAdd objCat</td>
<td>Creates the category objCat. objCat must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Specifies a dimension for the category. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Drill objDrillCat</td>
<td>Specifies a drill category for the category. objDrillCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Levels objLevel</td>
<td>Specifies a level for the category. objLevel can be the object name or object identifier. This argument is mandatory.</td>
</tr>
<tr>
<td>Parent objCat</td>
<td>Specifies a parent category. objCat can be the object name or object identifier. This argument is mandatory.</td>
</tr>
<tr>
<td>catopts</td>
<td>Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see &quot;catopts&quot; on page 171.</td>
</tr>
</tbody>
</table>
**Argument Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParentList objCats</td>
<td>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.</td>
</tr>
</tbody>
</table>

For more information about creating categories, see "CatMake" on page 72.

**Example**

This is an example of a CatAdd statement that creates the Back Packs category:

```plaintext
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**

The CatDelete verb removes a category from the model.

The user interface equivalent is to select the category and, from the Edit menu, click Delete.

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Locating objects uniquely" on page 13.

The syntax is as follows:

```
CatDelete objCat [Dimension objDim]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CatDelete objCat</td>
<td>Deletes the category objCat. objCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Is specified, if necessary, to uniquely identify the category. objDim can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

This is an example of a CatDelete statement that removes the Germany category.

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

**CatJoin**

The CatJoin verb joins a category to its specified parent category. This verb is required when creating alternate drill-down paths, and is a useful means of making similar categories unique.

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Locating objects uniquely" on page 13. There is no user interface equivalent.
CatJoin is required when using alternate drill-down paths. It links a category in the convergence level with the parent in the alternate drill-down path.

The syntax is as follows:

```
CatJoin objCat [Dimension objDim] Parent objCat
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CatJoin</td>
<td>Specifies the category <code>objCat</code> that is to be joined. <code>objCat</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension</td>
<td>Is specified, if necessary, to uniquely identify the category. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Parent</td>
<td>Specifies the parent that is to be joined. <code>objCat</code> can be the object name or object identifier. This argument is mandatory.</td>
</tr>
</tbody>
</table>

**Example**

This is an example of a `CatJoin` statement that joins the category 1014 to its parent, Canada.

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

**CatMake**

The `CatMake` verb creates a new category or updates an existing one.

The equivalent to create a category on the user interface is to open the category viewer (diagram), click the right side of an existing category, and drag to the right to add a category at that same level. The equivalent to modify an existing category is to change the settings on its property sheet, as required.

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

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Locating objects uniquely" on page 13.

The syntax is as follows:

```
CatMake objCat [Dimension objDim] [Drill objDrillCat Levels [objlevel] [Parent objlevel] [catopts] [ParentList objlevels EndList]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CatMake</td>
<td>Creates the category <code>objCat</code> or modifies it if it exists. <code>objCat</code> can be the object name, object identifier, or both. Include the object name if the category does not exist.</td>
</tr>
<tr>
<td>Dimension</td>
<td>When creating a category, this specifies its dimension. When updating a category, this is used, if necessary, to uniquely identify it. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Argument</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Drill <strong>objDrillCat</strong></td>
<td>When creating a category, this specifies its drill category. This argument is not necessary when updating a category. <strong>objDrillCat</strong> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Levels <strong>objLevel</strong></td>
<td>When creating a category, this specifies its level. This argument is not necessary when updating a category. <strong>objLevel</strong> can be the object name or object identifier.</td>
</tr>
<tr>
<td>Parent <strong>objCat</strong></td>
<td>When creating a category, this specifies its parent category. This argument is not necessary when updating a category. <strong>objCat</strong> can be the object name, or object identifier.</td>
</tr>
<tr>
<td><em>catopts</em></td>
<td>Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see &quot;catopts&quot; on page 171. If the category exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>ParentList <strong>objCats</strong></td>
<td>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. <strong>objCats</strong> is a list of one or more parent categories, each identified by object name or object identifier.</td>
</tr>
</tbody>
</table>

**Example**

This is an example of a CatMake statement that 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**

The CatMorph verb changes the type of the specified category.

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see “Locating objects uniquely” on page 13. There is no user interface equivalent.

CatMorph exists so that MDL is a complete representation of a Cognos Transformer model when a category is changed. For example, when a leaf category acquires a child it changes (or "morphs") 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.

The syntax is as follows:

CatMorph **objCat** [Dimension **objDim**] CatType **type**
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CatMorph objCat</td>
<td>Specifies the category objCat that is to be changed. objCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Is specified, if necessary, to uniquely identify the category. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>CatType type</td>
<td>Specifies the category type. The type can be Leaf, Special, Parent, Drill, or Root. The type is mandatory.</td>
</tr>
</tbody>
</table>

**CatMoveVertical**

The CatMoveVertical verb moves a category to a new position within a level.

The user interface equivalent is to open the category viewer and drag the category from one position to another within a level.

This verb does not change the object identifier of the category. Rather, it determines how the category displays in the dimension viewer and its position in the .mdc file, which affects how it is viewed in the reporting components.

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see “Locating objects uniquely” on page 13.

The syntax is as follows:

CatMoveVertical objCat [Dimension objDim] Parent objLevel Sibling objCat

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CatMoveVertical objCat</td>
<td>Specifies the category that is to be moved. objCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Is specified, if necessary, to uniquely identify the category. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Parent objLevel</td>
<td>Specifies the parent level of the category in its new position. The parent must be specified even if it remains the same. objLevel can be the object name or object identifier.</td>
</tr>
<tr>
<td>Sibling objCat</td>
<td>Specifies the sibling category that becomes the next category below the moved category. objCat can be the object name or object identifier. This argument is mandatory.</td>
</tr>
</tbody>
</table>

**Example**

This is an example of a CatMoveVertical statement that moves the category Sunblock to the position above the category Tents.
**CatUpdate**

The `CatUpdate` verb updates an existing category.

Its user interface equivalent is to modify the required settings on the `Category` property sheet.

For more information about updating categories, see “CatMake” on page 72.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CatUpdate objCat</strong></td>
<td>Specifies the category to update. <code>objCat</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Dimension objDim</strong></td>
<td>Is specified, if necessary, to uniquely identify the category. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>catopts</strong></td>
<td>Optional parameters that describe the category in greater detail. For the complete list of options, see “catopts” on page 171. Previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td><strong>ParentList objCats</strong></td>
<td>Specifies parents of the category. This can be used to add one or more new parents for the category. <code>objCats</code> are the added parent categories, each identified by its object name or object identifier.</td>
</tr>
</tbody>
</table>

**CatUpdateAll**

The `CatUpdateAll` verb changes the last use value of all categories in the model to the current date. There is no user interface equivalent that allows you to globally change this setting on the `General` tab of all `Category` property sheets.

The syntax is as follows:

```
CatUpdateAll
```

**CleanHouse**

The `CleanHouse` verb deletes categories with a last use date that is earlier than the date specified.

The equivalent on the user interface is the `Clean House` command on the `Tools` menu.

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 on the `Category` property sheet in the user interface.
The syntax is as follows:

```
CleanHouse [Dimension objDim] date
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension</td>
<td>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.</td>
</tr>
<tr>
<td>date</td>
<td>Specifies the date, in the format YYYYMMDD. To delete all categories, specify a future date.</td>
</tr>
</tbody>
</table>

**Example**

This example removes all categories in the **Products** dimension with a **Last Use** date prior to 20070101 (January 1, 2007).

```
CleanHouse Dimension "Products" 20070101
```

**CognosPackageAdd**

The `CognosPackageAdd` verb creates a package or report data source definition in the model. The package or report definition is a reference to a package or report that is defined in IBM Cognos. You can then create a data source definition that has columns based on items that are selected from the package or report.

If you include a data source ID for the new package or report in the syntax, and the object already exists, you will receive an error message.

To insert a dimension from a package (the **Insert Dimension From Package** command on the **Edit** menu), you must first add the package and data source and then add a dimension from the data source. For more information about adding a dimension from a data source, see "DimAdd" on page 116.

The syntax for a query based on an IBM Cognos package or report is as follows:

```
CognosPackageAdd objDataSource [cognospackage_opts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objDataSource</td>
<td>Creates the data source objDataSource. objDataSource must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td>cognospackage_opts</td>
<td>Optional parameters that can describe the package or report in greater detail. For the complete list of options, see &quot;cognospackageopts&quot; on page 181.</td>
</tr>
</tbody>
</table>

**Example**

This example adds the IBM Cognos package data source Go Sales and Retailers to the model.
CognosPackageAdd 103 "GO Sales and Retailers" SourceType Package
SourcePath "/content/package[@name='GO Sales and Retailers']"
PackageTimeStamp "/content/package[@name='GO Sales and Retailers']/model[@name='model']"

This example adds the IBM Cognos report database GSR_rpt_1 to the model.
CognosPackageAdd 115 "GSR_rpt_1" SourceType Report SourcePath "
/content/package[@name='GO Sales and Retailers']/report[@name='GSR_rpt_1']"
"PackageTimeStamp"/content/package[@name='GO Sales and Retailers']/
/report[@name='GSR_rpt_1']"

CognosPackageDelete

The CognosPackageDelete verb removes a package or report data source from the model.

The user interface equivalent is to click **Delete** on the **Edit** menu when a data source is selected.

For more information about data sources, see the Cognos Transformer **User Guide** and the Framework Manager **User Guide**.

The syntax is as follows:
CognosPackageDelete objDataSource

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CognosPackageDelete objDataSource</td>
<td>Deletes the data source <strong>objDataSource</strong>. <strong>objDataSource</strong> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

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

CognosPackageMake

The CognosPackageMake verb creates a package or report data source definition in the model. If one already exists, it is updated. If one does not exist, it is added. The package or report definition is a reference to a package or report that is defined in IBM Cognos. You can then create a data source definition which has columns based on items that are selected from the package or report.

**Note:** To insert a dimension from a package (the **Insert Dimension From Package** command on the **Edit** menu, you must first add the package and data source and then add a dimension from the data source. For more information about adding a dimension from a data source, see "**DimAdd**" on page 116.

The syntax for a query based on an IBM Cognos package or report is as follows:
CognosPackageMake objDataSource [cognospackage_opts]
**CognosPackageMake**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CognosPackageMake</strong></td>
<td>Creates the data source <strong>objDataSource</strong>. <strong>objDataSource</strong> must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td>cognospackage_opts</td>
<td>Optional parameters that can describe the package or report in greater detail. For the complete list of options, see &quot;cognospackageopts&quot; on page 181.</td>
</tr>
</tbody>
</table>

**Example**

This example adds the IBM Cognos package data source Go Sales and Retailers to the model.

```
CognosPackageMake 103 "GO Sales and Retailers" SourceType Package
SourcePath "/content/package[@name='GO Sales and Retailers']" PackageTimeStamp "/content/package[@name='GO Sales and Retailers']/model[@name='model']"
```

This example adds the IBM Cognos report data base GSR_rpt_1 to the model.

```
CognosPackageMake 115 "GSR_rpt_1" SourceType Report
SourcePath "/content/package[@name='GO Sales and Retailers']/report[@name='GSR_rpt_1']"
PackageTimeStamp "/content/package[@name='GO Sales and Retailers']/report[@name='GSR_rpt_1']"
```

**CognosPackageUpdate**

The **CognosPackageUpdate** verb modifies a package or report data source definition in the model. The package or report definition is a reference to a package or report that is defined in IBM Cognos. You can then create a data source definition which has columns based on items that are selected from the package or report.

**Note:**
- If you include a data source ID for the new package or report in the syntax, and the object doesn’t exist, you will receive an error message.
- To insert a dimension from a package (the **Insert Dimension From Package** command on the **Edit** menu, you must first add the package and data source and then add a dimension from the data source. For more information about adding a dimension from a data source, see "DimAdd" on page 116.

The syntax for a query based on an IBM Cognos package or report is as follows:

```
CognosPackageUpdate objDataSource [cognospackage_opts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CognosPackageUpdate</strong></td>
<td>Creates the data source <strong>objDataSource</strong>. <strong>objDataSource</strong> must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td>cognospackage_opts</td>
<td>Optional parameters that can describe the package or report in greater detail. For the complete list of options, see &quot;cognospackageopts&quot; on page 181.</td>
</tr>
</tbody>
</table>
**Example**

This example modifies the IBM Cognos package data source Go Sales and Retailers.

CognosPackageUpdate 103 "GO Sales and Retailers" SourceType
Package SourcePath "/content/package[@name='GO Sales and Retailers']" PackageTimeStamp "/content/package[@name='GO Sales and Retailers']/model[@name='model']"

This example modifies the IBM Cognos report database GSR_rpt_1.

CognosPackageUpdate 115 "GSR_rpt_1" SourceType Report SourcePath "/content/package[@name='GO Sales and Retailers']/report[@name='GSR_rpt_1']" PackageTimeStamp "/content/package[@name='GO Sales and Retailers']/report[@name='GSR_rpt_1']"

**CognosPackageDatasourceConnectionAdd**

The CognosPackageDatasourceConnectionAdd verb defines the data source signon to be used when the data source has an ambiguous signon: multiple connections, each with multiple signons.

**Example**

The equivalent on the user interface is to define a Cognos Transformer data source signon when prompted.

The syntax is as follows:

```
CognosPackageDatasourceConnectionAdd "Content Store data source name"
CognosPackageConnection "connection" CognosPackageConnectionSignon "ContentStore signon name" Database Transformer data source ID 
"Transformerdatasource signon name"
CognosPackageAlwaysUseTransformerSignon {True|False}
CognosPackagePowercubeSource {True|False}
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CognosPackageDatasourceConnectionAdd &quot;Content Store data source name&quot;</td>
<td>Specifies that a Cognos Transformer signon is added for the data source defined in Content Manager.</td>
</tr>
<tr>
<td>CognosPackageConnection &quot;connection&quot;</td>
<td>Specifies the connection to the data source defined in Content Manager.</td>
</tr>
<tr>
<td>CognosPackageConnectionSignon &quot;Content Store signon name&quot;</td>
<td>Specifies the signon name defined in Content Manager.</td>
</tr>
<tr>
<td>Database Transformer data source ID &quot;Transformer data source signon name&quot;</td>
<td>Specifies the Cognos Transformer data source ID and new Cognos Transformer data source signon name.</td>
</tr>
<tr>
<td>CognosPackageAlwaysUseTransformerSignon {True</td>
<td>False}</td>
</tr>
<tr>
<td>CognosPackagePowercubeSource {True</td>
<td>False}</td>
</tr>
</tbody>
</table>
This example adds the Cognos Transformer signon "cogtr_signon" for the
oracle_gosales data source with data source connection "connection1" and signon
"signon1" defined in Content Manager. Queries will always use the "cogtr_signon"
signon. The data source is not an IBM Cognos PowerCube.

```
CognosPackageDatasourceConnectionAdd "oracle_gosales"
CognosPackageConnection "connection1"
CognosPackageConnectionSignon "signon1" Database 181 "cogtr_signon"
CognosPackageAlwaysUseTransformerSignon True CognosPackagePowercubeSource False
```

### CognosPackageDatasourceConnectionDelete

The CognosPackageDatasourceConnectionDelete verb deletes a previously-created
Cognos Transformer data source signon.

The equivalent on the user interface is to click Delete Signon on the edit menu
when a signon is selected.

The syntax is as follows:

```
CognosPackageDatasourceConnectionDelete "Content Store data source name"
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CognosPackageDatasourceConnectionDelete</td>
<td>Specifies that a Cognos Transformer signon is deleted data source defined in Content Manager.</td>
</tr>
</tbody>
</table>

### Example

This example deletes the Cognos Transformer signon "cogtr_signon".

```
CognosPackageDatasourceConnectionDelete "oracle_gosales"
```

### CognosPackageDatasourceConnectionMake

The CognosPackageDatasourceConnectionMake verb creates the data source signon
to be used when the data source has an ambiguous signon: multiple connections,
each with multiple signons.

The equivalent on the user interface is to define a Cognos Transformer data source
signon when prompted.

The syntax is as follows:

```
CognosPackageDatasourceConnectionMake "Content Store data source name"
DataSource "Package ID or name"
CognosPackageConnection "connection"
CognosPackageConnectionSignon "ContentStore signon name"
Database Transformer data source ID
"Transformerdatasource signon name"
CognosPackageAlwaysUseTransformerSignon {True|False}
CognosPackagePowercubeSource {True|False}
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CognosPackageDatasourceConnectionMake</td>
<td>Specifies that a Cognos Transformer signon is created for the data source defined in Content Manager.</td>
</tr>
<tr>
<td>Argument</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DataSource</strong> &quot;Package ID or name&quot;</td>
<td>Specifies the ID or name of the package with which the object is associated.</td>
</tr>
<tr>
<td><strong>CognosPackageConnection</strong> &quot;connection&quot;</td>
<td>Specifies the connection to the data source defined in Content Manager.</td>
</tr>
<tr>
<td><strong>CognosPackageConnectionSignon</strong> &quot;Content Store signon name&quot;</td>
<td>Specifies the signon name defined in Content Manager.</td>
</tr>
<tr>
<td><strong>Database</strong> Transformer data source ID &quot;Transformer data source signon name&quot;</td>
<td>Specifies the Cognos Transformer data source ID and the new Cognos Transformer signon name. This argument is optional and appears when the package is associated with a Cognos Transformer signon.</td>
</tr>
<tr>
<td><strong>CognosPackageAlwaysUseTransformerSignon</strong> [True</td>
<td>False]</td>
</tr>
<tr>
<td><strong>CognosPackagePowercubeSource</strong> [True</td>
<td>False]</td>
</tr>
</tbody>
</table>

**Example**

This example creates the Cognos Transformer signon "cogtr_signon" for the oracle_gosales data source with data source connection "connection1" and signon "signon1" defined in Content Manager. Queries will always use the "cogtr_signon" signon. The data source is not a PowerCube.

```mdl
CognosPackageDatasourceConnectionMake "oracle_gosales" DataSource "Package"
CognosPackageConnection "connection1"
CognosPackageConnectionSignon "signon1" Database 181 "cogtr_signon"
CognosPackageAlwaysUseTransformerSignon True
CognosPackagePowercubeSource False
```

This example creates the data source signon when no Cognos Transformer signon is specified.

```mdl
CognosPackageDatasourceConnectionMake "great_outdoors_sales"
DataSource "Package"
CognosPackageConnectionSignon "great_outdoors_sales_2" False
CognosPackagePowercubeSource False
```

This example creates the data source signon when Cognos Transformer signon is specified.

```mdl
CognosPackageDatasourceConnectionMake "great_outdoors_sales"
DataSource "Package"
CognosPackageConnection "great_outdoors_sales"
CognosPackageConnectionSignon "great_outdoors_sales_2"
Database "TransformerSignon"
CognosPackageAlwaysUseTransformerSignon False
CognosPackagePowercubeSource False
```
**CognosPackageDatasourceConnectionUpdate**

The CognosPackageDatasourceConnectionUpdate verb updates the data source signon created when the data source has an ambiguous signon: multiple connections, each with multiple signons.

The equivalent on the user interface is to modify the Cognos Transformer data source signon property sheet.

The syntax is as follows:

```
CognosPackageDatasourceConnectionUpdate "Content Store data source name"
CognosPackageConnection "connection"
CognosPackageConnectionSignon "ContentStore signon name"
Database Transformer data source ID "Transformerdata source signon name"
CognosPackageAlwaysUseTransformerSignon {True|False}
CognosPackagePowercubeSource {True|False}
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CognosPackageDatasourceConnectionUpdate</td>
<td>Specifies that a Cognos Transformer signon is updated for the data source defined in Content Manager.</td>
</tr>
<tr>
<td>&quot;Content Store data source name&quot;</td>
<td></td>
</tr>
<tr>
<td>CognosPackageConnection</td>
<td>Specifies the connection to the data source defined in Content Manager.</td>
</tr>
<tr>
<td>&quot;connection&quot;</td>
<td></td>
</tr>
<tr>
<td>CognosPackageConnectionSignon</td>
<td>Specifies the signon name defined in Content Manager.</td>
</tr>
<tr>
<td>&quot;Content Store signon name&quot;</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>Specifies the Cognos Transformer data source ID and Cognos Transformer data source signon name.</td>
</tr>
<tr>
<td>Transformer data source ID &quot;Transformerdata source signon name&quot;</td>
<td></td>
</tr>
<tr>
<td>CognosPackageAlwaysUseTransformerSignon</td>
<td>Specifies whether queries use the new Cognos Transformer signon or the signon defined in Content Manager.</td>
</tr>
<tr>
<td>{True</td>
<td>False}</td>
</tr>
<tr>
<td>CognosPackagePowercubeSource</td>
<td>Specifies whether the data source is a PowerCube.</td>
</tr>
<tr>
<td>{True</td>
<td>False}</td>
</tr>
</tbody>
</table>

**Example**

This example updates the Cognos Transformer signon "cogtr_signon" for the oracle_gosales data source with data source connection "connection1" and signon "signon1" defined in Content Manager. Queries will no longer always use the "cogtr_signon" signon. The data source is not a PowerCube.

```
CognosPackageDatasourceConnectionUpdate "oracle_gosales"
CognosPackageConnection "connection1"
CognosPackageConnectionSignon "signon1"
Database 181 "cogtr_signon"
CognosPackageAlwaysUseTransformerSignon False
CognosPackagePowercubeSource False
```
**CognosPackageFilterAdd**

The CognosPackageFilterAdd verb imports filters defined within IBM Cognos package or report data sources.

The syntax is as follows:

```
CognosPackageFilterAdd "filter name"
DATASOURCE data source ID
CognosPackageFilterRef "filter reference ID"
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CognosPackageFilterAdd &quot;filter name&quot;</td>
<td>Specifies the name of the filter to be added in the Cognos Transformer model.</td>
</tr>
<tr>
<td>DATASOURCE data source ID</td>
<td>Specifies the data source ID.</td>
</tr>
<tr>
<td>CognosPackageFilterRef &quot;filter reference ID&quot;</td>
<td>Specifies the reference ID of the filter to be added in the Cognos Transformer model.</td>
</tr>
</tbody>
</table>

**Example**

The following example adds the package filter “Tents1” in the Cognos Transformer model, using the IBM Cognos filter reference `oracle_gosales.Tents` from the IBM Cognos data source with ID 105.

```
CognosPackageFilterAdd "Tents1" DATASOURCE 105
CognosPackageFilterRef "[oracle_gosales].[Tents]"
```

**CognosPackageFilterDelete**

The CognosPackageFilterDelete verb deletes filters imported with IBM Cognos package or report data sources.

The syntax is as follows:

```
CognosPackageFilterDelete "filter name"
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CognosPackageFilterDelete &quot;filter name&quot;</td>
<td>Specifies the name of the filter to be deleted from the Cognos Transformer model.</td>
</tr>
</tbody>
</table>

**Example**

The following example deletes the filter “Tents1” from the Cognos Transformer model.

```
CognosPackageFilterDelete "Tents1"
```

**CognosPackageFilterMake**

The CognosPackageFilterMake verb imports filters defined within IBM Cognos package or report data sources.

The syntax is as follows:
**CognosPackageFilterMake**  
*“filter name”*  
**DATASOURCE**  data source ID  
**CognosPackageFilterRef**  *“filter reference ID”*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CognosPackageFilterMake</strong></td>
<td><em>“filter name”</em> Specifies the name of the filter to be added in the Cognos Transformer model.</td>
</tr>
<tr>
<td><strong>DATASOURCE</strong></td>
<td>data source ID Specifies the data source ID.</td>
</tr>
<tr>
<td><strong>CognosPackageFilterRef</strong></td>
<td><em>“filter reference ID”</em> Specifies the reference ID of the filter to be added in the Cognos Transformer model.</td>
</tr>
</tbody>
</table>

**Example**

The following example creates the package filter “Tents1” in the Cognos Transformer model, using the IBM Cognos filter reference *oracle_gosales.Tents* from the IBM Cognos data source with ID 105.

```plaintext
CognosPackageFilterMake "Tents1" DATASOURCE 105  
CognosPackageFilterRef "[oracle_gosales].[Tents]"
```

**CognosPackageFilterUpdate**

The CognosPackageFilterUpdate verb updates filters imported with IBM Cognos package or report data sources.

The syntax is as follows:

**CognosPackageFilterUpdate**  *“filter name”*  
**DATASOURCE**  data source ID  
**CognosPackageFilterRef**  *“filter reference ID”*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CognosPackageFilterUpdate</strong></td>
<td><em>“filter name”</em> Specifies the name of the filter to be updated in the Cognos Transformer model.</td>
</tr>
<tr>
<td><strong>DATASOURCE</strong></td>
<td>data source ID Specifies the data source ID.</td>
</tr>
<tr>
<td><strong>CognosPackageFilterRef</strong></td>
<td><em>“filter reference ID”</em> Specifies the reference ID of the filter to be added in the Cognos Transformer model.</td>
</tr>
</tbody>
</table>

**Example**

The following example updates the filter “Tents1” to use the package filter “*[oracle_gosales].[Tents2]*”.

```plaintext
CognosPackageFilterUpdate "Tents1" DATASOURCE 105  
CognosPackageFilterRef "[oracle_gosales].[Tents2]"
```
**ColumnAdd**

The `ColumnAdd` verb adds a column to a data source.

Its user interface equivalent is the **Insert Column** command on the **Edit** menu.

For more information about creating columns, see the "ColumnMake" section on page 86.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnAdd objCol</td>
<td>Creates the column <code>objCol</code>. <code>objCol</code> must be the object name, and can include the object identifier.</td>
</tr>
<tr>
<td>DataSource objDataSource</td>
<td>Places the column within a data source. <code>objDataSource</code> can be the data source object name, object identifier, or both.</td>
</tr>
<tr>
<td>Origin origin</td>
<td>Specifies the origin. <code>origin</code> must be one of <strong>Source</strong>, <strong>Generated</strong>, <strong>Calculated</strong>, or <strong>Manual</strong>. This argument is mandatory.</td>
</tr>
<tr>
<td>Offset offset</td>
<td>Specifies the amount by which the column is offset. <code>offset</code> can be 0 or greater. This argument is mandatory.</td>
</tr>
<tr>
<td>colopts</td>
<td>Parameters that describe the column in greater detail. All are optional except <code>Column objCol</code>, which is mandatory. Some options are set by default if you do not set them. For the complete list of options, see &quot;colopts&quot; on page 182.</td>
</tr>
</tbody>
</table>

**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**

The `ColumnDelete` verb removes a column.

The user interface equivalent is to click **Delete** on the **Edit** menu when a column is selected.

The syntax is as follows:

```
ColumnDelete objCol [DataSource objDataSource]
```
**Argument Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ColumnDelete objCol</strong></td>
<td>Specifies the column that is to be deleted. <em>objCol</em> must be the object name, and can include its object identifier.</td>
</tr>
<tr>
<td><strong>DataSource objDataSource</strong></td>
<td>Is specified, if necessary, to uniquely identify the column. <em>objDim</em> can be the dimension object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the column Branch Code.

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

---

**ColumnListUpdate**

The ColumnListUpdate verb reorders the columns in a data source.

The equivalent operation on the user interface is to drag and drop the columns as required.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataSource objDataSource</strong></td>
<td>Specifies the data source to be reordered. <em>objDataSource</em> can be the data source object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>StartList objCols</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

**Example**

This example reorders the columns in the Excel data source "All Staff Count" 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**

The ColumnMake verb creates a column or updates an existing one.

The user interface equivalent, if the column exists, is to modify its Column property sheet. For a new column, click Insert Column on the Edit menu.

Column names cannot contain an at sign (@).
The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnMake objCol</td>
<td>Creates the column objCol or modifies it if it exists. objCol can be the column object name, object identifier, or both. Include the object name if the column does not exist.</td>
</tr>
<tr>
<td>DataSource objDataSource</td>
<td>When creating a column, this specifies the data source and where it is to be placed. When updating a column this is used, if necessary, to uniquely identify it. objDataSource can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Origin origin</td>
<td>Specifies the origin. origin must be one of Source, Generated, Calculated, or Manual. This argument is mandatory.</td>
</tr>
<tr>
<td>Offset offset</td>
<td>Specifies the amount by which the column is offset. offset can be equal or greater than 0. This argument is mandatory.</td>
</tr>
<tr>
<td>colopts</td>
<td>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 do not set them. For the complete list of options, see [colopts] on page 182. If the column exists, previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

**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**

The ColumnUpdate verb updates an existing column.

The equivalent action on the user interface is to modify the Column property sheet.

For more information about updating columns, see "ColumnMake" on page 86.

The syntax is as follows:

ColumnUpdate objCol [DataSource objDataSource] [colopts]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnUpdate objCol</td>
<td>Specifies the column to update. objCol can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Argument</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DataSource</strong></td>
<td><em>objDataSource</em> Is specified, if necessary, to uniquely identify the column. <em>objDataSource</em> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>colopts</strong></td>
<td>Optional parameters that describe the column in greater detail. For the complete list of options, see &quot;colopts&quot; on page 182. Previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

**Example**

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

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

**CreateColumns**

The `CreateColumns` verb creates columns for data sources. There is no user interface equivalent because columns are created automatically in the Data Sources list on the user interface.

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

The syntax is as follows:

`CreateColumns objQueries`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateColumns</strong></td>
<td><em>objQueries</em> <em>objQueries</em> 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.</td>
</tr>
</tbody>
</table>

**Example**

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

`CreateColumns "Products (CSV)"`

**CreateFiles**

The `CreateFiles` verb creates an .mdc file for each PowerCube in the model.

The user interface equivalent is the Create PowerCube toolbar button or the Create PowerCubes command on the Run menu.

For other ways to create and update .mdc files, see "CreateFromCubes" on page 89, "CreateFromQueries" on page 89, and "UpdatePowerCubes" on page 157.

The syntax is as follows:

`CreateFiles`
**Example**

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

CreateFiles

**CreateFromCubes**

The CreateFromCubes verb creates an .mdc file for each specified PowerCube.

The user interface equivalent is to click **Create Selected PowerCubes** on the Run menu when one or more PowerCubes are selected.

For other ways to create and update .mdc files, see "CreateFiles" on page 88, "CreateFromQueries," and "UpdatePowerCubes" on page 157.

The syntax is as follows:

\[
\text{CreateFromCubes} \ [\text{objCubes}] \\
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objCubes</td>
<td>Specifies one or more PowerCubes for which .mdc files are to be created. PowerCubes can be identified by object name or object identifier.</td>
</tr>
</tbody>
</table>

**Example**

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

CreateFromCubes "cube1"

**CreateFromQueries**

The CreateFromQueries verb creates an .mdc file for each PowerCube in the model by retrieving data from only specific data sources. There is no user interface equivalent.

If no data sources are specified, nothing happens.

If one or more of the specified data sources are not set for **PowerCube Creation** on the **General** tab of the **Data Source** property sheet in the user interface, an error is issued. This can occur with structural data sources because they do not contain transactional data.

For other ways to create and update .mdc files, see "CreateFromCubes," "CreateFiles" on page 88, and "UpdatePowerCubes" on page 157.

The syntax is as follows:

\[
\text{CreateFromQueries} \ [\text{objQueries}] \\
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objQueries</td>
<td>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.</td>
</tr>
</tbody>
</table>
Example

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

CreateFromQueries "All Staff Count(Excel)"

**CubeAdd**

The `CubeAdd` verb creates the PowerCube object in the Cognos Transformer model.

To create or update an .mdc file, see ["CreateFiles" on page 88](#).

The user interface equivalent is to select the **Insert** option on the **Edit** menu when the **PowerCubes** pane is selected.

For more information about creating PowerCubes, see ["CubeMake" on page 99](#).

The syntax is as follows:

```
CubeAdd objCube [powercubeopts][ DimensionView objView]
[MeasureInclude objMeasure meaopt]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CubeAdd</strong> objCube</td>
<td>Creates the PowerCube <code>objCube</code>. <code>objCube</code> must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td>powercubeopts</td>
<td>Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see <a href="#">&quot;powercubeopts&quot; on page 216</a>.</td>
</tr>
<tr>
<td><strong>DimensionView</strong> objDim objView</td>
<td>Specifies the dimension and dimension view. <code>objDim</code> can be the object name or object identifier. <code>objView</code> can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in <code>DimensionView &quot;Years&quot; &quot;All Categories&quot;</code> <code>DimensionView &quot;Products&quot; &quot;All Categories&quot;</code></td>
</tr>
<tr>
<td><strong>MeasureInclude</strong> objMeasure meaopt</td>
<td>Specifies the measure. <code>objMeasure</code> can be the object name, object identifier, or both. <code>meaopt</code> must be one of Yes, Indirect, or No. Repeat this argument to specify multiple measures, as in <code>MeasureInclude 159 Yes</code> <code>MeasureInclude 175 No</code></td>
</tr>
</tbody>
</table>

**Example**

This example creates the **Great Outdoors Company Sales** PowerCube object.

`CubeAdd "Great Outdoors Company Sales" MdFile "c:\outdoors.mdc"`

**CubeDelete**

The `CubeDelete` verb removes the PowerCube object from the Cognos Transformer model.
The user interface equivalent is to click **Delete** on the **Edit** menu when a cube is selected.

The **CubeDelete** verb only deletes the PowerCube object, not the .mdc file.

The syntax is as follows:

```
CubeDelete objCube
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeDelete objCube</td>
<td>Specifies the PowerCube to be deleted. <code>objCube</code> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the PowerCube called Great Outdoors Company Sales.

```
CubeDelete "Great Outdoors Company Sales"
```

**CubeGroupAdd**

The **CubeGroupAdd** verb creates the PowerCubeGroup object in the Cognos Transformer model.

To create or update an .mdc file, see “CreateFiles” on page 88.

The user interface equivalent is to select the **Insert** option from the **Edit** menu when the PowerCubes pane is selected, and then to specify a dimension and level on the Cube Group tab.

If you do not include the options **SegmenterLevel** and **SegmenterDimension**, the **CubeGroupAdd** verb creates a PowerCube rather than a PowerCubeGroupCube object. For more information, see “powercubeopts” on page 216.

For more information about updating cube groups, see “CubeGroupMake” on page 97.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeGroupAdd objCubeGroup</td>
<td>Specifies the cube group that is to be created. <code>objCubeGroup</code> must be the object name, and can also be the object identifier.</td>
</tr>
<tr>
<td>powercubeopts</td>
<td>Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see “powercubeopts” on page 216.</td>
</tr>
<tr>
<td>Argument</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DimensionView</strong></td>
<td>Specifies the dimension and dimension view. <code>objDim</code> can be the object name or object identifier. <code>objView</code> can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in DimensionView &quot;Years&quot; &quot;All Categories&quot; DimensionView &quot;Products&quot; &quot;All Categories&quot;</td>
</tr>
<tr>
<td><strong>MeasureInclude</strong></td>
<td>Specifies the measure. <code>objMeasure</code> can be the object name, object identifier, or both. <code>meaopt</code> must be one of Yes, Indirect or No. Repeat this argument to specify multiple measures, as in MeasureInclude 159 Yes MeasureInclude 175 No</td>
</tr>
</tbody>
</table>

**Example**

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

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

**CubeGroupCubeAdd**

The `CubeGroupCubeAdd` verb creates the PowerCube object for a `PowerCubeGroup` object in the Cognos Transformer model.

**Example**

To create or update an .mdc file, see "CreateFiles" on page 88.

The user interface equivalent is to select the **Insert** option from the **Edit** menu when the PowerCubes pane is selected, and then to provide a dimension and level on the **Cube Group** tab.

In the user interface, a PowerCubeGroupCube object is created for every category under the Drill category in the dimension, unless some categories have been excluded, 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" on page 94.

The syntax is as follows:

```plaintext
CubeGroupCubeAdd objCubeGroupCube
CubeGroup objCubeGroup [ Dimension objDim]
Code objCat [ powercubeopts]
```
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeGroupCubeAdd objCubeGroup</td>
<td>Creates the PowerCube group cube objCubeGroupCube, objCubeGroupCube must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td>CubeGroup objCubeGroup</td>
<td>Specifies a cube group for the PowerCube. objCubeGroup can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Is specified, if necessary, to uniquely identify the category. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Code objCat</td>
<td>Specifies the category on which the PowerCube is based. objCat can be the object name, object identifier, or both. This argument is mandatory.</td>
</tr>
<tr>
<td>powercubeopts</td>
<td>Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see &quot;powercubeopts&quot; on page 216.</td>
</tr>
</tbody>
</table>

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**

The `CubeGroupCubeDelete` verb removes the PowerCube from a PowerCubeGroupCube object in the Cognos Transformer model. It does not delete the .mdc file.

**Example**

The user interface equivalent is to click **Delete** on the **Edit** menu when a PowerCube in a cube group is selected.

This verb can only be used if the model contains a dimension view that apexes, cloaks, suppresses, or excludes the category at the apex of the cube.

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

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>objCubeGroupCube</code></td>
<td>Deletes the cube group <code>objCubeGroupCube</code>.</td>
</tr>
<tr>
<td></td>
<td><code>objCubeGroupCube</code> can be the object name, object</td>
</tr>
<tr>
<td></td>
<td>identifier, or both.</td>
</tr>
</tbody>
</table>

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

```plaintext
ViewMake "Environmental Line" Dimension "Products" ViewSecurity 0 Apex "Product Line" Suppressed "Environmental Line" CubeGroupCubeDelete "Environmental Line" CubeGroup "cubegroup1"
```

**CubeGroupCubeListUpdate**

The `CubeGroupCubeListUpdate` verb reorders the PowerCubes in a `PowerCubeGroup` object.

The user interface equivalent is to drag PowerCubes into the appropriate cube group.

The syntax is as follows:

```
CubeGroupCubeListUpdate CubeGroup `objCubeGroup` [StartList `objCubeGroupCubes` EndList]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CubeGroupCubeListUpdate</code> CubeGroup <code>objCubeGroup</code></td>
<td>Specifies the cube group. <code>objCubeGroup</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><code>StartList</code> <code>objCubeGroupCubes</code></td>
<td>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.</td>
</tr>
</tbody>
</table>

**Example**

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

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

**CubeGroupCubeMake**

The `CubeGroupCubeMake` verb creates a PowerCube in a `PowerCubeGroup` object, or updates it if it exists. It does so by defining a PowerCube group cube object in the Cognos Transformer model.

To create or update an .mdc file, see "CreateFiles" on page 88.
The user interface equivalent, if the cube already exists within a cube group, is to modify the Cube property sheet. For a new cube in a cube group, select the Insert option on the Edit menu when the PowerCubes pane is selected, and then provide a dimension and level on the Cube Group tab.

The cube group must exist before you can create a PowerCube for the group.

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Locating objects uniquely" on page 13.

The syntax is as follows:

```plaintext
CubeGroupCubeMake objCubeGroupCube [CubeGroup objCubeGroup][Dimension objDim] Code objCat [powercubeopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CubeGroupCubeMake</strong> objCubeGroupCube</td>
<td>Creates the PowerCube group cube <code>objCubeGroupCube</code> or modifies it if it exists. <code>objCubeGroupCube</code> can be the object name, object identifier, or both. Include the object name if the cube group cube does not exist.</td>
</tr>
<tr>
<td><strong>CubeGroup</strong> objCubeGroup</td>
<td>When creating a cube in a cube group, this specifies the cube group. <code>objCubeGroup</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Dimension</strong> objDim</td>
<td>Is specified, if necessary, to uniquely identify the category. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Code</strong> objCat</td>
<td>Specifies the category on which the cube is based. <code>objCat</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>powercubeopts</strong></td>
<td>Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see &quot;powercubeopts&quot; on page 216.</td>
</tr>
</tbody>
</table>

**Example**

This example defines the PowerCube group cube Outdoor Products.

```plaintext
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**

The `CubeGroupCubeUpdate` verb updates an existing PowerCube within a PowerCubeGroup object in the Cognos Transformer model. It does not modify existing .mdc files.

The user interface equivalent is to modify the Cube Group property sheet.
For more information about updating cubes in cube groups, see “CubeGroupCubeMake” on page 94.

The syntax is as follows:

```
CubeGroupCubeUpdate objCubeGroupCube [ 
  CubeGroup objCubeGroup] [powercubeopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeGroupCubeUpdate objCubeGroupCube</td>
<td>Specifies the cube group cube to update. <code>objCubeGroupCube</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>CubeGroup objCubeGroup</td>
<td>Specifies the cube group. <code>objCubeGroup</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>powercubeopts</td>
<td>Optional parameters that describe the cube in greater detail. For the complete list of options, see “powercubeopts” on page 216. Previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

**Example**

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

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

**CubeGroupDelete**

The `CubeGroupDelete` verb removes a `PowerCubeGroup` object from the Cognos Transformer model. It does not delete the .mdc file.

The user interface equivalent is to click **Delete** on the **Edit** menu when a cube group is selected.

The syntax is as follows:

```
CubeGroupDelete objCubeGroup
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeGroupDelete objCubeGroup</td>
<td>Specifies the cube group that is to be deleted. <code>objCubeGroup</code> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the `PowerCube` cubegroup1.

```
CubeGroupDelete "cubegroup1"
```
**CubeGroupMake**

The CubeGroupMake verb creates a PowerCubeGroup object or updates an existing one, by defining the PowerCube object in the Cognos Transformer model.

To create or update an .mdc file, see ["CreateFiles" on page 88](#).

The user interface equivalent, for an existing cube group, is to modify its Cube Group property sheet. For a new cube group, select the Insert option from the Edit menu when the PowerCubes pane is selected, and then provide a dimension and level on the Cube Group tab.

If you do not include the options SegmenterLevel and SegmenterDimension, the CubeGroupMake verb will create a PowerCube rather than a PowerCubeGroup object. For more information, see ["powercubeopts" on page 216](#).

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CubeGroupMake</strong></td>
<td>objCubeGroup creates the cube group <strong>objCubeGroup</strong> or modifies it if it exists. <strong>objCubeGroup</strong> can be the object name, object identifier, or both. Include the object name if the category does not exist.</td>
</tr>
<tr>
<td><em>powercubeopts</em></td>
<td>Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see <a href="#">&quot;powercubeopts&quot; on page 216</a>. If the cube exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td><strong>DimensionView</strong></td>
<td>objDim objView specifies the dimension and dimension view. <strong>objDim</strong> can be the object name or object identifier. <strong>objView</strong> can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in <code>DimensionView &quot;Years&quot; &quot;All Categories&quot; DimensionView &quot;Products&quot; &quot;All Categories&quot;</code></td>
</tr>
<tr>
<td><strong>MeasureInclude</strong></td>
<td>objMeasure meaopt specifies the measure. <strong>objMeasure</strong> can be the object name, object identifier, or both. <strong>meaopt</strong> must be one of Yes, Indirect, or No. Repeat this argument to specify multiple measures, as in <code>MeasureInclude &quot;Cost&quot; Yes MeasureInclude &quot;Revenue&quot; No</code></td>
</tr>
</tbody>
</table>

**Example**

This example defines the cube group cubegroup1. The cubes within this group are created from the Products dimension, at the Product Line level with a focus of Product Type.
CubeGroupUpdate verb updates an existing PowerCubeGroup object by modifying the PowerCube in the Cognos Transformer model. It does not affect existing .mdc files.

The user interface equivalent is to modify the Cube Group property sheet.

For more information about updating cube groups, see "CubeGroupMake" on page 97.

The syntax is as follows:

\texttt{CubeGroupUpdate \textit{objCubeGroup}} \ [\textit{powercubeopts}] \ [\textit{DimensionView} \textit{objDim} \textit{objView}] \ [\textit{MeasureInclude} \textit{objMeasure} \textit{meaopt}]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeGroupUpdate \textit{objCubeGroup}</td>
<td>Specifies the PowerCube group to update. \textit{objCubeGroup} can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>\textit{powercubeopts}</td>
<td>Optional parameters that describe the cube in greater detail. For the complete list of options, see &quot;powercubeopts&quot; on page 216. Previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>DimensionView \textit{objDim} \textit{objView}</td>
<td>Specifies the dimension and dimension view. \textit{objDim} can be the object name or object identifier. \textit{objView} can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in DimensionView &quot;Years&quot; &quot;All Categories&quot; DimensionView &quot;Products&quot; &quot;All Categories&quot;</td>
</tr>
<tr>
<td>MeasureInclude \textit{objMeasure} \textit{meaopt}</td>
<td>Specifies the measure. \textit{objMeasure} can be the object name, object identifier, or both. \textit{meaopt} must be one of Yes, Indirect, or No. Repeat this argument to specify multiple measures, as in MeasureInclude 159 Yes MeasureInclude 175 No</td>
</tr>
</tbody>
</table>
**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**

The CubeMake verb creates a PowerCube or updates an existing one by defining the PowerCube object in the Cognos Transformer model.

To create or update an .mdc file, see "CreateFiles" on page 88.

The user interface equivalent, if the cube exists, is to modify the Cube property sheet. For a new cube, when the PowerCubes pane is selected, select the Insert option on the Edit menu.

The syntax is as follows:

```
CubeMake objCube [powercubeopts] [DimensionView objDim objView][MeasureInclude objMeasuremeaopt]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeMake objCube</td>
<td>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.</td>
</tr>
<tr>
<td>powercubeopts</td>
<td>Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see &quot;powercubeopts&quot; on page 216. If the cube exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>DimensionView objDim objView</td>
<td>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 &quot;Years&quot; &quot;All Categories&quot; DimensionView &quot;Products&quot; &quot;All Categories&quot;</td>
</tr>
<tr>
<td>MeasureInclude objMeasure meaopt</td>
<td>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</td>
</tr>
</tbody>
</table>

---

**Example**

This example defines the PowerCube Great Outdoors Company Sales.
CubeMake "Great Outdoors Company Sales" MdcFile "c:\outdoors.mdc" Status OK
CubeCreation On Optimize Categories Compress False DatabaseInfo "Local;"
IncrementalUpdate False ServerCube False CubeStamp 860096730 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

The CubeUpdate verb updates an existing PowerCube by modifying the PowerCube object in the Cognos Transformer model.

To create or update an .mdc file, see "CreateFiles" on page 88.

The user interface equivalent is to modify the PowerCube property sheet.

For more information about updating PowerCubes, see "CubeMake" on page 99.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CubeUpdate objCube</td>
<td>Specifies the PowerCube to update. objCube can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>powercubeopts</td>
<td>Optional parameters that describe the cube in greater detail. For the complete list of options, see &quot;powercubeopts&quot; on page 216. If the cube exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>DimensionView objDim objView</td>
<td>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 &quot;Years&quot; &quot;All Categories&quot; DimensionView &quot;Products&quot; &quot;All Categories&quot;</td>
</tr>
<tr>
<td>MeasureInclude objMeasure meaopt</td>
<td>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</td>
</tr>
</tbody>
</table>
Example

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

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

CurrencyAdd

The CurrencyAdd verb adds a currency definition to the currency table for the model.

The user interface equivalent is to right-click a currency item in the currency table and then select Add New Currency.

The syntax is as follows.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrencyAdd objCurrency</td>
<td>Adds the currency <code>objCurrency</code> to the list of currencies in the currency table. <code>objCurrency</code> must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Is specified, if necessary, to uniquely identify the level <code>objLevel</code>. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Drill objDrillCat</td>
<td>Is specified, if necessary, to uniquely identify the date level <code>objLevel</code>. <code>objDrillCat</code> can be the drill category object name, object identifier, or both.</td>
</tr>
<tr>
<td>Levels objLevel</td>
<td>Specifies the level of the date category <code>objCat</code>. This indicates the granularity of the rate data, such as daily or monthly. <code>objLevel</code> can be the object name or object identifier. The level is mandatory.</td>
</tr>
<tr>
<td>currencyrecordopts</td>
<td>Optional parameters that describe the currency in greater detail. Some options are set by default if you do not set them. For the complete list of options, see &quot;currencyrecordopts&quot; on page 190</td>
</tr>
<tr>
<td>CurrencyRateList</td>
<td>Indicates the start of conversion rates.</td>
</tr>
</tbody>
</table>
Arguments Description

**EffectiveDate**  
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 country or region is part of the European Monetary Union (EMU).

**ConversionRate**  
`rate` is the exchange rate for that date. It can include decimals; for example, 3.42556.

**RateIsBySource**  
`RateIsBySource` indicates whether the conversion rate is sourced from an external data source or was manually entered. The default is `False`.

**CurrencyTableType**  
`CurrencyTableType` specifies the table type that applies to this currency record for the time period. `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`.

**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**

The `CurrencyDelete` verb removes a currency from the currency table.

The user interface equivalent is to manually remove a currency from the currency table.

The syntax is as follows:

```
CurrencyDelete objCurrency
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CurrencyDelete</strong></td>
<td>Deletes the currency <code>objCurrency</code>. <code>objCurrency</code> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>
Example

This example deletes Canadian Dollars (C$) from the currency table.

CurrencyDelete "C$"

**CurrencyMake**

The CurrencyMake verb adds a currency to the currency table or updates an existing currency.

The user interface equivalent is to modify the property sheet for an existing currency. For a new currency, right-click a currency in the Currency Table property sheet and then select Add New Currency.

A currency table must exist before you can create currencies. To create a currency table, see "CurrencyTableMake" on page 106.

Currency conversions are only applied to measures if the measure definition contains the option IsCurrency True.

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Locating objects uniquely" on page 13.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrencyMake objCurrency</td>
<td>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.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Is specified, if necessary, to uniquely identify the level objLevel. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Drill objDrillCat</td>
<td>Is specified, if necessary, to uniquely identify the date level objLevel. objDrillCat can be the drill category object name, object identifier, or both.</td>
</tr>
<tr>
<td>Levels objLevel</td>
<td>Specifies the level of the date category objCat. This indicates the granularity of the rate data, such as daily or monthly. objLevel can be the object name or object identifier. The level is mandatory.</td>
</tr>
<tr>
<td>Argument</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>currencyrecordopts</td>
<td>Optional parameters that describe the currency in greater detail. Some options are set by default if you do not set them. For the complete list of options, see &quot;currencyrecordopts” on page 190. If the currency exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>CurrencyRateList</td>
<td>Indicates the start of conversion rates.</td>
</tr>
<tr>
<td>EffectiveDate [objCat]</td>
<td>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 country or region is part of the European Monetary Union (EMU).</td>
</tr>
<tr>
<td>ConversionRate rate</td>
<td>rate is the exchange rate for that date. It can include decimals; for example, 3.42556.</td>
</tr>
<tr>
<td>RateIsBySource [True</td>
<td>False] RateIsBySource indicates whether each rate is sourced from the external data source or was manually entered. The default is False.</td>
</tr>
<tr>
<td>CurrencyTableType tabletype</td>
<td>CurrencyTableType specifies the table type that applies to the currency record for the time period. 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.</td>
</tr>
</tbody>
</table>

**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
```
**CurrencyTableAdd**

The `CurrencyTableAdd` verb creates a currency table in your Cognos Transformer model.

The user interface equivalent is to select **Currency Table** from the **File** menu.

If you specify the option `CurrencyTableType`, the object name of the currency table is optional.

For more information about creating Currency Tables, see "CurrencyTableMake" on page 106.

The syntax is as follows:

```
CurrencyTableAdd [objCurrencyTable] [currencytableopts] Associations assocopts
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CurrencyTableAdd</code></td>
<td><code>objCurrencyTable</code> creates the currency table <code>objCurrencyTable</code>. <code>objCurrencyTable</code> can be the object name, object identifier, or both. Include the name for clarity in your model.</td>
</tr>
<tr>
<td><code>currencytableopts</code></td>
<td>Optional parameters that specify columns in an external data source. For the complete list of options, see &quot;currencytableopts&quot; on page 193. If the currency exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td><code>Associations</code></td>
<td>Required parameters that specify associations for the currency table. For the complete list of options, see &quot;assocopts&quot; on page 170.</td>
</tr>
</tbody>
</table>

**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**

The `CurrencyTableDelete` verb removes an existing currency table from the model.
The user interface equivalent is the **Remove Currency Table** option on the **File** menu.

The syntax is as follows:

```plaintext
CurrencyTableDelete objCurrencyTable
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrencyTableDelete</td>
<td>Deletes the currency table <code>objCurrencyTable</code>.</td>
</tr>
<tr>
<td><code>objCurrencyTable</code></td>
<td><code>objCurrencyTable</code> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the base currency table from the Cognos Transformer model.

```plaintext
CurrencyTableDelete "CurrencyBaseTable"
```

### CurrencyTableMake

The **CurrencyTableMake** verb creates a currency table or updates an existing currency table.

The user interface equivalent is to modify the properties on the **Currency Table** property sheet.

On the Cognos Transformer user interface, the currency table has a single property sheet, even when Euro triangulation is in effect. In MDL, the Base Currency and Euro Currency tables display as two separate objects. Each has its own `CurrencyTableMake` statement.

The default object names for the two types of table are `CurrencyBaseTable` and `CurrencyEuroTable`. You can specify a different object name, but on the user interface, the default object always displays.

If you specify the option `CurrencyTableType`, the object name of the currency table is optional.

The syntax is as follows:

```plaintext
CurrencyTableMake [objCurrencyTable] [currencytableopts] Associations assocopts
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CurrencyTableMake</strong></td>
<td>Creates the currency table <code>objCurrencyTable</code> or modifies it if it exists.</td>
</tr>
<tr>
<td><code>objCurrencyTable</code></td>
<td><code>objCurrencyTable</code> can be the object name, object identifier, or both. Include the object name if the currency table does not exist.</td>
</tr>
<tr>
<td><code>currencytableopts</code></td>
<td>Optional parameters that specify columns in an external data source. For the complete list of options, see &quot;currencytableopts&quot; on page 193. If the currency exists, previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>
### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associations assocopts</td>
<td>Optional parameters that specify associations for the currency table. For the complete list of options, see &quot;assocopts&quot; on page 170.</td>
</tr>
</tbody>
</table>

### Example

This example defines a currency table.

```plaintext
CurrencyTableMake B69 "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

The `CurrencyTableUpdate` verb updates an existing currency table in the model.

The user interface equivalent is to modify the properties on the **Currency Table** property sheet.

If you specify the option `CurrencyTableType`, the object name of the currency table is optional.

For more information about updating Currency Tables, see ["CurrencyTableMake"](on page 106).

The syntax is as follows:

```plaintext
CurrencyTableUpdate [objCurrencyTable] [currencytableopts] Associations assocopts
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrencyTableUpdate objCurrencyTable</td>
<td>Updates the currency table <code>objCurrencyTable</code>. <code>objCurrencyTable</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>currencytableopts</td>
<td>Optional parameters that specify columns in an external data source. For the complete list of options, see [&quot;currencytableopts&quot;](on page 193). If the currency exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>Associations assocopts</td>
<td>Optional parameters that specify associations for the currency table. For the complete list of options, see [&quot;assocopts&quot;](on page 170).</td>
</tr>
</tbody>
</table>
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

The `CurrencyUpdate` verb updates an existing currency in a currency table.

The user interface equivalent is to modify the properties on the Currency Table property sheet.

For more information about updating currencies, see "CurrencyMake" on page 103.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrencyUpdate objCurrency</td>
<td>Modifies the currency <code>objCurrency</code>. <code>objCurrency</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Is specified, if necessary, to uniquely identify the level <code>objLevel</code>. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Drill objDrillCat</td>
<td>Is specified, if necessary, to uniquely identify the date level <code>objLevel</code>. <code>objDrillCat</code> can be the drill category object name, object identifier, or both.</td>
</tr>
<tr>
<td>Levels objLevel</td>
<td>Specifies the level of the date category <code>objCat</code>. This indicates the granularity of the rate data, such as daily or monthly. <code>objLevel</code> can be the object name or object identifier. The level is mandatory.</td>
</tr>
<tr>
<td>currencyrecordopts</td>
<td>Optional parameters that describe the currency in greater detail. For the complete list of options, see &quot;currencyrecordopts&quot; on page 190. Previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>CurrencyRateList</td>
<td>Indicates the start of conversion rates.</td>
</tr>
</tbody>
</table>
**Argument Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EffectiveDate</strong></td>
<td>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 <em>EffectiveDate</em>. <em>objCat</em> 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 country or region is part of the European Monetary Union (EMU).</td>
</tr>
<tr>
<td><strong>ConversionRate</strong></td>
<td><em>rate</em> is the exchange rate for that date. It can include decimals; for example, 3.42556.</td>
</tr>
<tr>
<td><strong>RateIsBySource</strong></td>
<td><em>RateIsBySource</em> indicates whether each rate is sourced from the external data source or was manually entered. The default is false.</td>
</tr>
<tr>
<td><strong>CurrencyTableType</strong></td>
<td><em>CurrencyTableType</em> specifies the table type that applies to the currency record for the time period. <em>tabletype</em> can be one of <em>BaseTable</em>, <em>EuroTable</em>, or <em>OtherTable</em>. The default is <em>BaseTable</em> unless <em>EffectiveDate</em> is followed by zero (0), in which case the default is <em>EuroTable</em>.</td>
</tr>
</tbody>
</table>

**Example**

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

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

**CustomViewAdd**

The `CustomViewAdd` verb creates a custom view in the model.

The user interface equivalent for creating a custom view is to select the **Create Custom View** command in the diagram.

The syntax is as follows:

```
CustomViewAdd "customview" DimensionView "All Categories"
DimensionView objView MeasureInclude objMeasure {Yes|No}
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CustomViewAdd</strong></td>
<td>Adds the custom view &quot;custom view&quot;. &quot;custom view&quot; must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td><strong>DimensionView</strong></td>
<td>Specifies that all categories are initially included in the custom view.</td>
</tr>
</tbody>
</table>
**DimensionView**

Specifies the dimension and dimension view. `objView` can be the object name, object identifier, or both. Repeat this argument to specify multiple views.

**MeasureInclude**

Specifies the measure. `objMeasure` can be the object name, object identifier, or both. Repeat this argument to specify multiple measures, as in

| MeasureInclude 159 Yes MeasureInclude 175 No |

**Example**

CustomViewAdd "Authors" DimensionView 149 "All Categories" DimensionView 195 "Authors~User View" MeasureInclude 225 Yes

---

**CustomViewChildListUpdate**

The `CustomViewChildListUpdate` verb defines the list of descendant custom views. You can only update the list of descendant custom views in MDL.

The syntax is as follows:

`CustomViewChildListUpdate custom view ID StartList custom view ID Endlist`

**Example**

CustomViewChildListUpdate 11529 StartList 11537 Endlist
CustomViewChildListUpdate 11537 StartList Endlist

---

**CustomViewDelete**

The `CustomViewDelete` verb deletes a custom view from the model.

The user interface equivalent for deleting a custom view is to right-click the custom view, and then click **Delete**.

The syntax is as follows:

`CustomViewDelete "customview"`

**Example**

CustomViewDelete "Authors"

---

**CustomViewMake**

The `CustomViewMake` verb creates a custom view in the model.
The user interface equivalent for creating a custom view is to select the **Create Custom View** command in the diagram.

The syntax is as follows:

```
CustomViewMake "customview" DimensionView objView
MeasureInclude objMeasure {Yes|No}
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CustomViewMake</strong> &quot;custom view&quot;</td>
<td>Adds the custom view &quot;custom view&quot;. &quot;custom view&quot; must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td><strong>DimensionView</strong> objView</td>
<td>Specifies the dimension and dimension view. objView must be the object name and object identifier. Repeat this argument to specify multiple views.</td>
</tr>
<tr>
<td><strong>MeasureInclude</strong> objMeasure [Yes</td>
<td>No]</td>
</tr>
</tbody>
</table>

**Example**

CustomViewMake "Authors" DimensionView 149 "All Categories" DimensionView 195 "Authors"User View" MeasureInclude 225 Yes

**CustomViewUpdate**

The **CustomViewUpdate** verb modifies a custom view in the model.

The user interface equivalent for creating a custom view is to right-click the custom view, and then modify the custom view properties.

The syntax is as follows:

```
CustomViewUpdate "customview" DimensionView objView MeasureInclude objMeasure {Yes|No}
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CustomViewUpdate</strong> &quot;custom view&quot;</td>
<td>Adds the data source &quot;custom view&quot;. &quot;custom view&quot; must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td><strong>DimensionView</strong> objView</td>
<td>Specifies the dimension and dimension view. objView can be the object name, object identifier, or both. Repeat this argument to specify multiple views.</td>
</tr>
<tr>
<td><strong>MeasureInclude</strong> objMeasure [Yes</td>
<td>No]</td>
</tr>
</tbody>
</table>
Example
CustomViewUpdate "Authors" DimensionView 149 "All Categories"
DimensionView 195 "Authors~User View" MeasureInclude 225 No

**DataSourceAdd**

The `DataSourceAdd` verb creates a data source other than a package or report in the model. This verb is not used to create a package or report data source.

For information about creating a package or report data source, see "CognosPackageAdd" on page 76.

The user interface equivalent for creating a data source is to select the Insert command on the Edit menu when a data source is selected.

The user interface equivalent for creating a query based on an IBM Cognos package or report data source is to select the Add Query From Package or Report command on the Edit menu when a data source is selected.

**Note:**
- If you include a data source ID for the new data source in the syntax, and the object already exists, you will receive an error message.
- To insert a dimension from a package (the Insert Dimension From Package command on the Edit menu), you must first add the data source and then add a dimension from the data source. For more information about adding a dimension from a data source, see "DimAdd" on page 116.

For more information about creating data sources, see "DataSourceMake" on page 114.

The syntax for a query based on an IBM Cognos package or report is as follows:

```
DataSourceAdd objDataSource Sourcetype CognosSourceQuery PackageReportSource
sourceID"source name" [appqueryopts][OrgName objCol
Origin (Source|Generated|Calculated|Manual) Offset offset][colopts]
```

The syntax for all other data source types is as follows:

```
DataSourceAdd objDataSource [appqueryopts][OrgName objCol
Origin (Source|Generated|Calculated|Manual) Offset offset][colopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DataSourceAdd</code></td>
<td><code>objDataSource</code> creates the data source <code>objDataSource</code>. <code>objDataSource</code> must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td><code>appqueryopts</code></td>
<td>Optional parameters that describe the data source in greater detail. Some options are set by default if you do not set them. For the complete list of options, see <code>appqueryopts</code>.</td>
</tr>
</tbody>
</table>
### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OrgName objCol Origin</strong> ([Source</td>
<td>Generated</td>
</tr>
<tr>
<td><strong>Offset offset</strong></td>
<td>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 about structured format, see <a href="#">Structured MDL</a>.</td>
</tr>
</tbody>
</table>

| colopts | Optional parameters that describe the columns in greater detail. Some options are set by default if you do not set them. For the complete list of options, see ["colopts" on page 182](#). |

### Examples

This example adds a query based on the IBM Cognos package data source Go Sales and Retailers to the model.

```mdl
DataSourceAdd "GOSRQuery" SourceType CognosSourceQueryPackageReportSource 103 "GO Sales and Retailers" Separator "," CharacterSet Default DecimalSep "." Thousandsep "," Columns True Timing PopYes CreateDefault AutoSummary True SetCurrent True ServerSource False Speed False Presummarized False
```

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

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

### DataSourceDelete

The `DataSourceDelete` verb removes a data source other than a package or report from the model. This verb does not delete package or report data sources.

For information about deleting package and report data sources, see ["CognosPackageDelete" on page 77](#).

The user interface equivalent is to click **Delete** on the **Edit** menu when a data source is selected.

For more information about IBM Cognos data sources, see the Cognos Transformer User Guide and the Framework Manager User Guide.

The syntax is as follows:

```mdl
DataSourceDelete objDataSource
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataSourceDelete objDataSource</strong></td>
<td>Deletes the data source <code>objDataSource</code>. <code>objDataSource</code> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>
Example

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

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

**DataSourceMake**

The `DataSourceMake` verb creates a data source other than a package or report or updates an existing one. This verb does not create package or report data sources.

For information about creating package and report data sources, see "CognosPackageMake" on page 77.

The user interface equivalent, if the data source exists, is to modify the Data Source property sheet. For a new data source, when the data source is selected, the equivalent is to select the Insert command on the Edit menu.

The syntax for a query based on an IBM Cognos package or report is:

```sql
DataSourceMake datasource ID objDataSource SourceType CognosSourceQuery PackageReportSource source ID "source name" [appqueryopts]
```

The syntax for all other data source types is as follows:

```sql
DataSourceMake datasourceID objDataSource [appqueryopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>objDataSource</code></td>
<td>Creates the data source <code>objDataSource</code> or modifies it if it exists. <code>objDataSource</code> can be the object name, object identifier, or both. Include the object name if the category does not exist.</td>
</tr>
<tr>
<td><code>appqueryopts</code></td>
<td>Optional parameters that describe the data source in greater detail. Some options are set by default if you do not set them. For the complete list of options, see <code>appqueryopts</code>. If the data source exists, previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

**Examples**

This example adds the data source Go Sales and Retailers to the model.

```sql
DataSourceMake 103 "Go Sales and Retailers" SourceType Package SourcePath "/content/package[@name='GO Sales and Retailers']
PackageTimeStamp "/content/package[@name='GO Sales and Retailers200701011200']/model[@name='model']"
```

This example adds a query based on the data source Go Sales and Retailers to the model.

```sql
DataSourceMake 105 "Go Sales and Retailers" SourceType CognosSourceQuery
PackageReportSource 103 "GO Sales and Retailers" Separator "," CharacterSet Default DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault AutoSummary True SetCurrent True ServerSource False Speed False Presummarized False
```

This example defines the data source All Staff Count.
**DataSourceUpdate**

The **DataSourceUpdate** verb updates an existing data source other than a package or report in the model. This verb does not update package or report data sources.

For information about creating package and report data sources, see
"CognosPackageUpdate" on page 78.

The user interface equivalent is to modify the **Data Source** property sheet.

**Note:** If you include a data source ID for the updated data source in the syntax, and the object does not already exist, you will receive an error message.

The syntax is as follows:

**DataSourceUpdate** objDataSource [appqueryopts]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DatasourceUpdate</strong></td>
<td>Specifies the data source to update. <strong>objDataSource</strong> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>appqueryopts</strong></td>
<td>Optional parameters that describe the data source in greater detail. For the complete list of options, see <strong>appqueryopts</strong>. Previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

**Example**

This example disables AutoSummary for the data source Go Sales and Retailers~1.

**DataSourceUpdate** 105 "Go Sales and Retailers~1" Separator "," CharacterSet Default DecimalSep "," ThousandSep "," Columns True Timing PopYesCreateDefault AutoSummary True SetCurrent True ServerSource False Speed False Presummarized False

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**

The **DeletionListUpdate** verb updates the list of objects to be removed from the model.

The user interface equivalent in Cognos Transformer is to delete objects from the model by means of an incremental update.

Do not change this syntax in generated MDL, because this action could cause unexpected results in your model.
The syntax is as follows:

**DeletionListUpdate**  
`deletionsopts`  
**Endlist**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| **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` |

**Example**

This example deletes the category with object identifier 1001.

```
DeletionListUpdate Category 1001 Endlist
```

**DimAdd**

The `DimAdd` verb creates a dimension in the model.

The user interface equivalent is to select the **Insert** command on the **Edit** menu when a dimension is selected.

For more information about creating dimensions, see [“DimMake” on page 121](#).

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| **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 do not set them. For the complete list of options, see `dimopts`. |
| **Associations**  
`assocopts` | Parameters that specify associations for the dimension.  
For the complete list of options, see [“assocopts” on page 170](#) |

**Example**

This example creates the Years dimension.

```
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**

The `DimCalcDefAdd` verb creates a dimension calculation definition in the model.
The user interface equivalent is the Add command on the Calculation tab of the Dimension property sheet.

For more information about creating dimension calculation definitions, see “DimCalcDefMake” on page 118.

The syntax is as follows:

\[
\text{DimCalcDefAdd} \hspace{1em} \text{objDimCalcDef} \hspace{1em} \text{Dimension} \hspace{1em} \text{objDim} \hspace{1em} \text{[Calc expropts]} \hspace{1em} \text{[GroupCalculateCategory \{True|False\}]} \hspace{1em} \text{[Set string StartList objCats EndList]} 
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DimCalcDefAdd objDimCalcDef</td>
<td>Creates the dimension calculation definition objDimCalcDef. objDimCalcDef must be the object name and can also be the object identifier.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Specifies the dimension where the calculated category is to be located. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Calc expropts</td>
<td>Specifies the calculation. expropts are a mixture of calculation keywords and objects. The calculation keywords that can be used are Average, Max, and Min, as well as the Cognos Transformer-specific functions Change, Percent-Growth 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, &quot;Go Water Bottle@Category@4805&quot;.</td>
</tr>
<tr>
<td>GroupCalculateCategory {True</td>
<td>False}</td>
</tr>
<tr>
<td>Set string StartList objCats</td>
<td>Creates one or more sets and specifies the categories in each set. string is the set name, such as Set 1, Set 2, and so on. objCats can be the object name or the object identifier. This argument is repeated for each set; for example Set &quot;Set 1&quot; StartList 4793 4797 EndList Set &quot;Set 2&quot; StartList 4799 EndList</td>
</tr>
</tbody>
</table>

Example

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

\[
\text{DimCalcDefAdd 'share ('Set 1", "GO Sport Line")' Dimension 2947 Calc share ('"Set 1"Set", "GO Sport Line@Category@4789") Set "Set 1" StartList 4793 4797 4801 4803 EndList} 
\]
**DimCalcDefDelete**

The DimCalcDefDelete verb removes a dimension calculation definition from the model.

The user interface equivalent is the **Remove** command on the **Calculation** tab of the **Dimension** property sheet.

The syntax is as follows:

```
DimCalcDefDelete objDimCalcDef [Dimension objDim]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DimCalcDefDelete objDimCalcDef</strong></td>
<td>Deletes the dimension calculation definition <strong>objDimCalcDef</strong>. <strong>objDimCalcDef</strong> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Dimension objDim</strong></td>
<td>Is specified, if necessary, to uniquely identify <strong>objDimCalcDef</strong>. <strong>objDim</strong> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

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

```
DimCalcDefDelete 27195
```

**DimCalcDefMake**

The DimCalcDefMake verb creates or updates a dimension calculation definition in the model.

The user interface equivalents are the **Add** and **Modify** commands on the **Calculation** tab of the **Dimension** property sheet.

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 “DimCalcDefAdd” on page 116.

Calculated categories are defined as regular categories, not special categories.

The syntax is as follows:

```
DimCalcDefMake objDimCalcDef [Dimension objDim] GroupCalculateCategory {True|False}][
Calc expropts] [Set string StartList objCats EndList]
```
### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DimCalcDefMake <code>objDimCalcDef</code></td>
<td>Creates the dimension calculation definition <code>objDimCalcDef</code> or modifies it if it exists. Include the object name if the definition does not exist.</td>
</tr>
<tr>
<td>Dimension <code>objDim</code></td>
<td>Is specified, if necessary, to uniquely identify <code>objDimCalcDef</code>. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>GroupCalculateCategory</td>
<td>Specifies 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 displays in the Category Calculation Definition dialog box when you create a category definition. The default is False.</td>
</tr>
<tr>
<td>Calc <code>expropts</code></td>
<td>Specifies the calculation. <code>expropts</code> are a mixture of calculation keywords and objects. The calculation keywords that can be used are Average, Max, and Min, as well as the Cognos Transformer-specific functions Change, Percent-Growth and Share. The objects are specified as object name followed by an at sign (<code>@</code>), the object type, and, optionally, an at sign (<code>@</code>) and the object identifier. The object type can be Category, Level, or Drill. For example, &quot;Go Water Bottle@Category@4805&quot;.</td>
</tr>
<tr>
<td>Set <code>string</code> StartList <code>objCats</code> EndList</td>
<td>Creates one or more sets and specifies the categories in each set. <code>string</code> is the set name, such as Set 1, Set 2, and so on. <code>objCats</code> can be the object name or the object identifier. This argument is repeated for each set; for example Set &quot;Set 1&quot; StartList 4795 4797 EndList Set &quot;Set 2&quot; StartList 4799 EndList</td>
</tr>
</tbody>
</table>

### Example

This example defines the dimension calculation 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 EndList
Set "Set 2" StartList 4799 EndList
```

### DimCalcDefUpdate

The DimCalcDefUpdate verb modifies a dimension calculation definition in the model.
The user interface equivalent is the **Modify** command on the **Calculation** tab of the **Dimension** property sheet.

For more information about updating dimension calculation definitions, see "DimCalcDefMake" on page 118.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DimCalcDefUpdate objDimCalcDef</td>
<td>Modifies the dimension calculation definition <code>objDimCalcDef</code>. <code>objDimCalcDef</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Is specified, if necessary, to uniquely identify <code>objDimCalcDef</code>. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>GroupCalculateCategory {True</td>
<td>False}</td>
</tr>
<tr>
<td>Calc exprots</td>
<td>Specifies the calculation. <code>exprots</code> are mixture of calculation keywords and objects. The calculation keywords that can be used are Average, Max, and Min, as well as the Cognos Transformer-specific functions Change, Percent-Growth 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, &quot;Go Water Bottle@Category@4805&quot;.</td>
</tr>
</tbody>
</table>
| Set string StartList objCats EndList | Creates one or more sets and specifies the categories in each set. `string` is the set name, such as Set 1, Set 2, and so on. `objCats` can be the object name or the object identifier. This argument is repeated for each set; for example  
  
  Set "Set 1" StartList 4795 4797 EndList  
  Set "Set 2" StartList 4799 EndList |

**Example**

This example modifies dimension calculation Share ("Set 1", "GO Sport Line") by adding the category with object identifier 4789 into Set 1.
**DimDelete**

The DimDelete verb removes a dimension from the model.

The user interface equivalent is to click **Delete** on the **Edit** menu when a dimension is selected.

The syntax is as follows:

```
DimDelete objDim
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DimDelete objDim</td>
<td>Deletes the dimension <em>objDim</em>. <em>objDim</em> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the dimension Products.

```
DimDelete "Products"
```

**DimensionListUpdate**

The DimensionListUpdate verb reorders the list of dimensions in the model.

The user interface equivalent is to drag and drop dimensions into the required positions.

The syntax is as follows:

```
DimensionListUpdate objDims EndList
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DimensionListUpdate objDims</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Example**

This example reorders the list so that Products and Years display as the first two dimensions.

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

**DimMake**

The DimMake verb creates a dimension or updates an existing one.
The user interface equivalent, if the dimension exists, is to modify the **Dimension** property sheet. For a new dimension, select the **Insert** option on the **Edit** menu when a dimension is selected.

Dimensions require a root category, a drill category and two default dimension views. These are created automatically when a dimension is added on the user interface, but they are only created in MDL if you use the verb `ModelEnsureCompleteness`.

The syntax is as follows:

```
DimMake objDim [dimopts] Associations assocopts

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DimMake</strong> objDim</td>
<td>Creates the dimension <code>objDim</code> or modifies it if it exists. <code>objDim</code> can be the object name, object identifier, or both. Include the object name if the category does not exist.</td>
</tr>
<tr>
<td>dimopts</td>
<td>Optional parameters that describe the dimension in greater detail. Some options are set by default if you do not set them. For the complete list of options, see <code>dimopts</code>. If the dimension exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>Associations assocopts</td>
<td>Parameters that specify associations for the dimension. For the complete list of options, see &quot;assocopts&quot; on page 170</td>
</tr>
</tbody>
</table>

**Example**

This example defines the dimension `Products`.

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

**DimUpdate**

The `DimUpdate` verb updates an existing dimension.

The user interface equivalent is to modify the **Dimension** property sheet.

For more information about updating dimensions, see "DimMake" on page 121.

The syntax is as follows:

```
DimUpdate objDim [dimopts] [Associationsassocopts]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DimUpdate</strong> objDim</td>
<td>Specifies the dimension to update. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>dimopts</td>
<td>Optional parameters that describe the dimension in greater detail. For the complete list of options, see <code>dimopts</code>. Previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>
### Arguments and Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associations</td>
<td>Optional parameters that specify associations for the dimension. For the complete list of options, see &quot;assocopts&quot; on page 170.</td>
</tr>
</tbody>
</table>

### Example

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

```
DimUpdate "Years" EarliestDate 19900101 LatestDate 20100101
```

#### DrillCatMake

The `DrillCatMake` verb creates or updates a drill category.

### Example

The user interface equivalent is to modify the Drill Category property sheet. Drill categories are created automatically when you build models on the user interface.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DrillCatMake</td>
<td>Creates the drill category <code>objDrillCat</code> or changes it if it exists. <code>objDrillCat</code> can be the object name, object identifier, or both. Include the object name if the category does not exist.</td>
</tr>
<tr>
<td>Dimension</td>
<td>When creating a drill category, this specifies its dimension. When updating a drill category this is used, if necessary, to uniquely identify it. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Root</td>
<td>When creating a drill category, this specifies its root category. <code>objRootCat</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>JoiningLevel</td>
<td>When creating a drill category, this specifies the convergence level. <code>objLevel</code> can be the object name or object identifier.</td>
</tr>
</tbody>
</table>

`catopts`  
Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see "catopts" on page 171. If the category exists, previously set options are retained unless you change them with this command.

This example defines the drill category By Product Line.
**EventEnd**

The EventEnd verb signals the end of an update event that was started by the EventStart verb. The status of the PowerCubes involved in the event reverts to Free. There is no user interface equivalent for this verb.

In IBM Cognos, you do not specify a client/server platform type.

The syntax is as follows:

```
EventEnd
```

**Example**

This example ends the current update event.

```
EventEnd
```

**EventStart**

The EventStart verb 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. There is no user interface equivalent for this verb.

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

In IBM Cognos, you do not specify a client/server platform type.

The syntax is as follows:

```
EventStart [EventName]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventName</td>
<td>The event name can be up to 256 characters in length. It cannot contain a carriage return.</td>
</tr>
</tbody>
</table>

**Example**

This example starts the Batch Update event.

```
EventStart "Batch Update"
```

**FilterCat**

The FilterCat verb excludes (or filters out) a category from the specified dimension view. The user interface equivalent is the **Exclude** option on the **Diagram** menu.

Using FilterCat on an already-excluded category removes the filter.
Note: Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see “Locating objects uniquely” on page 13.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FilterCat</strong> /<em>objView</em>/</td>
<td>Specifies the dimension view containing the category to be excluded or filtered out. /<em>objView</em>/ can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Dimension</strong> /<em>objDim</em>/</td>
<td>Is specified, if necessary, to uniquely identify the view. /<em>objDim</em>/ can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Category</strong> /<em>objCat</em>/</td>
<td>Specifies the category to be excluded or filtered out. /<em>objCat</em>/ can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

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

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

**LevelAdd**

The LevelAdd verb creates a new level in the specified model dimension.

The user interface equivalent is to select the **Insert** option on the **Edit** menu when a level is selected.

For more information about creating levels, see “LevelMake” on page 127.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LevelAdd</strong> /<em>objLevel</em>/</td>
<td>Creates the level /<em>objLevel</em>/. /<em>objLevel</em>/ must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td><strong>Dimension</strong> /<em>objDim</em>/</td>
<td>Specifies a dimension for the level. /<em>objDim</em>/ can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Drill</strong> /<em>objDrillCat</em>/</td>
<td>Specifies a drill category for the level. /<em>objDrillCat</em>/ can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Parent</strong> /<em>objLevel</em>/</td>
<td>Specifies a parent for the level. /<em>objLevel</em>/ can be the object name, or object identifier.</td>
</tr>
</tbody>
</table>
**Argument Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>levelopts</strong></td>
<td>Optional parameters that describe the level in greater detail. Some options are set by default if you do not set them. For the complete list of options, see <a href="#">levelopts</a>.</td>
</tr>
<tr>
<td><strong>DrillList objDrillCats</strong></td>
<td>Specifies the multiple drill categories objDrillCats. objDrillCats is one or more drill categories, each identified by object name or object identifier.</td>
</tr>
<tr>
<td><strong>Associations assocopts</strong></td>
<td>Parameters that specify associations for the level. For the complete list of options, see <a href="#">assocopts</a> on page 170.</td>
</tr>
</tbody>
</table>

**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**

The LevelDelete verb removes a level from the specified model dimension.

The user interface equivalent is to click **Delete** on the **Edit** menu when a level is selected.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LevelDelete objLevel</strong></td>
<td>Deletes the level objLevel. objLevel can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Dimension objDim</strong></td>
<td>Is specified, if necessary, to uniquely identify the level. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Drill objDrillCat</strong></td>
<td>Is specified, if necessary, to uniquely identify the level. objDrillCat can be the object name or object identifier.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the level Product Line.

```
LevelDelete "Product Line" Dimension "Products" Drill "By Product Line"
```
The `LevelMake` verb creates a level or updates an existing one.

The user interface equivalent, if the level exists, is to modify the Level property sheet. For a new level, select the Insert option on the Edit menu when a level is selected.

Level names cannot contain an at sign (`@`).

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LevelMake objLevel</td>
<td>Creates the level <code>objLevel</code> or modifies it if it exists. <code>objLevel</code> can be the object name, object identifier, or both. Include the object name if the category does not exist.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>When creating a level, this specifies its dimension. When updating a level this is used, if necessary, to uniquely identify it. <code>objDim</code> can be the object name or object identifier.</td>
</tr>
<tr>
<td>Drill objDrillCat</td>
<td>When creating a level, this specifies its drill category. When updating a level this is used, if necessary, to uniquely identify it. <code>objDrillCat</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Parent objLevel</td>
<td>When creating a level, this specifies its parent. When updating a level this is used, if necessary, to uniquely identify it. <code>objLevel</code> can be the object name or object identifier.</td>
</tr>
<tr>
<td>levelopts</td>
<td>Optional parameters that describe the level in greater detail. Some options are set by default if you do not set them. For the complete list of options, see <code>levelopts</code>. If the category exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>DrillList objDrillCats</td>
<td>Specifies the multiple drill categories <code>objDrillCats</code>. <code>objDrillCats</code> is one or more drill categories, each identified by object name or object identifier.</td>
</tr>
<tr>
<td>Associations assocopts</td>
<td>Parameters that specify associations for the level. For the complete list of options, see &quot;assocopts&quot; on page 170.</td>
</tr>
</tbody>
</table>

**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
```
The `LevelMoveAfter` verb switches the positions of two levels.

The user interface equivalent is to drag and drop the level into the required position.

Using this verb does not change the object identifier for the level or its position in the .mdl file. It only affects the visual display of the level on the user interface and its position in the .mdc file, which affects how it is viewed in the reporting components.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LevelMoveAfter objLevel</strong></td>
<td>Specifies the level that is to be placed after. <code>objLevel</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Dimension objDim</strong></td>
<td>Is specified, if necessary, to uniquely identify the level. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Drill objDrillCat</strong></td>
<td>Is specified, if necessary, to uniquely identify the level. <code>objDrillCat</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Child objLevel</td>
<td>Specifies the level that is to be placed before. <code>objLevel</code> can be the object name or object identifier.</td>
</tr>
</tbody>
</table>

**Example**

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

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

The `LevelMoveBefore` verb switches the positions of two levels.

The user interface equivalent is to drag the level into the required position.

Using this verb does not change the object identifier for the level or its position in the .mdl file. It only affects the visual display of the level on the user interface and its position in the .mdc file, which affects how it is viewed in the reporting components.

The syntax is as follows:

```
LevelMoveBefore objLevel [Dimension objDim] [Drill objDrill]
Child objLevel
```
### LevelMoveBefore

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objLevel</td>
<td>Specifies the level that is to be placed before. objLevel can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>objDim</td>
<td>Is specified, if necessary, to uniquely identify the level. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>objDrillCat</td>
<td>Is specified, if necessary, to uniquely identify the level. objDrillCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>objLevel</td>
<td>Specifies the level that is to be placed after. objLevel can be the object name or object identifier. The child is mandatory.</td>
</tr>
</tbody>
</table>

#### Example

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

**LevelMoveBefore** "Product Line" **Drill** "By Product Line" **Child** "Product ID"

---

### LevelNewDrill

The `LevelNewDrill` verb creates a new drill path in a level.

The user interface equivalent is to drag levels and columns to form the new path.

The syntax is as follows:

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objLevel</td>
<td>Specifies the level that is to have a new drill path. objLevel can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>objDim</td>
<td>Is specified, if necessary, to uniquely identify the level. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>objDrillCat</td>
<td>Specifies the old drill category. objDrillCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>objDrillCat</td>
<td>Specifies the new drill category. objDrillCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>objLevel</td>
<td>Specifies the joining level. objLevel can be the object name or object identifier.</td>
</tr>
</tbody>
</table>

#### Example

This example adds a new drill path Territory in the level State with a joining level of City.
The **LevelUpdate** verb updates an existing level.

The user interface equivalent is to modify the **Level** property sheet.

For more information about updating levels, see ["LevelMake" on page 127](#).

The syntax is as follows:

```
LevelUpdate objLevel [Dimension objDim] [Drill objDrillCat] [levelopts] [DrillList objDrillCatsEndList] [Associations assocopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LevelUpdate</strong> objLevel</td>
<td>Specifies the level to update. <em>objLevel</em> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Dimension</strong> objDim</td>
<td>Is specified, if necessary, to uniquely identify the level. <em>objDim</em> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Drill</strong> objDrillCat</td>
<td>Is specified, if necessary, to uniquely identify the level. <em>objDrillCat</em> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>levelopts</td>
<td>Optional parameters that describe the level in greater detail. For the complete list of options, see &quot;levelopts&quot;. Previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td><strong>DrillList</strong> objDrillCats</td>
<td>Specifies the multiple drill categories <em>objDrillCats</em>. <em>objDrillCats</em> is one or more drill categories, each identified by object name or object identifier.</td>
</tr>
<tr>
<td>Associations assocopts</td>
<td>Optional parameters that specify associations for the level. For the complete list of options, see &quot;assocopts&quot; on page 170.</td>
</tr>
</tbody>
</table>

**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
```

**MDCClear**

The **MDCClear** verb clears the processing status for the specified PowerCube. The default status is New.

There is no user interface equivalent, although the effects can be verified on the **Processing** tab of the **PowerCube** property sheet.
The syntax is as follows:

**MDCClear** *objCube*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDCClear <em>objCube</em></td>
<td>Specifies the PowerCube that is to have its status flags cleared. <em>objCube</em> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

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

**MDCClear** "Great Outdoors Sales"

**MeasureAdd**

The **MeasureAdd** verb adds a measure to the model. If the measure object already exists, an error message is issued.

The user interface equivalent is to click the **Insert** option on the **Edit** menu when a measure is selected.

Every measure requires an **AllocationAdd** statement that contains the measure name and the option **TypeDefault**.

Cognos 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**" on page 65.

For more information about creating measures, see "**MeasureMake**" on page 132.

The syntax is as follows:

**MeasureAdd** *objMeasure* [*meaopts Associations*] *asscopts*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MeasureAdd</strong> <em>objMeasure</em></td>
<td>Creates the measure <em>objMeasure</em>. <em>objMeasure</em> must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td><em>meaopts</em></td>
<td>Optional parameters that describe the measure in greater detail. Some options are set by default if you do not set them. For the complete list of options, see [<em>meaopts</em>].</td>
</tr>
<tr>
<td><strong>Associations</strong> <em>asscopts</em></td>
<td>Parameters that specify associations for the measure. For the complete list of options, see [<em>asscopts</em>] on page 170.</td>
</tr>
</tbody>
</table>

**Example**

This example adds the Staff Count measure from the Staff Count column.
MeasureDelete

The MeasureDelete verb removes a measure from the model.

The user interface equivalent is to click Delete on the Edit menu when a measure is selected.

The syntax is as follows:

```
MeasureDelete objMeasure
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeasureDelete objMeasure</td>
<td>Deletes the category objMeasure. objMeasure can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

Example

This example deletes the measure Profit Margin %.

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

MeasureListUpdate

The MeasureListUpdate verb reorders the items in the list of measures.

The user interface equivalent is to drag the measures into the required order.

The syntax is as follows:

```
MeasureListUpdate objMeasures EndList
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeasureListUpdate objMeasures</td>
<td>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.</td>
</tr>
</tbody>
</table>

Example

This example changes the order of the list of measures, placing Product Cost and then Revenue at the top.

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

MeasureMake

The MeasureMake verb creates a measure or updates an existing one.
The user interface equivalent, if the measure exists, is to modify the Measure property sheet. For a new measure, click the Insert option on the Edit menu when a measure is selected.

Every measure requires an AllocationAdd statement that contains the measure name and the option Type Default. Cognos Transformer creates this statement automatically when a measure is added on the user interface or when you use the verb ModelEnsureCompleteness. Otherwise, you must create the statement. For more information, see “AllocationAdd” on page 65.

Measure names cannot contain an at sign (@).

The syntax is as follows:

```
MeasureMake objMeasure [meaopts] Associations assocopts
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeasureMake objMeasure</td>
<td>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.</td>
</tr>
<tr>
<td>meaopts</td>
<td>Optional parameters that describe the measure in greater detail. Some options are set by default if you do not set them. For the complete list of options, see <a href="#">meaopts</a> If the category exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>Associations assocopts</td>
<td>Parameters that specify associations for the measure. For the complete list of options, see <a href="#">assocopts</a> on page 170.</td>
</tr>
</tbody>
</table>

**Example**

The following statements were generated when a sample model created on the user interface was saved as an .mdl file.

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 "%" MeasureInfo "Percent profit" DrillThrough False EndList
```

This example creates or updates a Revenue measure that allows drill-through to an IBM Cognos Impromptu report (dt_cust.imr).

```
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

The MeasureUpdate verb updates an existing measure.

The user interface equivalent is to modify the Measure property sheet.

For more information about updating measures, see “MeasureMake” on page 132.

The syntax is as follows:

```
MeasureUpdate objMeasure [meaopts] [Associations assocopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeasureUpdate objMeasure</td>
<td>Specifies the measure to update. objMeasure can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>meaopts</td>
<td>Optional parameters that describe the measure in greater detail. For the complete list of options, see meaopts. If the category exists, previously set options are retained unless you change them with this command.</td>
</tr>
<tr>
<td>Associations assocopts</td>
<td>Optional parameters that specify associations for the measure. For the complete list of options, see “assocopts” on page 170.</td>
</tr>
</tbody>
</table>

Example

This example changes the timing of the Profit Margin % measure.

```
MeasureUpdate "Profit Margin %" Timing Before_Rollup
```

ModelEnsureCompleteness

The ModelEnsureCompleteness verb scans all dimensions and cubes in the model and creates necessary default settings and objects, including root categories, drill categories, and default dimension views.

There is no user interface equivalent. This is done automatically by the Cognos Transformer component.

Using ModelEnsureCompleteness avoids the need to create Dimension views and associate them with PowerCubes and security objects in the model. Include this verb near the end of your user-defined models, to ensure that each .mdl file is valid.

The syntax is as follows:

```
ModelEnsureCompleteness
```

NewModel

The NewModel verb creates, names, describes, and opens a Cognos Transformer model.

The user interface equivalent is to click the New option on the File menu.
Cognos Transformer saves the model file in the My Documents/Transformer/Models directory, unless a server path is specified.

Running `NewModel` will close previously opened files without saving them. If you want to save a file, you must use the verb `SaveMDL` or `SavePY` before you use the `NewModel` verb.

The syntax is as follows:

```
NewModel objModel [appqueryopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>NewModel</code></td>
<td>Specifies the model that is to be opened. <code>objModel</code> must be the model name.</td>
</tr>
<tr>
<td><code>appqueryopts</code></td>
<td>Optional parameters that describe the model in greater detail. Some options are set by default if you do not set them. For the complete list of options, see <a href="#">appqueryopts</a>.</td>
</tr>
</tbody>
</table>

**Example**

This example creates a model called PowerPlay® Sample, and provides a description.

```
NewModel "PowerPlay Sample" AppInfo "This model used the tutorial."
```

---

**OpenDef**

Opens .def, .gen, and .dat files.

The user interface equivalent is the Open dialog box, in editions of Cognos Transformer where you can select PowerPlay definition files (*def & *.gen).

Support for this format was retained to ensure compatibility with earlier versions.

The syntax is as follows:

```
OpenDef filename
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OpenDef</code></td>
<td>Specifies the file name, including path if desired.</td>
</tr>
</tbody>
</table>

**Example**

This example opens the "file.def" PowerPlay definition file.

```
OpenDef "file.def"
```

---

**OpenMDL**

The `OpenMDL` verb opens the specified .mdl file. Execution of the remaining MDL script continues after the file is opened.
The user interface equivalent is the **Open** dialog box, where you can select **Model files (*.py? and *.mdl)**.

You can only open one model at a time. Any model that was open before the OpenMDL command is issued will be closed without being saved.

To save an open model, you must use the SaveMDL or SavePY command before using the OpenMDL command.

The syntax is as follows:

```plaintext
OpenMDL filename
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMDL filename</td>
<td>Specifies the file name, including path if desired, for the current model. The file name should have the extension .mdl.</td>
</tr>
</tbody>
</table>

**Example**

This example opens the model outdoors.mdl, located in the root directory of the c: drive.

```plaintext
OpenMDL "c:\outdoors.mdl"
```

---

**OpenPY**

The **OpenPY** verb opens the specified .py? file. Execution of the remaining MDL script continues after the file is opened.

The question mark in the extension .py? is replaced by the character that is used in your release of Cognos Transformer, such as .pyj.

You can only open one model at a time. Any model that was open before the OpenPY command is issued will be closed without being saved.

To save an open model, you must use the SaveMDL or SavePY command before the OpenPY command.

The user interface equivalent is the **Open** dialog box, where you can select **Model files (*.py? and *.mdl)**.

The syntax is as follows:

```plaintext
OpenPY filename
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenPY filename</td>
<td>Specifies the file name, including path if desired, for the current model. The file name should have the extension .py?, where ? is dependent on your version of Cognos Transformer.</td>
</tr>
</tbody>
</table>
Example

This example opens Model.pyj in the root directory of the c: drive and saves it as an .mdl file.

`OpenPY "C:\Model.pyj" SaveMDL "C:\Model.mdl"`

**PopulateFromQueries**

The `PopulateFromQueries` verb populates the model with only those categories found in the specified data sources.

The user interface equivalent is to right-click one or more data sources and, from the Run menu, click Generate Categories.

The syntax is as follows:

```
PopulateFromQueries objQueries
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>objQueries</code></td>
<td>Specifies one or more data sources to be used. <code>objQueries</code> can be the object identifiers or object names.</td>
</tr>
</tbody>
</table>

Example

This example populates the model using only the All Staff Count (Excel) data source.

`PopulateFromQueries "All Staff Count (Excel)"`

**PopulateModel**

The `PopulateModel` verb populates the model with categories using all data sources in the model.

The user interface equivalent is the Generate Categories option on the Run menu.

In IBM Cognos software, you do not specify the platform type.

The syntax is as follows:

```
PopulateModel
```

**PowerCubeCustomViewListUpdate**

The `PowerCubeCustomViewListUpdate` verb defines the list of custom views assigned to a cube.

The syntax is as follows:

```
PowerCubeCustomViewListUpdate Cube PowerCube ID StartList PowerCube ID Endlist
```

Example

```
PowerCubeCustomViewListUpdate Cube 11535 StartList 11537 Endlist
```
PowerCubeDelete

The PowerCubeDelete verb removes the specified PowerCube or cube group by deleting the PowerCube objects from the model. This verb does not affect existing .mdc files.

The user interface equivalent is to click Delete on the Edit menu when a PowerCube or cube group is selected.

The syntax is as follows:

```
PowerCubeDelete {objCube|objCubeGroup}
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>objCube</code></td>
<td>Deletes the PowerCube <code>objCube</code> or the PowerCube group <code>objCubeGroup</code>. <code>objCube</code> and <code>objCubeGroup</code> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the PowerCube **Great Outdoors Sales**.

```
PowerCubeDelete "Great Outdoors Sales"
```

PowerCubeListUpdate

The PowerCubeListUpdate verb reorders the PowerCubes and cube groups listed in the model.

The user interface equivalent is to drag PowerCubes and cube groups into the required order.

The syntax is as follows:

```
PowerCubeListUpdate [objCubes][objCubeGroups] EndList
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>objCubes</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>objCubeGroups</code></td>
<td>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.</td>
</tr>
</tbody>
</table>

**Example**

This example changes the order of the Great Outdoors Sales and cubegroup1 cubes.

```
PowerCubeListUpdate "cubegroup1" "Great Outdoors Sales" EndList
```
**PromptAdd**

The PromptAdd verb imports prompts defined in IBM Cognos package and report data sources.

*PromptAdd* promptname *DataSource* datasource ID *promptopts*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prompt name</td>
<td>Specifies the name of the prompt.</td>
</tr>
<tr>
<td>data source name</td>
<td>Specifies the name of the IBM Cognos package, report, or query data source.</td>
</tr>
<tr>
<td>promptopts</td>
<td>Specifies the prompt type and prompt value. Simple-value, multi-value, and range prompts are type 'xsdString'; Member Unique Name (MUN) prompts, for SAP and DMR data sources only, are type 'memberUniqueName'.</td>
</tr>
</tbody>
</table>

**Example**

This example adds a prompt named Product Name1 with type xsdString and value BugShield Natural for the package 574942~1.

```
PromptAdd "Product Name1" DataSource "574942~1" PromptType "xsdString"
PromptValue "BugShield Natural"
```

**PromptDelete**

The PromptDelete verb deletes prompts defined in Cognos Transformer model.

*PromptDelete* promptname

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prompt name</td>
<td>Specifies the name of the prompt.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the Product Name1 prompt from the model.

```
PromptDelete "Product Name1"
```

**PromptMake**

The PromptMake verb creates prompts that are defined in IBM Cognos package and report data sources.

*PromptMake* promptname *DataSource* datasource ID *promptopts*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prompt name</td>
<td>Specifies the name of the prompt.</td>
</tr>
<tr>
<td>data source name</td>
<td>Specifies the name of the IBM Cognos package, report, or query data source.</td>
</tr>
</tbody>
</table>
### Argument Description

**promptopts**  
Specifies the prompt type and prompt value. Simple-value, multi-value, and range prompts are type "xsdString"; Member Unique Name (MUN) prompts, for SAP and DMR data sources only, are type "memberUniqueName".

### Example

This example creates a prompt named Product Name 1 with type xsdString and value BugShield Natural for the package 574942~1.

PromptMake "Product Name1" DataSource "574942"1" PromptType "xsdString" PromptValue "BugShield Natural"

---

### PromptUpdate

The **PromptUpdate** verb updates prompts in the Cognos Transformer model.

#### Example

**PromptUpdate** promptname **DataSource** datasource ID **promptopts**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prompt name</td>
<td>Specifies the name of the prompt.</td>
</tr>
<tr>
<td>data source name</td>
<td>Specifies the name of the IBM Cognos package, report, or query data source.</td>
</tr>
<tr>
<td>promptopts</td>
<td>Specifies the prompt type and prompt value. Simple-value, multi-value, and range prompts are type &quot;xsdString&quot;; Member Unique Name (MUN) prompts, for SAP and DMR data sources only, are type &quot;memberUniqueName&quot;.</td>
</tr>
</tbody>
</table>

This example updates a prompt named Product Name 1 with type xsdString and value SmallShield Artificial for the package 574942~1.

PromptUpdate "Product Name1" DataSource "574942"1" PromptType "xsdString" PromptValue "SmallShield Artificial"

---

### ReportPartitions

The **ReportPartitions** verb writes a partition status report to the log file for a specified PowerCube. There is no user interface equivalent.

The syntax is as follows:

**ReportPartitions** **objCube**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReportPartitions <strong>objCube</strong></td>
<td>Specifies the PowerCube that is to have a partition status report written to its log file.</td>
</tr>
</tbody>
</table>
**Example**

This example writes a partition report to the log file for the Great Outdoors Sales PowerCube.

ReportPartitions "Great Outdoors Sales"

---

**RootCatMake**

The `RootCatMake` verb creates a root category or updates an existing one.

The user interface equivalent is to modify the **Root Category** property sheet. Root categories are created automatically when modeling on the user interface.

The syntax is as follows:

```
RootCatMake objRootCat [Dimension objDim][catopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RootCatMake objRootCat</code></td>
<td>Creates the category <code>objRootCat</code> or changes it if it exists. <code>objRootCat</code> can be the object name, object identifier, or both. Include the object name if the category does not exist.</td>
</tr>
<tr>
<td><code>Dimension objDim</code></td>
<td>Specifies the dimension of a new root category or, if necessary, uniquely identifies an existing root category. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><code>catopts</code></td>
<td>Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see page 171. If the category exists, previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

---

**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**

The `RootCatUpdate` verb updates an existing root category.

The user interface equivalent is to modify the **Root Category** property sheet.

For more information about updating root categories, see "RootCatMake."

The syntax is as follows:

```
RootCatUpdate objRootCat Dimension [objDim][catopts]
```
**Argument Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RootCatUpdate</strong> objRootCat</td>
<td>Specifies the root category to update. objRootCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Dimension</strong> objDim</td>
<td>Required. Is specified to uniquely identify the root category. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>catopts</strong></td>
<td>Optional parameters that describe the category in greater detail. For the complete list of options, see &quot;catopts&quot; on page 171. Previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

**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**

The **SaveMDL** verb saves the current model as an .mdl file.

The user interface equivalent is to save the model with an .mdl extension.

The syntax is as follows:

```
SaveMDL filename
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SaveMDL</strong> filename</td>
<td>Specifies the file name, including path if desired, for the current model. The file name should have the extension .mdl.</td>
</tr>
</tbody>
</table>

**Example**

This example opens a .py? file and saves it as an .mdl file.

```
OpenPY "C:\model.pyj" SaveMDL "C:\model.mdl"
```

**SavePY**

The **SavePY** verb saves the current model as a .py? file, where the ? is replaced by the character that is used in your release of Cognos Transformer.

The user interface equivalent is to save the model with a .py? extension.

The syntax is as follows:

```
SavePY filename
```
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SavePY filename</td>
<td>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 Cognos Transformer.</td>
</tr>
</tbody>
</table>

**Example**

This example saves the open model in .py format in the root directory of c: drive, and calls it model.pyj.

SavePY "C:\model.pyj"

**SecurityNamespaceAdd**

The SecurityNamespaceAdd verb defines the IBM Cognos namespace from which the IBM Cognos security objects (groups, roles, users) are retrieved.

The user interface equivalent is to click Assign Security in the Custom View dialog box.

The syntax is as follows:

```
SecurityNameSpaceAdd "namespace" SecurityNamespaceCAMID 'CAMID("::")'
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityNameSpaceAdd &quot;namespace&quot;</td>
<td>Specifies the namespace from which to retrieve the IBM Cognos security objects. The namespace name is defined in IBM Cognos Configuration.</td>
</tr>
<tr>
<td>SecurityNamespaceCAMID 'CAMID(&quot;::&quot;)'</td>
<td>Defines the search path for the CAMID, or security object ID.</td>
</tr>
</tbody>
</table>

**Example**

This example adds the namespace "authors" to the membership in the model.

SecurityNameSpaceAdd "authors" SecurityNamespaceCAMID 'CAMID("::")'

**SecurityNamespaceDelete**

The SecurityNamespaceDelete verb deletes the IBM Cognos namespace from the model.

The user interface equivalent is to click Unassign Security in the Custom View dialog box.

The syntax is as follows:

```
SecurityNameSpaceDelete "namespace name"
```
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityNamespaceDelete &quot;namespace name&quot;</td>
<td>Specifies the namespace to be deleted from the model. The namespace name is defined in Cognos Configuration.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the namespace "authors" from the membership in the model.

`SecurityNamespaceDelete "authors"`

---

**SecurityNamespaceMake**

The `SecurityNamespaceMake` verb defines the Cognos namespace from which the IBM Cognos security objects (groups, roles, users) are retrieved.

The user interface equivalent is to click Assign Security in the Custom View dialog box.

The syntax is as follows:

`SecurityNamespaceMake "namespace name" SecurityNamespaceCAMID 'CAMID(":"')`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityNamespaceMake &quot;namespace name&quot;</td>
<td>Specifies the namespace from which to retrieve the IBM Cognos security objects. The namespace name is defined in IBM Cognos Configuration.</td>
</tr>
<tr>
<td>SecurityNamespaceCAMID 'CAMID(&quot;:&quot;')</td>
<td>Defines the search path for the CAMID, or security object ID.</td>
</tr>
</tbody>
</table>

**Example**

This example adds the namespace "authors" to the membership in the model.

`SecurityNamespaceMake "authors" SecurityNamespaceCAMID 'CAMID(":"')`

---

**SecurityNamespaceUpdate**

The `SecurityNamespaceUpdate` verb updates the namespace from which the IBM Cognos security objects (groups, roles, users) are retrieved.

The user interface equivalent is to modify the assigned security membership in the Custom View dialog box.

The syntax is as follows:

`SecurityNamespaceUpdate "namespace name" SecurityNamespaceCAMID 'CAMID(":"')`
Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityNameSpaceUpdate &quot;namespace name&quot;</td>
<td>Specifies the namespace from which to retrieve the IBM Cognos security objects. The namespace name is defined in IBM Cognos Configuration.</td>
</tr>
<tr>
<td>SecurityNamespaceCAMID 'CAMID(&quot;;&quot;)'</td>
<td>Defines the search path for the CAMID, or security object ID.</td>
</tr>
</tbody>
</table>

**Example**

This example updates the authors namespace.

SecurityNameSpaceUpdate "authors" SecurityNamespaceCAMID'CAMID(";")'

### SecurityObjectAdd

The SecurityObjectAdd verb defines a security object to be imported into the model from the last defined namespace.

The syntax is as follows:

```
SecurityObjectAdd security object ID security object name SecurityNamespace namespace
SecurityObjectDisplayName security object name
SecurityObjectType SecurityType_Group|SecurityType_Role|SecurityType_User
CustomViewList custom view list
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityObjectAdd security object ID security object name</td>
<td>Specifies the security object to be imported into the model from the last defined namespace.</td>
</tr>
<tr>
<td>SecurityNamespace namespace</td>
<td>Specifies the namespace where the security object exists. References can be the ID or the namespace name.</td>
</tr>
<tr>
<td>SecurityObjectDisplayName security object name</td>
<td>Specifies the security object name that is displayed in the model.</td>
</tr>
<tr>
<td>SecurityObjectType SecurityType_Group</td>
<td>SecurityType_Role</td>
</tr>
<tr>
<td>CustomViewList custom view list</td>
<td>Specifies the custom views to which the security object is assigned.</td>
</tr>
</tbody>
</table>

**Example**

The following example adds the security role "Authors"(ID 11531) from the namespace 11533 and assigns it to custom view 11529.

SecurityObjectAdd 11531 'CAMID (";Authors")' SecurityNamespace 11533
SecurityObjectDisplayName "Authors" SecurityObjectType SecurityType_Role
CustomViewList 11529 EndList
**SecurityObjectDelete**

The *SecurityObjectDelete* verb removes a security object from the model.

The syntax is as follows:

```
SecurityObjectDelete securityobjectID security object name
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>SecurityObjectDelete</em> security object ID security object name</td>
<td>Specifies the security object to be removed from the model.</td>
</tr>
</tbody>
</table>

**Example**

The following example removes the security role "Authors"(ID 11531) from the model.

```
SecurityObjectDelete 11531 'CAMID ("Authors")'
```

**SecurityObjectMake**

The *SecurityObjectMake* verb defines a security object to be imported into the model from the last defined namespace.

The syntax is as follows:

```
SecurityObjectMake securityobjectID security object name SecurityNamespace namespace SecurityObjectDisplayName security object name SecurityObjectType SecurityType_Group|SecurityType_Role|SecurityType_User CustomViewList custom view list
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>SecurityObjectMake</em> security object ID security object name</td>
<td>Specifies the security object to be imported into the model from the last defined namespace.</td>
</tr>
<tr>
<td><em>SecurityNamespace</em> namespace</td>
<td>Specifies the namespace where the security object exists. References can be the ID or the namespace name.</td>
</tr>
<tr>
<td><em>SecurityObjectDisplayName</em> security object name</td>
<td>Specifies the security object name that is displayed in the model.</td>
</tr>
<tr>
<td><em>SecurityObjectType</em> SecurityType_Group</td>
<td>SecurityType_Role</td>
</tr>
<tr>
<td><em>CustomViewList</em> custom view list</td>
<td>Specifies the custom views to which the security object is assigned.</td>
</tr>
</tbody>
</table>

**Example**

The following example imports the security role "Authors"(ID 11531) from the namespace 11533 and assigns it to custom view 11529.

```
SecurityObjectMake 11531 11533 'Authors' 11529
```
SecurityObjectUpdate

The SecurityObjectUpdate verb updates a security object in the model.

The syntax is as follows:

SecurityObjectUpdate securityobjectID security object name
SecurityNamespace namespace SecurityObjectDisplayName securityobject name
SecurityObjectType SecurityType_Group | SecurityType_Role | SecurityType_User
CustomViewList custom view list

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityObjectUpdate</td>
<td>Specifies the security object in the model to be updated.</td>
</tr>
<tr>
<td>security object ID security object name</td>
<td></td>
</tr>
<tr>
<td>SecurityNamespace namespace</td>
<td>Specifies the namespace where the security object exists. References can be the ID or the namespace name.</td>
</tr>
<tr>
<td>SecurityObjectDisplayName security object name</td>
<td>Specifies the security object name that is displayed in the model.</td>
</tr>
<tr>
<td>SecurityObjectType SecurityType_Group</td>
<td>SecurityType_Role</td>
</tr>
<tr>
<td>CustomViewList custom view list</td>
<td>Specifies the custom views to which the security object is assigned.</td>
</tr>
</tbody>
</table>

Example

The following example updates the security role "Authors"(ID 11531) from the namespace 11533 by assigning it to an additional custom view 11539).

SecurityObjectUpdate 11531 'CAMID ("Authors")' SecurityNamespace 11533
SecurityObjectDisplayName "Authors" SecurityObjectType SecurityType_Role
CustomViewList 11529 11539 EndList

SignonAdd

The SignonAdd verb is used to add IBM Cognos or data source signon objects in the Cognos Transformer model.

The user interface equivalent is to click the **Insert Signon** option on the **Edit** menu, and follow the prompts to specify the **Signon** properties.

There is no order requirement to the syntax for SignonAdd.

The syntax is as follows:

SignonAdd "signonname" UserId "user name"
password "password" AutoLogon {True|False}
SignonNamespace "namespace id" SignonType"Cognos"
The syntax for a data source signon is as follows:

```plaintext
SignonAdd "signonname" PromptForPassword {False|True} UserId "user name" password "password" SignonType "DataSource"
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignonAdd objSignon</td>
<td>Adds the signon objSignon. objSignon must be the object name, and can include the object identifier.</td>
</tr>
<tr>
<td>signonopts</td>
<td>Optional parameters that describe the signon object in greater detail. Some options are set by default if you do not set them. For the complete list of options, see <a href="#">signonopts</a>.</td>
</tr>
</tbody>
</table>

**Example**

This example creates two signons and two data source signons:

- LDAP1 is an IBM Cognos signon with AutoLogon enabled.
- LDAP2 is an IBM Cognos signon with AutoLogon disabled.
- DataSource1 is a data source signon that does not prompt the user to enter a password.
- DataSource2 is a data source signon that prompts the user to enter a password.

```plaintext
SignonAdd "LDAP1" UserId "user1" password "pwd1" AutoLogon True
SignonNamespace "LDAP" SignonType "Cognos"
SignonAdd "LDAP2" UserId "user2" password "pwd2" AutoLogon False
SignonNamespace "LDAP2" SignonType "Cognos"
SignonAdd "DataSource1" PromptForPassword False UserId "gosales" password "gosales" SignonType "DataSource"
SignonAdd "DataSource2" PromptForPassword True UserId "gosales" password "gosales" SignonType "DataSource"
```

**SignonDelete**

The `SignonDelete` verb deletes an IBM Cognos or data source signon from the model.

The user interface equivalent is to click **Delete** on the **Edit** menu when an object in the **Signon List** is selected.

The syntax is as follows:

```plaintext
SignonDelete objSignon
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignonDelete objSignon</td>
<td>Specifies the signon that is to be deleted. objSignon can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the signon I40_DBASE_NTV_PP74_SAMPLE.

```plaintext
SignonDelete "I40_DBASE_NTV_PP74_SAMPLE"
```
The signonmake verb is used to create IBM Cognos or data source signon objects.

The user interface equivalent is to click the Insert Signon option on the Edit menu, and follow the prompts to specify the Signon properties.

There is no order requirement to the syntax for signonmake.

The syntax is as follows:

```
SignonMake "signonname" UserId "user name" password "password"
   AutoLogon {True|False} SignonNamespace "namespace id" SignonType "Cognos"
```

The syntax for a data source signon is as follows:

```
SignonMake "signonname" PromptForPassword {False|True} UserId "user name"
   password "password" SignonType "DataSource"
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objSignon</td>
<td>Creates the signon objSignon. objSignon must be the object name, and can include the object identifier.</td>
</tr>
<tr>
<td>signonopts</td>
<td>Optional parameters that describe the signon object in greater detail. Some options are set by default if you do not set them. For the complete list of options, see signonopts.</td>
</tr>
</tbody>
</table>

Example

This example creates two IBM Cognos signons and two data source signons:

- LDAP1 is a Cognos signon with AutoLogon enabled.
- LDAP2 is a Cognos signon with AutoLogon disabled.
- DataSource1 is a data source signon that does not prompt the user to enter a password.
- DataSource2 is a data source signon that prompts the user to enter a password.

```
SignonMake "LDAP1" UserId "user1" password "pwd1" AutoLogon True
   SignonNamespace "LDAP" SignonType "Cognos"
SignonMake "LDAP2" UserId "user2" password "pwd2" AutoLogon False
   SignonNamespace "LDAP" SignonType "Cognos"
SignonMake "DataSource1" PromptForPassword False UserId "gosales"
   password "gosales" SignonType "DataSource"
SignonMake "DataSource2" PromptForPassword True UserId "gosales"
   password "gosales" SignonType "DataSource"
```

The signonupdate verb updates IBM Cognos or data source signon objects in the model.

The user interface equivalent is to modify the Signon property sheet.

The syntax for a Cognos signon is as follows:

```
SignonUpdate "signonname" UserId "user name" password "password"
   AutoLogon {True|False} SignonNamespace "namespace id" SignonType "Cognos"
```
The syntax for a data source signon is as follows:

```
SignonUpdate "signonname" PromptForPassword {False|True} UserId "username"
    password "password" SignonType "DataSource"
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignonListUpdate objSignons</td>
<td>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. <code>objSignons</code> can be the object names or object identifiers.</td>
</tr>
<tr>
<td>signonopts</td>
<td>Optional parameters that describe the database connection in greater detail. For the complete list of options, see <code>signonopts</code>. Previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

**Example**

This example updates the LDAP3 IBM Cognos signon object.

```
SignonUpdate "LDAP3" UserId "user3" password "pwd4" AutoLogon False
SignonNamespace "LDAP" SignonType "Cognos"
```

**SourceListUpdate**

The `SourceListUpdate` verb reorders the items in the list of data sources for the model.

The user interface equivalent is to drag the data source objects into the required position in the list.

The syntax is as follows:

```
SourceListUpdate objDataSources EndList
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceListUpdate objDataSources</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Example**

This example reorders 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**

The `SpecialCatAdd` verb creates a special category in the specified dimension.

The user interface equivalent is to open the Diagram, click the right side of a root or special category, and drag the connection to the right.
For more information about creating special categories, see "SpecialCatMake" on page 152.

The syntax is as follows:

**SpecialCatAdd**  
`objSpecialCat [Dimension objDim] Parent objCat [catopts]`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpecialCatAdd</td>
<td>Creates the special category <code>objSpecialCat</code>. <code>objSpecialCat</code> must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Specifies a dimension <code>objDim</code> for the special category. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Parent objCat</td>
<td>Specifies the parent category <code>objCat</code>. <code>objCat</code> can be the object name or object identifier.</td>
</tr>
<tr>
<td>catopts</td>
<td>Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see &quot;catopts&quot; on page 171.</td>
</tr>
</tbody>
</table>

**Example**

This example creates the special category Current Month.

```
SpecialCatAdd "Current Month" Parent "Time"
```

**SpecialCatDelete**

The `SpecialCatDelete` verb removes a special category from the model.

The user interface equivalent is to click **Delete** on the **Edit** menu when a special category is selected.

If a special category has children, the children must be deleted before the special category is deleted or else an error message is issued. This is different from the user interface, where deleting a special category automatically deletes its children.

The syntax is as follows:

**SpecialCatDelete**  
`objSpecialCat [Dimension objDim]`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpecialCatDelete</td>
<td>Deletes the special category <code>objSpecialCat</code>. <code>objSpecialCat</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Is specified, if necessary, to uniquely identify the category. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>
**Example**

This example deletes the special category Current Month.

`SpecialCatDelete "Current Month"`

**SpecialCatMake**

The `SpecialCatMake` verb creates a special category or updates an existing one.

The user interface equivalent, if the special category exists, is to modify the **Special Category** property sheet. For a new special category, open the **Diagram**, click the right side of the root or special category, and drag the connection to the right.

Calculated categories are defined as regular categories, not special categories, in MDL. For more information, see "DimCalcDefMake" on page 118.

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Locating objects uniquely" on page 13.

The syntax is as follows:

```
SpecialCatMake objSpecialCat [Dimension objDim][Parent objLevel][catopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SpecialCatMake objSpecialCat</strong></td>
<td>Creates the category <code>objSpecialCat</code> or modifies it if it exists. <code>objSpecialCat</code> can be the object name, object identifier, or both. Include the object name if the special category does not exist.</td>
</tr>
<tr>
<td><strong>Dimension objDim</strong></td>
<td>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. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Parent objLevel</strong></td>
<td>When creating a category, this specifies its parent and is mandatory. This argument is not needed when updating a category. <code>objLevel</code> can be object name or object identifier.</td>
</tr>
<tr>
<td><strong>catopts</strong></td>
<td>Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see &quot;catopts&quot; on page 171. If the category exists, previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

**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

The SpecialCatUpdate verb updates an existing special category.

The user interface equivalent is to modify the **Special Category** property sheet.

For more information about updating special categories, see “SpecialCatMake” on page 152.

The syntax is as follows:

```
SpecialCatUpdate objSpecialCat [Dimension objDim][catopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpecialCatUpdate objSpecialCat</td>
<td>Updates the special category objSpecialCat. objSpecialCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Is specified, if necessary, to uniquely identify the special category. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>catopts</td>
<td>Optional parameters that describe the category in greater detail. For the complete list of options, see “catopts” on page 171. Previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

**Example**

This example turns off the RollUp attribute for the special category Current Month.

```
SpecialCatUpdate "Current Month" RollUp False
```

SubDimRootMake

The SubDimRootMake verb creates a subdimension or updates the category at the root of an existing subdimension.

The user interface equivalent, if the subdimension exists, is to modify the **Category** property sheet of the category that is at the root of the subdimension. For a new dimension, click the Create/Delete Subdimension option on the Diagram menu when a category is selected.

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 an .mdl file, when the MDL is generated for the subdimension, the following changes occur:

- A new SubDimRootMake statement replaces the CatMake statement for the category.
- New DrillCatMake and LevelMake statements are created and placed immediately after the SubDimRootMake statement.
- All of the categories in the subdimension remain but have new parents, drills, and levels.
Note: Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Locating objects uniquely" on page 13.

The syntax is as follows:
```
SubDimRootMake objCat [Dimension objDim] [Parent objLevel] [Drill objDrillCat] [Levels objLevel] [catopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubDimRootMake objCat</td>
<td>Specifies the category that is to be at the root of the subdimension. objCat can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>When creating a subdimension or updating the root category of a subdimension, this argument is used if necessary, to uniquely identify it. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Parent objLevel</td>
<td>When creating a subdimension, this specifies the parent and must be included. objLevel can be object name or object identifier.</td>
</tr>
<tr>
<td>Drill objDrillCat</td>
<td>When creating a subdimension, this specifies the drill category and is optional. objDrillCat can be the drill category object name, object identifier, or both.</td>
</tr>
<tr>
<td>Levels objLevel</td>
<td>When creating a subdimension, this specifies its level and must be included. objLevel can be the object name or object identifier.</td>
</tr>
<tr>
<td>catopts</td>
<td>Optional parameters that describe objCat in greater detail. Some options are set by default if you do not set them. For the complete list of options, see &quot;catopts&quot; on page 171. If the category exists, previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

Example

This example defines the root category 2006 for the subdimension Years.
```
SubDimRootMake "2006" Dimension "Years" Parent "By Time" Drill "By Time"
Levels "Year" Label "2006"
```

SubDimRootUpdate

The SubDimRootUpdate verb updates the category at the root of an existing subdimension.

The user interface equivalent is to modify the property sheet of the category that is at the root of the subdimension.

For more information about subdimensions, see "SubDimRootMake" on page 153.

The syntax is as follows:
```
SubDimRootUpdate objCat Dimension objDim [catopts]
```
### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SubDimRootUpdate objCat</strong></td>
<td>Updates the category that is at the root of the subdimension. <em>objCat</em> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Dimension objDim</strong></td>
<td>Specifies the dimension in which the subdimension is located. <em>objDim</em> can be the object name, object identifier, or both. This argument is mandatory.</td>
</tr>
<tr>
<td><strong>catopts</strong></td>
<td>Optional parameters that describe the category in greater detail. For the complete list of options, see &quot;catopts&quot; on page 171. Previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

### Example

This example modifies the category Sport Wear, which is at the root of a subdimension, by adding the description "Sports clothing".

```
SubDimRootUpdate "Sport Wear" Dimension "Products" Parent "GO Sport Line" Drill "By Product Line" Levels "Product Type" CatInfo "Sports clothing"
```

### SummarizeCat

The `SummarizeCat` verb summarizes the data for the descendants of a specified category in any PowerCube that is created using the specified dimension view.

The user interface equivalent is to click the `Summarize` option on the `Diagram` menu when a category is selected.

Using `SummarizeCat` on an already-excluded category removes the filter.

**Note:** Category object names differ between MDL and the user interface. MDL uses the category code as the object name. For more information, see "Locating objects uniquely" on page 13.

The syntax is as follows:

```
SummarizeCat objView [Dimension objDim] Category objCat
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SummarizeCat objView</strong></td>
<td>Specifies the dimension view. <em>objView</em> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Dimension objDim</strong></td>
<td>Must be specified if the view is not referenced by object identifier. <em>objDim</em> can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td><strong>Category objCat</strong></td>
<td>Specifies the category to be summarized. <em>objCat</em> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>
Example

This example summarizes the Belgium category in the Europe dimension view.
SummarizeCat "Europe" Dimension "Locations" Category "Belgium"

SummarizeLevel

The SummarizeLevel verb summarizes the data values for the specified level in a view.

The user interface equivalent is to click the Summarize option on the Diagram menu when a level is selected.

The syntax is as follows:
SummarizeLevel objView [Dimension objDim][Drill objDrill] Levels objLevel

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SummarizeLevel objView</td>
<td>Specifies the view. objView can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Must be specified if the view is not referenced by object identifier. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Drill objDrill</td>
<td>Is specified, if necessary, to uniquely identify the view. objDrill can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Levels objLevel</td>
<td>Specifies the level to be summarized. objLevel can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

Example

This example summarizes the Branch Code level in the Europe dimension view.
SummarizeLevel "Europe" Dimension "Locations" Drill "By Region" Levels "Branch Code"

UpdateForwardReference

The UpdateForwardReference verb may be used to resolve forward-referencing problems that can occur when you change a model by using a new verb in the model definition file. There is no user interface equivalent.

For more information about setting the VerbOutput option, see “Cognos Transformer and verb MDL” on page 7.

Processing times may be slower when you use the UpdateForwardReference verb.

The syntax is as follows:
UpdateForwardReference
**UpdatePowerCubes**

The `UpdatePowerCubes` verb allows you to update the metadata in the .mdc files for existing PowerCubes without updating the data.

The user interface equivalents are the **Update PowerCubes** and **Update Selected PowerCubes** options on the **Run** menu.

The syntax is as follows:

```
UpdatePowerCubes [objCube][updatepowercubeopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UpdatePowerCubes</strong> objCube</td>
<td>Updates the PowerCube <code>objCube</code>. <code>objCube</code> can be the object name, object identifier, or both. If no PowerCube is specified, all are updated.</td>
</tr>
<tr>
<td><code>updatepowercubeopts</code></td>
<td>Optional parameters that describe the PowerCube in greater detail. These parameters are: DrillThrough {True</td>
</tr>
</tbody>
</table>

**Example**

This example updates the metadata in the cube1.mdc file.

```
UpdatePowerCubes mycube.mdc UserClasses True
```

**ViewAdd**

The `ViewAdd` verb adds a dimension view. If the object already exists in the model, an error message is issued.

The user interface equivalent is to click **Add New View** in the menu option that is accessed by right-clicking the relevant portion of the view pane (**Diagram**).

The syntax is as follows:

```
ViewAdd objView [Dimension objDim] [Type viewtype] [viewopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ViewAdd</strong> objView</td>
<td>Adds the dimension view <code>objView</code>. <code>objView</code> must be the object name and can include the object identifier.</td>
</tr>
<tr>
<td><strong>Dimension</strong> objDim</td>
<td>Is specified to locate the dimension view within a dimension. <code>objDim</code> can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>
**Argument Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong> viewtype</td>
<td>Specifies the type of view. viewtype can be All, Omit or Custom. If not specified, the default is Custom.</td>
</tr>
<tr>
<td>viewopts</td>
<td>Optional parameters that describe the view in greater detail. Some options are set by default if you do not set them. For the complete list of options, see viewopts.</td>
</tr>
</tbody>
</table>

**Example**

This example creates a Europe view for the Locations dimension and applies the Apex action. The ViewSecurity argument is blank (null), indicating that this is a dimension view.

```
ViewAdd "Europe" Dimension "Locations" ViewSecurity "" Apex "Europe"
```

For examples of custom views that use IBM Cognos security objects rather than user classes, see the security chapter in the IBM Cognos Transformer User Guide.

---

**ViewDelete**

The ViewDelete verb removes a dimension view from the model.

The user interface equivalent is to click **Delete** in the menu that is accessed by right-clicking a dimension view.

You cannot delete a default view.

The syntax is as follows:

```
ViewDelete objView [Dimension objDim]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewDelete objView</td>
<td>Deletes the dimension view objView. objView can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Must be specified if the view is not referenced by object identifier. objDim can be the object name, object identifier, or both.</td>
</tr>
</tbody>
</table>

**Example**

This example deletes the dimension view Europe.

```
ViewDelete "Europe" Dimension "Locations"
```

**ViewListUpdate**

The ViewListUpdate verb reorders a dimension view list.

The user interface equivalent is to drag the dimension views into the required order in the list.
You cannot move the default views. All Categories and Omit Dimensions always display as the first and second items in the list.

The syntax is as follows:

```
ViewListUpdate [Dimension objDim] StartList objViews EndList
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension objDim</td>
<td>Must be specified if the views are not referenced by object identifier. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>StartList objViews</td>
<td>Lists the dimension views in the desired order. The dimension views can be identified by object name or object identifier.</td>
</tr>
</tbody>
</table>

**Example**

This example moves the views Far East and Europe to the top of the custom views in the dimension view list. The default views still always display first.

```
ViewListUpdate Dimension "Locations" StartList "Far East" "Europe" EndList
```

**ViewMake**

The ViewMake verb creates a dimension view or updates an existing one.

The user interface equivalent, when an existing view is selected, is to click Options on the Diagram menu. For a new view, click the Add New View option in the menu that is accessed by right-clicking a dimension view.

The syntax is as follows:

```
ViewMake objView [Dimension objDim][Type viewtype][viewopts]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewMake objView</td>
<td>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.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>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.</td>
</tr>
<tr>
<td>Type {All</td>
<td>Omit</td>
</tr>
<tr>
<td>viewopts</td>
<td>Optional parameters that describe the view in greater detail. Some options are set by default if you do not set them. For the complete list of options, see [viewopts]. If the view exists, previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>
Example

This example defines a Europe view, which shows only the data for that region. The ViewSecurity argument is blank (null), indicating that this is a dimension view.

ViewMake "Europe" Dimension "Locations" ViewSecurity 0 Apex "Europe"

This example shows the two default views for the Products dimension. Cognos Transformer creates these default views for each dimension in the model. Again, ViewSecurity is blank or null.

ViewMake "All Categories" Dimension "Products" Type All ViewSecurity 0
ViewMake "Omit Dimension" Dimension "Products" Type Omit ViewSecurity 0

ViewUpdate

The ViewUpdate verb updates an existing dimension view.

The user interface equivalent is to click Options on the Diagram menu when an existing view is selected.

You cannot update the default views that Cognos Transformer creates for each dimension (Omit Dimension and All Categories).

For more information about updating views, see "ViewMake" on page 159.

The syntax is as follows:

ViewUpdate objView [Dimension objDim][Type viewtype][viewopts]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewUpdate objView</td>
<td>Specifies the view to update. objView can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Dimension objDim</td>
<td>Must be specified if the view is not referenced by object identifier. objDim can be the object name, object identifier, or both.</td>
</tr>
<tr>
<td>Type viewtype</td>
<td>Specifies the type of view. viewtype can be All, Omit, or Custom. If not specified, the existing view type is retained.</td>
</tr>
<tr>
<td>viewopts</td>
<td>Optional parameters that describe the view in greater detail. For the complete list of options, see viewopts. Previously set options are retained unless you change them with this command.</td>
</tr>
</tbody>
</table>

Example

This example applies a summarize operation to the Belgium category in the Europe dimension view.

ViewUpdate "Europe" Dimension "Locations" Summary "Belgium"
Chapter 7. MDL options

This section describes the Model Definition Language (MDL) options you can use with IBM Cognos Transformer to define and manipulate objects.

Each grouping applies to one or more Cognos Transformer verbs and the objects on which they operate. The option descriptions include some or all of the following:

- Purpose of the option group
- The location of the equivalent functionality on the Windows interface (UI)
- Verb syntax
- Notes
- Examples

Before you begin, please review the Chapter 5, “Syntax conventions,” on page 61 and the topic that explains when to use object identifiers or category codes: “Locating objects uniquely” on page 13.

For more information about Cognos Transformer features and functionality, see the IBM Cognos Transformer User Guide.

**appqueryopts**

Use appqueryopts to set the options that apply to the following MDL verbs:

- DataSourceAdd
- DataSourceMake
- DataSourceUpdate
- NewModel

You can use MDL to specify a package, report, or query from an IBM Cognos data source and, optionally, any filter references.

**AppInfo**

The AppInfo option allows you to add a description that provides details about the query.

The syntax is as follows:

```markdown
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**

The CharacterSet option specifies the type of character set used in the query.

The syntax is as follows:

```markdown
CharacterSet charset
```

where `charset` is one of Default, ANSI, OEM, Unicode or Multibyte.
Example

CharacterSet Multibyte

CognosSource

The CognosSource option specifies an IBM Cognos package or report data source in the model. All package or report definitions are added before query definitions.

The syntax is as follows:

```
CognosSource id "name"
```

where `name` is the name of the package in the Cognos Transformer model. All packages, reports, and queries must have unique names.

Example

```
CognosSource 103 "Go Sales and Retailers"
```

Columns

The Columns option adds information about the type of column used by the data source.

This option relates to the structure of the source data and should not be manually set.

The syntax is as follows:

```
Columns {True|False}
```

where `False` indicates that the data is fixed length.

Example

```
Columns True
```

DataRange

The DataRange option adds information about the table ranges used from the source data.

This option applies only to queries made against Excel databases.

The syntax is as follows:

```
DataRange string
```

where `string` is the name of the Excel data range.

Example

```
DataRange "reptotal_rng"
```

DecimalSep

The DecimalSep option specifies the character used as the decimal separator for the data values.

The syntax is as follows:

```
DecimalSep string
```
where \textit{string} is the decimal separator.

\textbf{Example}

\texttt{DecimalSep "."}

\textbf{ImrName}

The \texttt{ImrName} option specifies the drill-through target for an IQD data source.

This option is required for drill-through to an Impromptu report from an IQD data source.

The syntax is as follows:

\texttt{ImrName \textit{string}}

where \textit{string} is a file name and optional path.

\textbf{Example}

\texttt{ImrName "C:installation_directory\PowerCubes and Reports\bsc_msrs.imr"}

\textbf{Isolation}

The \texttt{Isolation} option provides information about the IQD data source.

This option applies only to IQD data sources.

The syntax is as follows:

\texttt{Isolation \textit{number}}

where \textit{number} is a number from 0 to 6 that corresponds to the following list:

<table>
<thead>
<tr>
<th>Isolation level</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>0</td>
</tr>
<tr>
<td>Read Uncommitted</td>
<td>1</td>
</tr>
<tr>
<td>Read Committed</td>
<td>2</td>
</tr>
<tr>
<td>Cursor Stability</td>
<td>3</td>
</tr>
<tr>
<td>Reproducible Read</td>
<td>4</td>
</tr>
<tr>
<td>Phantom Protection</td>
<td>5</td>
</tr>
<tr>
<td>Serializable</td>
<td>6</td>
</tr>
</tbody>
</table>

\textbf{Example}

\texttt{Isolation 2}

\textbf{ModelStamp}

The \texttt{ModelStamp} option assigns a numeric code to the model.

This option is for internal use only.

The syntax is as follows:

\texttt{ModelStamp \textit{bignum}}
Example
ModelStamp 827358629

ModelNonRollupHierarchies
The ModelNonRollupHierarchies option specifies how to define hierarchies based on non-rollup children of the root category in a time dimension when there is only one non-rollup child.

Starting with IBM Cognos Transformer version 10.2.0, all non-rollup children of the time dimension root category are defined as separate hierarchies in a PowerCube.

In PowerCubes created with Transformer versions earlier than 10.2.0, when there is only one non-rollup child of the root category in a time dimension, that category is not considered to be a base of a hierarchy. Instead, each of its children defines a hierarchy.

The syntax is as follows:
ModelNonRollupHierarchies {True|False}

where True means that the rule for defining hierarchies in a time dimension introduced in Cognos Transformer version 10.2.0 is applied.

If the ModelNonRollupHierarchies option does not exist in the MDL, the value is based on the MDL Version value. If the version is earlier than 10.2.0, False is used. If the version is 10.2.0 or later True is used. If no Version is specified in the MDL, the value defaults to the build version of Cognos Transformer, which results in using True.

Example
ModelNonRollupHierarchies True

PackageReportSource
The PackageReportSource option specifies a reference to the package on which an IBM Cognos query is defined.

The syntax is as follows:
PackageReportSource source ID "source name"

Example
PackageReportSource 103 "Go Sales and Retailers"

PackageTimeStamp
The PackageTimeStamp option specifies the version of the package last used.

The syntax is as follows:
PackageTimeStamp "pathandtimestamp"

Examples
PackageTimeStamp "/content/package[@name='GO Sales and Retailers200701011200']/model[@name='model']"
PackageTimeStamp "/content/package[@name='Go Sales']/model[@name='2008-04-16T15:41:14.078Z']"
**PreSummarized**

The PreSummarized option specifies whether measures are externally rolled up before they are brought into the model.

The syntax is as follows:

```
PreSummarized {True|False}
```

**Example**

```
PreSummarized False
```

**SegmenterPrompt**

The SegmenterPrompt option specifies the prompt used to segment a SAP BW fact query during stream extract.

The syntax is as follows:

```
SegmenterPrompt string
```

where `string` is the name of the prompt used for segmenting the query.

**Example**

```
SegmenterPrompt "WhichBrand"
```

**SegmenterPromptEnabled**

The SegmenterPromptEnabled option specifies whether stream extract is to be used for eligible SAP BW fact queries.

The syntax is as follows:

```
SegmenterPromptEnabled {True|False}
```

**Example**

```
SegmenterPromptEnabled True
```

**Separator**

The Separator option specifies the character used to separate character-delimited fields in the data source.

This option applies only to sources that use delimited-field text or delimited-field text with column titles.

The syntax is as follows:

```
Separator string
```

**Example**

```
Separator ","
```

**SetCurrent**

The SetCurrent option specifies whether the source data sets the current date in the time dimension for the model.

The syntax is as follows:

```
SetCurrent {True|False}
```
**Example**
SetCurrent False

**Source**
The Source option specifies the name and location of the source file for the model.

The syntax is as follows:
```
Source string
```
where `string` is a file name and optional path.

**Example**
Source "c:installation_directory\cubes and reports\prodinfo.csv"

**SourceInfo**
The SourceInfo option specifies additional descriptive information about the source file for the model.

The syntax is as follows:
```
SourceInfo 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."

**SourcePath**
The SourcePath option specifies the path to the IBM Cognos package or report within Content Manager.

The syntax is as follows:
```
SourcePath "path"
```

**Example**
SourcePath "\content\package [@name='Go Sales and Retailers']"

**SourceSignonList**
The SourceSignonList option specifies the signons that apply to the data source.

The syntax is as follows:
```
SourceSignonList signonlist EndList
```
where `signonlist` is the list of signons for a data source, where these are required. Signons can be the object name, object identifier, or both.

**Example**
SourceSignonList 105 EndList

**SourceType**
The SourceType option allows you to specify any supported data source for the model.
For an IQD data source, the keyword is `DataSource`. However, for an IBM Cognos single query, this keyword must be qualified by specifying `SourceType CognosSourceQuery`.

Similarly, the keyword for an IBM Cognos package or report is `CognosSource`. It must be qualified by specifying `SourceType package` or `SourceType report`.

The syntax is as follows:

```
SourceType sourceType
```

where `sourceType` is one of `FlatFileColNames`, `FlatFile`, `DataSource`, `Access`, `AccessQuery`, `ExcelCrosstab`, `ExcelDatabase`, `PowerHousePortable`, `FixedASCII`, `FixedASCIINoCrlf`, `Dictionary` or, for IBM Cognos data sources, one of either `CognosSourceQuery`, `package`, or `report`.

**Example**

The following example specifies an IBM Cognos package as the source type:

```
CognosSource 103 "GO Sales" SourceType package SourcePath
"/content/package[@name='GO Sales']"
```

The following example specifies an IBM Cognos report as the source type:

```
CognosSource 104 "GO Sales" SourceType report SourcePath
"/content/package[@name='GO Sales']/report[@name='GO SalesReport']"
```

The following example specifies a single query created using an IBM Cognos report as the source:

```
DataSource 116 "Time" Separator "," SourceType CognosSourceQuery CharacterSet Default DecimalSep "." ThousandSep "," Columns True Timing Pop Yes Create Default
Package Report Source 104 "GO Sales Report" Set Current True Speed False
Presummarized False
```

**Speed**

The `Speed` option allows you to specify whether query optimization or uniqueness verification checks are enabled for your model.

The syntax is as follows:

```
Speed {True|False}
```

where `True` enables the Maximize Data Access Speed processing option. `False` enables the Verify Category Uniqueness processing option.

**Example**

```
Speed False
```

**SQL**

The `SQL` option represents the query as a Structured Query Language (SQL) string.

This option cannot be changed by the user. Any changes must be made in the source.

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.

The syntax is as follows:

```
SQL ' string [string...]'
```

**Example**

The following represents the SQL for the Main IQD data source:

```
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" * T3."PROD_COST") as c7,' '(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")
AND ' > 0.65)' THEN ('"20% - 65%")' WHEN (((T2."QTY" * T2."PRICE") -
'(T2."QTY" * T3."PROD_COST") / (T2."QTY" * T2."PRICE")
' THEN ('"Over 65%")' ELSE (''ERROR'')' END) as c10 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**

The Stamp option assigns a numeric code to the model for internal use.

This option is for internal use only.

The syntax is as follows:

```
Stamp number
```

**Example**

```
Stamp 890332424
```

**StreamExtractSize**

The StreamExtractSize option specifies the size of the stream extract buffer used for an SAP BW fact query.

The syntax is as follows:

```
StreamExtractSize number
```

where number is the size in MB of the stream extract buffer, or 0. The number 0 disables stream extracts for the query.

**Example**

```
StreamExtractSize 10
```

**SuppressNull**

The SuppressNull option specifies whether to suppress null values in IBM Cognos packages based on a SAP BW data source.

The syntax is as follows:

```
SuppressNull suppressnull
```

where suppressnull is one of SuppressNullYes, SuppressNullNo, or SuppressNullModel.
Example
SuppressNull SuppressNullYes

**ThousandSep**
The ThousandSep option specifies which character demarcates numbers in the thousands, or larger.

The syntax is as follows:

**ThousandSep** *string*

**Example**
ThousandSep "," 

**Timing**
The Timing option specifies when categories are generated during cube-building.

The syntax is as follows:

**Timing** *datasourcetiming*

where *datasourcetiming* is one of PopNoCreateNo, PopYesCreateNo, PopNoCreateDefault, PopYesCreateDefault, PopNoCreateYes, PopYesCreateYes. These, in turn, relate to the following Windows interface settings:

<table>
<thead>
<tr>
<th><em>datasourcetiming</em></th>
<th>Windows interface setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>PopNo</td>
<td>Generate Categories is not selected</td>
</tr>
<tr>
<td>PopYes</td>
<td>Generate Categories is selected</td>
</tr>
<tr>
<td>Create Default</td>
<td>PowerCube Creation and Default are selected</td>
</tr>
<tr>
<td>CreateYes</td>
<td>PowerCube Creation and Create the PowerCubes are selected</td>
</tr>
<tr>
<td>CreateNo</td>
<td>PowerCube Creation and Generate Categories Only are selected</td>
</tr>
</tbody>
</table>

**Example**
Timing PopYesCreateNo

**UpdateCycle**
The UpdateCycle option is set by Cognos Transformer for internal use only.

The syntax is as follows:

**UpdateCycle** *bignum*

**Version**
The Version option is set by Cognos Transformer. It should not be set manually.

The syntax is as follows:

**Version** *string*

**Example**
Version "7.4 1012"
assocopts

Use assocopts to set the options that apply to the following MDL verbs:
AssociationAdd, AssociationMake, AssociationUpdate, CurrencyTableAdd,
CurrencyTableMake, CurrencyTableUpdate, DimensionAdd, DimensionMake,
DimensionUpdate, LevelAdd, LevelMake, LevelUpdate, MeasureAdd, MeasureMake, and
MeasureUpdate.

**AssociationContext**

The `AssociationContext` option specifies the context for an object in the overall model.

The syntax is as follows:

```
AssociationContext objcontext
```

where `objcontext` can be the object name or object identifier or both.

**Example**

```
AssociationContext 2979 "Branch Code"
```

**AssociationReferenced**

The `AssociationReferenced` option specifies the reference used for the association.

The syntax is as follows:

```
AssociationReferenced objref
```

where `objref` is the name of the association

**Example**

```
AssociationReferenced "Branch Code"
```

**AssociationRole**

The `AssociationRole` option specifies the role played by the associated object in the model.

The syntax is as follows:

```
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` is one of `Role_CountryCode`, `Role_Date`, `Role_Label`, or
`Role_Rate`.

**Example**

```
AssociationRole Role_Label
```

**AssociationType**

The `AssociationType` option specifies the type of object represented by the association.

The `Type_Query` option is used for Impromptu Query Definition (IQD) data sources.
The syntax is as follows:

**AssociationType** *objtype*

where *objtype* is one of *Type_PowerCube* or *Type_Query*.

**Example**

AssociationType Type_PowerCube

---

**catopts**

Use *catopts* to set the options that apply to the following MDL verbs: CatAdd, CatMake, CatUpdate, SpecialCatAdd, SpecialCatMake, SpecialCatUpdate, DrillCatMake, RootCatMake, RootCatUpdate, SubDimRootMake, and SubDimRootUpdate.

---

**Blanks**

The *Blanks* option indicates the presence of missing values for regular categories.

For more information about how null and missing values are treated in the supported IBM Cognos reporting applications, see the Measures chapter of the Cognos Transformer *User Guide*.

This option applies to regular categories only.

The syntax is as follows:

**Blanks** {True|False}

where true means the category is blank (has missing values).

**Example**

Blanks False

---

**Calc**

The *Calc* option specifies a calculation and the model objects to which it applies.

The Calculation keywords supported by Cognos Transformer are as follows:

<table>
<thead>
<tr>
<th>Calculation keyword</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>Number</td>
<td>Divides.</td>
</tr>
<tr>
<td>-</td>
<td>Date or Number</td>
<td>Subtracts.</td>
</tr>
<tr>
<td>+</td>
<td>Character, Date, or Number</td>
<td>Adds.</td>
</tr>
<tr>
<td>^</td>
<td>Number</td>
<td>Returns a number raised to the power of a second number.</td>
</tr>
<tr>
<td>Accumulate</td>
<td>Number</td>
<td>Computes the running total.</td>
</tr>
<tr>
<td>AccPOB</td>
<td>Number</td>
<td>Computes the running total as a percentage of the total.</td>
</tr>
<tr>
<td>Calculation keyword</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Average</td>
<td>Number</td>
<td>Averages.</td>
</tr>
<tr>
<td>Max</td>
<td>Number</td>
<td>Represents a maximum instance.</td>
</tr>
<tr>
<td>Min</td>
<td>Number</td>
<td>Represents a minimum instance.</td>
</tr>
<tr>
<td>*</td>
<td>Number</td>
<td>Multiplies.</td>
</tr>
<tr>
<td>Percent-Growth</td>
<td>Number</td>
<td>Computes the percentage change that parameter 2 is, relative to parameter 1.</td>
</tr>
<tr>
<td>Share</td>
<td>Number</td>
<td>Computes the share that parameter 2 is, relative to parameter 1.</td>
</tr>
<tr>
<td>Change</td>
<td>Number</td>
<td>Computes the difference between two parameters (parameter 2 minus parameter 1).</td>
</tr>
</tbody>
</table>

The syntax is as follows:

```
Calc calculation
```

where `calculation` is a combination of `objects` and `keywords`.

`objects` consist of the object name followed by the at sign (\@), the object type and, optionally, the object identifier. The object type can be `Category`, `Level`, or `Drill`, such as `GO Water Bottle@Category@4805`.

**Example**

The following examples represent two commonly encountered calculation specifications:

- `Calc Percent-Growth ("Prior YTD","YTD")`
- `Calc Share ("GO Water Bottle@Category@4805, "Sport Wear@Category@4799")`

**CalcDef**

The `CalcDef` option references a calculation defined elsewhere in the model.

The `CalcDef` option can serve as a forward reference for the `DimCalcDef` object. A `CatMake` statement that references a calculation can appear in the MDL file before the calculation object is defined using the `DimCalcDefMake` statement.

The syntax is as follows:

```
CalcDef objDimCalcDef
```

where `objDimCalcDef` can be the object name or object identifier.
Example
CalcDef 6375

CatInfo
The CatInfo option adds a category description to the model.

The CatInfo option describes regular, special, drill, and root categories.

The syntax is as follows:
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! Determine why these items are profitable
and use the information to improve the performance of others in the group."

ContextLevel
The ContextLevel option specifies the level on which a relative time period is
based, establishing the context for the target period.

The string parameter is context-sensitive.

The syntax is as follows:
ContextLevel string

where objDimCalcDef can be the object name or object identifier.

Example
ContextLevel "Quarter"

ContextOffset
The ContextOffset option specifies the number of periods by which the relative
time period is offset (or removed) from the current period, where 0 corresponds to
the current period. For grouped categories, this value defined the range of relative
time categories in the group.

The syntax is as follows:
ContextOffset number

Example
ContextOffset -1

Current
The Current option specifies the date category that will serve as the current
period.

The syntax is as follows:
Current

Example
Current
**Date**

The `Date` option specifies the start date for a date category. This option is required.

The syntax is as follows:

```plaintext
Date date
```

where `date` is the start of the date period.

**Example**

```
Date 19990101
```

**DateDrill**

The `DateDrill` option specifies the drill category for the time dimension.

The syntax is as follows:

```plaintext
DateDrill objDrillCat
```

where `objDrillCat` is the object name or object identifier.

**Example**

```
DateDrill "By Time"
```

**ExtraWeek**

The `ExtraWeek` option specifies the conditions under which Cognos Transformer adds extra days as a separate week, to resynchronize a lunar quarter or year with the regular calendar. You can specify that an extra week never be added, or only added if there are exactly 7 extra days. Or, you can specify some other condition that will trigger the addition of an extra week, such as 4 or more remainder days.

The syntax is as follows:

```plaintext
ExtraWeek extraweek
```

where `extraweek` is one of None, 7only, , 6orMore 5orMore, or 4orMore.

**Example**

```
ExtraWeek 4orMore
```

**Filtered**

The `Filtered` option specifies whether categories are excluded from the model.

The syntax is as follows:

```plaintext
Filtered {True|False}
```

**Example**

```
Filtered True
```

**Format**

The `Format` option specifies the number of decimal places in a numeric measure.

The syntax is as follows:

```plaintext
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**

The Inclusion option specifies whether categories are included or suppressed.

This parameter applies to regular, special, and drill categories.

The syntax is as follows:

\[ \text{Inclusion} \text{inclusion} \]

where inclusion is one of Default, Generate, Suppress, Retain, or Filtered.

The Windows interface equivalents of these are as follows:

- Default (when needed)
- Always Include
- Suppress (on the Diagram menu)
- Include When Needed or Suppress Blank Categories
- Exclude (on the Diagram menu)

**Example**

This example suppresses the specified category:

\[ \text{Inclusion} \text{Suppress} \]

**IsKeyOrphanage**

The IsKeyOrphanage option specifies whether the category in a manual level is a placeholder that will serve as the parent category for any newly generated categories that display in the next lower source level.

Only one orphanage is allowed in a lower manual level for each source level category.

For easier reference, you can specify that the category code of the orphanage be shown for a particular category.

The syntax is as follows:

\[ \text{IsKeyOrphanage} \ (\text{True}|\text{False}) \]

**Example**

\[ \text{IsKeyOrphanage False} \]

**IsTruncated**

The IsTruncated option specifies whether an overly long category identifier has been truncated.

The syntax is as follows:
IsTruncated \{True|False\}

Example
IsTruncated True

Label
The Label option adds a descriptive name to clearly identify a category.

The syntax is as follows:
Label stringwithnewline

Example
Label "Day Tripper"

LastUse
The LastUse option specifies the date when a category was created or last updated in the model.

The syntax is as follows:
LastUse date

where date is in the format YYYYMMDD.

Example
Lastuse 20070324

NewPartition
The NewPartition option specifies the partition level for a category.

The syntax is as follows:
NewPartition uns

where uns is an unsigned number that represents the partition level of that category.

Example
NewPartition 1

HideValue
The HideValue option hides meaningless measure values, either at the root or at any other category level in a scenario dimension.

Use this option with DimDefaultCategory to specify a new default category at a lower level, having meaningful measure values, where you want the cube to open.

The syntax is as follows:
HideValue \{True|False\}

Example
RootCatUpdate "Products" Dimension "Products" Inclusion Generate ... HideValue True
Orphanage
The Orphanage option is automatically set by Cognos Transformer when a manual category is added. It is for internal use only.

The syntax is as follows:
Orphanage

Example
Orphanage

PartialWeek
The PartialWeek option specifies how partial-week categories are to be allocated or split when they span two higher-level time periods.

The syntax is as follows:
PartialWeek partialweek

where partialweek is one of First, Last, Largest, Split, SplitIfGreater, or None. The None parameter does not have a Windows interface equivalent. The Windows interface equivalents of the other parameters are as follows:
• First Period
• Last Period
• Largest Period
• Always Split
• Split > 1 Day)

Example
This example splits the categories into two weeks if there is more than one day in the extra partial week:
PartialWeek SplitIfGreater

Primary
The Primary option specifies the primary drill category for a set of categories.

The syntax is as follows:
Primary objCat

where objCat can be the object name or object identifier.

Example
Primary "20071201-20071231"

PrimaryDrill
The PrimaryDrill option specifies whether the category is in the primary drill path.

The syntax is as follows:
PrimaryDrill {True|False}
Example
PrimaryDrill True

Rollup
The Rollup option specifies whether the measure values from the category should be rolled up, or summarized.

The syntax is as follows:
Rollup {True|False}

Example
Rollup True

RunningPeriods
The RunningPeriods option specifies the number of relative time periods for an N-period running total.

The syntax is as follows:
RunningPeriods uns

where uns is the number of running periods to include.

Example
RunningPeriods 0

Share
The Share option applies a Cognos Transformer-specific calculation that computes the proportionate share of an item relative to its parent category.

For more information, see the Cognos Transformer User Guide.

The syntax is as follows:
Share objCat

where objCat is the object name or object identifier.

Example
Share "Back Packs"

ShortName
The ShortName option specifies a shorter name for the category.

You can use this option with regular, special, drill, and root categories.

The syntax is as follows:
ShortName string

Example
ShortName "ENV"
Sign

The Sign option specifies whether the sign of a measure value associated with a category is to be reversed, changing it from a negative to a positive number, or from a positive to a negative number.

This option is useful if you need to convert values into the form required by specific financial reports, such as debits and credits, or assets and liabilities.

You can use this option with regular and special categories.

The syntax is as follows:

```
Sign {True|False}
```

where True means that the sign is reversed.

Example

```
Sign True
```

SourceValue

The SourceValue option specifies the name by which a source column is identified.

This option applies to regular categories only.

The syntax is as follows:

```
SourceValue stringwithnewline
```

Example

```
SourceValue "Environmental Line"
```

SplitWeek

The SplitWeek option specifies whether extra days should be divided between two weeks.

This option is for internal use only.

The syntax is as follows:

```
SplitWeek {True|False}
```

Example

```
SplitWeek False
```

Suppressed

The Suppressed option specifies whether the category should have its values suppressed (hidden).

You can use this option with regular and special categories.

The syntax is as follows:

```
Suppressed {True|False}
```

where True means that the category values are suppressed (hidden).
Example
Suppressed True

TargetLevel
The TargetLevel option specifies the target category for a relative time definition or, for to-date or N-period running totals, the granularity of periods that you set to compute the totals.

The syntax is as follows:
TargetLevel string

Example
TargetLevel "Month"

TargetOffset
The TargetOffset option sets the number of periods by which the relative time category is offset (or removed) from the current period. The effect of your selection depends on whether you are creating a relative time category for a Single Category, a Period To-Date, or an N-Period Running Total.

The syntax is as follows:
TargetOffset number

Example
TargetOffset -1

TimeAggregate
The TimeAggregate option specifies the type of relative time, such as To-Date, Running Total, or a grouped version of these.

The syntax is as follows:
TimeAggregate timeaggregate

where timeaggregate is one of None, Single, ToDate, ToDate_Grp, Running, or Running_Grp.

Example
TimeAggregate ToDate

ToDateLevel
The ToDateLevel option specifies the level to which the Period To-Date relative time category applies.

The syntax is as follows:
ToDateLevel objLevel

where objLevel is the object name.

Example
ToDateLevel "Quarter"
**WeekBegins**
The `WeekBegins` option specifies the day that each week in the model begins.

The syntax is as follows:

```
WeekBegins day
```

where `day` is one of Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, or Saturday.

**Example**
WeekBegins Sunday

**YearBegins**
The `YearBegins` option specifies the date that the fiscal year begins, where this differs from the start of the calendar year.

The syntax is as follows:

```
YearBegins date
```

where `date` is in the format YYYYMMDD

**Example**
YearBegins 20070401

---

**cognospackageopts**
Use `cognospackageopts` to set the options that apply to the following MDL verbs: `CognosPackageAdd`, `CognosPackageMake`, and `CognosPackageUpdate`.

**Description**
The `Description` option adds a description of the source column associated with the package.

The syntax is as follows:

```
Description objCol
```

where `objCol` is the object name of a column.

**Example**
Description "Product types"

**PackageTimeStamp**
The `PackageTimeStamp` option specifies the version of the package last used.

The syntax is as follows:

```
PackageTimeStamp "pathandtimestamp"
```

**Examples**
PackageTimeStamp "/content/package[@name='GO Sales and Retailers2007011200']/model[@name='model']"
PackageTimeStamp "/content/package[@name='Go Sales']/model[@name='2008-04-16T15:41:14.078Z']"
SourcePath
The SourcePath option specifies the path to the IBM Cognos package or report within Content Manager.

The syntax is as follows:
SourcePath "path"

Example
SourcePath "/content/package[@name='Go Sales and Retailers']"

SourceType
The SourceType option specifies which type of metadata source is being referenced in Content Manager. The valid values are package or report.

The syntax is as follows:
SourceType sourceType

where sourceType is one of package or report.

Example
The following example specifies an IBM Cognos package as the source type:
CognosSource 103 "GO Sales" SourceType package SourcePath "/content/package[@name='GO Sales']"

The following example specifies an IBM Cognos report as the source type:
CognosSource 104 "GO Sales" SourceType report SourcePath "/content/package[@name='GO Sales']/report[@name='GO SalesReport']"

colopts
Use colopts to set the options that apply to the following MDL verbs: ColumnAdd, ColumnMake, ColumnUpdate, DataSourceAdd, DataSourceMake, DataSourceUpdate, SourceAttributeAdd, SourceAttributeMake, and SourceAttributeUpdate.

Calc
The Calc column option defines the calculation and the objects involved in the calculation.

The syntax is as follows:
Calc keywords objects

where objects are the object name followed by the at sign (@) and the object identifier; for example, "Revenue@259"

The keywords used with the Calc column option are as follows:

<table>
<thead>
<tr>
<th>Calculation keyword</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>Number</td>
<td>Divided by</td>
</tr>
<tr>
<td>=</td>
<td>Boolean</td>
<td>Equals</td>
</tr>
<tr>
<td>Calculation keyword</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>&gt;</td>
<td>Boolean</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Boolean</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>Boolean</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Boolean</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>(</td>
<td></td>
<td>Left parenthesis</td>
</tr>
<tr>
<td>-</td>
<td>Date or Number</td>
<td>Minus</td>
</tr>
<tr>
<td>*</td>
<td>Number</td>
<td>Multiplied by</td>
</tr>
<tr>
<td>+</td>
<td>Character, Date, or Number</td>
<td>Adds numbers or concatenates strings</td>
</tr>
<tr>
<td>^</td>
<td>Number</td>
<td>Returns a number raised to the power of a second number</td>
</tr>
<tr>
<td>)</td>
<td></td>
<td>Right parenthesis</td>
</tr>
<tr>
<td>Absolute</td>
<td>Number</td>
<td>Converts numbers to their unsigned value</td>
</tr>
<tr>
<td>Add-Days</td>
<td>Date</td>
<td>Returns the date resulting from adding a number of days to a date</td>
</tr>
<tr>
<td>Add-Months</td>
<td>Date</td>
<td>Returns the date resulting from adding a number of months to a date</td>
</tr>
<tr>
<td>Add-Years</td>
<td>Date</td>
<td>Returns the date resulting from adding a number of years to a date</td>
</tr>
<tr>
<td>Age</td>
<td>Date</td>
<td>Age</td>
</tr>
<tr>
<td>Ceiling</td>
<td>Number</td>
<td>Returns a number rounded to the next highest integer</td>
</tr>
<tr>
<td>Char-Length</td>
<td>Number</td>
<td>Returns the number of characters in a string</td>
</tr>
<tr>
<td>Date-To-Days-From-1900</td>
<td>Number</td>
<td>Returns the number of days from January 1, 1900 inclusive</td>
</tr>
<tr>
<td>Day</td>
<td>Number</td>
<td>Returns the day of the month (1-31) from a date</td>
</tr>
<tr>
<td>Calculation keyword</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Days-From-1900-To-Date-Time</td>
<td>Date</td>
<td>Returns the date obtained from converting a number of days from January 1, 1900 inclusive to a date</td>
</tr>
<tr>
<td>Days-To-End-Of-Month</td>
<td>Date</td>
<td>Returns the number of days to the last day of the month represented by a date</td>
</tr>
<tr>
<td>Else</td>
<td>Boolean</td>
<td>Else</td>
</tr>
<tr>
<td>First-Of-Month</td>
<td>Date</td>
<td>Returns the first day of a month from a date or datetime</td>
</tr>
<tr>
<td>First-Word</td>
<td>Character</td>
<td>Returns the first word of a string</td>
</tr>
<tr>
<td>Floor</td>
<td>Number</td>
<td>Returns a number rounded down to the next lowest integer</td>
</tr>
<tr>
<td>If</td>
<td>Boolean</td>
<td>If</td>
</tr>
<tr>
<td>Integer-Divide</td>
<td>Number</td>
<td>Returns the integer obtained by truncating the result of an integer divided by a second integer</td>
</tr>
<tr>
<td>IsNull</td>
<td>Character, Date, or Number</td>
<td>Is null</td>
</tr>
<tr>
<td>Last-Of-Month</td>
<td>Date</td>
<td>Returns the last day of a month from a date or datetime</td>
</tr>
<tr>
<td>Left</td>
<td>Character</td>
<td>Returns a specific number of characters, starting at the left of the string</td>
</tr>
<tr>
<td>Lower</td>
<td>Character</td>
<td>Converts uppercase characters to lowercase</td>
</tr>
<tr>
<td>Mod</td>
<td>Number</td>
<td>Returns the remainder (modulus) of an integer divided by a second integer</td>
</tr>
<tr>
<td>Month</td>
<td>Number</td>
<td>Returns the month number from a date, datetime, or interval</td>
</tr>
<tr>
<td>Months-Between</td>
<td>Number</td>
<td>Returns the integer number of months from a date to a second date</td>
</tr>
<tr>
<td>Calculation keyword</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Not</td>
<td>Boolean</td>
<td>Not</td>
</tr>
<tr>
<td>Number-To-String</td>
<td>Character</td>
<td>Returns the string representation of a number rounded to the next integer</td>
</tr>
<tr>
<td>Or</td>
<td>Boolean</td>
<td>Or</td>
</tr>
<tr>
<td>Position</td>
<td>Number</td>
<td>Returns the starting position of a string in a second string</td>
</tr>
<tr>
<td>Reverse</td>
<td>Character</td>
<td>Returns a string with the order of the characters reversed</td>
</tr>
<tr>
<td>Right</td>
<td>Character</td>
<td>Returns a specific number of characters, starting at the right of the string</td>
</tr>
<tr>
<td>Round-Down</td>
<td>Number</td>
<td>Returns a number rounded down</td>
</tr>
<tr>
<td>Round-Near</td>
<td>Number</td>
<td>Returns a number rounded to the nearest integer</td>
</tr>
<tr>
<td>Round-Up</td>
<td>Number</td>
<td>Returns a number rounded up</td>
</tr>
<tr>
<td>Round-Zero</td>
<td>Number</td>
<td>Returns a number rounded towards zero</td>
</tr>
<tr>
<td>SqRt</td>
<td>Number</td>
<td>Returns the square root of a number</td>
</tr>
<tr>
<td>String-To-Number</td>
<td>Number</td>
<td>Converts a string to a number</td>
</tr>
<tr>
<td>SubString</td>
<td>Character</td>
<td>Returns a substring from a string</td>
</tr>
<tr>
<td>Then</td>
<td>Boolean</td>
<td>Then</td>
</tr>
<tr>
<td>Today</td>
<td>Date</td>
<td>Returns the current date according to your computer clock</td>
</tr>
<tr>
<td>Trim-Leading</td>
<td>Character</td>
<td>Returns a string with leading spaces removed</td>
</tr>
<tr>
<td>Trim-Trailing</td>
<td>Character</td>
<td>Returns a string with trailing spaces removed</td>
</tr>
<tr>
<td>Calculation keyword</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Upper</td>
<td>Character</td>
<td>Converts lowercase characters to uppercase</td>
</tr>
<tr>
<td>Year</td>
<td>Number</td>
<td>Returns the year from the date</td>
</tr>
<tr>
<td>Years-Between</td>
<td>Number</td>
<td>Returns the number of years from a date to another date</td>
</tr>
</tbody>
</table>

**Example**

Calc "Order Qty" * "Price"

**Class**

The **Class** option specifies the data class associated with the column.

The syntax is as follows:

```
Class [dataclass]
```

where `dataclass` is one of Default or no setting, Description, Date, Quantity, Ignore, or Member. The Windows interface equivalents are:

- Unspecified
- Text
- Date
- Numeric
- Ignore
- Array Member

**Example**

```
Class Quantity
```

**ColSrcType**

The **ColSrcType** option defines the type of data source for the column.

This option is for internal use only.

The syntax is as follows:

```
ColSrcType colsrcertype
```

where `colsrcertype` is one of SAPMeasure, SAPBusinessKey, SAPLevelKey, SAPOther, RLDimensional, OLAPDIMENSIONAL, or SAPCalcmeasure.

**Example**

```
ColSrcType RLDimensional
```

**Column**

The **Column** option specifies the original name for the column in the data source.

You must specify the column.
If you change a column name in MDL, the object name may differ from the one on the Windows interface. This is because MDL takes the object name from the Original Name box on the Column property sheet, whereas Cognos Transformer takes the object name from the Column Name box of the Column property sheet.

The syntax is as follows:

```
Column objCol
```

where `objCol` is the object name.

**Example**

```
Column "Product ID"
```

### ColumnInfo

The ColumnInfo option provides a description of the column in the model.

The syntax is as follows:

```
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 product sales"
```

### Dateconstant

The Dateconstant option must be inserted before any date constant used in a calculation.

The syntax is as follows:

```
Dateconstant 'date'
```

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

**Example**

```
months-between (dateconstant '2007-01-01', today)
```

### DateLevel

The DateLevel option specifies the level of the time dimension associated with the column.

The syntax is as follows:

```
DateLevel datelevel
```

where `datelevel` is one of None, Year, Quarter, Month, Week, or Day.

**Example**

```
DateLevel Day
```
Decimals

The Decimals option specifies the number of decimal spaces in the source data.

You should set this option to the actual value in the data source.

You can also set this option on the Windows interface, by using the Decimal Places box on the Format tab of the Measure property sheet.

The syntax is as follows:

Decimals uns

Example

Decimals 2

Detail

The Detail option specifies the degree of detail for the column.

You can select from a context-sensitive list of valid codes in the Windows interface.

For example, if you do not want to include all of the categories in a level, you can set the lowest level of detail to include. That way, you can avoid unwanted entries or duplicate names that need to be qualified by the name of their owning category, such as Brand (OF Tents).

The syntax is as follows:

Detail string

where string can be "Unspecified"

Example

Detail "Unspecified"

Format

The Format option specifies the date input format used by the date values in the column that populates the time dimension.

The syntax is as follows:

Format [dateformat]

where dateformat is one of PreDefined, YMD, DMY, MDY, YM, MY, Y, Q, M, W, or D.

Example

Format YMD

InputScale

The InputScale option specifies the number used to scale the data values in the column.

The syntax is as follows:

InputScale number

where number is not greater than 16.
Example
InputScale 1

Offset
The Offset option specifies the number by which the column position is offset.

The syntax is as follows:
Offset uns

Example
Offset 2

Origin
The Origin option specifies the type of data source used for the column.

You should not manually set this option.

The syntax is as follows:
Origin {Source|Generated|Calculated|Manual}

where Generated is the default; Source specifies an IQD data source; Calculated specifies a calculated column; and Manual specifies a manually created column in the model.

Example
Origin Generated

Scale
The Scale column option specifies the output scale.

This option is either read directly from the data source or specified in the measure definition. For more information, see “Scale” on page 214.

The syntax is as follows:
Scale uns

where uns is an unsigned number that specifies the output scale.

Example
Scale 0

Size
The Size column option specifies the size of certain types of data source.

This column option only applies when the SourceType of the associated data source is PowerHousePortable, FixedASCII, or FixedASCIINoCrLf.

The syntax is as follows:
Size uns

Example
Size 1
Storage

The Storage option specifies the actual storage type used by the source data for the column.

This column option relates directly to the source data.

The syntax is as follows:

Storage storagetype

where storagetype is one of Default, Int8, Int16, Int32, Float64, Text, Character, or LongText.

Example

Storage Int32

TimeArray

The TimeArray option specifies the type of array used to group items for the time dimension.

The syntax is as follows:

TimeArray timearray

where timearray is one of Off, Quarter, Month, Month13, Quarter6, Month18, Quarter8, or Month24.

Example

TimeArray Month13

TimeArrayCol

The TimeArrayCol option specifies the name of the array used to group items for the time dimension.

The syntax is as follows:

TimeArrayCol string

Example

TimeArrayCol "Time"

TimeArrayMonth

The TimeArrayMonth option specifies the month number on which the time array is based.

The syntax is as follows:

TimeArrayMonth uns

Example

TimeArrayMonth 2

currencyrecordopts

Use currencyrecordopts to set the options that apply to the following MDL verbs: CurrencyAdd, CurrencyMake, and CurrencyUpdate.
**BaseCurrency**

The `BaseCurrency` option defines the default currency stored in the Currency Table for the model.

The syntax is as follows:

```
BaseCurrency
```

where the Windows interface equivalent is to specify a `<Base Default>` name in the Currency Record dialog box, accessible by clicking the Properties button on the Currency Table dialog box.

**Example**

```
BaseCurrency
```

**CountryCode**

The `CountryCode` option specifies the 3-character international code used to represent each country or region listed in the Currency Table.

The syntax is as follows:

```
CountryCode code
```

where `code` is a unique identifier for a country or region.

**Example**

```
CountryCode "USA"
```

**CurrencyCountryLabel**

The `CurrencyCountryLabel` option specifies the name used to uniquely identify each country or region listed in the Currency Table.

The syntax is as follows:

```
CurrencyCountryLabel label
```

where `label` is a unique identifier for a country or region.

**Example**

```
CurrencyCountryLabel "Canada"
```

**CurrencyDecimals**

The `CurrencyDecimals` option specifies the number of decimal places used when displaying the converted currency values for each country or region listed in the Currency Table. However, this value may not be the same as the decimal place setting used in your reporting application. Depending on your implementation, it may be overridden from the Measure property sheet, or at run time.

The syntax is as follows:

```
CurrencyDecimals decimals
```

where `decimals` can be a number from 0 to 9. The default used with the Great Outdoors.mdl sample is 0. The Cognos Transformer Currency Table default is 2.

**Example**

```
CurrencyDecimals 5
```
CurrencyFormatOverride

The CurrencyFormatOverride option specifies whether the currency format settings can be overridden elsewhere in the IBM Cognos modeling or reporting applications.

The syntax is as follows:

```
CurrencyFormatOverride {True|False}
```

Example

```
CurrencyFormatOverride True
```

CurrencyIsEMU

The CurrencyIsEMU option specifies whether the country or region is a member of the European Monetary Union (EMU) and therefore uses the euro rather than a national currency.

The syntax is as follows:

```
CurrencyIsEMU {True|False}
```

Example

```
CurrencyIsEMU True
```

CurrencySymbol

The CurrencySymbol option specifies the symbol used with each currency listed in the Currency Table.

The syntax is as follows:

```
CurrencySymbol symbol
```

where the default label is *.

Example

```
CurrencySymbol "$"
```

EmuEntryDate

The EMUEntryDate option specifies the date when the country or region became a member of the European Monetary Union (EMU) and started using the euro rather than a national currency.

The syntax is as follows:

```
EmuEntryDate date
```

where date is in YYYYMMDD format

Example

```
EmuEntryDate 19990101
```

EuroBaseCurrency

The EuroBaseCurrency option establishes the euro as the base currency in the Currency Table for this model.

The syntax is as follows:
currencytableopts

Use currencytableopts to set the options that apply to the following MDL verbs: CurrencyTableAdd, CurrencyTableMake, CurrencyTableUpdate, and CurrencyTableDelete.

**CurrencyCountryCodeColumn**

The `CurrencyCountryCodeColumn` option specifies the source column that provides the country or region code used in the Currency Table.

The syntax is as follows:

```
CurrencyCountryCodeColumn objCol
```

where `objCol` is the object name and you must specify `CurrencyCountryCodeColumn`, `CurrencyLabelColumn`, or both.

**Example**

```
CurrencyCountryCodeColumn "Country_Code"
```

**CurrencyDateColumn**

The `CurrencyDateColumn` option specifies the source column that provides the effective date for the conversion rate. It is a mandatory parameter when converting currencies based on information in the Currency Table.

The syntax is as follows:

```
CurrencyDateColumn objCol
```

where `objCol` is the object name.

**Example**

```
CurrencyDateColumn "Currency Date"
```

**CurrencyLabelColumn**

The `CurrencyLabelColumn` option specifies the source column that provides the currency label used in the Currency Table.

The syntax is as follows:

```
CurrencyLabelColumn objCol
```

where `objCol` is the object name and you must specify `CurrencyCountryCodeColumn`, `CurrencyLabelColumn`, or both.

**Example**

```
CurrencyLabelColumn "Currency_Label"
```
**CurrencyRateColumn**

The `CurrencyRateColumn` option specifies the source column that provides the conversion rates. It is a mandatory parameter when converting currencies based on information in the Currency Table.

The syntax is as follows:

```
CurrencyRateColumn objCol
```

where `objCol` is the object name.

**Example**

```
CurrencyRateColumn "Conversion_Rate"
```

**CurrencyTableType**

The `CurrencyTableType` option specifies the type of Currency Table used in the model.

The syntax is as follows:

```
CurrencyTableType type
```

where `type` is one of `BaseTable`, `EuroTable`, or `OtherTable`.

**Example**

```
CurrencyTableType BaseTable
```

**deletionopts**

Use `deletionopts` to set the options that apply to the MDL verb `DeletionListUpdate`.

**CalcDef**

The `CalcDef` option specifies a calculation definition used in the dimension.

The syntax is as follows:

```
CalcDef objDimCalcDef
```

where `objDimCalcDef` is the object identifier.

**Example**

```
CalcDef 6375
```

**Column**

The `Column` option specifies a column used in the dimension.

The syntax is as follows:

```
Column objCol
```

where `objCol` is the object identifier.

**Example**

```
Column 1092
```
**Cube**

The *Cube* option specifies a cube used in the model.

The syntax is as follows:

```
Cube objCub
```

where *objCub* is the object identifier.

**Example**

```
Cube 1601
```

**DataSource**

The *DataSource* option specifies a data source used for the model.

The syntax is as follows:

```
DataSource objDataSource
```

where *objDataSource* is the object identifier.

**Example**

```
DataSource 4213
```

**Dimension**

The *Dimension* option specifies a dimension used in the model.

The syntax is as follows:

```
Dimension objDim
```

where *objDim* is the object identifier.

**Example**

```
Dimension 1844
```

**Levels**

The *Level* option specifies a level in a dimension used in the model.

The syntax is as follows:

```
Levels objLev
```

where *objLev* is the object identifier.

**Example**

```
Levels 1385
```

**Measure**

The *Measure* option specifies a measure used in the model.

The syntax is as follows:

```
Measure objMeasure
```

where *objMeasure* is the object identifier.
Example
Measure 1500

Signon
The Signon option specifies a signon used in the model.

The syntax is as follows:

\texttt{Signon \textit{objSignon}}

where \textit{objSignon} is the object identifier.

Example
Signon 9876

View
The View option specifies a dimension or security view used in the model.

For more information about custom views based on IBM Cognos security objects, see the security chapter of the Cognos Transformer \textit{User Guide}.

The syntax is as follows:

\texttt{View \textit{objView}}

where \textit{objView} is the object identifier.

Example
View 1092

dimopts
The dimopts options apply to the following MDL verbs: DimAdd, DimMake, and DimUpdate.

Association
The Association option specifies the associated column from the data source.

The syntax is as follows:

\texttt{Association \textit{objCol}}

where \textit{objCol} is the object name.

Example
Association "Time Category Code"

DaysInWeek
The DaysInWeek option specifies each day in the week as a unique, but non-sequential number.

To specify the number of days in each week, add the appropriate day codes. For example, a Monday-to-Friday week is represented by \((2 + 4 + 8 + 16 + 32) = 62\). A seven-day Sunday-to-Saturday week (the default) is represented by the number 127.
The syntax is as follows:

\[ \text{DaysInWeek } bignum \]

where \( bignum \) is based the sum of the bit representations for the days in the week, as follows:
- Sunday = 1
- Monday = 2
- Tuesday = 4
- Wednesday = 8
- Thursday = 16
- Friday = 32
- Saturday = 64

**Example**

DaysInWeek 62

**DimDefaultCategory**

The `DimDefaultCategory` option specifies a new default category, at a lower level in the cube, where a scenario dimension opens, to show report users meaningful data values.

The syntax is as follows:

\[ \text{DimDefaultCategory } objDimDefaultCategory \]

where `objDimDefaultCategory` is the object identifier.

**Example**

DimUpdate Dimension 185 "Products" DimType Regular NewCatsLock False ExcludeAutoPartitioning False DimDefaultCategory 293

**DimInfo**

The `DimInfo` option provides additional information about the dimension.

The syntax is as follows:

\[ \text{DimInfo } string [string...] \]

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**

The `DimType` option specifies the dimension type. Note that time dimensions are defined as a specific type (Date) but scenario dimensions are not.

The syntax is as follows:

\[ \text{DimType } \{ \text{Regular|Date|NonStandard} \} \]

where NonStandard is only used for time dimensions in which the associated data source has PreSummarized set to True, and the Association option is either blank or not set.
**Example**

DimType Date

**EarliestDate**

The EarliestDate option specifies the earliest valid date for the model.

The syntax is as follows:

\[ \text{EarliestDate date} \]

where \( date \) is in the format YYYYMMDD. The default is 19000101.

**Example**

EarliestDate 19000101

**ExcludeAutoPartitioning**

The ExcludeAutoPartitioning option specifies whether auto-partitioning will be used as a model optimization strategy.

The syntax is as follows:

\[ \text{ExcludeAutoPartitioning \{True|False\}} \]

where the default is False.

**Example**

ExcludeAutoPartitioning True

**LatestDate**

The LatestDate option specifies the latest valid date for the model.

The syntax is as follows:

\[ \text{LatestDate date} \]

where \( date \) is in the format YYYYMMDD. The default is 99991231.

**Example**

LatestDate 99991231

**ManualPeriods**

The ManualPeriods option specifies whether the current time period will be manually or automatically set.

The syntax is as follows:

\[ \text{ManualPeriods \{True|False\}} \]

where the default is False, which is equivalent to selecting the **Automatically Set Current Time Period** option on the Windows interface. True means that the current time period will be manually set.

**Example**

ManualPeriods False
**NewCatsLock**

The NewCatsLock option specifies whether new categories will be automatically created in the dimension as they are encountered in the data source.

The syntax is as follows:

```
NewCatsLock {True|False}
```

where the default is False. True means that the dimension is locked.

**Example**

```
NewCatsLock False
```

---

**filteropts**

Use filteropts to set the options that apply to the following MDL verbs: CognosPackageFilterAdd, CognosPackageFilterDelete and CognosPackageFilterUpdate.

**CognosPackageFilterRef**

The CognosPackageFilterRef option specifies a valid filter reference in an IBM Cognos package or report data source.

```
CognosPackageFilterRef filter reference
```

**Example**

```
CognosPackageFilterRef "[gosales_goretailers].[Asia Pacific]"
```

**DATASOURCE**

The DATASOURCE option specifies the ID of the IBM Cognos package or report data source.

```
DATASOURCE datasourceID
```

**Example**

```
DATASOURCE 103
```

**CognosPackageFilterDelete**

The CognosPackageFilterDelete option is used to delete filters from the MDL script.

The syntax is as follows:

```
CognosPackageFilterDelete filter_name
```

**Example**

```
CognosPackageFilterDelete "Tents1"
```

**CognosPackageFilterUpdate**

The CognosPackageFilterUpdate option is used to update filters that are part of the MDL script.

The syntax is as follows:

```
CognosPackageFilterUpdate "filter_name" DATASOURCE query_id
Cognos PackageFilterRef "package_reference_name"
```
where query_id is the object id of the query that is based on the package and package_reference_name is the name of the package followed by the name of the filter. The package that is in the package_reference_name must be in the data source that is being queried.

**Example**

`CognosPackageFilterUpdate "Tents2" DATASOURCE 105 CognosPackageFilterRef "[oracle_gosales].[Tents]"

**ExpMark**

The `ExpMark` option specifies the start of the SQL string used to query the column in the data source. The end of the string is denoted by `ExpMarkEnd`.

The syntax is as follows:

```
ExpMark SQLstring ExpMarkEnd
```

**FilterDescription**

The `FilterDescription` option adds a description to explain the filter used when querying a column in the data source.

The syntax is as follows:

```
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**

Use `levelopts` to set the options that apply to the following MDL verbs: `LevelAdd`, `LevelMake`, and `LevelUpdate`.

**Association**

The `Association` option defines the connection between a level and columns in a data source.

The syntax is as follows:

```
Association objCol
```

where `objCol` is the object name.

**Examples**

`Association "Time Category Code"

**Blanks**

The `Blanks` option indicates how missing values are represented within that level.

For more information about how null and missing values are treated in the supported IBM Cognos reporting applications, see the Measures chapter of the Cognos Transformer User Guide.
The syntax is as follows:

\texttt{Blanks string}

where "N/A" represents missing values, or those which are "Not Available".

**Example**

\texttt{Blanks "N/A"}

**CategoryCode**

The CategoryCode option specifies the code used to uniquely identify a category at that level.

The syntax is as follows:

\texttt{CategoryCode objCol}

where \texttt{objCol} is the object name.

**Example**

\texttt{CategoryCode "Product Line"}

**CatLabFormat**

The CatLabFormat option specifies the format used for the date category labels at that level.

The syntax is as follows:

\texttt{CatLabFormat string}

where \texttt{string} is the desired format.

**Example**

The following format string produces the label 2006 Q3:

\texttt{CatLabFormat 'YYYY "Q"Q'}

**DateFunction**

The DateFunction option specifies the time period calculation used to generate that level in the time dimension.

The syntax is as follows:

\texttt{DateFunction datefunction}

where \texttt{datefunction} is one of None, Year, LunarYear, Quarter, LunarQuarter, Month, LunarMonth, LunarMonth445, LunarMonth454, LunarMonth544, Week, or Day.

**Example**

The following date function generates lunar months, where each 13-week lunar quarter consists of a 4-week month, a 4-week month, and a 5-week month:

\texttt{CDateFunction LunarMonth445}

**Description**

The Description option adds a description of the source column associated with the level.
The syntax is as follows:

**Description objCol**

where *objCol* is the object name of a column.

**Example**

Description "Product types"

**Filtered**

The *Filtered* option specifies whether the categories in the level are excluded (hidden).

The syntax is as follows:

*Filtered* {True|False}

**Example**

Filtered True

**Format**

The *Format* option specifies the date format used for the time categories in the level.

The syntax is as follows:

*Format* *string*

where *string* is a recognized date format code, such as YMD.

**Example**

Format "YMD"

**Generate**

The *Generate* option specifies the generation date to use when determining which categories to generate.

The syntax is as follows:

*Generate* *generatedate*

where *generatedate* is one of Default, All, or Need. On the Windows interface, selecting the check box generates all categories in the period. Clearing the check box is equivalent to the Need setting.

**Example**

Generate All

**Inclusion**

The *Inclusion* option specifies how the categories in a level display in the model and in the resulting cubes and reports.

The syntax is as follows:

*Inclusion* *inclusion*

where *inclusion* is one of Default, Generate, Suppress, Retain, or Filtered.
On the Windows interface, equivalents display on the **Inclusion** or **Diagram** menus. Note that Generate equates to **Always Include**, Retain equates to **Include When Needed** or **Suppress Blank Categories**, and Filtered equates to **Exclude**.

**Example**

Inclusion Suppress

**LevelInfo**

The **LevelInfo** option adds a description for the level.

The syntax is as follows:

*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 reseller type influences the amount of sales."

**Label**

The **Label** option specifies a label for the level.

The syntax is as follows:

*Label objCol*

where *objCol* is the object name.

**Example**

Label "Customer Name"

**NewCatsLock**

The **NewCatsLock** option specifies whether new categories will be automatically created in the level as they are encountered in the data source.

The syntax is as follows:

*NewCatsLock {True|False}*

where the default is False. True means that the dimension is locked.

**Example**

NewCatsLock False

**OrderBy**

The **OrderBy** option specifies the value to use as the basis for ordering categories, within the specified drill category.

The syntax is as follows:

*OrderBy Drill objDrillCat Value*  
*string*

where *objDrillCat* can be the object name, object identifier, or both; and *string* is what displays in the **Order Value** box on the Windows interface.
**Example**

OrderBy Drill 6881 Value "200712"

**SortOrder**

The SortOrder option specifies the type of data used as the basis for ordering the categories, such as text. It is followed by the SortAs option, which determines the sorting sequence, which can be in either ascending or descending order.

If the data is alphanumeric, use the Character storage type. If the data is purely numeric, use the Float64 storage type.

The syntax is as follows:

`SortOrder Drill objDrillCat Column objCol SortOrder storagetype SortAs {Ascending|Descending}`

where `objDrillCat` and `objCol` can be the object name, object identifier, or both; and `storagetype` is one of Default, Int8, Int16, Int32, Float64, Text, Character, or Longtext.

**Example**

SortOrder Character

**SortAs**

The SortAs option determines the sorting sequence, which can be in either ascending or descending order. It follows the SortOrder option, which specifies the type of data used as the basis for sorting.

If the data is alphanumeric, use the Character storage type. If the data is purely numeric, use the Float64 storage type.

The syntax is as follows:

`SortOrder Drill objDrillCat Column objCol SortOrder storagetype SortAs {Ascending|Descending}`

where `objDrillCat` and `objCol` can be the object name, object identifier, or both; and `storagetype` is one of Default, Int8, Int16, Int32, Float64, Text, Character, or Longtext.

**Example**

SortAs Ascending

**Partition**

The Partition option specifies the partition number assigned to that level.

The syntax is as follows:

`Partition uns`

where `uns` is an unsigned number from 0 to 15.

**Example**

Partition 3
**RefreshLabel**

The `RefreshLabel` option specifies whether to refresh the label for the level.

The syntax is as follows:

```
RefreshLabel {True|False}
```

**Example**

`RefreshLabel True`

---

**RefreshDescription**

The `RefreshDescription` option specifies whether to refresh the description for the level.

The syntax is as follows:

```
RefreshDescription {True|False}
```

**Example**

`RefreshDescription False`

---

**RefreshShortName**

The `RefreshShortName` option specifies whether to refresh the short name for the level.

The syntax is as follows:

```
RefreshShortName {True|False}
```

**Example**

`RefreshShortName False`

---

**Share**

The `Share` option specifies the ancestor level with which the current level has a shared, or proportional, relationship.

The `Share` function computes the share that an item is, relative to its parent. For more information about this Cognos Transformer-specific function, see the Cognos Transformer User Guide.

The syntax is as follows:

```
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**

The `ShortName` option specifies the short name that identifies the level.

The syntax is as follows:

```
ShortName objCol
```
where objCol is the object name.

**Example**

ShortName "CUST_SHRTN"

**Source**
The **Source** option specifies the data column that serves as the source for this level.

The syntax is as follows:

```
Source objCol
```

where objCol is the object name.

**Example**

Source "Customer Number"

**Suppressed**
The **Suppressed** option specifies whether data is to be suppressed (hidden) for this level.

The syntax is as follows:

```
Suppressed {True|False}
```

**Example**

Suppressed True

**TimeRank**
The **TimeRank** option specifies the relative ranking of user-defined levels in the time dimension.

Normally this value is automatically set by Cognos Transformer to remove ambiguities that can arise, for example, in relative time calculations. It is the value relative to other date value ranks rather than the absolute value, that is important. The Windows interface equivalent is the **Time Level Ranking** field.

The syntax is as follows:

```
TimeRank uns
```

where uns is an unsigned number representing the time-level ranking.

**Example**

TimeRank 10

**UniqueCategories**
The **UniqueCategories** option specifies whether the categories in this level are unique.

The syntax is as follows:

```
UniqueCategories {True|False}
```

**Example**

UniqueCategories True
**UniqueMove**

The `UniqueMove` option specifies whether the moved categories are unique.

The syntax is as follows:

```
UniqueMove {True|False}
```

**Example**

```
UniqueMove True
```

**meaopts**

Use `meaopts` to set the options that apply to the following MDL verbs: MeasureAdd, MeasureMake, and MeasureUpdate.

**ActivityMeasure**

The `ActivityMeasure` option specifies the criteria used to determine the category count.

By default, the non-zero-value categories associated with all measures in the model are counted.

To create a Category Count, you can also use the `Dimension` or `Rollup` options. For more information about these alternatives, see "Dimension" on page 210 and "Rollup" on page 213.

The syntax is as follows:

```
ActivityMeasure objMeasure
```

where `objMeasure` is the object name. The default, when no ActivityMeasure is specified but one or more measures exist for a category, is to return a count.

**Example**

The scope of this category count is any existing measure:

```
MeasureMake 159 "Revenue"  Levels 139 RollupCategoryCount Storage DefaultScale 0 Decimals 0 Sign False IsCurrency False DrillThrough False EndList
```

The following category count is for the specific measure referenced using `ActivityMeasure`:

```
ActivityMeasure "Sales"
Rollup CategoryCount Storage Default Scale 0 Decimals 0 Sign False IsCurrency False DrillThrough False EndList
```

**Allocation**

The `Allocation` option specifies the type of allocation and a measure to which it applies.

The syntax is as follows:

```
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.
Example

Allocation None

Association

The Association option specifies the source column associated with this measure.

The syntax is as follows:

Association objCol

where objCol is the column name.

Example

Association "Cost"

Calc

The Calc option specifies a calculation and the measure(s) to which it applies.

The syntax is as follows:

Calc keywords objects

where calc is a mixture of objects and keywords.

Objects consist of the object name, optionally followed by the at sign (@) and the object identifier, such as "Revenue@259". If object identifiers are suppressed, objects are identified by their object names only.

The measure calculation keywords supported by Cognos Transformer are as follows:

<table>
<thead>
<tr>
<th>Calculation keyword</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISNULL</td>
<td>N/A</td>
<td>zero</td>
</tr>
<tr>
<td>IF</td>
<td>N/A</td>
<td>conditional IF expression</td>
</tr>
<tr>
<td>THEN</td>
<td>N/A</td>
<td>conditional THEN expression</td>
</tr>
<tr>
<td>ELSE</td>
<td>N/A</td>
<td>conditional ELSE expression</td>
</tr>
<tr>
<td>AND</td>
<td>N/A</td>
<td>both</td>
</tr>
<tr>
<td>OR</td>
<td>N/A</td>
<td>either</td>
</tr>
<tr>
<td>NULL</td>
<td>N/A</td>
<td>does not exist</td>
</tr>
<tr>
<td>!=</td>
<td>N/A</td>
<td>not equal to</td>
</tr>
<tr>
<td>=</td>
<td>N/A</td>
<td>equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>N/A</td>
<td>less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>N/A</td>
<td>less than or equal to</td>
</tr>
</tbody>
</table>
The following example represents an if-then-else calculation definition for "Revenue":

Measure 173 "Revenue" calc IF (ISNULL("Cost@177"+"Quantity@181")) THEN ("Cost@177") ELSE ("Quantity@181")

Decimals

The Decimals option specifies the number of decimal spaces in the source data.

You should set this option to the actual value in the data source.

You can also set this option on the Windows interface by using the Decimal Places box on the Format tab of the Measure property sheet.

The syntax is as follows:

Decimals uns
**Example**

Decimals 2

**Dimension**

The Dimension option specifies the drill category, levels, and/or dimension to which the measure applies.

If the measure is a Category Count, you can also use the ActivityMeasure or Rollup options. For more information about these alternatives, see "ActivityMeasure" on page 207 and "Rollup" on page 213.

The syntax is as follows:

```
Dimension objDim Drill objDrillCat Levels objLevel
```

where `objDim`, `objDrillCat`, and `objLevel` are the relevant object identifiers.

**Example**

Dimension "Products" Levels "Product ID"

**DrillThrough**

The DrillThrough option specifies whether drill-through actions are enabled and lists the supported target reports, such as the relevant .imr file for Impromptu Query Definition (IQD) data sources.

The syntax is as follows:

```
DrillThrough {True|False} [ string ] [ string ]... EndList
```

where `string` is a report name with optional path.

**Example**

In this example, the drill-through operation is not selected:

```
DrillThrough False EndList
```

In this example, the drill-through operation is enabled to the specified .imr report:

```
DrillThrough True "c:\installation_directory\bsc_msrs.imr" EndList
```

**DuplicateRollup**

The DuplicateRollup option specifies how records with duplicate values will be rolled up or summarized.

The syntax is as follows:

```
DuplicateRollup duplicatesrollup
```

where `duplicatesrollup` is one of Default, Sum, Minimum, Maximum, Average, First, or Last.

**Example**

In this example, the rollup function is explicitly set to add (sum) all duplicate records:

```
DuplicateRollup Sum
```
**DuplicateWeight**

The `DuplicateWeight` option, along with the time state, specifies the weighting factors and functions used to obtain the weighted average of duplicate records for a measure.

The syntax is as follows:

```
DuplicateWeight objMeasure
```

where `objMeasure` is the object name.

**Example**

In this example, the `Revenue` measure is the weighting factor to use when determining the average rollup value of a set of duplicate records:

```
DuplicateWeight "Revenue"
```

---

**Exclude**

The `Exclude` option, followed by a list of excluded objects, restricts the scope of drill-through options so that only the non-excluded level(s) can serve as a starting point for drill-through operations. Use with the `DrillThrough` option to ensure that report users navigate directly to the relevant portion of the cube before they begin a drill-through operation.

For each or cube or measure for which you want to restrict drill-through by level, you must specify the relevant dimension and levels, and any drill-through targets 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 of the Windows interface.

The syntax is as follows:

```
Exclude [Dimension objDim] [Drill objDrill] Levels objLevel
```

where `objDim`, `objLevels`, and `objDrill` can be the unique object identifiers, names, or both.

**Example**

In this example, the `State` and `Market` levels in the `Region` and `Sales` dimensions are excluded for the specified cube drill-through operations:

```
DrillThrough True ... "C:\path_to_cube.mdc" "Description text" Exclude Dimension "Region" Exclude Levels 219 "State" Exclude Dimension "Sales" Exclude Levels "Market" EndExclude "C:\path_to_cube2.mdc" "Description text2" EndList
```

---

**Format**

The `Format` option specifies the measure format and the number of decimal places to use.

The syntax is as follows:

```
Format string
```

where `string` is the format, followed by a tilde (`~`), and the number of decimal places.
Example
In this example, the measure format is #,#0 and the number of decimal places is set to 2:
Format ",#0"2

IgnoreMissingValue
The IgnoreMissingValue option specifies whether 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) applies.

You cannot use the IgnoreMissingValue option with 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.

Null and missing data values are always excluded from Min and Max calculations for rollups, whether they are set by Cognos Transformer to display as 0 or na (the NA setting).

If you do not specify this option, its value is assumed to be False.

For more information about specifying measures, see "MeasureMake" on page 132.

The syntax is as follows:

\[\text{TimeStateRollup } \text{type}\]

where \(\text{type}\) is one of Average or Weighted Average.

Example
This example uses structured MDL to ignore missing values for a Weighted Average time state rollup:

```plaintext
Measure 173 "Cost" TimeStateRollup Average TimeStateWeight "Outlet"
\textbf{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"
```

This example uses verb MDL to ignore missing values for an Average time state rollup:

```plaintext
MeasureCreate 173 "Profit" TimeStateRollup Average \textbf{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"
```

IsCurrency
The IsCurrency option specifies whether the measure is a currency value.

The syntax is as follows:

\[\text{IsCurrency \{True\|False\}}\]
**Example**
IsCurrency True

**IsFolder**
The IsFolder option specifies whether the measure is a folder that groups several measures.

The syntax is as follows:
IsFolder {True|False}

**Example**
IsFolder True

**Label**
The Label option specifies the label used for the measure.

The syntax is as follows:
Label string

**Example**
Label "Qty"

**MeasureInfo**
The MeasureInfo option provides a description of the measure.

The syntax is as follows:
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**
The Missing option specifies how records with missing or null values are to be represented in the reporting components.

The syntax is as follows:
Missing [missingvalue]

where missingvalue is one of Default, Zero, or NA (reports show na).

**Example**
Missing Zero

**Rollup**
The Rollup option specifies how measure values are summarized or rolled up, across each reporting time period.
The External rollup option indicates that the measure was rolled-up before it was imported into Cognos Transformer, and so retains the summarization method established in the data source.

For a Category Count measure, you can also use the ActivityMeasure or Dimension options. For more information about these alternatives, see "ActivityMeasure" on page 207 and "Dimension" on page 210.

The syntax is as follows:

```
Rollup [rollup]
```

where `rollup` is one of CategoryCount, Default, Sum, Minimum, Maximum, Average, Count, CountAll, Any, None, or External.

**Example**

Rollup Average

**Parent**

The Parent option uniquely identifies the parent object of the measure, where the model contains a hierarchy of measures.

The syntax is as follows:

```
Parent objMeasure
```

where `objMeasure` is the unique identifier of the parent.

**Example**

Parent 4719

**Scale**

The Scale option specifies the output scale for the measure.

The syntax is as follows:

```
Scale uns
```

where `uns` is an unsigned number and zero (0) signifies no change in scale.

**Example**

Scale 0

**ShortName**

The ShortName option specifies the short name used to represent the measure.

The syntax is as follows:

```
ShortName string
```

**Example**

ShortName "QtySold"

**Sign**

The Sign option specifies whether the sign of a measure value is to be reversed, changing its value from a negative to a positive number, or from a positive to a
negative number. Use this option to convert values into the form required by specific financial reports, such as debits and credits, or assets and liabilities.

The syntax is as follows:

\[ \text{Sign} \ {\text{True}|\text{False}} \]

**Example**

Sign False

**Storage**

The Storage option specifies the data type used to store the source values for this measure.

The syntax is as follows:

\[ \text{StorageType} \ {\text{storagetype}} \]

where *storagetype* is one of Default, Int8, Int16, Int32, Float64, Text, Character, or LongText.

**Example**

Storage Float64

**TimeStateRollup**

The TimeStateRollup option specifies the function used to summarize or roll up measure values for a specific time period.

**Note:** Time-state rollups only occur at run time. This is why External can be specified as a rollup parameter, but not as a timerollup parameter.

The syntax is as follows:

\[ \text{TimeStateRollup} \ {\text{timerollup}} \]

where *timerollup* is one of Default, Minimum, Maximum, First, Last, Current, Average, or Summary.

**Example**

This example uses verb MDL to create an Average time state rollup for the Profit measure, and ignore any missing values:

```mdl
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"
```

**TimeStateWeight**

The TimeStateWeight option specifies the object used to obtain the weighting for a time state rollup. Note that the TimeStateRollup option must be explicitly set to Average.

The syntax is as follows:

\[ \text{TimeStateWeight} \ \text{objMeasure} \]

where *objMeasure* is the object name.
Example

This example uses structured MDL to create an average Cost measure, weighted by, or in proportion to, the measures associated with the Outlet object. Missing values are ignored.

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"

Timing

The Timing option specifies the timing that applies to the measure; that is, before or after roll-up.

The syntax is as follows:

```
Timing [measuretiming]
```

where measuretiming is one of Default, Before_Rollup, or After_Rollup.

Example

```
Timing After_Rollup
```

Weight

The Weight option specifies the measure to use for weighting the time-state rollup. This option requires an object name, whereas the WeightID option uses the object identifier. For both, the TimeStateRollup option must be set to Average.

The syntax is as follows:

```
Weight objMeasure
```

where objMeasure is the object name.

Example

```
Weight "Revenue"
```

WeightID

The WeightID option specifies the measure to use for weighting the time-state rollup. This option requires an object identifier, whereas the Weight option uses the object name. For both, the TimeStateRollup option must be set to Average.

The syntax is as follows:

```
WeightID objMeasure
```

where objMeasure is the object identifier.

Example

```
Weight 259
```

`powercubeopts`

Use powercubeopts to set the options that apply to the following MDL verbs: CubeAdd, CubeMake, CubeUpdate, CubeGroupAdd, CubeGroupMake, CubeGroupUpdate, CubeGroupCubeAdd, CubeGroupCubeMake, and CubeGroupCubeUpdate.
**BlockParentTotals**

The BlockParentTotals option specifies whether the total values for parent categories are included or hidden for the specified cube or cube group.

Use this option to display the word “denied” in the reporting component, rather than a total value that would represent an inaccurate rollup of only non-excluded categories. The Windows equivalent is to select the Block Total Values for Parent Categories with Excluded Children option when categories have been excluded from a custom view. For this to work properly, you must ensure that the appropriate security objects have been enabled.

The syntax is as follows:

```
BlockParentTotals {True|False}
```

where False is the default.

**Example**

```
BlockParentTotals True
```

---

**Caching**

The Caching option specifies whether crosstab caching is enabled. The option sets a flag in the .mdc cube file.

The syntax is as follows:

```
Caching {True|False}
```

**Example**

```
Caching True
```

---

**Compress**

The Compress option specifies whether the cube is compressed. This option may appear in legacy cubes but is not supported as an optimization technique for IBM Cognos.

The syntax is as follows:

```
Compress {True|False}
```

**Example**

```
Compress False
```

---

**Consolidate**

The Consolidate option specifies the record consolidation method used for the cube or cube group.

The syntax is as follows:

```
Consolidate consolidate
```

where consolidate is one of Default, Yes, No, or PreSorted.

**Example**

```
Consolidate Yes
```
**ConsolidatedRecords**

The ConsolidatedRecords option specifies the number of records consolidated during each partitioning pass.

The syntax is as follows:

```
ConsolidatedRecords bignum
```

where the default is 10,000,000 records.

**Example**

```
ConsolidatedRecords 10,000,000
```

**CubeCreation**

The CubeCreation option enables or disables the creation of specific cubes in the model.

The syntax is as follows:

```
CubeCreation makecube
```

where `makecube` is one of Default, Implicit, On, or Off.

**Example**

```
CubeCreation On
```

**CubeCycle**

The CubeCycle option is related to internal Cognos Transformer processing and should not be set.

The syntax is as follows:

```
CubeCycle bignum
```

**Example**

```
CubeCycle 604
```

**CubeStamp**

The CubeStamp option is related to internal Cognos Transformer processing and should not be set.

The syntax is as follows:

```
CubeStamp bignum
```

**Example**

```
CubeStamp 861985194
```

**CubeUpdateLock**

The CubeUpdateLock option specifies whether no cubes or all cubes are locked for update.

The syntax is as follows:

```
CubeUpdateLock {None|All}
```
Example
CubeUpdateLock None

Database
The Database option specifies the database providing the source data for a cube or cube group. For example, you can use this option to identify the source for an IQD signon.

The syntax is as follows:
```
Database objSignon
```

where `objSignon` can be the object name, object identifier, or both.

Example
Database 6317

DatabaseInfo
The DatabaseInfo option provides information about the type of database used with the cube or cube group.

The syntax is as follows:
```
DatabaseInfo string
```

where the database type and settings are separated by semi-colons.

Example
DatabaseInfo "Local;;"

DataSource
The DataSource option specifies a main Impromptu Query Definition (IQD) data source and one or more alternate data sources, with their associated options.

For more information on IBM Cognos data sources and options, see "appqueryopts" on page 161.

The syntax is as follows:
```
DataSource objDataSource AlternateSource filename [altdatasourceopts]
```

where `objDataSource` is the object name and `altdatasourceopts` is one or more of the following:
- Isolationnumber
- DmsSignonobjSignon
- IMRName string
- SourceSignonList signon signons... ] EndList
- SQL string
- Stampnumber

Example
The following example shows the SQL for an Impromptu Query Definition (IQD) data source:
**DeployLocations**

The DeployLocations option specifies a list of deployment locations for the PowerCube.

The syntax is as follows:

```
DeployLocations
value
Endlist
```

**Example**

`DeployLocations "C:\Cubes" Endlist`

**DeployType**

The DeployType option specifies the type of PowerCube deployment strategy to use for the PowerCube.

The syntax is as follows:

```
DeployType
deploytype
```

where `deploytype` is one of DeployNone, DeploySwapTogether, or DeploySwapSingle (not available in this release).

**Example**

`DeployType DeploySwapTogether`

**DetailLevel**

The DetailLevel option identifies the level that contains the categories on which individual cubes in a cube group are based. Use to create a cube for each category in the level. Duplicate level names in the list are qualified by the name of the owning category, as in Brand (OF Tent).

The syntax is as follows:

```
DetailLevel
objLevel
```

where `objLevel` can be the object name.

**Example**

`DetailLevel "Product Type"`

**DrillThrough**

The DrillThrough option specifies whether drill-through action is enabled for the cube or cube group, and lists the target report objects.

The syntax is as follows:

```
DrillThrough [True|False] reportnames EndList
```
where reportnames are the reports, as listed in the Custom Reports and Report Description boxes on the Windows interface.

Example

In this example, drill through is enabled for the specified .mdc cube file:
DrillThrough True "c:\installation_directory\all_regions.mdc" EndList

Exclude

The Exclude option is used, followed by a list of excluded objects, to restrict the scope of drill-through activity to targets related to a specific non-excluded drill category or dimension level. By so doing, the modeler can ensure that users navigate directly to the relevant portion of the cube before they begin their drill-through operation.

For each or cube or measure for which to want to restrict drill-through activity by level, you must specify the relevant dimension and levels, and any drill-through targets 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 of the Windows interface.

The syntax is as follows:
Exclude [Dimension objDim] [Drill objDrill] Levels objLevel

where objDim, objLevels, and objDrill can be the unique object identifiers, names, or both.

Example

In this example, the State and Market levels in the Region and Sales dimensions are excluded for the specified cube drill-through operations:
DrillThrough True ... "C:\path_to_cube.mdc" "Description text" Exclude Dimension "Region" Exclude Levels 219 "State" Exclude Dimension "Sales" Exclude Levels "Market" EndExclude "C:\path_to_cube2.mdc" "Description text2" EndList

IncrementalUpdate

The IncrementalUpdate option specifies whether incremental updates are enabled for this cube or cube group.

An object identifier is automatically assigned when the cube is incrementally updated.

The syntax is as follows:
IncrementalUpdate {True|False}

Example

IncrementalUpdate True

Information

The Information option provides a description of the cube or cube group.

The syntax is as follows:
Information string [string...]

Chapter 7. MDL options 221
where each string may be up to 256 characters and the total description may be up
to 4,095 characters.

**Example**

Information "GO Sales Sample PowerCube."

**MDCFile**

The MDCFile option specifies the path and file name used for the .mdc cube file.

The syntax is as follows:

```
MDCFile string
```

**Example**

MDCFile "c:\Temp\outdoors.mdc"

**MeasureName**

The MeasureName option specifies the name used for a cube measure or group of
measures.

The syntax is as follows:

```
MeasureName string
```

**Example**

MeasureName "Basic Measures"

**Optimize**

The Optimize option specifies the optimization strategy applied to the cube or cube
group. For more information about optimizing cube build times, see the Cognos
Transformer User Guide.

The syntax is as follows:

```
Optimize cubeoptimize
```

where `cubeoptimize` is one of Default, Categories, Datapasses, DirectCreate,
PriorMarking, or AutoPartition.

**Example**

Optimize Categories

**PartitionSize**

The PartitionSize option specifies the number of records included in each
partition.

The syntax is as follows:

```
PartitionSize bignum
```

**Example**

PartitionSize 500,000
**PassesNumber**

The `PassesNumber` option specifies the number of partitioning passes to run.

The syntax is as follows:

```
PassesNumber bignum
```

where `bignum` is any number; the default is 5.

**Example**

```
PassesNumber 4
```

**Password**

The `Password` option specifies an unencrypted cube password.

The syntax is as follows:

```
Password string
```

**Example**

```
Password "mypasswd"
```

**PublishAllowAccessToSuppressionOptions**

The `PublishAllowAccessToSuppressionOptions` option specifies whether the package published from Cognos Transformer allows the user to access null suppression options in the Web studios.

The syntax is as follows:

```
PublishAllowAccessToSuppressionOptions {True|False}
```

**Example**

```
PublishAllowAccessToSuppressionOptions False
```

**PublishAllowMultiEdgeSuppression**

The `PublishAllowMultiEdgeSuppression` option specifies whether the package published from Cognos Transformer allows the user to enable multi-edge suppression in the Web studios.

The syntax is as follows:

```
PublishAllowMultiEdgeSuppression {True|False}
```

**Example**

```
PublishAllowMultiEdgeSuppression False
```

**PublishAllowNullSuppression**

The `PublishAllowNullSuppression` option specifies whether the package published from Cognos Transformer allows the user to control null suppression in the Web studios.

The syntax is as follows:

```
PublishAllowNullSuppression {True|False}
```

**Example**

```
PublishAllowNullSuppression False
```
SegmenterDimension
The SegmenterDimension option specifies the name of the dimension on which a cube group is based or partitioned.

The syntax is as follows:
SegmenterDimension objDim

where objDim is the object name.

Example
SegmenterDimension "Products"

SegmenterLevel
The SegmenterLevel option specifies the name of the level on which a cube group is based or partitioned.

The syntax is as follows:
SegmenterLevel objLevel

where objLevel is the object name.

Example
SegmenterLevel "Product Line"

ServerCube
The ServerCube option specifies whether the cube is located on the server, which is relevant in versions of Cognos Transformer that support client/server processing.

The syntax is as follows:
ServerCube {True|False}

where True is equivalent to selecting On the Server in Cognos Transformer version 7.x. Client-server processing is not supported in Cognos Transformer version 8.x.

Example
ServerCube False

Status
The Status option specifies the processing status of the cube or cube group.

The syntax is as follows:
Status [cubestatus]

where cubestatus is one of New, OK, Inactive, Warnings, Invalid, Failed, Busy, or Missing.

Example
Status Inactive
**SummaryLevel**

The `SummaryLevel` option specifies the level at which category values are summarized.

The Windows interface equivalent is the **Summarize all external categories in the level** option on the **Cube Group** tab of the **Cube** property sheet.

The syntax is as follows:

```
SummaryLevel string
```

**Example**

```
SummaryLevel "Product Type"
```

**UseAlternateFilename**

The `UseAlternateFilename` option specifies whether to use an alternate file name for the cube.

The syntax is as follows:

```
UseAlternateFilename {True|False}
```

**Example**

```
UseAlternateFilename True
```

**TimeBasedPartitionedCube**

The `TimeBasedPartitionedCube` option specifies whether time-based partitioning is enabled for the cube.

The syntax is as follows:

```
TimeBasedPartitionedCube {True|False}
```

**Example**

```
TimeBasedPartitionedCube True
```

**promptopts**

Use `promptopts` to set the options that apply to data source prompts and the following MDL verbs: `PromptAdd`, `PromptMake`, and `PromptUpdate`.

**PromptType**

For data source prompts, the `PromptType` option specifies the type of prompt to be imported into the Cognos Transformer model.

The syntax is as follows:

```
PromptType {xsdString|memberUniqueName}
```

Use “xsdString” as the prompt type for simple-value prompts, multi-value prompts, and range prompts.

**Examples**

```
PromptType "xsdString" PromptType "memberUniqueName"
```
**PromptValue**

For data source prompts, the `PromptValue` option specifies the value of the prompts imported into the Cognos Transformer model.

The syntax is as follows:

```
PromptValue value
```

**Examples**

The following example specifies a prompt value of 2 for a simple-value prompt.

```
PromptValue 2
```

The following example specifies prompt values of 0, 1, and 2 for a multi-value prompt.

```
PromptValue StartList "0""1""2" EndList
```

The following example specifies prompt values of 1 through 22 for a range prompt.

```
PromptValue StartRange "1""22" EndRange
```

The following example specifies the prompt value for a Member Unique Name prompt.

```
PromptValue StartList "[ZQ1AUTCTY WORLD].[AMERICA ZQ1AUTCNT]\"EndList
```

---

**signonopts**

Use `signonopts` to set the options that apply to data sources and the following MDL verbs: `SignonAdd`, `SignonMake`, and `SignonUpdate`.

**PromptForPassword**

For data source signons, the `PromptForPassword` option specifies whether the user is prompted to enter their signon password.

The syntax is as follows:

```
PromptForPassword {True|False}
```

**Example**

```
PromptForPassword True
```

**UserId**

The `UserId` option specifies the user ID portion of the signon.

The syntax is as follows:

```
UserId "user name"
```

**Example**

```
UserId "corpadm"
```

**password**

The `password` option specifies the signon password in unencrypted form.

The syntax is as follows:

```
password "password"
```
Example
password "my_pass"

**AutoLogon**
For IBM Cognos signons, the AutoLogon option specifies whether to prompt for credentials as part of the logon process.

The syntax is as follows:
```plaintext
AutoLogon {True|False}
```

**Example**
AutoLogon True

**SignonNamespace**
For IBM Cognos signons, the SignonNamespace option specifies the ID of namespace for the signon.

The syntax is as follows:
```plaintext
SignonNamespace "namespace id"
```

**Example**
SignonNamespace "Cognos"
SignonNamespace "LDAP"

**SignonType**
The SignonType option specifies whether the signon is an IBM Cognos or data source signon.

The syntax is as follows:
```plaintext
SignonType {Cognos|DataSource}
```

**Example**
SignonType Cognos

**SignonInfo**
The SignonInfo option provides a description of the signon.

The syntax is as follows:
```plaintext
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 the IBM Cognos data source."

viewopts

Use viewopts to set the options that apply to the following MDL verbs: ViewAdd, ViewMake, and ViewUpdate. These act on dimension views and custom security views. For more information about the latter, see the Security section of the Cognos Transformer User Guide.

Chapter 7. MDL options  227
**Apex**
The Apex option specifies which category is to be apexed and become the top level. For more information, see the Cognos Transformer User Guide.

The syntax is as follows:
```
Apex objCat
```
where \( objCat \) specifies the category and is the object name, object identifier, or both.

**Example**
```
Apex "Europe"
```

**Cloak**
The Cloak option specifies which category is to be cloaked or hidden. For more information, see the Cognos Transformer User Guide.

The syntax is as follows:
```
Cloak objCat
```
where \( objCat \) specifies the category and is the object name, object identifier, or both.

**Example**
```
Cloak "Far East"
```

**Filter**
The Filter option specifies which category is to be excluded, or filtered, from the cube. For more information, see the Cognos Transformer User Guide.

The syntax is as follows:
```
Filter objCat
```
where \( objCat \) specifies the category and is the object name, object identifier, or both.

**Example**
```
Filter "Europe"
```

**LevelCloak**
The LevelCloak option specifies which level and drill category are to be cloaked or hidden.

The syntax is as follows:
```
LevelCloak [Dimension objDimension][Drill objDrill] Levels objLevel
```
where \( objDimension \) specifies the dimension and is the object name, object identifier or both; \( objDrill \) specifies the drill and is the object name, object identifier, or both; \( objLevel \) specifies the level and is the object name, object identifier or both. Drill is needed if the level name is not unique in the entire model. Dimension is needed if the Drill is needed and the Drill name is not unique in the entire model.

For more information, see the IBM Cognos Transformer User Guide.
```
LevelCloak Dimension "Location" Drill "By Location" Levels "Division"
LevelCloak Dimension "Office" Drill "By Office" Levels "Division"
```
LevelFilter

The LevelFilter option specifies which level and drill category are to be excluded, or filtered, from the cube.

The syntax is as follows:

```
LevelFilter [Dimension objDimension][Drill objDrill] Levels objLevel
```

where `objDimension` specifies the dimension and is the object name, object identifier or both; `objDrill` specifies the drill and is the object name, object identifier, or both; `objLevel` specifies the level and is the object name, object identifier or both. Drill is needed if the level name is not unique in the entire model. Dimension is needed if the Drill is needed and the Drill name is not unique in the entire model.

For more information, see the IBM Cognos Transformer User Guide.

Example

```
LevelFilter Dimension "Location" Drill "By Location" Levels "Division"
LevelFilter Dimension "Office" Drill "By Office" Levels "Division"
```

LevelSummary

The LevelSummary option specifies which level and drill category are to be summarized.

The syntax is as follows:

```
LevelFilter [Dimension objDimension][Drill objDrill] Levels objLevel
```

where `objDimension` specifies the dimension and is the object name, object identifier or both; `objDrill` specifies the drill and is the object name, object identifier, or both; `objLevel` specifies the level and is the object name, object identifier or both. Drill is needed if the level name is not unique in the entire model. Dimension is needed if the Drill is needed and the Drill name is not unique in the entire model.

For more information, see the IBM Cognos Transformer User Guide.

Example

```
LevelSummary Dimension "Location" Drill "By Location" Levels "Division"
LevelSummary Dimension "Office" Drill "By Office" Levels "Division"
```

LevelSuppressed

The LevelSuppressed option specifies which level is to be suppressed.

The syntax is as follows:

```
LevelFilter [Dimension objDimension][Drill objDrill] Levels objLevel
```

where `objDimension` specifies the dimension and is the object name, object identifier or both; `objDrill` specifies the drill and is the object name, object identifier, or both; `objLevel` specifies the level and is the object name, object identifier or both. Drill is needed if the level name is not unique in the entire model. Dimension is needed if the Drill is needed and the Drill name is not unique in the entire model.

For more information, see the IBM Cognos Transformer User Guide.
Example
LevelSuppressed Dimension "Location" Drill "By Location" Levels "Division"
LevelSuppressed Dimension "Office" Drill "By Office" Levels "Division"

Name
The Name option specifies a name for the view.

The syntax is as follows:

\textbf{Name} \textit{objView}

where \textit{objView} is the object name.

Example
Name "Europe"

NotCloak
The NotCloak option specifies which category is not to be cloaked or hidden. For more information, see the Cognos Transformer User Guide.

The syntax is as follows:

\textbf{NotCloak} \textit{objCat}

where \textit{objCat} specifies the category and is the object name, object identifier, or both.

Example
NotCloak "Far East"

NotFilter
The NotFilter option specifies which category is not to be excluded, or filtered, from the cube. For more information, see the Cognos Transformer User Guide.

The syntax is as follows:

\textbf{NotFilter} \textit{objCat}

where \textit{objCat} specifies the category and is the object name, object identifier, or both.

Example
NotFilter "Customer Type"

NotSummary
The NotSummary option specifies which category is not to be summarized. For more information, see the Cognos Transformer User Guide.

The syntax is as follows:

\textbf{NotSummary} \textit{objCat}

where \textit{objCat} specifies the category and is the object name, object identifier, or both.

Example
NotSummary "Region"
**NotSuppressed**

The `NotSuppressed` option specifies which category is not to be suppressed. For more information, see the Cognos Transformer User Guide.

The syntax is as follows:

```
NotSuppressed
objCat
```

where `objCat` specifies the category and is the object name, object identifier, or both.

**Example**

```
NotSuppressed "Region"
```

**Summary**

The `Summary` option specifies which category is to be summarized. For more information, see the Cognos Transformer User Guide.

The syntax is as follows:

```
Summary objCat
```

where `objCat` specifies the category and is the object name, object identifier, or both.

**Example**

```
Summary "Europe"
```

**Suppressed**

The `Suppressed` option specifies which category is to be suppressed. For more information, see the Cognos Transformer User Guide.

The syntax is as follows:

```
Suppressed objCat
```

where `objCat` specifies the category and is the object name, object identifier, or both.

**Example**

```
Suppressed "Far East"
```
Chapter 8. Structured MDL

There are two forms of MDL: verb MDL, which is the main focus of the Developer Guide, and structured MDL, which is briefly described in this section, under the following topics:

- "History of MDL"
- "Comparison of structured and verb keywords"
- "Using structured MDL" on page 235

Like verb MDL, the structured form of MDL can provide a complete and functionally independent definition of any Cognos Transformer model.

History of MDL

Model Definition Language (MDL) was created to store Cognos Transformer models so they could be opened in future versions of Cognos Transformer or related IBM Cognos modeling components. In its original form, MDL contained statements that defined the model in a fixed order: that is, structured MDL.

Verbs were added to enhance the functionality of the language and to support software developers by providing an independent, parallel, workable format: that is, verb MDL.

To maintain compatibility between product versions, structured MDL continues to be the default form. However, if you are building models using MDL, add the cogtr.ini setting VerbOutput=1, so that Cognos Transformer automatically generates verb output.

For more information about global preference settings, see the Cognos Transformer User Guide.

Comparison of structured and verb keywords

The following table lists Cognos 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.

Note: Most but not all of the currently-supported verbs are included in this list.

<table>
<thead>
<tr>
<th>Cognos Transformer object</th>
<th>Structured keyword</th>
<th>Verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Name</td>
<td>NewModel</td>
</tr>
<tr>
<td>Data Source</td>
<td>DataSource, CognosSource</td>
<td>DataSourceAdd, DataSourceDelete, SourceListUpdate, DataSourceMake, DataSourceUpdate</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 2007, 2013
<table>
<thead>
<tr>
<th>Cognos Transformer object</th>
<th>Structured keyword</th>
<th>Verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>Filter, FilterRef</td>
<td>CognosPackageFilterAdd, CognosPackageFilterMake, CognosPackageFilterDelete, CognosPackageFilterUpdate</td>
</tr>
<tr>
<td>Prompt</td>
<td>Prompt</td>
<td>PromptAdd, PromptDelete, PromptMake, PromptUpdate</td>
</tr>
<tr>
<td>Column</td>
<td>OrgName</td>
<td>ColumnAdd, ColumnDelete, ColumnListUpdate, ColumnMake, ColumnUpdate</td>
</tr>
<tr>
<td>Dimension</td>
<td>Dimension</td>
<td>DimAdd, DimDelete, DimMake, DimUpdate, DimensionListUpdate</td>
</tr>
<tr>
<td>Subdimension</td>
<td>SubDimCat</td>
<td>SubDimRootMake, SubDimRootUpdate</td>
</tr>
<tr>
<td>Root Category</td>
<td>Root</td>
<td>RootCatMake, RootCatUpdate</td>
</tr>
<tr>
<td>Drill Category</td>
<td>Drill</td>
<td>DrillCatMake</td>
</tr>
<tr>
<td>Level</td>
<td>Levels</td>
<td>LevelAdd, LevelDelete, LevelMake, LevelMoveAfter, LevelMoveBefore, LevelNewDrill, LevelUpdate, FilterLevel</td>
</tr>
<tr>
<td>Category</td>
<td>Category</td>
<td>CatAdd, CatDelete, CatJoin, CatMake, CatMorph, CatMoveLevel, CatMoveVertical, CatUpdate, FilterCat</td>
</tr>
<tr>
<td>Special Category</td>
<td>SpecialCategory</td>
<td>SpecialCatAdd, SpecialCatDelete, SpecialCatMake, SpecialCatUpdate</td>
</tr>
<tr>
<td>Dimension View</td>
<td>ViewName, DimensionView</td>
<td>ApexCat, ViewAdd, ViewDelete, ViewListUpdate, ViewMake, ViewUpdate, CustomViewChildListUpdate</td>
</tr>
<tr>
<td>Measure</td>
<td>Measure</td>
<td>MeasureAdd, MeasureDelete, MeasureMake, MeasureListUpdate, MeasureUpdate</td>
</tr>
</tbody>
</table>
## 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 or in conjunction with structured statements, as long as the verb statements are at the end of the model. After a verb statement, no structured statements can be used.

### Example

The following is an example of a model in structured MDL format:

```plaintext
Name "New Model" ModelCodePage "windows-1252" AutoAccess False
SynchroCycle 0 SynchroStamp 0 UpdateCycle 1 ModelStamp 1189458791
ClientStamp 1189458791 ServerStamp 0 Version "8.3.790.0"
ModelCategoryOrderDefault OrderUsePreference ModelOrderedByDefault False
CognosSource 103 "GO Sales and Retailers" SourceType Package
```

<table>
<thead>
<tr>
<th>Cognos Transformer object</th>
<th>Structured keyword</th>
<th>Verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PowerCube Group</td>
<td>CubeGroup</td>
<td>CubeGroupAdd, CubeGroupDelete, CubeGroupMake, CubeGroupUpdate, PowerCubeDelete, PowerCubeListUpdate</td>
</tr>
<tr>
<td>PowerCube Group Cube</td>
<td>CubeGroupCube</td>
<td>CubeGroupCubeAdd, CubeGroupCubeDelete, CubeGroupCubeListUpdate, CubeGroupCubeMake, CubeGroupCubeUpdate</td>
</tr>
<tr>
<td>Signon</td>
<td>Signon</td>
<td>SignonDelete, SignonListUpdate, SignonMake, SignonUpdate, CognosPackageDatasourceConnectionAdd, CognosPackageDatasourceConnectionMake, CognosPackageDatasourceConnectionUpdate, CognosPackageDatasourceConnectionDelete</td>
</tr>
<tr>
<td>Dimension Calculation Definition</td>
<td>DimCalcDef</td>
<td>DimCalcDefAdd, DimCalcDefDelete, DimCalcDefMake, DimCalcDefUpdate</td>
</tr>
<tr>
<td>Currency Table</td>
<td>CurrencyTable</td>
<td>CurrencyTableAdd, CurrencyTableDelete, CurrencyTableMake, CurrencyTableUpdate</td>
</tr>
<tr>
<td>Currency</td>
<td>CurrencyRecord</td>
<td>CurrencyAdd, CurrencyDelete, CurrencyMake, CurrencyUpdate, CurrencyTableAdd</td>
</tr>
<tr>
<td>Security View</td>
<td>SecurityNameSpace and SecurityObject</td>
<td>NameSpaceMake, SecurityObjectMake</td>
</tr>
</tbody>
</table>
Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing
3755 Riverside Dr
Ottawa, ON K1V 1B7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.
Trademarks

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
Index

Special characters
.dat files
  OpenDef 135
.def files
  OpenDef 135
.gen files
  OpenDef 135

1

A
ActivityMeasure 207
adding associations
  AssociationAdd 68
allocating
  measure allocation options 207
allocating measures
  AllocationAdd 65
Allocation 207
allocations
  creating 19
Apex 228
apexing categories
  ApexCat 67
AppInfo 161
appqueryopts 161
archiving models 6
Association 196, 200, 208
AssociationContext 170
AssociationReferenced 170
AssociationRole 170
associations
  adding 68
  AssociationContext 170
  AssociationUpdate 69
creating or updating using AssociationMake 69
deleting 68
MDL options 170
AssociationType 170
assocopts 170
auto-partitioning
  ConsolidatedRecords 218
  PassesNumber 223
AutoDesign 70
AutoLogon 227

B
BaseCurrency 191
binary files
  models in .py? format 3
Blanks 171, 200
BlockParentTotals 217
buffers
  StreamExtractSize 168

C
Caching 217
Calc 171, 208
Calc column option 182
CalcDef 172, 194
calculated categories
  Calc 171
  CalcDef 172
dimension calculation definitions 118
Share 178
calculated columns 182
calculated measures 208
calculation definitions
  CalcDef 194
  DimCalcDefAdd 117
  DimCalcDefMake 118
  DimCalcDefUpdate 120
  DimCalcDefDelete 118
categories
  apexed 228
  apening 67
  Calc 171
  CalcDef 172
  CatAdd 70
  CatDelete 71
  CatJoin 71
  CatMake 72
  CatMorph 73
  CatMoveVertical 74
  CatUpdate 75
  CatUpdateAll 75
cloaked 228
current time period 173
date 174
DateDrill 174
default cube-open level for scenario dimensions 197
descriptions 173
excluding (filtering) 228
ExtraWeek 174
FilterCat 124
filters 174
generate all in period 202
hiding meaningless values in scenario dimensions 176
Inclusion 175
IsTruncated 175
Label 176
LastUse 176
lowest detail level 220
missing values 171
naming 11
NewPartition 176
options 171
order values 177
orphanages 175
PartialWeek 177
PopulateFromQueries 137
PopulateModel 137
Primary 177
PrimaryDrill 177
prohibit automatic creation of new categories 199, 203
relative time target level 180

© Copyright IBM Corp. 2007, 2013

245
categories (continued)
  reversing the sign of measure values 179
Rollup 178
RootCatMake 141
RootCatUpdate 141
RunningPeriods 178
setting global sorting preferences 14
Share 178
ShortName 178
SourceValue 179
SpecialCatAdd 150
SpecialCatDelete 151
SpecialCatMake 152
SpecialCatUpdate 153
SplitWeek 179
SummarizeCat 155
Suppressed 179
TargetOffset 179
TimeAggregate 180
ToDateLevel 180
unique moves 207
uniqueness 206
WeekBegins 181
YearBegins 181
Category 233
category count measures 210
category counts
  ActivityMeasure 207
Rollup 214
category generation
Timing 169
category names 11
CategoryCode 201
CatInfo 173
CatLabFormat 201
catopts 171
CharacterSet 161
Class 186
CleanHouse 75
  automating 58
Cloak 228
CognosPackageAdd 76
CognosPackageDataSourceConnectAdd 79
CognosPackageDataSourceConnectDelete 80
CognosPackageDataSourceConnectMake 80
CognosPackageDataSourceConnectUpdate 82
CognosPackageDelete 77
CognosPackageFilterAdd 83, 199
CognosPackageFilterDelete 83
CognosPackageFilterMake 83
CognosPackageFilterRef 199
CognosPackageFilterUpdate 84, 199
CognosPackageMake 77
 cognospackageopts 181
CognosPackageUpdate 78
CognosSource 162
colopts 182
ColSrcType 186
Column 186, 194
ColumnInfo 187
columns
  array type 190
calculated 182
ColumnAdd 85
ColumnDelete 85
ColumnListUpdate 86
ColumnMake 86
columns (continued)
ColumnUpdate 87
CreateColumns 88
data class 186
date 187
date input formats 188
DateLevel 187
decimal places 188
degree of detail 188
descriptions 187
input scales 188
labels 203
naming 11
Offset 189
options 182
Origin 189
original name 186
Scale 189
Size 189
storage type 190
TimeArrayMonth 190
Columns 162
compatibility
  migrating between versions using .mdl-format models 3
Compress 217
Consolidate 217
ConsolidatedRecords 218
ContextLevel 173
ContextOffset 173
conventions
  used for data types 61
conversion rates
  updating 59
corruption 6
CountryCode 191
Cube 195, 233
cube groups
  CubeGroupAdd 91
  CubeGroupCubeAdd 92
  CubeGroupCubeDelete 93
  CubeGroupCubeListUpdate 94
  CubeGroupCubeMake 94
  CubeGroupCubeUpdate 95
  CubeGroupDelete 96
  CubeGroupMake 97
  CubeGroupUpdate 98
  naming 11
  SegmenterDimension 224
  SegmenterLevel 224
cube updates
  EventEnd 124
  EventStart 124
  CubeCreation 218
  CubeCycle 218
  CubeGroup 233
  CubeGroupCube 233
cubes
  auto-partitioning record consolidation 218
  BlockParentTotals 217
  Caching 217
  Compress 217
  Consolidate 217
  CubeCreation 218
  CubeCycle 218
  CubeDelete 91
  CubeMake 99
  CubeStamp 218

cubes (continued)
  CubeUpdate 100
  CubeUpdateLock 218
  database signons 219
descriptions 221
drill through 220
drill through 220
file name 222
generating in .mdc-format using CreateFiles 88
generating in .mdc-format using CreateFromCubes 89
generating in .mdc-format using CreateFromQueries 89
IncrementalUpdate 221
lowest detail level 220
measure names 222
optimization 222
output types 219
partitioning passes 223
partitioning status report 140
PartitionSize 222
passwords 223
PowerCubeDelete 138
PowerCubeListUpdate 138
data source types changing 57
ColSrcType 186
data sources changing 57
DataSourceAdd 112
DataSourceDelete 113
DataSourceMake 114
DataSourceUpdate 115
naming 11
query types and options 161
SourceListUpdate 150
Database 219
data bases
  Source 166
data range
  DataRange 162
data source types
  changing 57

D
data classes 186
data files
  DataFiles 187
deleting associations
  AssociationDelete 68
database sources
  DataSource 195
data sources changing 57
DateTime 173
date based partitioning 225
date formats 174
date formats
  Date 174
date formats 201
date input formats 188, 202
date formats 174
decimal places
  DecimalSep 162
date formats 174
decimal places
  Unit 174
decimal separators
  DecimalSep 162
Decimals 188, 209
DecimalSep 162
degree of detail 187, 188
deleting associations
  AssociationDelete 68
deletion options
  CalcDef 194
calculated 194
  Column 194
cube 194
data source 195
dimension 195
levels 195
measure 195
Deletion options (continued)

Signon 196
View 196
DeletionListUpdate 115
deletionopts 194
deletions
options 194
updating list 115
DeployLocations 220
deployment
DeployLocations 220
DeployType 220
Description 181, 202
descriptions
adding 57
category 173
column 187
dimension 197
measure 213
PowerCubes and cube groups 221
query 161
signon 227
Detail 188
DetailLevel 220
DimCalcDef 233
DimDefaultCategory
opening level for scenario dimensions 197
Dimension 195, 210, 233
dimension views
 Apex 228
 Cloak 228
 Exclude or Filter 228
 names 230
 options 228
 use as the basis for new PowerCubes 59
 ViewAdd 157
 ViewDelete 158
 ViewListUpdate 158
 ViewMake 159
 ViewUpdate 160
dimensions
 Association 196
cube-open category 197
DaysInWeek 196
descriptions 197
DimAdd 116
DimDelete 121
DimensionListUpdate 121
DimMake 122
DimType 197
DimUpdate 122
EarliestDate 198
ExcludeAutoPartitioning 198
LatestDate 198
ManualPeriods 198
naming 11
options 196
prohibit automatic creation of new categories 199
SubDimRootMake 153
SubDimRootUpdate 154
ViewAdd 157
ViewDelete 158
ViewListUpdate 158
ViewMake 159
ViewUpdate 160
DimensionView 233
DimInfo 197
dimopts 196
DimType 197
documentation scope
 supported functionality xiii
Drill 233
drill categories
 DrillCatMake 123
 JoiningLevel argument 123
drill through
 restrictions using Exclude 211
 using Exclude to restrict scope 221
drill-through to IQD option
ImrName 163
DrillThrough 210, 220
DuplicateRollup 210
DuplicateWeight 211
EarliestDate 198
EmuEntryDate 192
euro
 European Monetary Union setting 192
EuroBaseCurrency 192
EventEnd 124
EventStart 124
Exclude 211, 221
ExcludeAutoPartitioning 198
ExpMark 200
ExtraWeek 174
field delimiters
 Separator 165
file formats
 converting 58
Filter 228
FilterDescription 200
Filtered 174, 202
filteropts 199
filters
 CognosPackageFilterAdd 199
 CognosPackageFilterUpdate 199
ExpMark 200
FilterDescription 200
 options 199
Format 174, 188, 202, 211
fragmentation 6
Generate 202
HideValue 176
IgnoreMissingValue 212
IMRName 163
inclusion
 levels 202
Inclusion 175, 202
incremental updates
  disabling 59
IncrementalUpdate 221
Information 221
InputScale 188
IQD drill-through targets
  ImrName 163
IsCurrency 212
IsFolder 213
IsKeyOrphanage 175
Isolation 163
IsTruncated 175
Label 176, 203, 213
Last Use 176
LatestDate 198
LevelCloak 228
LevelFilter 229
LevelInfo 203
levelopts 200
levels
  associations 200
  blank substitutions 200
  category code columns 201
  cloaking 228
  date formats 201
  date functions 201
  date input formats 202
  descriptions 203
  DetailLevel 220
  filtering 229
  filters 202
  generate all categories in period 202
  inclusion 202
LevelAdd 125
LevelDelete 126
LevelMake 127
LevelMoveAfter 128
LevelMoveBefore 128
LevelNewDrill 129
LevelUpdate 130
naming 11
options 200
order by 204
ordering categories 203
partitions 204
prohibit automatic creation of new categories 203
refresh 205
share objects 205
short names 205
SortAs 204
source column description 181, 202
source columns 206
SummarizeLevel 156
summarizing 229
supress 206
suppressed 229
time-level ranking 206
unique moves 207
  using Exclude to restrict drill-through 211
using Exclude to restrict drill-through scope 221
Levels 195, 233
LevelSummary 229
LevelSuppressed 229
line length
  maximum 62

M
ManualPeriods 198
MDCClear 130
MDCFile 222
MDL statements
  description and purpose 3
meaopts 207
Measure 195, 233
measure values
  hiding in scenario dimensions 176
MeasureInfo 213
MeasureName 222
measures
  allocating 65
  allocation options 207
  calculated 208
  category count 207, 210, 214
  decimal place format 174
  decimal places 209
  descriptions 213
  drill through 210
duplicate rollup 210
duplicate weight rollups 211
format 211
ignoring missing values for specific rollup types 212
IsCurrency 212
isFolder 213
label 213
MeasureAdd 131
MeasureDelete 132
MeasureListUpdate 132
MeasureMake 133
MeasureUpdate 134
naming 11
null or missing values 213
options 207
output scale 214
parent 214
regular (from a column) 208
reversing the sign 215
Rollup 214
short names 214
storage types 215
time-state rollups 215
time-state weighting 215
Timing 216
using Exclude to restrict drill-through 211
Weight 216
WeightID 216
migrating models 6
Missing 213
missing values
  Blanks 171
  ignoring for specified rollup types 212
  NA option 213
model objects
  AssociationContext 170
deletion options 194
ModelEnsureCompleteness 134
ModelNonRollupHierarchies 164
models
  create using NewModel 134
files in .py? and .mdl format 3
models (continued)
   ModelEnsureCompleteness 134
   ModelStamp 163
   open using OpenMDL 136
   open using OpenPY 136
   options 161
   PopulateFromQueries 137
   PopulateModel 137
   SaveMDL 142
   SavePY 142
   setting global processing options 14
   Stamp 168
   ModelStamp 163
   modifying models
   directly 6
   with a script 6
   within the model 6
   multi-edge suppression
   PublishAllowMultiEdgeSuppression 223

N
Name 230, 233
new features
   version 10.2.0 1
   version 10.2.1 1
new features version 6.5
   category count measure 210
NewCatsLock 199, 203
NewPartition 176
NotFilter 230
NotSummary 230
NotSuppressed 231
null suppression
   PublishAllowAccessToSuppressionOptions 223
   PublishAllowNullSuppression 223
null values
   ignoring for specified rollup types 212
   NA option 213

O
object identifiers
   creating 10
   excluding 10
object names
   rules governing 10
   uniqueness 11
object types 9
objects
   AssociationContext 170
   Offset 189
   OpenDef 135
   OpenMDL 136
   OpenPY 136
optimization
   ExcludeAutoPartitioning option 198
   PassesNumber 223
Optimize 222
optimizing
   query speed 167
OrderBy 203, 204
OrgName 233
Origin 189
original names 186
Orphanage 177
orphanages
   IsKeyOrphanage 175

P
PackageReportSource 164
PackageTimeStamp 164, 181
Parent 214
PartialWeek 177
Partition 204
partitioning
   ConsolidatedRecords 218
   NewPartition 176
   PassesNumber 223
partitioning status
   ReportPartitions 140
   PartitionSize 222
   PassesNumber 223
   password 226
   Password 223
passwords
   unencrypted 226
   performance 6
PopulateFromQueries 137
PopulateModel 137
PowerCube groups
   CubeGroupAdd 91
   CubeGroupCubeAdd 92
   CubeGroupCubeDelete 93
   CubeGroupCubeListUpdate 94
   CubeGroupCubeMake 94
   CubeGroupCubeUpdate 95
   CubeGroupDelete 96
   CubeGroupMake 97
   CubeGroupUpdate 98
   PowerCubeCustomViewListUpdate 137
   powercubeopts 217
PowerCubes
   auto-partitioning record consolidation 218
   BlockParentTotals 217
   Caching 217
   changing optimization setting 58
   changing output locations 58
   changing output types 58
   Compress 217
   Consolidate 217
   CubeAdd 90
   CubeCreation 218
   CubeCycle 218
   CubeDelete 91
   CubeMake 99
   CubeStamp 218
   CubeUpdate 100
   CubeUpdateLock 218
   database signons 219
   descriptions 221
   drill through 220
   file name 222
   generating in .mdc-format using CreateFiles 88
   generating in .mdc-format using CreateFromCubes 89
   generating in .mdc-format using CreateFromQueries 89
   IncrementalUpdate 221
   lowest detail level 220
   measure names 222
   naming 11
   optimization 222
   output types 219
PowerCubes (continued)
  partitioning passes 223
  partitioning status report 140
  PartitionSize 222
  passwords 223
  PowerCubeDelete 138
  PowerCubeListUpdate 138
  powercubeopts 217
  processing status 224
  revert to default status using MDClear 130
  SegmenterDimension 224
  SegmenterLevel 224
  ServerCube 224
  source files 219
  summarizing external category values 225
  time-based partitioning 225
  UpdatePowerCubes 157
  updating all in model 57
  updating selected 57
  using Exclude to restrict drill-through scope 221
  using temporary filenames 225
  precision 209
  preferences
    global processing options 14
    running AutoDesign by default 70
  PreSummarized 165
  Primary 177
  PrimaryDrill 177
  processing
    revert to default status using MDClear 130
  PromptAdd 139
  PromptDelete 139
  PromptForPassword 226
  PromptMake 139
  promptopts 225
  PromptUpdate 140
  PromptValue 226
  PublishAllowAccessToSuppressionOptions 223
  PublishAllowNullSuppression 223
  PowerCube updates
    EventEnd 124
    EventStart 124
  Q
  queries
    character sets 161
    DataRange 162
    decimal separators 162
    descriptions 161
    drill-through to IQD 163
    isolation level 163
    naming 11
    options 161
    PackageReportSource 164
    PreSummarized 165
    Source 166
    SourceType 167, 182
    Speed 167
    SQL string 167
    Timing 169
    Query 233

  R
  ranges
    DataRange 162
    RefreshDescription 205
    RefreshLabel 205
    RefreshShortName 205
  relative time
    context offset 173
    context period 173
    drill categories 174
    number of running periods 178
    TargetLevel 180
    TargetOffset 180
    TimeAggregate 180
    ToDateLevel 180
  remarks
    creating in MDL file 62
    ReportPartitions 140
    restricted characters
      in category names 11
      in column names 12
      in measure names 133
  reversing the sign
    financial reporting 215
    financial reporting option 179
  Rollup 178, 214
  rollups
    average weighting 216
    duplicate 210
    duplicate weights 211
    external 214
    ignoring missing values for specific types 212
    PreSummarized 165
    regular 214
    regular timing 216
    time-state 215
    time-state weighting 215
  Root 233
  root categories
    RootCatMake 141
    RootCatUpdate 141
    RunningPeriods 178
  S
  SaveMDL 142
  SavePY 142
  Scale 189, 214
  scaling
    measure output 214
  scenario dimensions
    cube-open category 197
    SecurityNamespaceAdd 143
    SecurityNamespaceDelete 143
    SecurityNamespaceMake 144
    SecurityNamespaceUpdate 144
    SecurityObjectAdd 145
    SecurityObjectDelete 146
    SecurityObjectMake 146
    SecurityObjectUpdate 147
    SegmenterDimension 224
    SegmenterLevel 224
    SegmenterPrompt 165
    SegmenterPromptEnabled 165
  scenario dimensions
    HideValue 176
Separator 165
separators
   ThousandSep 169
ServerCube 224
SetCurrent 165
setting
   global processing preferences 14
Share 178, 205
ShortName 178, 205, 214
Sign 179, 215
Signon 196, 233
SignonAdd 147
SignonDelete 148
SignonInfo 227
SignonMake 149
SignonNamespace 227
signonopts 226
signons
   creating 19
   database specification 219
   descriptions 227
   prompt for passwords 226
   unencrypted passwords 226
SignonType 227
SignonDelete 148
Size 189
SortAs 204
sorting
   setting global preferences 14
Source 166, 206
source columns 206
source files 219
SourceInfo 166
SourceSignonList 166
SourceType 167, 182
SourceValue 179
special categories
   SpecialCatAdd 150
   SpecialCatDelete 151
   SpecialCatMake 152
   SpecialCatUpdate 153
SpecialCategory 233
Speed 167
SplitWeek 179
SQL 167
Stamp 168
status
   revert to defaults using MDCClear 130
Status 224
Storage 190, 215
stream extract
   SegmenterPrompt 165
   SegmenterPromptEnabled 165
StreamExtractSize 168
structured MDL
   introduction 233
SubDimCat 233
subdimensions
   naming 11
   SubDimRootMake 153
   SubDimRootUpdate 154
SubDimRootMake 153
SubDimRootUpdate 154
SummarizeCat 155
SummarizeLevel 156
summarizing
   external category values 225
Summary 231
SummaryLevel 225
Suppressed 179, 206, 231
suppression
   PublishAllowAccessToSuppressionOptions 223
   PublishAllowMultiEdgeSuppression 223
   PublishAllowNullSuppression 223
SuppressNull 168
syntax conventions 61
T
   tables
      DataRange 162
      TargetLevel 180
      TargetOffset 180
      ThousandSep 169
time categories
   Date 174
time dimensions
   automatically set current period 198
   DimType 197
   EarliestDate 198
   LatestDate 198
   manually set current period 198
time periods
   Current 173
time-state rollups
   average weighting 216
   TimeAggregate 180
   TimeArray 190
   TimeArrayCol 190
   TimeArrayMonth 190
   TimeRank 206
   TimeStateRollup 215
   TimeStateWeight 215
Timing 169, 216
TimeBasedPartitionedCube 225
ToDateLevel 180
U
   UniqueCategories 206
   UniqueMove 207
   uniqueness
      rules for naming objects 13
   UpdateCycle 169
   UpdateForwardReference 156
   UpdatePowerCubes 157
   UseAlternateFilename 225
   UserId 226
V
   values
      SuppressNull 168
   verb MDL 7
   Version 169
   versions
      supported by this documentation xiii
View 196
View names 12
ViewAdd 157
ViewDelete 158
ViewListUpdate 158
ViewMake 159
ViewName 233
viewopts 228
views
  Apex 228
  Cloak 228
  Exclude or Filter 228
ViewUpdate 160

W
week categories
  ExtraWeek 174

Y
YearBegins 181

WeekBegins 181
weeks
  DaysInWeek option 196
  PartialWeek 177
  Weight 216
  WeightID 216