

IBM Cognos Analytics  
Version 12.0.x

*Transformer Automation Guide*



©

## Product Information

This document applies to IBM Cognos Analytics version 12.0.0 and may also apply to subsequent releases.

## Copyright

Licensed Materials - Property of IBM

© Copyright IBM Corp. 2005, 2023.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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](http://www.ibm.com/legal/copytrade.shtml)" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

The following terms are trademarks or registered trademarks of other companies:

- Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- 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.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft product screen shot(s) used with permission from Microsoft.

© **Copyright International Business Machines Corporation .**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Chapter 1. Overview.....</b>	<b>1</b>
Object Creation Overview.....	1
Type Library Overview for Visual Basic.....	2
Type Library Overview for Visual C++ and Visual Studio 2009 C#.....	2
Type Library Overview for Transformer .....	3
Methods Overview.....	4
Value Lists and Constants Overview.....	4
Transformer Object Hierarchy Map.....	5
Transformer Object Hierarchy Map A.....	7
Transformer Object Hierarchy Map B.....	7
Transformer Object Hierarchy Map C.....	8
Transformer Object Hierarchy Map D.....	8
Transformer Object Hierarchy Map E.....	8
Software Development Kit Preference Settings.....	9
IBM Cognos Series 7 Models.....	9
 <b>Chapter 2. Collections.....</b>	 <b>11</b>
Associations Collection.....	12
CalculationDefinitions Collection.....	13
Categories Collection.....	14
CategorySets Collection.....	15
ChildCubes Collection.....	15
Columns Collection.....	16
CubeCustomViews Collection.....	17
Cubes Collection.....	17
CurrencyRates Collection.....	18
CurrencyRecords Collection.....	18
CurrencyTables Collection.....	19
CustomViews Collection.....	20
DataSources Collection.....	21
DimensionLevels Collection.....	22
Dimensions Collection.....	22
DrillDowns Collection.....	23
DrillThroughTargets Collection.....	24
Filters Collection.....	24
LevelCategories Collection.....	25
LevelDrillDowns Collection.....	26
Levels Collection.....	26
Measures Collection.....	27
Names Collection.....	28
Namespaces Collection.....	28
PackageDatasourceConnections Collection.....	29
Packages Collection.....	30
Prompts Collection.....	31
Queries Collection.....	31
Reports Collection.....	32
SecurityObjects Collection.....	33
Signons Collection.....	33
SuspendedModels Collection.....	34
Views Collection.....	35

<b>Chapter 3. Objects.....</b>	<b>37</b>
Application Object.....	39
Association Object.....	41
CalculationDefinition Object.....	42
Category Object.....	43
CategorySet Object.....	45
ChildCube Object.....	46
Column Object.....	48
CrossTabDataSource Object.....	50
Cube Object.....	51
CubeGroup Object.....	54
CurrencyRate Object.....	57
CurrencyRecord Object.....	58
CurrencyTable Object.....	59
CustomView Object.....	61
DataSource Object.....	62
DateDimension Object.....	64
DateDrillDown Object.....	66
DateLevel Object.....	68
DateWizard Object.....	71
DbDataSource Object.....	72
Dimension Object.....	74
DrillDown Object.....	76
DrillThroughTarget Object.....	77
Filter Object.....	78
FlatFileDataSource Object.....	79
IqdDataSource Object.....	81
Level Object.....	83
Measure Object.....	85
Model Object.....	88
Name Object.....	90
Namespace Object.....	91
Package Object.....	92
PackageDatasourceConnection Object.....	93
Prompt Object.....	94
Query Object.....	95
Report Object.....	96
SecurityObject Object.....	97
Signon Object.....	98
SpecialCategory Object.....	99
SuspendedModel Object.....	102
View Object.....	103
 <b>Chapter 4. Methods.....</b>	 <b>105</b>
Add Method ().....	108
Add Method (Categories).....	109
Add Method (CustomViews).....	110
Add Method (DrillThroughTargets).....	110
Add Method (Objects).....	111
AddDeployLocation Method.....	112
AddToCustomView Method.....	113
AddToFolder Method.....	113
AssociateWith Method.....	114
CheckLocalPowerCubes Method.....	115
CheckModel Method.....	116
CleanHouse Method.....	116

ClearDeployLocations Method.....	117
Close Method .....	117
ConnectWithCategory Method.....	118
CreateAlternateDrillDown Method.....	119
CreateDateDimension Method.....	119
CreateMDCFile Method.....	120
CreateMDCFiles Method.....	121
Delete Method.....	121
DeleteAllCustomViews Method.....	123
DeleteAllSecurityObjects Method.....	123
DeployCube Method.....	123
DeployCubes Method.....	124
DimensionAssociateWith Method.....	124
DoAutoDesign Method.....	125
FindCategoryByCatCode Method.....	126
GenerateCategories Method.....	126
GenerateDateCategories Method.....	127
GetDefaultCategory Method.....	128
GetViewStatus Method.....	128
IsExcludeDateDimension Method.....	129
IsExcludeDateLevel Method.....	129
IsExcludeDimension Method.....	130
IsExcludeLevel Method.....	131
Item Method.....	131
Item Method ().....	133
Item Method ().....	134
LoadCurrencyTable Method.....	135
Logoff Method.....	135
Logon Method.....	136
Move Method.....	136
MoveToCategory Method.....	137
MoveToLevel Method.....	138
NewModel Method.....	139
OpenModel Method.....	140
OpenSuspendedModel Method.....	140
PublishDatasource Method.....	141
PublishPackage Method.....	142
Remove Method.....	142
RemoveCubeCustomView Method.....	144
RemoveFromFolder Method.....	144
RemoveSuspendedModel Method.....	145
ResetPartitions Method.....	146
Save Method.....	146
SaveAs Method.....	146
SetAllocation Method.....	147
SetDefaultCategory Method.....	148
SetDeployType Method.....	149
SetExcludeDateDimension Method.....	150
SetExcludeDateLevel Method.....	151
SetExcludeDimension Method.....	151
SetExcludeLevel Method.....	152
SetViewStatus Method.....	153
TestBuild Method.....	153
Update Method.....	154
Verify Method.....	156

## **Chapter 5. Properties.....157**

ActivityMeasure Property.....	171
Aggregate Property.....	172
AllocationMeasure Property.....	173
AllocationType Property.....	174
AllowCurrencyConversion Property.....	175
AllowDrillThrough Property.....	175
AlternateQueryPath Property.....	176
AltMDCFile Property.....	177
AlwaysUseTransformerSignon Property.....	177
Apex Property.....	178
Application Property.....	178
AssociationRole Property.....	181
Associations Property.....	181
AssociationType Property.....	182
AutoLogon Property.....	183
AutoSummary Property.....	183
BlankSubstitute Property.....	184
BlockParentTotals Property.....	184
CacheCrossTabs Property.....	185
CalculationDefinitions Property.....	186
CAMID Property.....	186
CanAllocate Property.....	187
CanAllocateByMeasure Property.....	188
CanAllocateMeasure Property.....	189
Categories Property.....	190
Category Property.....	191
CategoryCount Property.....	191
CategoryCountLevel Property.....	192
CategorySets Property.....	193
CharacterType Property.....	193
ChildCategories Property.....	194
ChildCubes Property.....	194
ChildCustomViews Property.....	195
ChildMeasures Property.....	196
Code Property.....	196
Columns Property.....	197
ColumnsLoaded Property.....	198
CompressMDC Property.....	198
Connection Property.....	199
Consolidate Property.....	199
Context Property.....	200
ContextLevel Property.....	201
ContextOffset Property.....	202
ConvergenceLevel Property.....	202
Count Property.....	203
CountryCode Property.....	204
CubeCodePage Property.....	205
CubeCreation Property.....	207
CubeCustomViews Property.....	207
Cubes Property.....	208
CubeStamp Property.....	208
CurrencyCountryLabel Property.....	209
CurrencyDecimals Property.....	209
CurrencyFormatOverride Property.....	210
CurrencyIsEMU Property.....	211
CurrencyIsEuro Property.....	212
CurrencyRates Property.....	212
CurrencyRecord Property.....	213

CurrencyRecords Property.....	214
CurrencySymbol Property.....	214
CurrencyTable Property.....	215
CurrencyTables Property.....	215
CurrencyTableType Property.....	216
CurrentModel Property.....	217
CurrentValueIndex Property.....	217
CustomView Property.....	218
CustomViews Property.....	218
DataCharacterSet Property.....	219
DataClass Property.....	220
DataRange Property.....	220
DataSource Property.....	221
DataSourcePath Property.....	221
DataSources Property.....	222
DataSourceWindowsLocation Property.....	222
DataTemporaryFilePath Property.....	223
DateDegreeofDetail Property.....	224
DateDegreeofDetailLevelName Property.....	224
DateFormat Property.....	225
DateFunction Property.....	226
DateInputFormat Property.....	227
DateLevel Property.....	227
DateWizard Property.....	228
DecimalPoint Property.....	229
Decimals Property.....	229
DefaultCategoryOrderBy Property.....	230
DefaultDateFormat Property.....	230
Description Property.....	231
DesiredPartitionSize Property.....	232
DetachDataSource Property.....	233
DetailLevel Property.....	233
Dimension Property.....	234
DimensionInclude Property.....	234
DimensionLevels Property.....	235
DimensionName Property.....	236
Dimensions Property.....	236
DimensionView Property.....	237
DimensionViewType Property.....	238
DisplayName Property.....	239
DrillCode Property.....	240
DrillDowns Property.....	240
DrillInclusion Property.....	241
DrillThroughTargets Property.....	241
DuplicateRollup Property.....	242
DuplicateWeight Property.....	243
EarliestDate Property.....	244
EMUEntryDate Property.....	244
EnableMessageLogging Property.....	245
EnableTimePeriod Property.....	246
EstimatedRows Property.....	246
ExcludeAutoPartition Property.....	247
ExpressionText Property.....	248
External Property.....	249
FieldSeparator Property.....	250
FileName Property.....	250
Filters Property.....	251
FindCategoryByCatCode Property.....	252

Format Property.....	252
FormatDecimals Property.....	254
FullName Property.....	254
GenerateCategories Property.....	255
GenerateDateCategories Property.....	256
GenerateDates Property.....	256
GeneratePowerCube Property.....	257
GenerateTimePeriod Property.....	257
Group Property.....	258
GroupDimension Property.....	259
GroupLevel Property.....	259
HasSubdimension Property.....	260
HideValue Property.....	261
ID Property.....	261
IgnoreMissingValue Property.....	262
Inclusion Property.....	262
IncrementalUpdate Property.....	263
InputScale Property.....	264
IsAnyColumnMismatched Property.....	265
IsBad Property.....	265
IsExpressionValid Property.....	266
IsFolder Property.....	267
IsManual Property.....	267
IsMDCInUse Property.....	268
IsolationLevel Property.....	268
IsPrimary Property.....	269
IsTimeBasedPartitionedCube Property.....	270
KeyName Property.....	271
Label Property.....	271
LastUseDate Property.....	272
LatestDate Property.....	273
Level Property.....	274
LevelCategories Property.....	274
LevelDrillDowns Property.....	275
Levels Property.....	275
LocalPath Property.....	276
LogErrorLevel Property.....	276
LogFileAppend Property.....	277
LogFileName Property.....	278
LogFilesPath Property.....	278
Lunar Property.....	279
ManualCurrentPeriod Property.....	279
MaximizeSpeed Property.....	280
MaxNumPartLevels Property.....	281
MaxTransactionNumber Property.....	281
MDCFile Property.....	282
MeasureInclude Property.....	283
MeasureName Property.....	284
Measures Property.....	284
MeasureType Property.....	285
MissingValue Property.....	285
ModelName Property.....	286
ModelsPath Property.....	286
ModelTemporaryFilesPath Property.....	287
ModelType Property.....	288
MonthType Property.....	288
Name Property.....	289
Namespaces Property.....	290



NewCatsLocked Property.....	291
ObjectCAMID Property.....	291
ObjectName Property.....	292
Optimize Property.....	293
OrderByDescending Property.....	293
OrderByStorageType Property.....	294
Origin Property.....	295
OriginalName Property.....	295
Orphanage Property.....	296
OutputScale Property.....	297
Packages Property.....	298
PackagesDatasourceConnections Property.....	298
Parent Property.....	299
ParentCategories Property.....	301
Partition Property.....	302
Password Property.....	303
PatFile Property.....	303
Path Property.....	304
PopulateByDataSource Property.....	305
Position Property.....	305
PowerCubesPath Property.....	306
PowerPlayPath Property.....	306
Precision Property.....	307
PromptForPassword Property.....	308
Prompts Property.....	308
PromptValueType Property.....	309
QualifiedName Property.....	309
QuarterType Property.....	310
Queries Property.....	310
QyPath Property.....	311
Rate Property.....	312
RefName Property.....	312
RefreshDescription Property.....	313
RefreshLabel Property.....	313
RefreshShortName Property.....	314
RegularRollup Property.....	315
RegularWeight Property.....	316
Reports Property.....	316
ReverseSign Property.....	317
Rollup Property.....	318
RollupTiming Property.....	318
RowsAsSample Property.....	319
RowsChecked Property.....	320
RunningPeriods Property.....	320
SecurityObjects Property.....	321
Server Property.....	322
ServerModelPath Property.....	322
ServerPath Property.....	323
ServerQuery Property.....	323
ServicesBuildNumber Property.....	324
ServicesVersionText Property.....	324
SetsCurrentPeriod Property.....	325
ShortName Property.....	326
Signon Property.....	326
SignOnNamespace Property.....	327
Signons Property.....	328
SignonType Property.....	328
Size Property.....	329

SortComparisonRule Property.....	330
SourceType Property.....	330
SpecialCategoryCount Property.....	331
SQLExpression Property.....	332
Status Property.....	332
StorageType Property.....	333
StreamExtractAllowed Property.....	333
StreamExtractSize Property.....	334
SummaryLevel Property.....	334
SuppressNull Property.....	335
SuspendedModels Property.....	336
TargetLevel Property.....	336
TargetOffset Property.....	337
ThousandPoint Property.....	337
Time Property.....	338
TimeArrayColumn Property.....	339
TimeArrayStartMonth Property.....	339
TimeArrayType Property.....	340
TimeRank Property.....	341
TimeStamp Property.....	341
TimeStateRollup Property.....	342
TimeStateWeight Property.....	342
ToDateLevel Property.....	343
TransdaPath Property.....	344
TransformerSignon Property.....	344
Type Property.....	345
Unique Property.....	346
UniqueMove Property.....	347
UseAltMDCFile Property.....	348
User Property.....	349
UserCAMID Property.....	349
UserID Property.....	350
Value Property.....	350
ValuesCount Property.....	351
Version Property.....	352
Views Property.....	352
ViewType Property.....	353
WeekAdd Property.....	353
WeekSpan Property.....	354
WeekStart Property.....	355
WeekStartDay Property.....	356
WorkingDay Property.....	356
WorkingDays Property.....	357
YearStartDay Property.....	358
YearType Property.....	359

## **Chapter 6. Constants..... 361**

xtrAllocationType Value List.....	361
xtrAssociationRole Value List.....	361
xtrAssociationType Value List.....	362
xtrCharacterType Value List.....	363
xtrCubeConsolidate Value List.....	363
xtrCubeCreation Value List.....	364
xtrCubeOptimize Value List.....	364
xtrCubeStatus Value List.....	365
xtrCurrencyTableType Value List.....	366
xtrDataClass Value List.....	366

xtrDateCategoriesGeneration Value List.....	367
xtrDateFormat Value List.....	368
xtrDateLevel Value List.....	368
xtrDeployType Value List.....	369
xtrDuplicateRollup Value List.....	370
xtrGenerateOptions Value List.....	370
xtrInclusion Value List.....	371
xtrMeasureType Value List.....	372
xtrMissingValue Value List.....	372
xtrObjectType Value List.....	373
xtrOrigin Value List.....	375
xtrPowerCubeGeneration Value List.....	375
xtrPreferences Value List.....	376
xtrPromptValueType Value List.....	377
xtrRollup Value List.....	377
xtrRollupTiming Value List.....	378
xtrSecurityType Value List.....	379
xtrSourceType Value List.....	379
xtrSpecialFunction Value List.....	381
xtrStorage Value List.....	382
xtrTimeAggregate Value List.....	383
xtrTimeArrayType Value List.....	383
xtrTimeRollup Value List.....	384
xtrTimeType Value List.....	385
xtrViewStatus Value List.....	386
xtrViewType Value List.....	387
xtrWeekAdd Value List.....	387
xtrWeekDay Value List.....	388
xtrWeekSpan Value List.....	389
<b>Chapter 7. UI Equivalents.....</b>	<b>391</b>
Collections.....	391
Objects.....	392
Methods.....	394
Properties.....	396
Value Lists and Constants.....	415
<b>Chapter 8. Samples and Examples.....</b>	<b>419</b>
Open a Model and Specify an Order by Association Example.....	419
Open a Model and Add a Calculation Example.....	420
Create a Relative Time Category Example.....	420
Add a Cube Group Example.....	421
Add an Additional Data Source to a Model Example.....	422
Open a Model and Modify the Cube Properties Example.....	423
Create a Custom View Example.....	424
Open a Model and Add a Currency Record Example.....	425
Create a Cube Using DoAutoDesign and TestBuild Methods Example.....	426
Select, Change, and Update a Dimension Example.....	427
Delete a Level from a Level Collection Example.....	427
Move a Measure Object and Change the Revenue Measure Rollup Example.....	428
Create a Partition and Check the Model Example.....	428
Open a Model and Drill Through to a PowerCube Example.....	429
Add the Authors Role to a Custom View Example.....	429
Check for a Suspended Model Example.....	431
Open a Model and Create a Dimension View Example.....	431
Add a Cube Group to a Model Example.....	432
Create a Model and Update Properties for a Date Dimension Example.....	433

Create an Alternate Drill-down Path Example.....	434
Create a Category Count Measure and Add to Model Example.....	435
Use the DateWizard to Create a Time Dimension Example.....	435
Add a Dimension View to a Model Example.....	436
Move a Child Category to a Different Parent Example.....	437
Add a Table to a File and Load Data Example.....	438
Generate a Time Dimension Based On a Lunar Year Example.....	439
Move a Child Category to a Different Parent Category Example.....	440
Set Attributes for an Application Example.....	441
Add and Delete a Package Example.....	441
Add and Delete a Report Example.....	442
Create a Query Example.....	443
Create and Delete Filters for a Model Example.....	444
Create a Single-valued Prompt Example.....	445
Create a Multi-valued Prompt Example.....	446
Create a New Model and Publish a PowerCube Example.....	448
Copy and Activate a PowerCube Example.....	448
Create a Model Using a Signon and an IQD Data Source Example.....	449
Create a Model Using a Signon and Package Data Source Example.....	450
<b>Notices.....</b>	<b>453</b>
<b>Index.....</b>	<b>455</b>

---

# Chapter 1. Overview

Transformer OLE automation uses a program language interface to provide an alternative to the Transformer user interface. This document includes information about the UI equivalents of some Transformer OLE methods and properties, to help you get started.

OLE automation presents a Transformer model as a set of collections and objects that are modified by properties and acted upon by methods. Use automation to create and manage dimensions, levels, data sources, measures, categories, drill-down paths, and other model objects, and to create PowerCubes.

When you create a model in OLE, you must create your objects and assign values to them in a hierarchical sequence. For example, you cannot create a level until you have created the dimension in which the level resides.

You can use the hierarchy map [“Transformer Object Hierarchy Map” on page 5](#) to determine the order of object creation.

For general information about creating Transformer macros, see the following overview topics.

- [“Object Creation Overview” on page 1](#)
- [“Type Library Overview for Visual Basic” on page 2](#)
- [“Type Library Overview for Visual C++ and Visual Studio 2009 C#” on page 2](#)
- [“Type Library Overview for Transformer ” on page 3](#)
- [“Methods Overview” on page 4](#)
- [“Value Lists and Constants Overview” on page 4](#)

## Object Creation Overview

---

How you create objects within Transformer OLE depends on which editor you are using, which language you are using, and what you are trying to achieve.

Almost all examples shown in Transformer OLE documentation create objects in a generic fashion, as follows:

```
Dim objTransApp As Object
```

```
Dim objModel As Object
```

```
Dim objDataSource As Object
```

```
Dim objDimension As Object
```

Objects created in this generic fashion are compatible with VB.NET.

A limitation to the above method is that you cannot take advantage of the features of the Transformer type library. To access the type library, you must use Microsoft Visual Basic or a VB-compatible editor and create objects, as follows:

```
Dim objTransApp As Application
```

```
Dim objModel As Model
```

```
Dim objDataSource As DataSource
```

```
Dim objDimension As Dimension
```

## Type Library Overview for Visual Basic

---

A type library is a binary file that contains class interfaces and value lists (enumerators) recognized by an OLE automation server. Type libraries are supported by development tools, such as Microsoft Visual Basic, Visual C#, and Visual C++. Type libraries that are integrated into the development environment let you retrieve automation information when you compile and run an application.

The Transformer type library contains information about objects and collections, along with their properties and methods. It also gives you features, such as

- compile-time error checking rather than run-time error checking
- early binding of methods and properties rather than late binding, resulting in faster execution speed
- advanced help features during script creation

Use the information in the Transformer type library to build macro scripts in VB.NET, C#, or to create header files and implementation files in Visual C++. You can also use other applications, such as Word or Excel, to view the type library through the Visual Basic Editor included with these products.

Type libraries and OLE automation only work in Windows environments and not in other environments, such as UNIX.

The following procedures use Microsoft Visual Studio as the Integrated Development Environment (IDE).

### Procedure

1. Open a project.
2. From the **Project** menu, click **Add References**.
3. Click **Add** and in the **Add Reference** dialog box, select the **IBM Cognos Transformer Application Control**.

**Note:** If the entry isn't in the list, click **Browse** and select TransformerSDK.dll located in the bin folder in the Transformer installation location. TransformerSDK.dll is included with the Transformer installation.

4. From the **View** menu, click **Object Browser**.
5. From the Library list (in the top left corner), click **Interop.TransformerSDKLib**.
6. Find the object or collection you want, and select the corresponding methods and properties.
7. Use these methods and properties in your macro.

## Type Library Overview for Visual C++ and Visual Studio 2009 C#

---

A type library is a binary file that contains class interfaces and value lists (enumerators) recognized by an OLE automation server. Type libraries are supported by development tools, such as Microsoft Visual Basic, Visual C#, and Visual C++. Type libraries that are integrated into the development environment let you retrieve automation information when you compile and run an application.

The Transformer type library contains information about objects and collections, along with their properties and methods. It also gives you features, such as

- compile-time error checking rather than run-time error checking
- early binding of methods and properties rather than late binding, resulting in faster execution speed
- advanced help features during script creation

Use the information in the Transformer type library to build macro scripts in VB.NET, C#, or to create header files and implementation files in Visual C++. You can also use other applications, such as Word or Excel, to view the type library through the Visual Basic Editor included with these products.

Type libraries and OLE automation only work in Windows environments and not in other environments, such as UNIX.

The following procedures use Microsoft Visual Studio as the Integrated Development Environment (IDE).

## Procedure

1. Open Visual Studio.
2. Open a project.

The project must allow the inclusion of type libraries.

3. From the **Solution Explorer**, click **References**, right-click and click **Add Reference**.

**Tip:** Alternatively, from the **Project** menu, click **Add Reference**.

4. In the **Add Reference** dialog box, click **Browse** and select the TransformerSDK.dll file located in the bin folder where you installed Transformer Software Development Kit. TransformerSDK.dll is included as part of the Transformer installation.

To view the reference that you added, from the **Tools** menu, click **OLE/Com Object Viewer**. Under **Controls**, see **IBM Cognos Transformer Application Control**.

To view more details, right-click, and from the shortcut menu click **View type information**.

## Type Library Overview for Transformer

---

A type library is a binary file that contains class interfaces and value lists (enumerators) recognized by an OLE automation server. Type libraries are supported by development tools, such as Microsoft Visual Basic, Visual C#, and Visual C++. Type libraries that are integrated into the development environment let you retrieve automation information when you compile and run an application.

The Transformer type library contains information about objects and collections, along with their properties and methods. It also gives you features, such as

- compile-time error checking rather than run-time error checking
- early binding of methods and properties rather than late binding, resulting in faster execution speed
- advanced help features during script creation

Use the information in the Transformer type library to build macro scripts in VB.NET, C#, or to create header files and implementation files in Visual C++. You can also use other applications, such as Word or Excel, to view the type library through the Visual Basic Editor included with these products.

Type libraries and OLE automation only work in Windows environments and not in other environments, such as UNIX.

The following procedures use Microsoft Visual Studio as the Integrated Development Environment (IDE).

## Procedure

1. Open Visual Studio.
2. Open a project.

The project must allow the inclusion of type libraries.

3. From the **Project** menu, click **Add Class**, and select **MFC Categories/MFC Class From ActiveX Control** from the **Templates** list in the **Add Class** dialog box.

and in the **Add Class from ActiveX Control Wizard**, click **MFC Categories/MFC Class From ActiveX Control**.

4. In the **Available ActiveX controls** box, select the **IBM Cognos Transformer Application Control<version#>**, where version# is the version number of the Transformer Software Development Kit.

If the **Add class from Registry** is selected, the TransformerSDK.dll file must be specified.

5. Select the interfaces you want, and click **Finish**.
6. Click **Class View** to view the selected classes and their objects in the workspace.
7. Click **File Viewer** to view the object definitions in the different wrapper classes.

## Methods Overview

---

In OLE automation, methods are used to create, modify, or remove objects.

Some methods included with Transformer behave like subroutines and some behave like functions. You can determine the type of method in one of the following ways:

- If you are using a Visual Studio Development tool, look up the method in the Object Browser.
- Use the Oleview tool. To access Oleview, from the **Start** menu, click **Run Menu** and type **oleview**. Click **OK**. Under **Controls**, find **IBM Cognos Transformer Application Control**. To view the type information, right-click, and from the shortcut menu click **View Type Information**.

The TransformerSDK type information shows parentheses around parameters for both subroutine and function methods, but they are not always needed. Methods that define subroutines do not need to use parentheses and do not return values. Functions must use parentheses and return values. This also varies depending on the implementation language and if generic objects are used.

## Value Lists and Constants Overview

---

Many properties and methods in Transformer OLE use a set of enumerated data types or value lists. Each value list contains a set of constants that define the action of the applicable property or method.

The names of value lists are prefixed with the letters 'xtr'. The constants themselves are prefixed with the letters 'tr'. For example, here is the xtrCubeCreation value list used to set cube creation options:

- trCubeCreationDefault
- trCubeCreationON
- trCubeCreationOFF

You can use constants of value lists in both set and return operations. This example returns a constant of the xtrObjectType value list:

```
If objDimension1.ObjectType = xtrObjectType.trDateDimension  
Then
```

```
.
```

```
.
```

```
End If
```

This example uses a constant of the xtrObjectType value list to add a DateDimension object to a collection:

```
objDimCollection.Add(xtrObjectType.trDateDimension)
```

Note that if you previously used IBM® CognosScript Editor to create Transformer OLE macros, the TranConst.inc file has been updated to work with Cognos® TransformerSDK. However, this editor application is being discontinued. This file cannot be used with VB.NET, C# or C++.

A copy of TranConst.inc is available in the Transformer installation location in the *installation\_location\templates\cogtr\TransformerSDK* folder.

Here is an example of an Include statement that references the TranConst.inc file.

```
$Include "TranConst.inc"
```

While all constants have numeric equivalents, we recommend that you reference the constant by name to avoid problems when you upgrade your TransformerSDK. When using any .NET technology, the enumeration 'type.enumeration' item notation must be used.



## Transformer Object Hierarchy Map

---

When you build a model, you must create your collections and objects and assign values to them in hierarchical sequence.

The following map lists all collections and objects and shows their order in the hierarchy.

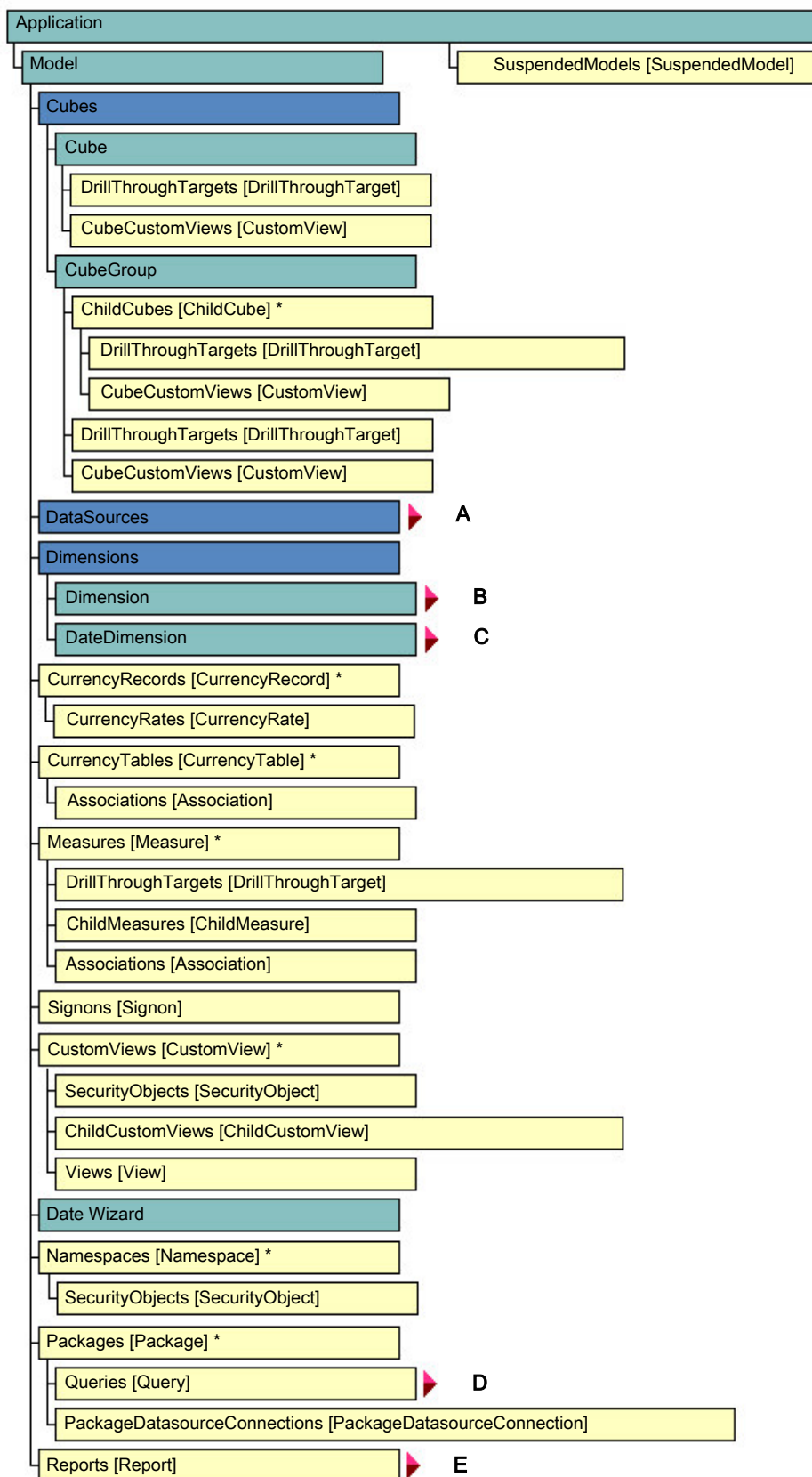



Figure 1. All objects

- Objects
- Collections

-  Collections and objects

Items with an asterisk (\*) apply to the object, not the collection.

## Transformer Object Hierarchy Map A

The following map shows the DataSources hierarchy.

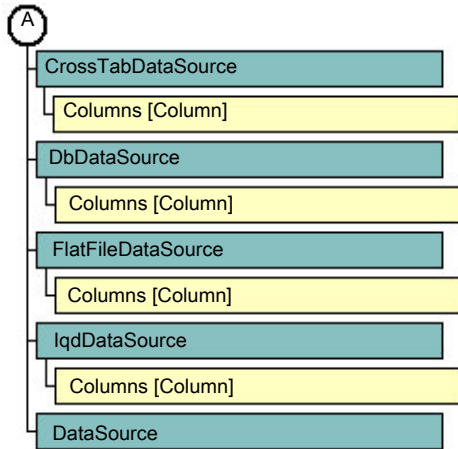





Figure 2. DataSources hierarchy

-  Objects
-  Collections
-  Collections and objects

## Transformer Object Hierarchy Map B

The following map shows the Dimension hierarchy.

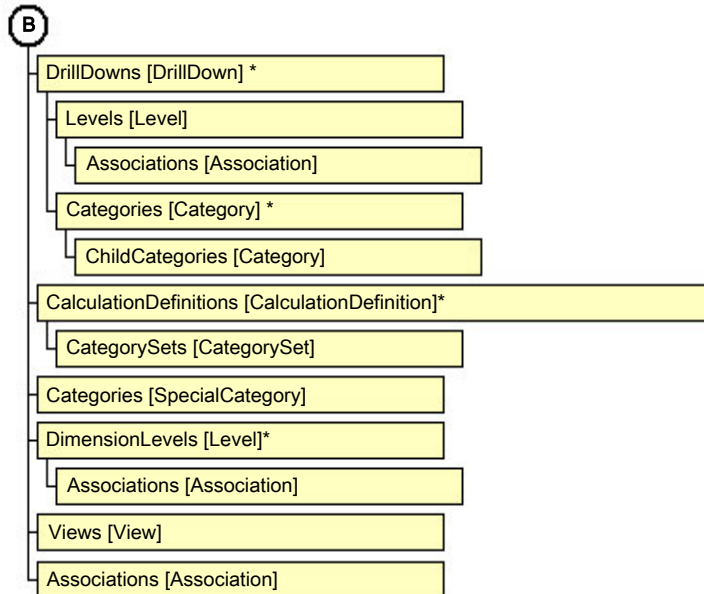





Figure 3. Dimension hierarchy

-  Objects
-  Collections
-  Collections and objects

Items with an asterisk (\*) apply to the object, not the collection.

## Transformer Object Hierarchy Map C

The following map shows the DateDimension hierarchy.

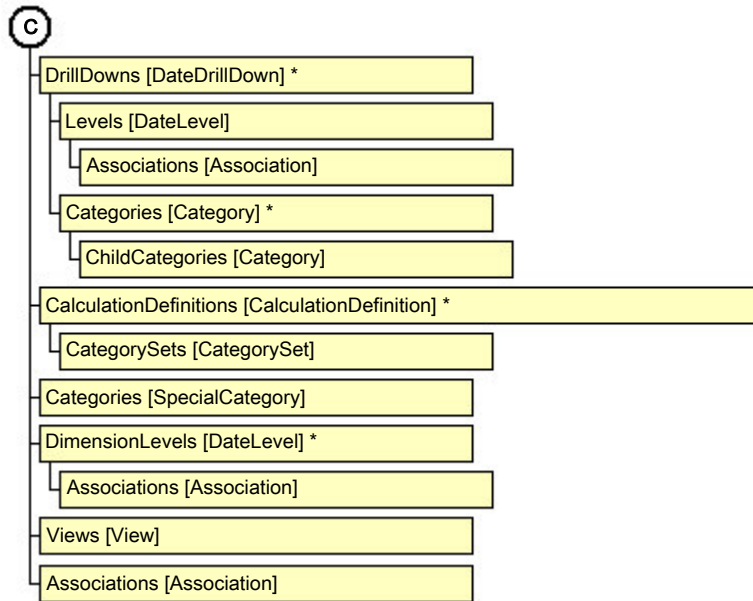


Figure 4. DateDimension hierarchy

- Objects
- Collections
- Collections and objects

Items with an asterisk (\*) apply to the object, not the collection.

## Transformer Object Hierarchy Map D

The following map shows the Query hierarchy.

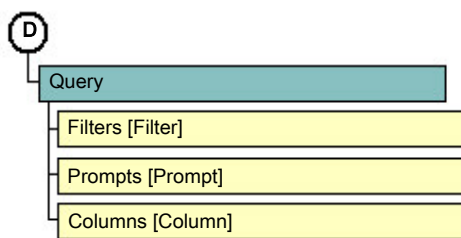


Figure 5. Query hierarchy

- Objects
- Collections
- Collections and objects

## Transformer Object Hierarchy Map E

The following map shows the Report hierarchy.

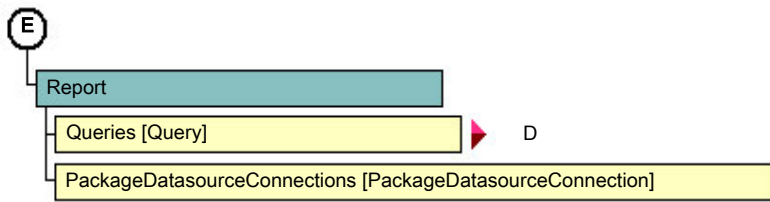





Figure 6. Report hierarchy

-  Objects
-  Collections
-  Collections and objects

## Software Development Kit Preference Settings

---

Make sure that you set the Software Development Kit preferences to guarantee that all the applications work consistently. The preferences can be set using the Application object properties.

## IBM Cognos Series 7 Models

---

Note that before using your IBM Cognos Series 7 models with this version of OLE automation, you must upgrade them.

For information about upgrading IBM Cognos Series 7 models, see the *Transformer User Guide*.



---

## Chapter 2. Collections

The following tables lists all the Transformer OLE automation collections.

Object	Description
<a href="#">Associations Collection</a>	Contains all the Association objects for a given object.
<a href="#">CalculationDefinitions Collection</a>	Contains all the CalculationDefinition objects for a given dimension.
<a href="#">Categories Collection</a>	Groups Category objects or SpecialCategory objects.
<a href="#">CategorySets Collection</a>	Contains all the CategorySet objects used by a CalculationDefinition object to calculate values.
<a href="#">ChildCubes Collection</a>	Groups ChildCube objects.
<a href="#">Columns Collection</a>	Groups Column objects in a data source.
<a href="#">CubeCustomViews Collection</a>	Contains CustomView objects that have access to a specific Cube, CubeGroup, or ChildCube object.
<a href="#">Cubes Collection</a>	Groups all Cube and CubeGroup objects in a model.
<a href="#">CurrencyRates Collection</a>	Groups CurrencyRate objects.
<a href="#">CurrencyRecords Collection</a>	Contains all the CurrencyRecord objects in a model.
<a href="#">CurrencyTables Collection</a>	Contains all CurrencyTable objects in a model.
<a href="#">CustomViews Collection</a>	Represents a collection of CustomView objects in a model.
<a href="#">DataSources Collection</a>	Contains all data sources in a model.
<a href="#">DimensionLevels Collection</a>	Contains a read-only list of unique levels in the related dimension.
<a href="#">Dimensions Collection</a>	Contains all Dimension and DateDimension objects in a model.
<a href="#">DrillDowns Collection</a>	Contains either DrillDown objects or DateDrillDown objects, but not both.
<a href="#">DrillThroughTargets Collection</a>	Groups related DrillThroughTarget objects.
<a href="#">Filters Collection</a>	Contains all Filter objects in a query.

Object	Description
<a href="#">LevelCategories Collection</a>	Contains a collection of categories for a specific level.
<a href="#">LevelDrillDowns Collection</a>	Represents a read-only collection of objects representing drill-down paths.
<a href="#">Levels Collection</a>	Groups Level objects or DateLevel objects, but not both.
<a href="#">Measures Collection</a>	Contains all Measure objects in a model.
<a href="#">Names Collection</a>	Contains a read-only group of Name objects.
<a href="#">Namespaces Collection</a>	Contains all Namespace objects in a model.
<a href="#">PackageDatasourceConnections Collection</a>	Contains all PackageDatasourceConnection objects.
<a href="#">Packages Collection</a>	Contains all Package objects in a model.
<a href="#">Prompts Collection</a>	Contains all Prompt objects in a query.
<a href="#">Queries Collection</a>	Contains all Query objects in a package or report.
<a href="#">Reports Collection</a>	Groups related Report objects.
<a href="#">SecurityObjects Collection</a>	Contains all SecurityObject objects in a model.
<a href="#">Signons Collection</a>	Groups related Signon objects.
<a href="#">SuspendedModels Collection</a>	Contains all SuspendedModel objects available to the application.
<a href="#">Views Collection</a>	Groups View objects.

## Associations Collection

The Associations collection contains all the Association objects for a given object.

### Discussion

Use this collection to add, select or remove an Association object.

Many objects in a model, such as levels and dimensions, maintain associations with external data through one or more Association objects. The association supplies values for source data, as well as for properties that represent labels, descriptions, sort names, and so on.

To return this collection, use the Associations property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method (Objects)</a>	Adds an Association object to the collection.



Method	Description
<a href="#">Item Method</a>	Selects a specific Association object in the collection.
<a href="#">Remove Method</a>	Removes a specific Association object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Association objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objAssociation = objDrill.ConvergenceLevel.Associations.Add()
```

## CalculationDefinitions Collection

The CalculationDefinitions collection contains all the CalculationDefinition objects for a given dimension.

### Discussion

Use this collection to add, select or remove a CalculationDefinition object.

CalculationDefinition objects consist of complex expressions that use the values of one or more categories as part of the calculation.

To return this collection, use the CalculationDefinitions property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method (Objects)</a>	Adds a CalculationDefinition object to the collection.
<a href="#">Item Method</a>	Selects a specific CalculationDefinition object in the collection.
<a href="#">Remove Method</a>	Removes a specific CalculationDefinition object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of CalculationDefinition objects in the collection.

Property	Description
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
calcDef = dimension.CalculationDefinitions.Add()
```

## Categories Collection

The Categories collection groups Category objects or SpecialCategory objects.

### Discussion

Use the Categories property of a DrillDown object to return a collection of regular categories.

Use the ChildCategories property of a Category object to return a collection of descendant categories.

Use the Categories property of the Dimension or DateDimension object to return a collection of special categories.

Calculated categories, which are associated with regular or special categories, may exist in either collection.

To generate categories for all dimensions use the GenerateCategories method. You can manually modify these collections. For example, use the Add method to add SpecialCategory objects to a collection, or set the Inclusion property to exclude specific Category objects.

You can manually create a Categories collection as part of building manual levels. This is useful where there are too many Category objects in a generated collection to be easily viewed in PowerPlay®.

A CalculationDefinition object uses the Categories collection of a CategorySet when building an expression. Use the Add method of the Categories collection each time you add a category to the CategorySet.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method (Categories)</a>	Adds a Category or SpecialCategory object to the collection.
<a href="#">Item Method</a>	Selects a specific Category or SpecialCategory object in the collection.
<a href="#">Remove Method</a>	Removes a specific Category or SpecialCategory object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Category or SpecialCategory objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objSpecCategory = _  
objModel.Dimensions("Time").Categories.Add(xtrObjectType.trSpecialCategory)
```

## CategorySets Collection

The CategorySets collection contains all the CategorySet objects used by a CalculationDefinition object to calculate values.

### Discussion

Use this collection to add, select or remove a CategorySet object. Each CategorySet object, in turn, contains a Categories collection.

To return this collection, use the CategorySets property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method (Objects)</a>	Adds a CategorySet object to the collection.
<a href="#">Item Method</a>	Selects a specific CategorySet object in the collection.
<a href="#">Remove Method</a>	Removes a specific CategorySet object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of CategorySet objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

### Examples

```
catSet = calcDef.CategorySets.Add()
```

## ChildCubes Collection

The ChildCubes collection groups ChildCube objects.

### Discussion

Use this collection to select a ChildCube object. Each cube in this collection is part of a CubeGroup object and represents a single category in the level associated with the CubeGroup.

To return this collection, use the ChildCubes property.

These tables list related methods and properties.

Method	Description
<a href="#">Item Method</a>	Selects a specific ChildCube object in the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of ChildCube objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objCubesByRegion = objModel.Cubes.Add(xtrObjectType.trCubeGroup)
objChildCube = objCubesByRegion.ChildCubes("Central Europe")
```

## Columns Collection

The Columns collection groups Column objects in a data source.

### Discussion

Use this collection to add, select or remove a Column object. Transformer data sources have their data arranged in columns that act as the source of values for levels.

To return this collection, use the Columns property of a data source object.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method (Objects)</a>	Adds a Column object to the collection.
<a href="#">Item Method</a>	Selects a specific Column object in the collection.
<a href="#">Remove Method</a>	Removes a specific Column object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Column objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
objColumn = objDataSource.Columns(1)
```

## CubeCustomViews Collection

The CubeCustomViews collection contains CustomView objects that are associated to a specific Cube, CubeGroup, or ChildCube object.

### Discussion

Use this collection to add, select or remove a CustomView object. Each Cube or ChildCube object defines a CubeCustomView collection. To return this collection, use the CubeCustomViews property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method (CubeCustomViews Collection)</a>	Adds a CustomView object to the collection.
<a href="#">Item Method</a>	Selects a specific CustomView object in the collection.
<a href="#">Remove Method</a>	Removes a specific CustomView object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of CustomView objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

### Examples

```
cube = model.Cubes.Item("Sales and Marketing")
cube.CubeCustomViews.Add(custom_view)
```

## Cubes Collection

The Cubes collection groups all Cube and CubeGroup objects in a model.

### Discussion

Use this collection to add, select or remove a Cube or CubeGroup object. The Cube object contains related Reports and CubeCustomViews collections. The CubeGroup object contains related ChildCubes, DrillThroughTargets, and CubeCustomViews collections.

To return this collection, use the Cubes property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method (Objects)</a>	Adds a Cube or CubeGroup object to the collection.
<a href="#">Item Method</a>	Selects a specific Cube or CubeGroup object in the collection.

Method	Description
<a href="#">Remove Method</a>	Removes a specific Cube or CubeGroup object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Cube and CubeGroup objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## CurrencyRates Collection

The Currency Rates collection groups CurrencyRate objects.

### Discussion

Use this collection to select a CurrencyRate object. Each object represents a conversion rate used to convert currencies.

A CurrencyRates, CurrencyRecords, and CurrencyTables collection are all required to convert currency. To return this collection, use the CurrencyRates property.

These tables list related methods and properties.

Method	Description
<a href="#">Item Method</a>	Selects a specific CurrencyRate object in the collection.
<a href="#">Remove Method</a>	Removes a specific CurrencyRate object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of CurrencyRate objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

### Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()
objCurrencyRate = objCurrencyRecord.CurrencyRates(intX)
```

## CurrencyRecords Collection

The CurrencyRecords collection contains all the CurrencyRecord objects in a model.

## Discussion

Use this collection to add, select or remove a CurrencyRecord object. Each object in the collection represents one row of currency information that is used to format and show currencies in PowerPlay.

A CurrencyRates, CurrencyRecords, and CurrencyTables collection are all required to convert currency.

To return this collection, use the CurrencyRecords property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a CurrencyRecord object to the collection.
<a href="#">Item Method</a>	Selects a specific CurrencyRecord object in the collection.
<a href="#">Remove Method</a>	Removes a specific CurrencyRecord object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of CurrencyRecord objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()
```

## CurrencyTables Collection

The CurrencyTables collection contains all CurrencyTable objects in a model.

## Discussion

Use this collection to add, select or remove a CurrencyTable object.

You can use external data sources to populate a currency table, or you can create and insert the currency records and rates. Both methods require data that sets the

- conversion date
- ISO country or region code
- conversion rate
- currency label (optional)

Transformer uses the currency table information to make the correct conversions when a user views cubes in PowerPlay. Currency conversion requires that you have at least one time dimension in the model. Transformer stores the currency information at the same level of detail as appears in the time dimension (by default, monthly). You can adjust it to a higher level if required.

For conversions to or from EMU currencies, you need two tables. One table must have the CurrencyTableType property set to trCurrencyTableBase, and the other must be set to trCurrencyTableEuro. A CurrencyTables collection can only contain one of each type of table.

CurrencyRates, CurrencyRecords, and CurrencyTables collections are all required to convert currency. To return this collection, use the CurrencyTables property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a CurrencyTable object to the collection.
<a href="#">Item Method</a>	Selects a specific CurrencyTable object in the collection.
<a href="#">Remove Method</a>	Removes a specific CurrencyTable object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of CurrencyTable objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objBaseTable = objModel.CurrencyTables.Add()
```

## CustomViews Collection

The CustomViews collection represents a collection of CustomView objects in a model.

### Discussion

Use this collection to add, select or remove a CustomView object. The CustomView object contains related SecurityObjects and Views collections.

To return this collection, use the CustomViews or ChildCustomViews property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a CustomView object to the collection.
<a href="#">Item Method</a>	Selects a specific CustomView object in the collection.
<a href="#">Remove Method</a>	Removes a specific CustomView object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.



Property	Description
<a href="#">Count Property</a>	Returns the number of CustomView objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
custom_view = model.CustomViews.Add()
custom_view = model.CustomViews.Add()
```

## DataSources Collection

The DataSources collection contains all data sources in a model.

### Discussion

Use this collection to add, select or remove a data source. This collection can include the following objects:

- CrossTabDataSource
- DataSource
- DbDataSource.
- FlatFileDataSource
- IqdDataSource

To return this collection, use the DataSources property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds an object to a collection that contains objects of more than one type.
<a href="#">Item Method</a>	Selects a specific data source in the collection.
<a href="#">Remove Method</a>	Removes a specific data source from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of DataSource objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
```

## DimensionLevels Collection

The DimensionLevels collection contains a read-only list of unique levels in the related dimension.

### Discussion

Use this collection to select a Level object from a dimension.

In cases where a dimension has alternate drill-down paths, an individual level may be included more than once in the dimension. However, this collection contains just one reference to that level, no matter how often it appears in the dimension.

To return this collection, use the DimensionLevels property.

These tables list related methods and properties.

Method	Description
<a href="#">Item Method</a>	Selects a specific object in the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

### Examples

```
objLevel = objModel.Dimensions("Products").DimensionLevels(1)
```

## Dimensions Collection

The Dimensions collection contains all Dimension and DateDimension objects in a model.

### Discussion

Use this collection to add, select or remove a Dimension or DateDimension object.

When you use the DoAutoDesign method to create dimensions, Transformer automatically adds dimensions to this collection based on relationships in the data sources. The DateWizard object adds DateDimension objects to this collection.

To return this collection, use the Dimensions property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a Dimension or DateDimension object to the collection.
<a href="#">Item Method</a>	Selects a specific Dimension or DateDimension object in the collection.

Method	Description
<a href="#">Remove Method</a>	Removes a specific Dimension or DateDimension object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Dimension and DateDimension objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objDimensions = objModel.Dimensions
```

## DrillDowns Collection

The DrillDowns collection contains either DrillDown objects or DateDrillDown objects, but not both.

### Discussion

Use this collection to select or remove a DrillDown or DateDrillDown object.

Transformer automatically adds one DrillDown or DateDrillDown object to each DrillDowns collection when the parent dimension is created. This object represents the primary drill-down path and you cannot remove it. Any other DrillDown or DateDrillDown object in the collection represents an alternate drill-down path.

Use the CreateAlternateDrillDown method of the Level object to add DrillDown or DateDrillDown objects to the collection.

To return this collection, use the DrillDowns property.

Method	Description
<a href="#">Item Method</a>	Selects a specific DrillDown or DateDrillDown object in the collection.
<a href="#">Remove Method</a>	Removes a specific DrillDown or DateDrillDown object from the collection. Not applicable to the first object in the collection. If applied to the first object, an exception is thrown.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of DrillDown or DateDrillDown objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objDrill = objModel.Dimensions("Retailers").DrillDowns(2)
```

## DrillThroughTargets Collection

The DrillThroughTargets collection groups related DrillThroughTarget objects.

### Discussion

Use this collection to add, select or remove a DrillThroughTarget object.

In previous releases, the documentation stated that the DrillThroughTarget object represents a drill-through object, such as an Impromptu report (.imr), Impromptu query definition file (.iqd), PowerPlay report (.ppr), PowerCube (.mdc), or macro script file (.mac). However, for certain operations, such as retrieving counts or other information about drill-through target objects, you must use MDL scripting.

To return the DrillThroughTargets collection for a Measure, Cube, or CubeGroup object, use the DrillThroughTargets property. When you add a DrillThroughTarget object to the DrillThroughTargets collection of a Measure, you restrict the drill-through functionality to that measure alone. When you add a DrillThroughTarget object to the DrillThroughTargets collection of a Cube or CubeGroup object, the drill-through functionality is available at any point in the cube or child cubes.

For models that use Impromptu query definition files (.iqd) as the data source, Transformer automatically adds the corresponding Impromptu report (.imr) to the DrillThroughTargets collection for each measure.

You must set the AllowDrillThrough property to True before drill-through capability is available.

Method	Description
<a href="#">Add Method (DrillThroughTargets)</a>	Adds a DrillThroughTarget object to the collection.
<a href="#">Item Method</a>	Selects a specific DrillThroughTarget object in the collection.
<a href="#">Remove Method</a>	Removes a specific DrillThroughTarget object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Report objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

### Examples

```
objReport = objMeasure.DrillThroughTargets.Add(strReportPath,  
"Default Report")
```

## Filters Collection

The Filters collection contains all Filter objects in a query.

## Discussion

Use this collection to select or remove a Filter object.

This collection is an IBM Cognos object. Each Query and Report object in a model may contain a Filters collection. A filter represents a condition that must be met before data can be retrieved from a data source. To return this collection, use the Filters property.

Use the Add method to add a new Filter object to the collection. Use the Remove method to remove a Filter object.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a Filter object to the collection.
<a href="#">Item Method</a>	Selects a specific Filter object in the collection.
<a href="#">Remove Method</a>	Removes a specific Filter object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Filter objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
new_package = model.Packages.Add()  
new_query = new_package.Queries.Add()  
new_filter = new_query.Filters.Add()
```

## LevelCategories Collection

The Levels Categories collection represents the collection of categories for a specific level.

## Discussion

Use this collection to select the categories associated with each level.

These tables list related methods and properties.

Method	Description
<a href="#">Item Method</a>	Selects a specific Category object in the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.

Property	Description
<a href="#">Count Property</a>	Returns the number of Category objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## LevelDrillDowns Collection

The LevelDrillDowns collection represents a read-only collection of objects representing drill-down paths.

### Discussion

Use this collection to select a DrillDown or DateDrillDown object in a dimension.

Most levels have a single drill-down path and therefore just one DrillDown or DateDrillDown object in this collection. In the case of a level that is the convergence of two or more drill-down paths, the collection has an equivalent number of entries.

A LevelDrillDowns collection can contain either DrillDown objects or DateDrillDown objects, but not both.

To return this collection, use the LevelDrillDowns property.

These tables list related methods and properties.

Method	Description
<a href="#">Item Method</a>	Selects a specific DrillDown or DateDrillDown object in the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of DrillDown or DateDrillDown objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

### Examples

```
objLevel = objDateDrillDown.Levels("Year")
Select Case objLevel.LevelDrillDowns(1).Categories(intX).KeyName
```

## Levels Collection

The Levels collection groups Level objects or DateLevel objects, but not both.

### Discussion

Use this collection to add, select or remove a Level or DateLevel object.

Each Dimension and DateDimension object in a model contains a Levels collection that represents steps in the drill-down path used to find information in PowerPlay. Transformer automatically creates Levels collections when you use the DoAutoDesign method.

To return this collection, use the Levels property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a Level or DateLevel object to the collection.
<a href="#">Item Method</a>	Selects a specific Level or DateLevel object in the collection.
<a href="#">Remove Method</a>	Removes a specific Level or DateLevel object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Level or DateLevel objects in the collection.

## Examples

```
objTimeDimension = objModel.Dimensions.Item("Time")objLevel  
= objTimeDimension.DrillDowns.Item(1).Levels.Item("Month")
```

## Measures Collection

The Measures collection contains all Measure objects in a model.

### Discussion

Use this collection to add, select or remove a Measure object.

Each Measure object in the collection represents numeric values that act as performance indicators in PowerPlay. Measures are always quantifiable. For example, valid measures include Revenue, Revenue per Employee, and Profit Margin. Transformer automatically creates the collection when you use the DoAutoDesign method.

A Measures collection can contain regular measures, calculated measures, or category count measures. To return this collection, use the Measures property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a Measure object to the collection.
<a href="#">Item Method</a>	Selects a specific Measure object in the collection.
<a href="#">Remove Method</a>	Removes a specific Measure object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.

Property	Description
<a href="#">Count Property</a>	Returns the number of Measure objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objMeasures = objModel.Measures
```

## Names Collection

The Names collection contains a read-only group of Name objects.

### Discussion

Use this collection to reference a group of strings representing names or messages. A Names collection is returned by the CheckModel method (Model object).

Each Name object in this collection contains a complete message string. You can use the return value of the Name property to read the results of a CheckModel procedure.

These tables list related methods and properties.

Method	Description
<a href="#">Item Method</a>	Selects a Name object in the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Name objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
MsgBox(objModel.CheckModel(intX).Name)
```

## Namespaces Collection

The Namespaces collection contains all Namespace objects in a model.

### Discussion

Use this collection to select or remove a Namespace object.

This IBM Cognos object represents an instance of an authentication provider which allows authentication and access control. To return this collection, use the Namespaces property. The collection count is 0 if the model does not use Namespaces. To use this object, you must use an IBM Cognos server. It allows access to user and group information which are members of the SecurityObjects object.



Use the Add method to add a new Namespace object to the collection. Use the Remove method to remove a Namespace object.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a Namespace object to the collection.
<a href="#">Item Method</a>	Selects a specific Namespace object in the collection.
<a href="#">Remove Method</a>	Removes a specific Namespace object from the collection

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Namespace objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
new_namespace = model.Namespaces.Add()
```

## PackageDatasourceConnections Collection

The PackageDatasourceConnections collection contains all PackageDatasourceConnection objects.

### Discussion

Use this collection to add, select, or remove a PackageDatasourceConnection object.

This IBM Cognos object requires that the IBM Cognos server is installed and running. PackageDatasourceConnection objects are created using IBM Cognos Connection. The collection count is 0 if the model does not use Packages.

To return this collection, use the PackageDatasourceConnections property.

Use the Add method to add a new PackageDatasourceConnection object to the collection. Use the Remove method to remove a PackageDatasourceConnection object.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a PackageDatasourceConnection object to the collection.
<a href="#">Item Method</a>	Selects a specific PackageDatasourceConnection object in the collection.
<a href="#">Remove Method</a>	Removes a specific PackageDatasourceConnection object from the collection

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of PackageDatasourceConnection objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
package = model.Packages.Add()
connection = package.PackageDatasourceConnections.Add()
```

## Packages Collection

The Packages collection contains all Package objects in a model.

### Discussion

Use this collection to select or remove a Package object.

This IBM Cognos requires that the IBM Cognos server is installed and running. By definition, a package can be a subset of a Transformer model that is made available to the IBM Cognos server. The collection count is 0 if the model does not use Packages.

To return this collection, use the Packages property.

Use the Add method to add a new Package object to the collection. Use the Remove method to remove a Package object.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a Package object to the collection.
<a href="#">Item Method</a>	Selects a specific Package object in the collection.
<a href="#">Remove Method</a>	Removes a specific Package object from the collection

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Package objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
packages = model.Packages
```

## Prompts Collection

---

The Prompts collection contains all Prompt objects in a query.

### Discussion

Use this collection to select or remove a Prompt object.

This IBM Cognos requires that the IBM Cognos server is installed and running. Prompts are associated with queries. To return this collection, use the Prompts property of a Query object.

Use the Add method to add a new Prompt object to the collection. Use the Remove method to remove a Prompt object.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a Prompt object to the collection.
<a href="#">Item Method</a>	Selects a specific Prompt object in the collection.
<a href="#">Remove Method</a>	Removes a specific Prompt object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Prompt objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

### Examples

```
new_report = model.Reports.Add()  
new_query = new_report.Queries.Add()  
new_prompt = new_query.Prompts.Add()
```

## Queries Collection

---

The Queries collection contains all Query objects in a package or report.

### Discussion

Use the Queries collection to select or remove a Query object.

This IBM Cognos requires that the IBM Cognos server is installed and running. A Query object represents a specification for a set of data retrieved from a data source. Transformer models may contain multiple queries as data sources.

To return this collection, use the Queries property.

Use the Add method to add a new Query object to the collection. Use the Remove method to remove a Query object.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a Query object to the collection.
<a href="#">Item Method</a>	Selects a specific Query object in the collection.
<a href="#">Remove Method</a>	Removes a specific Query object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Query objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
new_package = model.Packages.Add()
new_query = new_package.Queries.Add()
```

## Reports Collection

The Reports collection groups related Report objects.

### Discussion

Use this collection to add, select or remove a Report object.

A report is an IBM Cognos report that serves as a container data source. Multiple data source queries can be associated with a report. Reports are created by an IBM Cognos application, such as Query Studio and Reporting, using relational or DMR packages. Because reports are saved in IBM Cognos Connection, an IBM Cognos server is required to establish a data source connection. A report can contain one or more queries. The collection count is 0 if the model does not use reports.

To return this collection, use the Reports property.

Use the Add method to add a new Report object to the collection. Use the Remove method to remove a Report object.

Method	Description
<a href="#">Add Method ()</a>	Adds a Report object to the collection.
<a href="#">Item Method</a>	Selects a specific Report object in the collection.
<a href="#">Remove Method</a>	Removes a specific Report object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.

Property	Description
<a href="#">Count Property</a>	Returns the number of Report objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
reports = model.Reports
```

## SecurityObjects Collection

The SecurityObjects collection contains all SecurityObject objects in a model.

### Discussion

Use this collection to select or remove a SecurityObject object.

Use the SecurityObjects property of a CustomView or a Namespace object to retrieve this collection.

Use the Add method to add a new SecurityObject object to the collection. Use the Remove method to remove a SecurityObject object.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a Report object to the collection.
<a href="#">Item Method</a>	Selects a specific Report object in the collection.
<a href="#">Remove Method</a>	Removes a specific Report object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Report objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
new_namespace = model.Namespaces.Add()
securityObject = new_namespace.SecurityObjects.Add()
```

## Signons Collection

The Signons collection groups related Signon objects.

## Discussion

Use this collection to add, modify, or remove a Signon object.

Each object in the collection regulates access to data in an IqdDataSource object. To return this collection, use the Signons property.

When you use an Impromptu query definition file (.iqd) as a data source, Transformer uses information within the file to automatically create a new Signon object within the model. The new Signon object is assigned the logical database name that exists in the Impromptu query definition file (.iqd).

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a Signon object to the collection.
<a href="#">Item Method</a>	Selects a specific Signon object in the collection.
<a href="#">Remove Method</a>	Removes a specific Signon object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of Signon objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
signon = model.Signons.Add()
```

## SuspendedModels Collection

The SuspendedModels collection contains all SuspendedModel objects available to the application.

## Discussion

Use this collection to select or remove a SuspendedModel object.

If a model closes abruptly, as during a system outage, Transformer adds a SuspendedModel object for that model to this collection.

To return this collection, use the SuspendedModels property.

These tables list related methods and properties.

Method	Description
<a href="#">Item Method</a>	Selects a specific SuspendedModel object in the collection.
<a href="#">RemoveSuspendedModel Method</a>	Removes a specific SuspendedModel from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of SuspendedModel objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
If objTransApp.SuspendedModels.Count = 0 Then
```

## Views Collection

The Views collection groups View objects.

### Discussion

Use this collection to add, select or remove a View object.

In many organizations, you do not need to make all model data available to all users. One way to limit data is to create a view, which is a subset of the levels and categories in a dimension. You can group several views related to a dimension in a Views collection.

To return this collection, use the Views property.

These tables list related methods and properties.

Method	Description
<a href="#">Add Method ()</a>	Adds a View object to the collection.
<a href="#">Item Method</a>	Selects a specific View object in the collection.
<a href="#">Remove Method</a>	Removes a specific View object from the collection.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Count Property</a>	Returns the number of View objects in the collection.
<a href="#">Parent Property</a>	Returns the parent object.

## Examples

```
objDimension = objModel.Dimensions("Retailers")
objView = objDimension.Views.Add()
```





---

## Chapter 3. Objects

The following table lists all the Transformer OLE automation objects.

Object	Description
<a href="#">Application Object</a>	Represents an instance of the Transformer application.
<a href="#">Association Object</a>	Defines the relationship between a model object and the underlying data source.
<a href="#">CalculationDefinition Object</a>	Acts as a template to define calculated categories.
<a href="#">Category Object</a>	Defines a category used in a model.
<a href="#">CategorySet Object</a>	Groups categories used in a calculation definition.
<a href="#">ChildCube Object</a>	Defines a cube as a member of a cube group.
<a href="#">Column Object</a>	Represents a single data item in a data source.
<a href="#">CrossTabDataSource Object</a>	Represents an external file in crosstab format.
<a href="#">Cube Object</a>	Defines a cube.
<a href="#">CubeGroup Object</a>	Defines a set of related child cubes.
<a href="#">CurrencyRate Object</a>	Defines an exchange rate for currency conversion.
<a href="#">CurrencyRecord Object</a>	Defines how currency information appears.
<a href="#">CurrencyTable Object</a>	Defines a currency table used in currency conversion.
<a href="#">CustomView Object</a>	Represents a custom view.
<a href="#">DataSource Object</a>	Defines a generic data source.
<a href="#">DateDimension Object</a>	Organizes date data into logical groups.
<a href="#">DateDrillDown Object</a>	Defines primary and alternate drill-down paths for date or time values.
<a href="#">DateLevel Object</a>	Defines a date level in a model.
<a href="#">DateWizard Object</a>	Creates a DateDimension object.
<a href="#">DbDataSource Object</a>	Represents an external file in database format.
<a href="#">Dimension Object</a>	Organizes non-date data into logical groups.

<b>Object</b>	<b>Description</b>
<a href="#">DrillDown Object</a>	Defines primary and alternate drill-down paths used to navigate cubes.
<a href="#">DrillThroughTarget Object</a>	Represents a drill-through link to an external report or PowerCube.
<a href="#">Filter Object</a>	Represents a filter in the Transformer model.
<a href="#">FlatFileDataSource Object</a>	Represents an external file in ASCII format.
<a href="#">IqdDataSource Object</a>	Represents an Impromptu query definition file (.iqd).
<a href="#">Level Object</a>	Defines a level in a dimension.
<a href="#">Measure Object</a>	Represents quantitative values in a cube.
<a href="#">Model Object</a>	Defines a Transformer model.
<a href="#">Name Object</a>	References a text string representing a name or message.
<a href="#">Namespace Object</a>	Represents a namespace in the Transformer model.
<a href="#">Package Object</a>	Represents a package data source in the Transformer model.
<a href="#">PackageDatasourceConnection Object</a>	Represents a package data source connection in the Transformer model.
<a href="#">Prompt Object</a>	Represents a prompt in the Transformer model.
<a href="#">Query Object</a>	Represents a query that is part of a report or package.
<a href="#">Report Object</a>	Represents a report data source in the Transformer model.
<a href="#">SecurityObject Object</a>	Represents a security object in the Transformer model.
<a href="#">Signon Object</a>	Defines a database signon.
<a href="#">SpecialCategory Object</a>	Defines a category with a value that is not directly from a data source.
<a href="#">SuspendedModel Object</a>	Represents an existing incomplete model.
<a href="#">View Object</a>	Defines a partial view of a dimension.

## Application Object

The Application object represents an instance of the Transformer application.

### Discussion

Use this object to create or modify a model.

Most of the properties of the Application object set runtime preferences. After you set the preference properties, they remain in effect until you change them; that is, you do not have to set them each time you create the Application object.

To create this object, use the CreateObject function.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">NewModel Method</a>	Creates an empty model.
<a href="#">OpenModel Method</a>	Opens an existing model.
<a href="#">OpenSuspendedModel Method</a>	Opens a suspended model.
<a href="#">RemoveSuspendedModel Method</a>	Removes a suspended model from the collection.

Property	Description
<a href="#">PatFile Property</a>	Sets or returns the location of the pattern file <i>cogtr_locale.pat</i> , such as <i>cogtr_en.pat</i> , for the associated product locale when auto-design is used.
<a href="#">CurrentModel Property</a>	Returns the currently open Model object.
<a href="#">DataCharacterSet Property</a>	Sets or returns the default character set used by the application.
<a href="#">DataSourcePath Property</a>	Sets or returns the location where Transformer searches for data source files.
<a href="#">DataTemporaryFilesPath Property</a>	Sets or returns the name of the directory where Transformer creates temporary work files while it generates cubes.
<a href="#">DefaultDateFormat Property</a>	Sets or returns the default setting for the <i>DateInputFormat</i> property.
<a href="#">DetachDataSource Property</a>	Sets or returns whether the connection to the data source is maintained or released.
<a href="#">EnableMessageLogging Property</a>	Sets or returns whether Transformer messages are written to a log file.

Property	Description
<a href="#">LogErrorLevel Property</a>	Sets or returns the level of severity of error messages logged.
<a href="#">LogFileAppend Property</a>	Sets or returns whether Transformer appends messages to the log file or overwrites previous log messages.
<a href="#">LogFileName Property</a>	Sets or returns the name for the log file.
<a href="#">LogFilesPath Property</a>	Sets or returns the location where Transformer saves the log file.
<a href="#">MaxTransactionNumber Property</a>	Sets or returns the maximum number of records that Transformer processes before committing the changes to a cube.
<a href="#">ModelsPath Property</a>	Sets or returns the location where Transformer opens and saves model files.
<a href="#">ModelTemporaryFilesPath Property</a>	Sets or returns the location where Transformer creates temporary model files (.qy?).
<a href="#">Name Property</a>	Returns the name of the application.
<a href="#">PatFile Property</a>	Sets or returns the location of the pattern file cogtr_locale.pat, such as cogtr_en.pat, for the associated product locale when auto-design is used.
<a href="#">PowerCubesPath Property</a>	Sets or returns the location where Transformer creates PowerCube files (.mdc).
<a href="#">PowerPlayPath Property</a>	Sets or returns where the PowerPlay.exe is located.
<a href="#">RowsAsSample Property</a>	Sets or returns the number of rows that the DoAutoDesign method samples when creating a model.
<a href="#">RowsChecked Property</a>	Sets or returns the maximum number of rows that the DoAutoDesign method reads from the data source.
<a href="#">ServicesBuildNumber Property</a>	Returns the build number of the data access service used by Transformer.
<a href="#">ServicesVersionText Property</a>	Returns the version number of the data access service used by Transformer.
<a href="#">SortComparisonRule Property</a>	Sets or returns which comparison rule Transformer uses when sorting data.
<a href="#">SuspendedModels Property</a>	Returns a collection of suspended model objects.

Property	Description
<a href="#">TransdaPath Property</a>	Sets or returns where the Transda.exe is installed.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">Version Property</a>	Returns the version number of Transformer.

## Examples

```
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
```

## Association Object

The Association object defines the relationship between a model object and the underlying data source.

### Discussion

Use this object to manage the link between objects in a model and their underlying data sources.

Each Association object in a collection has a role. For example, an Association references source values, or displays short names, or defines a sort order. To specify a role for an Association, use the AssociationRole property.

Each Association object references a data source column. How an Association object uses a column depends on the role assigned to the Association. To specify a column, set the Label property to the name of the reference object.

To create an Association object, use the Add method of the Associations collection, or the AssociateWith or DimensionAssociateWith methods of an applicable object.

By default, categories appear in a level, and ultimately in a cube, in the order that they are encountered in the data source. To sort these objects, use an Association object and set the AssociationRole property to trAssociationOrderBy.

When Transformer sorts a level, it bases the sort on the column specified in the Label property. For example, you can sort a Product level by product\_name or product\_number.

Use the Context property to name the drill-down path to which the sort applies. Where a level represents the convergence of two or more drill-down paths, you can apply a different sort order to each one.

Use the OrderByDescending property to specify how the sort is ordered. If you use a column of numeric data to define the sort, use the OrderByStorageType to specify the size of the data type.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the Association object.
<a href="#">Update Method</a>	Updates the Association object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">AssociationRole Property</a>	Sets or returns the role performed by the Association object.
<a href="#">AssociationType Property</a>	Sets or returns the type of data source related to an Association object.
<a href="#">Context Property</a>	Sets or returns the drill-down path used to order categories within a level.
<a href="#">Label Property</a>	Sets or returns the name of the reference object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">RefreshDescription Property</a>	Sets a data source column for the Association object.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
objAssociation = objDrill.ConvergenceLevel.Associations.Add()
```

## CalculationDefinition Object

The CalculationDefinition object acts as a template to define calculated categories.

### Discussion

Use CalculationDefinition objects to add complex expressions that use the values of one or more categories as part of the calculation.

To create a CalculationDefinition object, use the Add method of the CalculationDefinitions collection.

The value for a CalculationDefinition object is based on a formula and the categories or category sets to which the formula applies. You compose the formula as an expression by using functions and operators and assign it to the ExpressionText property. The expression can reference more than one category set.

The categories described by the expression are generated when you execute the Update method. The expression is also checked for validity. If there are errors, such as duplicate categories, the update fails and Transformer issues error messages.

The Description property, although available as a property, does not do anything. Any change you make to the Description property does not persist.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the CalculationDefinition object.

Method	Description
<a href="#">Update Method</a>	Updates the CalculationDefinition object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">CategorySets Property</a>	Returns the CategorySets collection used by the CalculationDefinition object.
<a href="#">ExpressionText Property</a>	Sets or returns the contents of an expression that defines a value for the CalculationDefinition object.
<a href="#">Group Property</a>	Sets or returns whether the CalculationDefinition is grouped with the categories in the category set.
<a href="#">IsExpressionValid Property</a>	Returns whether an expression is valid.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
calcDef = dimension.CalculationDefinitions.Add()
```

## Category Object

The Category object defines a category used in a model.

### Discussion

Use this object to change the properties of a Category object in a model. Categories are generated from the data source when you use the GenerateCategories method. Categories are also generated when you create a cube.

A regular category contains structural data that describes or classifies details of an organization, for example a product line or vendor site. Categories usually relate to levels within a dimension. When you create a model, categories are populated with measure values from transactional data sources.

A calculated category uses expressions and category values to derive its own value. To create a calculated category, use the Add method of the Categories collection, and then assign it a value by using the ExpressionText property. You can also use CalculationDefinition objects and CategorySet objects to create calculated categories.

To return a top-level collection of regular categories in a dimension, use the Categories property of the DrillDown object. To return a collection of child categories, use the ChildCategories property of the Category object. To return a collection of special categories, use the Categories property of the parent dimension.

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">ConnectWithCategory Method</a>	Moves a child category to a new parent category in the same level.
<a href="#">Delete Method</a>	Deletes the Category object.
<a href="#">MoveToCategory Method</a>	Moves a child category to a different parent category.
<a href="#">MoveToLevel Method</a>	Moves a child category to a new level under the current parent category.
<a href="#">SetAllocation Method</a>	Changes the allocation type for the measure used by the object.
<a href="#">Update Method</a>	Updates the Category object.

Property	Description
<a href="#">AllocationMeasure Property</a>	Returns the Measure object used as a weighting factor.
<a href="#">AllocationType Property</a>	Returns how an object allocates a measure.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">CanAllocate Property</a>	Returns whether you can allocate any measure values to descendant levels and categories.
<a href="#">CanAllocateByMeasure Property</a>	Returns whether you can use the specified measure as a weighting factor when you allocate by measure.
<a href="#">CanAllocateMeasure Property</a>	Returns whether you can allocate the specified measure to descendant levels and categories.
<a href="#">ChildCategories Property</a>	Returns a Categories collection.
<a href="#">Code Property</a>	Sets or returns a unique identifying code for a category within a dimension.
<a href="#">Description Property</a>	Sets or returns the description of the Category object.
<a href="#">Dimension Property</a>	Returns the dimension for a Category object.
<a href="#">ExpressionText Property</a>	Sets or returns the contents of an expression that defines a value for the Category object.
<a href="#">Format Property</a>	Sets or returns how numeric values appear.



Property	Description
<a href="#">FormatDecimals Property</a>	Sets or returns the number of decimal places PowerPlay displays for the measure.
<a href="#">HideValue Property</a>	Specifies whether to hide the value of a category object. Default: false.
<a href="#">Inclusion Property</a>	Sets or returns when a category is included in a cube.
<a href="#">IsExpressionValid Property</a>	Returns whether an expression is valid.
<a href="#">IsPrimary Property</a>	Returns whether the Category object is the primary or alternate drill category.
<a href="#">KeyName Property</a>	Sets or returns the value that appears in the associated column in the data source.
<a href="#">Label Property</a>	Sets or returns the name of the object. In the Transformer user interface, the label is "Source Value", however, the default value is "".
<a href="#">LastUseDate Property</a>	Returns the date the Category object was last modified or used.
<a href="#">Level Property</a>	Returns the level for a Category object.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Orphanage Property</a>	Sets or returns whether a category is an orphanage.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">ParentCategories Property</a>	Returns a ParentCategories collection.
<a href="#">Partition Property</a>	Sets or returns a manual partition number.
<a href="#">ReverseSign Property</a>	Sets or returns whether PowerPlay reverses the sign of a measure.
<a href="#">ShortName Property</a>	Sets or returns a short name for the Category object.
<a href="#">Type Property</a>	Returns the object type.

### Examples

```
parentCategory = drillDown.Categories.Item(1)'select
"Camping Equipment"
```

## CategorySet Object

The CategorySet object groups categories used in a calculation definition.

## Discussion

In a CalculationDefinition object, some expressions can take a CategorySet object as a parameter. A category set contains one or more categories from an applicable dimension that provide values for the expression.

You must create the category set before you reference it in an expression.

A category set is a convenient way to group categories when you do not want to include the entire level in your formula. A category set can also contain categories from different levels in a dimension.

First create a CalculationDefinition object and then use the Add method to add a CategorySet to the CategorySets collection. Then, use the Add method to add categories to the Categories collection of the Category set. Finally, use the Label property to name the category set.

Use the Label property of the CategorySet object as a reference in the expression. Ensure that you enclose each category code string and label in an extra set of quotation marks.

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the CategorySet object.
<a href="#">Update Method</a>	Updates the CategorySet object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Categories Property</a>	Returns the collection of categories that are included in the CategorySet object.
<a href="#">Description Property</a>	Sets or returns the description of the CategorySet object.
<a href="#">Label Property</a>	Sets or returns a descriptive name that appears in PowerPlay.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
catSet = calcDef.CategorySets.Add()
```

## ChildCube Object

The ChildCube object defines a cube as a member of a cube group.

## Discussion

A cube group represents one level in a dimension. Each ChildCube object in the group reflects a single category in that level.

Use this object to change the properties of a ChildCube object. To return a ChildCube object from a collection, use the ChildCubes property of the CubeGroup object.

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">RemoveCubeCustomView Method</a>	Removes the custom view.
<a href="#">Update Method</a>	Updates the ChildCube object.

Property	Description
<a href="#">AllowDrillThrough Property</a>	Sets or returns whether a cube or measure can drill through to a cube or report.
<a href="#">AltMDCFile Property</a>	Sets or returns an alternate file name for the child cube.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Consolidate Property</a>	Sets or returns how a cube is consolidated.
<a href="#">CubeCreation Property</a>	Sets or returns whether the cube is created.
<a href="#">CubeCustomViews Property</a>	Returns a collection of CustomView objects.
<a href="#">Description Property</a>	Sets or returns the description of the ChildCube object.
<a href="#">DrillThroughTargets Property</a>	Returns a collection of all DrillThroughTargets objects associated with a ChildCube object.
<a href="#">IsMDCInUse Property</a>	Returns whether a child cube is in use or being rebuilt.
<a href="#">MDCFile Property</a>	Sets or returns the name of a PowerCube file (.mdc).
<a href="#">MeasureName Property</a>	Sets or returns a descriptive title that identifies a measure on the PowerPlay dimension line.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Optimize Property</a>	Sets or returns the option for the current cube optimization.

Property	Description
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Password Property</a>	Sets a case-sensitive password for the child cube.
<a href="#">Status Property</a>	Returns a problem status associated with the cube the last time it was created.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">UseAltMDCFile Property</a>	Sets or returns whether a temporary file name may be used.

### Examples

```
objCubesByRegion = objModel.Cubes.Add(xtrObjectType.trCubeGroup)objChildCube =
objCubesByRegion.ChildCubes("Central Europe")
```

## Column Object

The Column object represents a single data item in a data source.

### Discussion

Column objects in the model are linked to columns in their associated data source and act as the source of values for levels and measures after categories are generated.

Transformer uses columns to build dimensions and measures in the model. A column contains either text values that become categories in the model, or numeric values that become measure values. A column can also contain values that are used as labels, short names, or textual descriptions for categories.

For most of the supported data source types, Transformer automatically identifies columns in the data sources and assigns default column names and properties. You can change the assignments later, if you like. If you create columns manually, as when using a fixed-field text file, you need to identify only those data columns that are used in the model.

When the same column name appears in two or more data sources, Transformer associates these columns with a single level or measure. Transformer does not use matching columns to perform joins on the source files, but it does use each source file to populate the model with categories.

To include a time dimension in your model, at least one data source in the model must include a column that represents date values.

If you need more structural or numeric information in your model than is available from your data source, you can define a calculated column by using the ExpressionText property. A calculated column uses other columns, functions, and constants to derive new data for the model.

To create a Column object, use the Add method of the Columns collection.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the Column object.

Method	Description
<a href="#">Update Method</a>	Updates the Column object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">DataClass Property</a>	Sets or returns the data type of a source column.
<a href="#">DateDegreeofDetail Property</a>	Sets or returns the date level at which reporting will occur.
<a href="#">DateDegreeofDetailLevelName Property</a>	Sets or returns the date level that applies to an externally rolled up measure.
<a href="#">DateInputFormat Property</a>	Sets or returns how dates are displayed.
<a href="#">Decimals Property</a>	Returns the number of decimal places, if defined in the source data.
<a href="#">Description Property</a>	Sets or returns the description of the Column object.
<a href="#">ExpressionText Property</a>	Sets or returns the contents of an expression that defines a value for a calculated column.
<a href="#">InputScale Property</a>	Sets or a returns a scale value used to convert column numbers from decimal values to integer values.
<a href="#">IsExpressionValid Property</a>	Returns whether an expression is valid.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Origin Property</a>	Sets or returns the source of the Column object.
<a href="#">OriginalName Property</a>	Sets or returns the name of the column in the data source.
<a href="#">OutputScale Property</a>	Sets or a returns a scale value used to convert numbers from integer values to decimal values in PowerPlay.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Position Property</a>	Sets or returns the ordinal or starting position of the column in the data source.
<a href="#">Size Property</a>	Specifies the size of the column for some data sources.
<a href="#">StorageType Property</a>	Sets or returns the numeric storage type of the object, where applicable.

Property	Description
<a href="#">TimeArrayColumn Property</a>	Sets or returns the name of the first column in the array when the object represents a date array.
<a href="#">TimeArrayStartMonth Property</a>	Sets or returns the month in which a fiscal year begins when the object includes a date array.
<a href="#">TimeArrayType Property</a>	Sets or returns the type of array used for date values.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
objDataSource
= objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)objColumn
= objDataSource.Columns("EXPECTED_VOLUME")
```

## CrossTabDataSource Object

The Cross TabDataSource object represents an external file in crosstab format.

### Discussion

A CrossTabDataSource references a Lotus® 1-2-3 or Microsoft Excel spreadsheet. To create a crosstab data source, assign each column in the spreadsheet a range name.

To create a CrossTabDataSource object, use the Add method of the DataSources collection. Specify the parameter, trCrossTabDataSource, from the xtrObjectType value list.

Set the SourceType property to either trExcelCrossTab or trLotus123CrossTab.

When Transformer opens a crosstab data source, it creates a collection of columns in the model, with one column for each named range. Transformer determines the data class of each column (text, numeric, or date) based on a sampling of cell values in each range.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the CrossTabDataSource object.
<a href="#">Move Method</a>	Moves the CrossTabDataSource object to a different position in the DataSources collection.
<a href="#">Update Method</a>	Updates the CrossTabDataSource object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.

Property	Description
<a href="#">Columns Property</a>	Returns a Columns collection.
<a href="#">ColumnsLoaded Property</a>	Returns whether the columns in a data source have been used to build a model.
<a href="#">Description Property</a>	Sets or returns the description of the CrossTabDataSource object.
<a href="#">DecimalPoint Property</a>	Sets or returns the separator character used for a decimal point.
<a href="#">External Property</a>	Sets or returns whether the data source contains presummarized values.
<a href="#">GenerateCategories Property</a>	Sets or returns whether categories are generated for the data source.
<a href="#">GeneratePowerCube Property</a>	Sets or returns when a data source is referenced by a model.
<a href="#">IsAnyColumnMismatched Property</a>	Returns whether columns in the data source match the underlying data.
<a href="#">LocalPath Property</a>	Sets or returns the location of the local data source.
<a href="#">MaximizeSpeed Property</a>	Sets or returns whether category generation is optimized for speed.
<a href="#">Name Property</a>	Sets or returns the unique name for the data source.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">ServerQuery Property</a>	Sets or returns whether Transformer searches a data source to find the current period date.
<a href="#">SetsCurrentPeriod Property</a>	Sets or returns the type of data file a data source uses.
<a href="#">ThousandPoint Property</a>	Sets or returns the character used to separate numbers in thousands.
<a href="#">SourceType Property</a>	Returns the object type.

## Cube Object

---

The Cube object defines a cube.

## Discussion

Use this object to change the properties of a cube. Each cube defined in a model gives users a multidimensional view of the underlying data through PowerPlay. To create Cube objects, use the Add method of the Cubes collection.

Use the CreateMDCFile method to create the PowerCube file (.mdc). Use the CreateMDCFiles method of the Model object to create all .mdc files in the model.

Users need only log on once in a session to view any number of cubes and reports associated with their custom view.

To save changes to the properties of this object, use the Update method.

Users can publish either a data source or a package using the PublishDatasource or PublishPackage methods. When publishing a package, the data source is also published by default. Users have the option to choose whether to overwrite the data source or package when publishing. Users can assign custom views to each cube. Custom Views contain security information.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">AddDeployLocation Method</a>	Adds a new deployment location for Copy and Activate
<a href="#">ClearDeployLocations Method</a>	Removes all deployment locations.
<a href="#">CreateMDCFile Method</a>	Creates a PowerCube file (.mdc) based on the cube definition.
<a href="#">Delete Method</a>	Deletes the Cube object.
<a href="#">DeployCubes Method</a>	Deploys a PowerCube to all deployment locations.
<a href="#">PublishDatasource Method</a>	Publishes the Datasource connection for a PowerCube to the IBM Cognos Analytics server.
<a href="#">PublishPackage Method</a>	Publishes the Package for the PowerCube to the IBM Cognos Analytics server.
<a href="#">RemoveCubeCustomView Method</a>	Removes the custom view from the cube.
<a href="#">SetDeployType Method</a>	Sets the deployment type for Copy and Activate.
<a href="#">Update Method</a>	Updates the Cube object.

Property	Description
<a href="#">AllowDrillThrough Property</a>	Sets or returns whether a cube or measure can drill through to a cube or report.



Property	Description
<a href="#">AlternateQueryPath Property</a>	Sets or returns an alternative data source path used by the cube.  When using the AlternateQueryPath property, you must provide name of the data source and the path to the datasource.
<a href="#">AltMDCFile Property</a>	Sets or returns the alternate file name for the cube.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">BlockParentTotals Property</a>	Returns a denied value for parents of excluded children.
<a href="#">CacheCrossTabs Property</a>	Sets or returns whether summaries are cached for the initial PowerPlay crosstab in the cube.
<a href="#">CompressMDC Property</a>	Sets or returns whether the cube is compressed for compact storage.
<a href="#">Consolidate Property</a>	Sets or returns how a cube is consolidated.
<a href="#">CubeCreation Property</a>	Sets or returns whether the cube is created.
<a href="#">CubeCustomViews Property</a>	CubeCustomViews property description.
<a href="#">CubeStamp Property</a>	Returns a cube creation time-stamp.
<a href="#">DataSourceWindowsLocation Property</a>	Sets or returns the Windows location of the PowerCube on the IBM Cognos Analytics server.
<a href="#">Description Property</a>	Sets or returns the description of the Cube object.
<a href="#">DesiredPartitionSize Property</a>	Sets or returns the desired partition size when auto-partitioning is enabled.
<a href="#">DimensionView Property</a>	Sets or returns the View object associated with a cube or custom view.
<a href="#">DimensionViewType Property</a>	Sets or returns which dimensions and views belong in a cube.
<a href="#">DrillThroughTargets Property</a>	Returns a collection of DrillThroughTarget objects associated with a Cube object.
<a href="#">EstimatedRows Property</a>	Sets or returns an estimate of the number of records that the cube contains before auto-partitioning.
<a href="#">IncrementalUpdate Property</a>	Sets or returns whether the cube is incrementally updated from the data source.

Property	Description
<a href="#">IsMDCInUse Property</a>	Returns whether a child cube is in use or being rebuilt.
<a href="#">MaxNumPartLevels Property</a>	Sets or returns the maximum number of times Transformer reads the data source during cube partitioning.
<a href="#">MDCFile Property</a>	Sets or returns the name of a PowerCube file (.mdc).
<a href="#">MeasureInclude Property</a>	Sets or returns the name of a measure to include in the Cube object.
<a href="#">MeasureName Property</a>	Sets or returns a descriptive title that identifies a measure on the PowerPlay dimension line.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Optimize Property</a>	Sets or returns the current cube optimization option.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Password Property</a>	Sets a case-sensitive password for the cube.
<a href="#">Server Property</a>	Sets or returns whether a cube is processed locally or on a server.
<a href="#">Signon Property</a>	Sets or returns the Signon object used by the cube.
<a href="#">Status Property</a>	Returns a problem status associated with the cube the last time it was created.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">UseAltMDCFile Property</a>	Sets or returns whether a temporary file name may be used.

## Examples

```
objCube = objModel.Cubes.Item(1)
```

## CubeGroup Object

The CubeGroup object defines a set of related child cubes.

### Discussion

Use this object to define the contents of a cube group. A cube group represents one level in a dimension. Each ChildCube object in the group reflects a single category in that level.

To create a CubeGroup object, use the Add method of the Cubes collection, and set the Type parameter to trCubeGroup.

Use the GroupDimension and GroupLevel properties to determine what the cube group contains. The actual PowerCube files (.mdc) are created with the CreateMDCFile method.

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">CreateMDCFile Method</a>	Creates PowerCube files (.mdc) based on the cube group definition.
<a href="#">Delete Method</a>	Deletes the CubeGroup object and all the child cubes.
<a href="#">Update Method</a>	Updates the CubeGroup object.

Property	Description
<a href="#">AllowDrillThrough Property</a>	Sets or returns whether a cube or measure can drill through to a cube or report.
<a href="#">AlternateQueryPath Property</a>	Sets or returns an alternative data source used by cubes in the group.
<a href="#">AltMDCFile Property</a>	Sets or returns the alternate file name for the cube.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">CacheCrossTabs Property</a>	Sets or returns whether summaries are cached for the initial PowerPlay crosstab in a cube group.
<a href="#">ChildCubes Property</a>	Returns a collection of ChildCube objects from a cube group.
<a href="#">CompressMDC Property</a>	Sets or returns whether the cubes are compressed for compact storage.
<a href="#">Consolidate Property</a>	Sets or returns how a cube is consolidated.
<a href="#">CubeCreation Property</a>	Sets or returns whether the cube is created.
<a href="#">CubeCustomViews Property</a>	CubeCustomViews property description.
<a href="#">CubeStamp Property</a>	Returns a cube creation time-stamp.
<a href="#">Description Property</a>	Sets or returns the description of the CubeGroup object.
<a href="#">DesiredPartitionSize Property</a>	Sets or returns the desired partition size for each partition in the cubes when auto-partitioning is enabled.

Property	Description
<a href="#">DetailLevel Property</a>	Sets or returns the lowest detail level for cubes in a CubeGroup object.
<a href="#">DimensionView Property</a>	Sets or returns the View object associated with a cube or custom view.
<a href="#">DimensionViewType Property</a>	Sets or returns which dimensions and views belong in a cube group.
<a href="#">DrillThroughTargets Property</a>	Returns a collection of DrillThroughTarget objects associated with the CubeGroup object.
<a href="#">EstimatedRows Property</a>	Sets or returns an estimate of the number of records that the cube contains before auto-partitioning.
<a href="#">GroupDimension Property</a>	Sets or returns the dimension used to build the cube group.
<a href="#">GroupLevel Property</a>	Sets or returns the level whose categories become the individual cubes in the cube group.
<a href="#">IncrementalUpdate Property</a>	Sets or returns whether the cube group is incrementally updated from the data source.
<a href="#">IsMDCInUse Property</a>	Returns whether a child cube is in use or being rebuilt.
<a href="#">IsTimeBasedPartitionedCube Property</a>	Sets or returns whether a cube is specified as a time-based partitioned cube.
<a href="#">MaxNumPartLevels Property</a>	Sets or returns the maximum number of times Transformer reads the data source during cube partitioning.
<a href="#">MDCFile Property</a>	Sets or returns the name of a PowerCube file (.mdc).
<a href="#">MeasureInclude Property</a>	Sets or returns the name of a measure to include in the cube group.
<a href="#">MeasureName Property</a>	Sets or returns a descriptive title that identifies a measure on the PowerPlay dimension line.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Optimize Property</a>	Sets or returns the current cube optimization option.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Password Property</a>	Sets a case-sensitive password for the cube group.

Property	Description
<a href="#">Server Property</a>	Sets or returns whether a cube is processed locally or on a server.
<a href="#">Signon Property</a>	Sets or returns the Signon object used by the cube group.
<a href="#">Status Property</a>	Returns a problem status associated with the cube the last time it was created.
<a href="#">SummaryLevel Property</a>	Sets or returns which level to use to summarize external categories in the cube group.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">UseAltMDCFile Property</a>	Sets or returns whether a temporary file name may be used.

## Examples

```
objCubesByRegion = objModel.Cubes.Add(xtrObjectType.trCubeGroup)
```

## CurrencyRate Object

The CurrencyRate object defines an exchange rate for currency conversion.

### Discussion

CurrencyRate objects are either manually maintained or populated by a data source. Each currency rate is associated with a date category that sets the applicable period for a given rate.

CurrencyRate objects are automatically added to a CurrencyRecord object. The number of objects associated with a CurrencyRecord object corresponds to the date level specified by the DateLevel property as well as the date range defined in the time dimension. For example, with a level of detail set to month and a date range of two years, there are 24 currency rates in a currency record.

CurrencyRate, CurrencyRecord, CurrencyTable, and DateDimension objects are all needed for currency conversion to work.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the CurrencyRate object.
<a href="#">Update Method</a>	Updates the CurrencyRate object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.

Property	Description
<a href="#">Category Property</a>	Returns the date category that determines when the currency rate is valid.
<a href="#">CurrencyRecord Property</a>	Returns the CurrencyRecord object to which the CurrencyRate object applies.
<a href="#">CurrencyTable Property</a>	Returns the related CurrencyTable object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">PopulateByDataSource Property</a>	Sets or returns whether the currency rate is obtained through a data source or set within Transformer.
<a href="#">Rate Property</a>	Sets or returns the currency exchange rate.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()objCurrencyRate
= objCurrencyRecord.CurrencyRates(intX)
```

## CurrencyRecord Object

The Currency Record object defines how currency information appears.

### Discussion

Use this object to

- specify the date level for a currency rate
- define the information needed to format a currency for display in PowerPlay

This information includes the name of the currency, the currency code, the currency symbol, and the decimal places. The CurrencyDecimals and CurrencySymbol properties cannot be changed unless you set the CurrencyFormatOverride property to True.

If you use a data source to populate a currency table, CurrencyRecord objects are automatically added to the model, as are the currency rates associated with each record.

To explicitly create a CurrencyRecord object, use the Add method of the CurrencyRecords collection. In this case, although CurrencyRate objects are automatically generated, you must manually maintain the rates.

CurrencyRate, CurrencyRecord, CurrencyTable, and DateDimension objects are all needed for currency conversion to work.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the CurrencyRecord object.
<a href="#">Update Method</a>	Updates the CurrencyRecord object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">CountryCode Property</a>	Sets or returns an ISO code for the country or region to which the currency record applies.
<a href="#">CurrencyCountryLabel Property</a>	Sets or returns the currency country or region label.
<a href="#">CurrencyDecimals Property</a>	Sets or returns the number of decimal places used in the currency.
<a href="#">CurrencyFormatOverride Property</a>	Sets or returns whether you can override the standard format for a currency.
<a href="#">CurrencyIsEMU Property</a>	Sets or returns whether the record is an EMU currency record.
<a href="#">CurrencyIsEuro Property</a>	Sets or returns whether the currency record is the base euro currency.
<a href="#">CurrencyRates Property</a>	Returns the CurrencyRates collection.
<a href="#">CurrencySymbol Property</a>	Sets or returns the monetary symbol associated with the currency.
<a href="#">DateLevel Property</a>	Sets or returns the level in a time dimension to which the currency rates apply.
<a href="#">EMUEntryDate Property</a>	Sets or returns the date on which euro triangulation calculations began for a currency.
<a href="#">Label Property</a>	Sets or returns a descriptive name that appears in PowerPlay.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()
```

## CurrencyTable Object

The CurrencyTable object defines a currency table used in currency conversion.

## Discussion

Use this object to set up currency records and rates.

To create a CurrencyTable object, use the Add method of the CurrencyTables collection. If you use a data source to populate a currency table, CurrencyRecord objects are automatically added to the model, as are the currency rates associated with each record.

Alternatively, you can manually add currency records to a currency table. In this case, CurrencyRate objects are automatically generated for each new CurrencyRecord object, but you must manually maintain the rates.

CurrencyRate, CurrencyRecord, CurrencyTable, and DateDimension objects are all needed for currency conversion to work.

For conversions involving non-European Monetary Union (EMU) currencies or conversions involving EMU currencies before 1999, only one table is needed, with the CurrencyTableType property set to trCurrencyTableBase. The base table initially shows the default currency, such as US dollars. This is based on the regional setting of your operating system. The setting can be changed to any other currency, including the euro.

For conversions that involve EMU currencies after 1998, you need two tables. Set the CurrencyTableType property of one table to trCurrencyTableBase and set the same property in the second table to trCurrencyTableEuro. The euro table contains the fixed conversion rates used to convert values to, or from, the national currencies of EMU countries or regions by using the euro and base currencies as intermediate values in the conversion. This process is called euro triangulation. For euro triangulation to work, sort the date levels in ascending order and set the Unique property of each date level to True. Give the date field a zero value (0) for the records of EMU members that follow January 1st, 1999.

The required multiplication, division, and rounding operations are all performed in PowerPlay.

You can use several data sources to supply the currency data for your base table. However, all column names must match so that the information combines properly. The same requirement applies to the euro table, although you must assign different names to the columns associated with the euro table.

When you use a data source to populate a table, set the GeneratePowerCube property to trGenerationNoCreatePowerCubes and the GenerateCategories property to False. By doing so, any updates to the CurrencyTable object do not interfere with category generation in the rest of your model. You can leave these properties enabled if the names of the date columns in your base and euro currency data sources differ from the column name of your time dimension, and from each other. Otherwise, Transformer cannot differentiate between them.

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">AssociateWith Method</a>	Associates a source object with a CurrencyTable object.
<a href="#">Delete Method</a>	Deletes the CurrencyTable object.
<a href="#">Update Method</a>	Updates the CurrencyTable object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.



Property	Description
<a href="#">Associations Property</a>	Returns an Associations collection.
<a href="#">CurrencyTableType Property</a>	Sets or returns the type of currency table.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
objBaseTable = objModel.CurrencyTables.Add()
```

## CustomView Object

The CustomView object represents a custom view.

### Discussion

Use this object to provide basic security to the cubes generated from a model. Each custom view consists of security objects, such as users, groups and roles, who perform similar job functions, and therefore require similar information.

Each CustomView object automatically includes a collection of View objects: one for each dimension. The default view for each object in a collection is "All Categories". To change this view, you can use the DimensionInclude property to omit all categories or to create a custom view.

Custom views provide a specific subset of cube information to PowerPlay users. For example, a custom view may summarize an entire level or individual categories within the level. If you specify a custom view (trViewTypeCustom), you must also use the DimensionView property to return the View object associated with a Dimension. You can then use the Apex property to create a new root category, or the SetViewStatus method to associate the View object with selected levels or categories.

To complete a view with a custom view, you must associate a custom view with a cube. Use the Add method to add a CustomView object to the CubeCustomViews collection of the cube.

CustomView objects are defined in the model and are returned by the CustomViews property. You can modify properties of the Transformer CustomView object, such as DimensionInclude, to direct the scope of the custom view within Transformer.

To control access to all or parts of a cube, define groups of users in a namespace and then, in your Transformer model, associate the custom views with that cube.

For example, open a model that contains global performance data by product line, vendor, and region. Create a cube group in which each cube contains detailed performance data for one region and summarized data for all other regions. Custom views are defined and applied so that each sales team has the detailed information for their own region.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the CustomView object.
<a href="#">Update Method</a>	Updates the CustomView object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">ChildCustomViews Property</a>	Returns a collection of CustomView objects.
<a href="#">Description Property</a>	Returns the description of the CustomView object.
<a href="#">DimensionInclude Property</a>	Sets or returns the type of view for a custom view.
<a href="#">DimensionView Property</a>	Returns the View object associated with a custom view.
<a href="#">MeasureInclude Property</a>	Sets or returns a Boolean to determine if a measure applies to a CustomView object.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">SecurityObjects Property</a>	Returns the collection of SecurityObjects associated with the custom view.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">Views Property</a>	Returns a Views collection.

## Examples

```
custom_view = model.CustomViews.Add()
```

## DataSource Object

The DataSource object defines a generic data source.

### Discussion

Use this object to reference objects in the DataSources collection. Since the DataSource object represents a generic data source, the properties change to reflect the underlying data source.

The DataSource object does not have a type constant in the xtrObjectType value list. It derives the value for the Type property from the data source it currently represents. Because of this and other differences, it is not as effective as specific data source objects, such as an IqdDataSource.

To get the full functionality available with the DataSource object, use a statement like this in Visual Basic or VB-compatible editor:

```
Dim objDataSource As CognosTransformer.DataSource
```

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the DataSource object.
<a href="#">Move Method</a>	Moves the DataSource object to a different position in the DataSources collection.
<a href="#">Update Method</a>	Updates the DataSource object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Description Property</a>	Sets or returns the description of the DataSource object.
<a href="#">GenerateCategories Property</a>	Sets or returns whether categories are generated for the data source.
<a href="#">GeneratePowerCube Property</a>	Sets or returns when a data source is referenced by a model.
<a href="#">IsAnyColumnMismatched Property</a>	Returns whether columns in the data source match the underlying data.
<a href="#">MaximizeSpeed Property</a>	Sets or returns whether category generation is optimized for speed.
<a href="#">Name Property</a>	Sets or returns the unique name for the data source.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">ServerQuery Property</a>	Sets or returns whether data is processed locally or on a server.
<a href="#">SetsCurrentPeriod Property</a>	Sets or returns whether Transformer searches a data source to find the current period date.
<a href="#">SourceType Property</a>	Sets or returns the type of data file a data source uses.
<a href="#">Type Property</a>	Returns the type of the underlying data source the DataSource object is currently referencing.

## Examples

```
objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
```

## DateDimension Object

The DateDimension object organizes date data into logical groups.

### Discussion

A time dimension is a hierarchical structure that represents time-based information about a major aspect of a business. Each time dimension consists of a DrillDowns collection and one or more DateDrillDown objects, which represent drill-down paths.

Use this object to provide time context to any model. Many of the benefits of a cube, such as tracking profits and costs, are lost unless at least one DateDimension object is included. Also, without a time dimension, currency conversion is disabled.

Transformer automatically creates a time dimension based on date definitions in the data source when you use the DoAutoDesign method.

You can manually create a DateDimension object in two ways:

- Set all the applicable properties of the DateWizard object and assign a data source using the DimensionAssociateWith method. Then, use the CreateDateDimension method to complete the process. This process creates the dimension as well as a drill-down path and date levels. It also generates a set of relative time categories, such as year-to-date.
- Use the Add method of the Dimensions collection, and set the Type parameter to trDateDimension. You must set the Name property for the object. You can then use the AssociateWith method to assign a data source. This procedure creates the drill-down path to which you can then add date levels and relative time categories.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">AssociateWith Method</a>	Associates a source object with a DateDimension object.
<a href="#">CleanHouse Method</a>	Removes inactive categories from the time dimension.
<a href="#">Delete Method</a>	Deletes the DateDimension object.
<a href="#">GenerateDateCategories Method</a>	Populates a model with date categories specified by StartDate and EndDate.
<a href="#">Move Method</a>	Moves the time dimension to a different position in the Dimensions collection.
<a href="#">SetAllocation Method</a>	Changes the allocation type for the measure used by the object.
<a href="#">Update Method</a>	Updates the DateDimension object.

<b>Property</b>	<b>Description</b>
<a href="#">AllocationMeasure Property</a>	Returns the Measure object used as a weighting factor.
<a href="#">AllocationType Property</a>	Returns how an object allocates a measure.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Associations Property</a>	Returns an Associations collection.
<a href="#">CalculationDefinitions Property</a>	Returns a collection of CalculationDefinition objects.
<a href="#">CanAllocate Property</a>	Returns whether you can allocate any measure values to descendant levels and categories.
<a href="#">CanAllocateByMeasure Property</a>	Returns whether you can use the specified measure as a weighting factor when you allocate by measure.
<a href="#">CanAllocateMeasure Property</a>	Returns whether you can allocate the specified measure to descendant levels and categories.
<a href="#">Categories Property</a>	Returns the collection of categories that are included in the DateDimension object.
<a href="#">CategoryCount Property</a>	Returns the number of regular categories in the DateDimension object.
<a href="#">Description Property</a>	Sets or returns the description of the DateDimension object.
<a href="#">DimensionLevels Property</a>	Returns a DimensionLevels collection.
<a href="#">DrillDowns Property</a>	Returns a DrillDowns collection.
<a href="#">EarliestDate Property</a>	Sets or returns the earliest date in a range of date categories.
<a href="#">ExcludeAutoPartition Property</a>	Sets or returns whether the time dimension is excluded from the auto-partition process.
<a href="#">GenerateTimePeriod Property</a>	Sets or returns category generation options for a time dimension.
<a href="#">GetDefaultCategory Method</a>	Returns the default category belonging to a dimension.
<a href="#">LatestDate Property</a>	Sets or returns the latest date in a range of date categories.
<a href="#">ManualCurrentPeriod Property</a>	Sets or returns whether the current time period is set manually or by Transformer.

Property	Description
<a href="#">Name Property</a>	Sets or returns the name of the DateDimension object.
<a href="#">NewCatsLocked Property</a>	Sets or returns whether you can add new categories to the DateDimension object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">SetDefaultCategory Method</a>	Specifies a new default category for a dimension, other than the root or parent of a group of scenario categories.
<a href="#">SpecialCategoryCount Property</a>	Returns the number of drill, root, and special categories in a dimension.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">Views Property</a>	Returns a Views collection.
<a href="#">WorkingDay Property</a>	Sets or returns whether a day is part of the working week.
<a href="#">WorkingDays Property</a>	Sets or returns which days are part of the working week.

## Examples

```
objDateDim = objModel.Dimensions("Date")
```

## DateDrillDown Object

The DateDrillDown object defines primary and alternate drill-down paths for date or time values.

### Discussion

Use this object to modify the properties of the primary DateDrillDown object or to define alternate DateDrillDown objects.

Each time dimension has a DrillDowns collection. A primary DateDrillDown object is automatically added to that collection by Transformer when you build a time dimension. A collection contains only one primary drill-down path that you cannot delete. Any other DateDrillDown objects in the collection are alternate DateDrillDown objects that you create. An alternate drill-down path defines a different perspective on the data in PowerPlay.

You can use the IsPrimary property to change an alternate drill-down path to the primary drill-down path.

To add a DateDrillDown object to the collection, return the DateLevel object that you want as the lowest level of detail in the new drill-down path. Then use the CreateAlternateDrillDown method of the specified DateLevel object to create the new drill-down path. Because this level is common to the existing and new drill-down paths, it is called the convergence level.

To add more date levels to the alternate drill-down path, use the Add method of the newly created DateDrillDown object. You must then change the order of the collection to reflect the necessary drill-down hierarchy. To change the order, use the Move method to shift the additional levels to a position above the convergence level. For example, if the convergence level is set to Weeks and you add the Years level

to the Levels collection, use the Move method to move the Years level object to the first position in the collection.

Alternate drill-down paths always begin at the root category and extend to the selected convergence level. To correctly connect several parent categories to the same convergence category, each category in a convergence level must be unique and unambiguous. To specify category uniqueness, set the Unique property of the convergence level to True.

Because the category values at the convergence level and below are shared by all drill-down paths, removing or changing a category in one path at or below the convergence level immediately affects the same category in all other drill-down paths.

Categories that belong to multiple drill-down paths have more than one parent category. Use the ParentCategories property to return a collection of parent categories.

For information about alternate drill-down paths in time dimensions, see the Transformer online help.

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the DateDrillDown object.
<a href="#">Update Method</a>	Updates the DateDrillDown object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Categories Property</a>	Returns the collection of categories that are included in the DateDrillDown object.
<a href="#">ConvergenceLevel Property</a>	Returns the convergence level for the alternate drill-down path.
<a href="#">DrillCode Property</a>	Sets or returns a code that uniquely identifies the drill-down category within the entire dimension.
<a href="#">DrillInclusion Property</a>	Sets or returns whether the DateDrillDown object is included in the cube.
<a href="#">IsPrimary Property</a>	Sets or returns whether the drill-down path is the primary drill-down path.
<a href="#">Levels Property</a>	Returns a Levels collection.
<a href="#">Lunar Property</a>	Sets or returns whether the object is based on a lunar year.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.

Property	Description
<a href="#">ShortName Property</a>	Sets or returns a short name for the drill-down path.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">WeekAdd Property</a>	Sets or returns how many days are added to a lunar year.
<a href="#">WeekSpan Property</a>	Sets or returns how to treat a week that spans two years.
<a href="#">WeekStart Property</a>	Sets or returns the first day of the week.
<a href="#">YearStartDay Property</a>	Sets or returns the first day of a year.

## Examples

```
objLocationsDim = objModel.Dimensions("Sales
regions")objLevel = objLocationsDim.DrillDowns(1).Levels("Employee")objAltDrill
= objLevel.CreateAlternateDrillDown
```

## DateLevel Object

The DateLevel object defines a date level in a model.

### Discussion

Use the DateLevel object to present date-related context to PowerCube data, for example, Year, Quarter, or Month intervals.

Each DateDimension object in a model includes a Levels collection that contains DateLevel objects. There are two level types:

- Source levels, which contain categories that are generated from the data source. Each source level is linked to a column in the data source through an Association object.
- Manual levels, which are not associated with source columns. For example, you can create a manual level to contain orphan categories.

Use the Add method of the Levels collection to create a new DateLevel object.

Use the CreateAlternateDrillDown method to make a date level the convergence level in an alternate drill-down path.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Collection	Description
<a href="#">LevelCategories Collection</a>	Returns a collection of categories in a level.



Method	Description
<a href="#">AssociateWith Method</a>	Associates a source object with a DateLevel object.
<a href="#">CreateAlternateDrillDown Method</a>	Creates an alternate drill-down path in a dimension and makes the level the convergence level.
<a href="#">Delete Method</a>	Deletes the DateLevel object.
<a href="#">Move Method</a>	Moves the DateLevel object to a different position in the Levels collection.
<a href="#">SetAllocation Method</a>	Changes the allocation type for the measure used by the object.
<a href="#">Update Method</a>	Updates the DateLevel object.

Property	Description
<a href="#">AllocationMeasure Property</a>	Returns the Measure object used as a weighting factor.
<a href="#">AllocationType Property</a>	Returns how an object allocates a measure.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Associations Property</a>	Returns an Associations collection.
<a href="#">BlankSubstitute Property</a>	Sets or returns the default label for blank categories generated in the date level.
<a href="#">CanAllocate Property</a>	Returns whether you can allocate any measure values to descendant date levels and categories.
<a href="#">CanAllocateByMeasure Property</a>	Returns whether you can use the specified measure as a weighting factor when you allocate by measure.
<a href="#">CanAllocateMeasure Property</a>	Returns whether you can allocate the specified measure to descendant date levels and categories.
<a href="#">CategoryCount Property</a>	Returns the number of regular categories in the DateLevel object.
<a href="#">DateFormat Property</a>	Sets or returns how dates appear.
<a href="#">DateFunction Property</a>	Sets or returns which date categories are generated in a level.
<a href="#">Description Property</a>	Sets or returns the description of the DateLevel object.
<a href="#">GenerateDateCategories Property</a>	Sets or returns whether a date level generates date categories.

Property	Description
<a href="#">HasSubdimension Property</a>	Returns whether a level object contains a subdimension.
<a href="#">Inclusion Property</a>	Sets or returns when a category is included in a cube.
<a href="#">IsManual Property</a>	Returns whether the date level was manually created or was generated.
<a href="#">LevelDrillDowns Property</a>	Returns the LevelDrillDowns collection
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">NewCatsLocked Property</a>	Sets or returns whether you can add new categories to the DateLevel object.
<a href="#">OrderByDescending Property</a>	Sets or returns whether values appear in descending order.
<a href="#">OrderByStorageType Property</a>	Sets or returns how categories are sorted based on the storage type of a column.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Partition Property</a>	Sets or returns a manual partition number.
<a href="#">QualifiedName Property</a>	Returns the fully qualified name of the date level.
<a href="#">RefreshDescription Property</a>	Sets or returns whether descriptions are updated.
<a href="#">RefreshLabel Property</a>	Sets or returns whether labels are updated.
<a href="#">RefreshShortName Property</a>	Sets or returns whether short names are updated.
<a href="#">TimeRank Property</a>	Sets or returns the relative rank of date levels within a time dimension.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">Unique Property</a>	Sets or returns whether Transformer can identify each category in the date level by a unique source value.
<a href="#">UniqueMove Property</a>	Sets or returns how a unique DateLevel object is treated when the related category is moved.

## Examples

```
objDateDim = objModel.Dimensions("Date")objDateLevel
= objDateDim.DimensionLevels("Month")
```

## DateWizard Object

The DateWizard object creates a DateDimension object.

### Discussion

A time dimension is a hierarchical structure that represents time-based information about a business. This object provides an easy way to build DateDimension and DateLevel objects.

You must first use the DateWizard property of the Model object to return the DateWizard object. Next, set all the applicable properties of the DateWizard object and assign a data source using the DimensionAssociateWith method. Finally, use the CreateDateDimension method to create a new DateDimension object. To enable a lunar hierarchy set the MonthType, QuarterType, and YearType properties to lunar values.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">CreateDateDimension Method</a>	Creates a DateDimension object based on the properties of the DateWizard object.
<a href="#">DimensionAssociateWith Method</a>	Associates a source object with the new DateDimension object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">DimensionName Property</a>	Sets the name for the new DateDimension object.
<a href="#">EarliestDate Property</a>	Sets the earliest date in a range of date categories.
<a href="#">EnableTimePeriod Property</a>	Sets the level of detail for a time dimension.
<a href="#">GeneratePowerCube Property</a>	Sets whether the DateWizard object generates date categories.
<a href="#">LatestDate Property</a>	Sets the latest date in a range of date categories.
<a href="#">MonthType Property</a>	Sets how to calculate the month level of a time dimension.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">QuarterType Property</a>	Sets how to calculate the quarter level of a time dimension.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">WeekAdd Property</a>	Sets how many days are added to a lunar year.
<a href="#">WeekSpan Property</a>	Sets how to treat a week that spans two years.

Property	Description
<a href="#">WeekStartDay Property</a>	Sets the first day of the week.
<a href="#">WorkingDay Property</a>	Sets whether a day is part of the working week.
<a href="#">WorkingDays Property</a>	Sets which days are part of the working week.
<a href="#">YearStartDay Property</a>	Sets the first day of a year.
<a href="#">YearType Property</a>	Sets how to calculate the year level of a time dimension.

## Examples

```
objDateWizard = objModel.DateWizard
```

## DbDataSource Object

The DbDataSource object represents an external file in database format.

### Discussion

Use this object to connect to an external file in database format.

To create a DbDataSource object, use the Add method of the DataSources collection. Ensure that you set the Type parameter to the trDbDataSource constant. Set the SourceType property to the type of database file to open. Use the LocalPath and ServerPath properties, as applicable, to give the location of the data file.

DbDataSource objects contain a Columns collection. Columns are the source for levels in a dimension.

Transformer can use database information from the following:

- dBase
- Excel
- Access
- Paradox
- FoxPro
- Clipper

For example, you can use this property to specify a table from an Access database (.mdb).

You can use an Access query that points to an ODBC data source to get server-based data into Transformer. If you use an Access query as a source, the Columns collection is derived from the SQL query that references the database.

If you use Paradox tables with a sorting key other than ASCII, which is the default setting in the Jet engine for Paradox, you must include the primary index file (.px) in the same location as the Paradox database file (.db) before importing the tables. You must also modify the Windows registry key to a value corresponding to the sorting key used when creating the table. Valid values are

- ASCII
- International
- Norwegian-Danish
- Japanese

- Swedish-Finnish

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the DbDataSource object.
<a href="#">Move Method</a>	Moves the DbDataSource object to a different position in the DataSources collection.
<a href="#">Update Method</a>	Updates the DbDataSource object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Columns Property</a>	Returns a Columns collection.
<a href="#">ColumnsLoaded Property</a>	Returns whether the columns in a data source have been used to build a model.
<a href="#">DataRange Property</a>	Sets or returns the name of a database range in the data source.
<a href="#">DecimalPoint Property</a>	Sets or returns the separator character used for a decimal point.
<a href="#">Description Property</a>	Sets or returns the description of the DbDataSource object.
<a href="#">External Property</a>	Sets or returns whether values in the data source are treated as summarized or rolled up when used by the model.
<a href="#">GenerateCategories Property</a>	Sets or returns whether categories are generated for the data source.
<a href="#">GeneratePowerCube Property</a>	Sets or returns when a data source is referenced by a model.
<a href="#">IsAnyColumnMismatched Property</a>	Returns whether columns in the data source match the underlying data.
<a href="#">LocalPath Property</a>	Sets or returns the location of the local data source.
<a href="#">MaximizeSpeed Property</a>	Sets or returns whether category generation is optimized for speed.

Property	Description
<a href="#">Name Property</a>	Sets or returns the unique name for the data source.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">ServerPath Property</a>	Sets or returns the name and location of a data source file.
<a href="#">ServerQuery Property</a>	Returns whether data is processed locally or on a server.
<a href="#">SetsCurrentPeriod Property</a>	Sets or returns whether Transformer searches a data source to find the current period date.
<a href="#">SourceType Property</a>	Sets or returns the type of data file a data source uses.
<a href="#">ThousandPoint Property</a>	Sets or returns the character used to separate numbers in thousands.
<a href="#">Type Property</a>	Returns the object type. The object type returned is trDbQuery even though the xtrObjectType.trDbDataSource type must be used to add and retrieve an item from the DataSources collection.

## Dimension Object

The Dimension object organizes non-date data into logical groups.

### Discussion

Use a Dimension object to group Level and DrillDown objects that provide numeric or textual data. A dimension is a hierarchical structure that represents information about a major aspect of a business.

If you use the DoAutoDesign method, Transformer automatically creates dimensions and their associated Level and DrillDown objects based on patterns and relationships in the data sources.

Alternatively, you can manually create Dimension objects. First, use the Add method of the Dimensions collection. Then add Level objects to their applicable collections within the dimension.

For date or time data, use a DateDimension object instead.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">AssociateWith Method</a>	Associates a source object with a Dimension object.
<a href="#">CleanHouse Method</a>	Removes inactive categories from the dimension.

Method	Description
<a href="#">Delete Method</a>	Deletes the Dimension object.
<a href="#">FindCategoryByCatCode Method</a>	Returns the category object that contains the specified category code string.
<a href="#">GetDefaultCategory Method</a>	Returns the default category belonging to a dimension.
<a href="#">Move Method</a>	Moves the dimension to a different position in the Dimensions collection.
<a href="#">SetAllocation Method</a>	Changes the allocation type for the measure used by the object.
<a href="#">SetDefaultCategory Method</a>	Specifies a new default category for a dimension, other than the root or parent of a group of scenario categories.
<a href="#">Update Method</a>	Updates the Dimension object.

Property	Description
<a href="#">AllocationMeasure Property</a>	Returns the Measure object used as a weighting factor.
<a href="#">AllocationType Property</a>	Returns how an object allocates a measure.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Associations Property</a>	Returns an Associations collection.
<a href="#">CalculationDefinitions Property</a>	Returns a collection of CalculationDefinition objects.
<a href="#">CanAllocate Property</a>	Returns whether you can allocate any measure values to descendant levels and categories.
<a href="#">CanAllocateByMeasure Property</a>	Returns whether you can use the specified measure as a weighting factor when you allocate by measure.
<a href="#">CanAllocateMeasure Property</a>	Returns whether you can allocate the specified measure to descendant levels and categories.
<a href="#">Categories Property</a>	Returns the collection of special categories that are included in the Dimension object.
<a href="#">CategoryCount Property</a>	Returns a count of regular categories in the Dimension object.
<a href="#">Description Property</a>	Sets or returns the description of the Dimension object.

Property	Description
<a href="#">DimensionLevels Property</a>	Returns a DimensionLevels collection.
<a href="#">DrillDowns Property</a>	Returns a DrillDowns collection.
<a href="#">ExcludeAutoPartition Property</a>	Sets or returns whether the dimension is excluded from the auto-partition process.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">NewCatsLocked Property</a>	Sets or returns whether you can add new categories to the Dimension object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">SpecialCategoryCount Property</a>	Returns the number of drill, root, and special categories in a dimension.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">Views Property</a>	Returns a Views collection.

## Examples

```
objDimensions = objModel.DimensionsobjLocationsDim
= objDimensions.Item(3)
```

## DrillDown Object

The DrillDown object defines primary and alternate drill-down paths used to navigate cubes.

### Discussion

A DrillDowns collection can contain one primary drill-down path. Any other drill-down paths in the DrillDowns collection are alternate drill-down paths that you create. An alternate drill-down path defines a different perspective on the data in PowerPlay.

Use this object to modify the properties of the primary drill-down path or to define alternate drill-down paths. You cannot delete the default primary drill-down path created by Transformer.

To add a new alternate DrillDown object to the DrillDowns collection, apply the CreateAlternateDrillDown method to the Level object you want to be the convergence level in the alternate drill-down path. The level you select becomes the first level in the Levels collection of the new DrillDown object. You then add more levels to that Levels collection by using the Add method. Because the Add method always adds new levels to the end of the collection, use the Move method to shift the additional levels to a position before the convergence level.

Alternate drill-down paths always begin at the root category and extend to the selected convergence level. To correctly connect several parent categories to the same convergence category, each category in a convergence level must be unique and unambiguous. In other words, no two categories in the level can have the same value. For Transformer to accept an alternate drill-down path, you must set the Unique property of the convergence level to True.

Because the category values at the convergence level and below are shared by all drill-down paths, removing or changing a category in one path at or below the convergence level immediately affects the same category in all other drill-down paths.



Categories that belong to multiple drill-down paths have more than one parent category. Use the ParentCategories property to return a collection of parent categories.

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the DrillDown object.
<a href="#">Update Method</a>	Updates the DrillDown object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Categories Property</a>	Returns the collection of categories that are included in the DrillDown object.
<a href="#">ConvergenceLevel Property</a>	Returns the convergence level for the alternate drill-down path.
<a href="#">DrillCode Property</a>	Sets or returns a code that uniquely identifies the drill-down category within the entire dimension.
<a href="#">DrillInclusion Property</a>	Sets or returns whether the DrillDown object is included in the cube.
<a href="#">IsPrimary Property</a>	Sets or returns whether the drill-down path is the primary drill-down path.
<a href="#">Levels Property</a>	Returns a Levels collection.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">ShortName Property</a>	Sets or returns a short name for the drill-down path.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
objDrill = objModel.Dimensions("Retailers").DrillDowns(2)
```

## DrillThroughTarget Object

The DrillThroughTarget object represents a drill-through link to an external report or PowerCube.

## Discussion

Drill-through targets can be Impromptu reports (.imr), Impromptu query definition files (.iqd), PowerPlay reports (.ppr), PowerCubes (.mdc), or macro script files (.mac). Through OLE automation, the associated application opens directly from PowerPlay and shows the report from the specified vantage point.

Use the Add method of the DrillThroughTargets collection to create a DrillThroughTarget object. When you add a DrillThroughTarget object to the DrillThroughTargets collection of a Measure object, you restrict the drill-through capability to that measure alone. When you add a DrillThroughTarget object to the DrillThroughTargets collection of a Cube or CubeGroup object, the drill-through link is available anywhere in the cube or cubes.

The AllowDrillThrough property must be set to True before drill-through capability is available.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the DrillThroughTarget object.
<a href="#">isExcludeDimension Method</a>	Returns whether the given Dimension is excluded.
<a href="#">isExcludeLevel Method</a>	Returns whether the given Level is excluded.
<a href="#">SetExcludeDimension Method</a>	Sets the given Dimension to the isRestricted value.
<a href="#">SetExcludeLevel Method</a>	Sets the given Level to the isRestricted value.
<a href="#">Update Method</a>	Updates the DrillThroughTarget object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Description Property</a>	Sets or returns the description of the DrillThroughTarget object.
<a href="#">Name Property</a>	Sets or returns the name of the object. For a report, this needs to be a fully qualified file name.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
objReport = objMeasure.DrillThroughTargets.Add(strReportPath,  
"Default Report")
```

## Filter Object

The Filter object represents a filter in the Transformer model.

## Discussion

A Filters collection is returned by the Filters property (Query object). Each Filter object in the collection references a filter in the Transformer model.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the Filter object.
<a href="#">Update Method</a>	Updates the Filter object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">RefName Property</a>	Sets or returns the name of the filter in the data source.

## Examples

```
new_package = model.Packages.Add()new_query  
= new_package.Queries.Add()new_filter = new_query.Filters.Add()
```

## FlatFileDataSource Object

The FlatFileDataSource object represents an external file in ASCII format.

## Discussion

Use a FlatFileDataSource object when your data is in ASCII format and the columns are delimited by field separators.

To create a FlatFileDataSource object, use the Add method of the DataSources collection. Set the Type parameter to the trFlatFileDataSource constant. Set the SourceType property to the type of flat file to open. Use properties, such as FieldSeparator, DecimalPoint, and ThousandPoint, to describe the data format and structure. Use the LocalPath property or ServerPath property to give the location of the data file.

FlatFileDataSource objects contain a Columns collection. Columns are the source for levels in a dimension.

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the FlatFileDataSource object.
<a href="#">Move Method</a>	Moves the FlatFileDataSource object to a different position in the DataSources collection.
<a href="#">Update Method</a>	Updates the FlatFileDataSource object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">CharacterType Property</a>	Sets or returns the type of character set used by the data source.
<a href="#">Columns Property</a>	Returns a Columns collection.
<a href="#">ColumnsLoaded Property</a>	Returns whether the columns in a data source have been used to build a model.
<a href="#">DecimalPoint Property</a>	Sets or returns the character used for a decimal point in the flat file.
<a href="#">Description Property</a>	Sets or returns the description of the FlatFileDataSource object.
<a href="#">External Property</a>	Sets or returns whether the data source contains presummarized values.
<a href="#">FieldSeparator Property</a>	Returns the character used to mark the boundaries of fields in the flat file.
<a href="#">GenerateCategories Property</a>	Sets or returns whether categories are generated for the data source.
<a href="#">GeneratePowerCube Property</a>	Sets or returns when a data source is referenced by a model.
<a href="#">IsAnyColumnMismatched Property</a>	Returns whether columns in the data source match the underlying data.
<a href="#">LocalPath Property</a>	Sets or returns the location of the local data source.
<a href="#">MaximizeSpeed Property</a>	Sets or returns whether category generation is optimized for speed.
<a href="#">Name Property</a>	Sets or returns the unique name for the data source.
<a href="#">Parent Property</a>	Returns the parent object.

Property	Description
<a href="#">ServerPath Property</a>	Sets or returns the name and location of a data source file.
<a href="#">ServerQuery Property</a>	Sets or returns whether data is processed locally or on a server.
<a href="#">SetsCurrentPeriod Property</a>	Sets or returns whether Transformer searches a data source to find the current period date.
<a href="#">SourceType Property</a>	Sets or returns the type of data file a data source uses.
<a href="#">ThousandPoint Property</a>	Sets or returns the character used to separate numbers in thousands.
<a href="#">Type Property</a>	Returns the object type. A trFlatFileQuery type is returned, however, a trFlatFileDataSource needs to be used when retrieving or adding a data source to the DataSources collection.

## Examples

```
objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
```

## IqdDataSource Object

The IqdDataSource object represents an Impromptu query definition file (.iqd).

### Discussion

Use this object to set up access to an external file in Impromptu format.

To create an IqdDataSource object, use the Add method of the DataSources collection. Ensure that you set the Type parameter to the trIqdDataSource constant.

Set the SourceType property of the IqdDataSource to the constant, trQuery. Use the LocalPath or ServerPath properties, as applicable, to specify the location of the Impromptu query definition file.

IqdDataSource objects contain a Columns collection. Columns are the source for levels in a dimension.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the IqdDataSource object.
<a href="#">Move Method</a>	Moves the IqdDataSource object to a different position in the DataSources collection.
<a href="#">Update Method</a>	Updates the IqdDataSource object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Columns Property</a>	Returns a Columns collection.
<a href="#">ColumnsLoaded Property</a>	Returns whether the columns in a data source have been used to build a model.
<a href="#">DecimalPoint Property</a>	Sets or returns the character used for a decimal point in an Iqd file.
<a href="#">Description Property</a>	Sets or returns the description of the IqdDataSource object.
<a href="#">External Property</a>	Sets or returns whether values in the data source are treated as summarized or rolled up when used by the model.
<a href="#">GenerateCategories Property</a>	Sets or returns whether categories are generated for the data source.
<a href="#">GeneratePowerCube Property</a>	Sets or returns when a data source is referenced by a model.
<a href="#">IsAnyColumnMismatched Property</a>	Returns whether columns in the data source match the underlying data.
<a href="#">IsolationLevel Property</a>	Sets or returns the isolation level used to define permissible transactions.
<a href="#">LocalPath Property</a>	Sets or returns the location of the local data source.
<a href="#">MaximizeSpeed Property</a>	Sets or returns whether category generation is optimized for speed.
<a href="#">Name Property</a>	Sets or returns the unique name for the data source.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">ServerPath Property</a>	Sets or returns the name and location of a data source file.
<a href="#">ServerQuery Property</a>	Sets or returns whether data is processed locally or on a server.
<a href="#">SetsCurrentPeriod Property</a>	Sets or returns whether Transformer searches a data source to find the current period date.
<a href="#">SourceType Property</a>	Sets or returns the type of data file a data source uses.

Property	Description
<a href="#">SQLExpression Property</a>	Returns the SQL expression used to define an Impromptu query definition file (.iqd).
<a href="#">ThousandPoint Property</a>	Sets or returns the character used to separate numbers in thousands.
<a href="#">Type Property</a>	Returns the object type. trIqdQuery is the type returned even though the trIqdDataSource type must be used when retrieving or adding from the DataSources Collection.

## Examples

```
datasource = model.DataSources.Add(xtrObjectType.trIqdDataSource)
```

## Level Object

The Level object defines a level in a dimension.

### Discussion

Use the Level object to present all PowerCube data other than data directly related to dates.

Each Dimension object in a model includes a Levels collection that contains Level objects. There are two level types:

- Source levels, which contain categories that are generated from the data source. Each source level is linked to a column in the data source through an Association object.
- Manual levels, which are not associated with source columns. For example, you can create a manual level to contain orphan categories.

Use the Add method of the Levels collection to create a new Level object.

Use the CreateAlternateDrillDown method to make a level the convergence level in an alternate drill-down path.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">AssociateWith Method</a>	Associates a source object with the Level object.
<a href="#">CreateAlternateDrillDown Method</a>	Creates an alternate drill-down path in a dimension.
<a href="#">Delete Method</a>	Deletes the Level object.
<a href="#">Move Method</a>	Moves the level to a different position in the Levels collection.

Method	Description
<a href="#">SetAllocation Method</a>	Changes the allocation type for the measure used by the object.
<a href="#">Update Method</a>	Updates the Level object.

Property	Description
<a href="#">AllocationMeasure Property</a>	Returns the Measure object used as a weighting factor.
<a href="#">AllocationType Property</a>	Returns how an object allocates a measure.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Associations Property</a>	Returns an Associations collection.
<a href="#">BlankSubstitute Property</a>	Sets or returns the default label for blank categories generated in the Level object.
<a href="#">CanAllocate Property</a>	Returns whether you can allocate any measure values to descendant levels and categories.
<a href="#">CanAllocateByMeasure Property</a>	Returns whether you can use the specified measure as a weighting factor when you allocate by measure.
<a href="#">CanAllocateMeasure Property</a>	Returns whether you can allocate the specified measure to descendant levels and categories.
<a href="#">CategoryCount Property</a>	Returns the number of regular categories in the Level object.
<a href="#">Description Property</a>	Sets or returns the description of the Level object.
<a href="#">HasSubdimension Property</a>	Returns True if a Level object contains a subdimension.
<a href="#">Inclusion Property</a>	Sets or returns when a category is included in a cube.
<a href="#">IsManual Property</a>	Returns whether a level is associated with a source value.
<a href="#">LevelCategories Property</a>	Returns a collection of categories in a level.
<a href="#">LevelDrillDowns Property</a>	Returns a LevelDrillDowns collection.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">NewCatsLocked Property</a>	Sets or returns whether you can add new categories to the Level object.



Property	Description
<a href="#">OrderByDescending Property</a>	Sets or returns whether values appear in descending order.
<a href="#">OrderByStorageType Property</a>	Sets or returns how categories are sorted based on the storage type of a column.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Partition Property</a>	Sets or returns a manual partition number.
<a href="#">QualifiedName Property</a>	Sets or returns a qualified name for the Level object.
<a href="#">RefreshDescription Property</a>	Sets or returns whether descriptions are updated.
<a href="#">RefreshLabel Property</a>	Sets or returns whether labels are updated.
<a href="#">RefreshShortName Property</a>	Sets or returns whether short names are updated.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">Unique Property</a>	Sets or returns whether Transformer can identify each category in the level by a unique source value.
<a href="#">UniqueMove Property</a>	Sets or returns how a unique Level object is treated when the related category is moved.

## Examples

```
objTimeDimension = objModel.Dimensions.Item("Time")objLevel
= objTimeDimension.DrillDowns.Item(1).Levels.Item("Month")
```

## Measure Object

The Measure object represents quantitative values in a cube.

### Discussion

Use a measure to identify sets of numeric values in a model. Measures are quantifiable and supply performance indicators in PowerPlay.

You can create three types of measure object: regular, calculated, and category count. Each type of measure generates values in a different way. A regular measure uses an Association object. A calculated measure uses an expression defined in the ExpressionText property. A category count measure uses the setting for the CategoryCountLevel and ActivityMeasure properties.

A measure created by Transformer is always a regular measure. To create a calculated measure or category count measure, use the Add method of the Measures collection. Then use the CategoryCountLevel and ActivityMeasure properties to specify a category count measure, or the ExpressionText property to specify a calculated measure.

Measure values in lower levels are rolled up, or summarized, at the higher levels. Use the rollup properties of this object to specify how measure values are rolled up. For example, values may be summed, averaged or weighted.

In Transformer Series 7 Version 4 and subsequent releases, you can specify that null and missing values be ignored when Average or Weighted Average time state measures are rolled up. If missing values are represented as 'NA', set the IgnoreMissingValue property for the objMeasure object to TRUE, when you create or update the definition for a supported measure type.

**Note:** You must retain the default setting for First Period, Last Period, and Current Period. Null and missing values cannot be excluded from the rollup calculations for these measure types. Missing (null) data values are always excluded from Min and Max calculations for rollups, whether they are set by Transformer to display as '0' or 'n/a' (the NA setting).

When you have multiple data sources, you can use the SetAllocation method to define how measure values associated with one data source are viewed in categories associated with another data source.

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">AddToFolder Method</a>	Adds a measure to a measure folder.
<a href="#">AssociateWith Method</a>	Associates a source object with a Measure object.
<a href="#">Delete Method</a>	Deletes the Measure object.
<a href="#">Move Method</a>	Moves the measure to a different position in the Measures collection.
<a href="#">RemoveFromFolder Method</a>	Removes a child measure from the current measure folder.
<a href="#">Update Method</a>	Updates the Measure object.

Property	Description
<a href="#">ActivityMeasure Property</a>	Sets or returns the measure used by a category count measure.
<a href="#">AllowCurrencyConversion Property</a>	Sets or returns whether you can change a currency.
<a href="#">AllowDrillThrough Property</a>	Sets or returns whether a cube or measure can drill through to a cube or report.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Associations Property</a>	Returns an Associations collection.
<a href="#">CategoryCountLevel Property</a>	Sets or returns the Level object to which a category count applies.

Property	Description
<a href="#">ChildMeasures Property</a>	Returns a collection of child measures from a measure folder. An empty collection is returned if a measure folder does not contain any child measures. An exception is thrown if the measure is not a measure folder.
<a href="#">Description Property</a>	Sets or returns the description of the Measure object.
<a href="#">DrillThroughTargets Property</a>	Returns a collection of drill-through target objects associated with a Transformer model.
<a href="#">DuplicateRollup Property</a>	Sets or returns how duplicate measure values from consolidated records are rolled up.
<a href="#">DuplicateWeight Property</a>	Sets or returns the name of the measure that contains average weighting factors.
<a href="#">ExpressionText Property</a>	Sets or returns the contents of an expression that defines a value for a calculated measure. In the current release, if-then-else conditional expressions are now supported.
<a href="#">Format Property</a>	Sets or returns how numeric values appear.
<a href="#">FormatDecimals Property</a>	Sets or returns the number of decimal places PowerPlay displays for the measure.
<a href="#">IgnoreMissingValue Property</a>	Specifies whether to ignore null or missing values in a time state rollup. Default: false.
<a href="#">IsExpressionValid Property</a>	Returns whether an expression is valid.
<a href="#">IsFolder Property</a>	Sets or returns whether a measure is a measure folder.
<a href="#">Label Property</a>	Sets or returns a descriptive name that appears in PowerPlay.
<a href="#">MeasureType Property</a>	Returns whether a measure is regular, calculated, or a category count.
<a href="#">MissingValue Property</a>	Sets or returns what to show in place of a blank or null value.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">OutputScale Property</a>	Sets or a returns a scale value used to convert numbers from integer values to decimal values in PowerPlay.
<a href="#">Parent Property</a>	Returns the parent object.

Property	Description
<a href="#">Precision Property</a>	Sets or returns the number of decimal places for measures used in calculations.
<a href="#">RegularRollup Property</a>	Sets or returns the current regular rollup function for the Measure object.
<a href="#">RegularWeight Property</a>	Sets or returns a measure name used in a weighted average calculation.
<a href="#">ReverseSign Property</a>	Sets or returns whether PowerPlay reverses the sign of a measure.
<a href="#">RollupTiming Property</a>	Sets or returns when to perform calculations for calculated measures.
<a href="#">ShortName Property</a>	Sets or returns a short name for the Measure object.
<a href="#">StorageType Property</a>	Sets or returns the size of a numeric data type.
<a href="#">TimeStateRollup Property</a>	Sets or returns the date period used for time state rollups.
<a href="#">TimeStateWeight Property</a>	Sets or returns a measure name used in a weighted average calculation.
<a href="#">Type Property</a>	Returns the object type.

### Examples

```
objMeasure = objModel.Measures("Forecast")
```

## Model Object

The Model object defines a Transformer model.

### Discussion

A model is the highest level object in an Application object. From it, you directly open all the top-level collections.

You can build a new model and PowerCube by following these steps:

### Related Topics

These tables list related collections, methods, and properties.

Collection	Description
<a href="#">Namespaces Collection</a>	Contains all Namespace objects in a model.
<a href="#">SecurityObjects Collection</a>	Contains all SecurityObject objects in a model.

Method	Description
<a href="#">CheckLocalPowerCubes Method</a>	Checks the cubes defined in the model against their associated PowerCube files (.mdc).
<a href="#">CheckModel Method</a>	Executes a manual model check.
<a href="#">CleanHouse Method</a>	Removes inactive categories from the model.
<a href="#">Close Method</a>	Closes the model.
<a href="#">CreateMDCFiles Method</a>	Creates PowerCube files (.mdc) based on all cube and cube group definitions in the model.
<a href="#">DeleteAllCustomViews Method</a>	Deletes all custom views for the model.
<a href="#">DeleteAllSecurityObjects Method</a>	Removes security objects from the model.
<a href="#">DeployCubes Method</a>	Deploys all model PowerCubes to all deployment locations.
<a href="#">DoAutoDesign Method</a>	Generates dimensions, levels, drill-down paths, cubes, and measures.
<a href="#">GenerateCategories Method</a>	Populates a model with categories.
<a href="#">LoadCurrencyTable Method</a>	Loads a currency table into the model.
<a href="#">ResetPartitions Method</a>	Removes current cube partitions.
<a href="#">Save Method</a>	Saves the model.
<a href="#">SaveAs Method</a>	Saves the model with a new name.
<a href="#">TestBuild Method</a>	Creates a small test model or cube.
<a href="#">Update Method</a>	Updates the Model object.

Property	Description
<a href="#">CubeCodePage Property</a>	Sets or returns the cube code page setting for the model used to build the cube.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Cubes Property</a>	Returns a collection of Cube and CubeGroup objects.
<a href="#">CurrencyRecords Property</a>	Returns a CurrencyRecords collection.
<a href="#">CurrencyTables Property</a>	Returns a CurrencyTables collection.
<a href="#">DataSources Property</a>	Returns a DataSources collection.

Property	Description
<a href="#">DateWizard Property</a>	Returns the DateWizard object.
<a href="#">DefaultCategoryOrderBy Property</a>	Sets the default sort order (ascending) for all categories in the model.
<a href="#">Description Property</a>	Sets or returns the description of the Model object.
<a href="#">Dimensions Property</a>	Returns a Dimensions collection.
<a href="#">FileName Property</a>	Returns the name of a model file as it appears in a Windows folder or Windows Explorer.
<a href="#">FullName Property</a>	Returns the name and location of a model file.
<a href="#">Measures Property</a>	Returns a Measures collection.
<a href="#">ModelType Property</a>	Returns the file extension of a model file as it appears in a Windows folder or Windows Explorer.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Namespaces Property</a>	Returns a Namespaces collection.
<a href="#">Packages Property</a>	Returns a Packages collection.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Path Property</a>	Returns the location of a model file.
<a href="#">SecurityObjects Property</a>	Returns the SecurityObjects collection.
<a href="#">ServerModelPath Property</a>	Sets or returns the location of the server for a model.
<a href="#">Signon Property</a>	Returns a Signons collection.
<a href="#">Size Property</a>	Specifies the model size in bytes.
<a href="#">Time Property</a>	Returns the time stamp of a model as it appears in a Windows folder or Windows Explorer.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
objModel = objTransApp.OpenModel(strModelPath)
```

## Name Object

The Name object references a text string representing a name or message.

## Discussion

A Names collection is returned by the CheckModel method (Model object).

Each Name object in this collection contains a complete message string. You can use the return value of the Name property to read the results of a CheckModel procedure.

## Related Topics

These tables list related collections, methods, and properties.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Name Property</a>	Returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
MsgBox(objModel.CheckModel(intX).Name)
```

## Namespace Object

The Namespace object represents a namespace in the Transformer model.

## Discussion

A Namespace collection is returned by the Namespaces property in a Model object.

Each Namespace object in a collection references a namespace in the Transformer model.

Namespaces are used to create SecurityObjects. SecurityObjects are added to CustomViews to provide user and group-level security in a Transformer PowerCube.

Before a Namespace object can be used, you must log on to the applicable namespace using the Application object's Logon method. The user must provide a namespace, username, and password.

To use a Namespace object either the Name property needs to be set to the namespace or the ID property needs to be set to the namespace ID. If security is being applied to a user, the User property must also be set to the username of the user. If security is being applied to another object, such as a group or role, the ObjectName property must be set to the object name. After setting these properties appropriately, other properties can be used to retrieve security information.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the Namespace object.
<a href="#">Update Method</a>	Updates the Namespace object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">CAMID Property</a>	Returns the CAMID of the namespace.
<a href="#">ID Property</a>	Sets or returns the namespace ID.
<a href="#">Name Property</a>	Returns the name of the namespace.
<a href="#">ObjectCAMID Property</a>	Returns the CAMID of the object in the namespace set by the ObjectName property.
<a href="#">ObjectName Property</a>	Sets or returns the name of a namespace object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">SecurityObjects Property</a>	Returns a collection of SecurityObjects.
<a href="#">User Property</a>	Sets or returns a username associated with the namespace.
<a href="#">UserCAMID Property</a>	Returns the CAMID of the user set by the User property.

## Examples

```
new_namespace = model.Namespaces.Add()
```

## Package Object

The Package object represents a package data source in the Transformer model.

### Discussion

A Package collection is returned by the Packages property of the Model object. Each Package object in the collection references a package data source in the Transformer model.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the Package object.
<a href="#">Update Method</a>	Updates the Package object.
<a href="#">Verify Method</a>	Verifies the Package object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.



Property	Description
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">PackagesDatasourceConnections Property</a>	Returns the collection of PackageDatasourceConnection objects associated with the package.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Path Property</a>	Sets or returns the path to the package in IBM Cognos Connection.
<a href="#">Queries Property</a>	Returns the collection of Query objects associated with the package.
<a href="#">TimeStamp Property</a>	Sets or returns the time stamp of the package.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
new_package = model.Packages.Add()
```

## PackageDatasourceConnection Object

The PackageDatasourceConnection object represents a package data source connection in the Transformer model.

## Discussion

When a model is based on a data source that has multiple connections or multiple signons, an ambiguity occurs that must be resolved. The system checks the model to find a package data source connection that will resolve the ambiguity. The data source, connection, and signon belonging to the data source connection are used in such cases.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the PackageDatasourceConnection object.
<a href="#">Update Method</a>	Updates the PackageDatasourceConnection object.
<a href="#">Verify Method</a>	Verifies the PackageDatasourceConnection object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.

Property	Description
<a href="#">AlwaysUseTransformerSignon Property</a>	When true, the system uses the Transformer signon before using the Content Manager signon. When false, the Content Manager signon is used by default.
<a href="#">Connection Property</a>	Sets or returns the Content Manager connection.
<a href="#">DataSource Property</a>	Sets or returns the Content Manager data source.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Signon Property</a>	Sets or returns the Content Manager signon.
<a href="#">TransformerSignon Property</a>	Sets or returns the Transformer signon object associated with a package data source connection, which is the IBM Cognos signon.

## Examples

```
package = model.Packages.Add()connection
= package.PackageDatasourceConnections.Add()
```

## Prompt Object

The Prompt object represents a prompt in the Transformer model.

### Discussion

A Prompts collection is returned by the Prompts property (Query object). Each Prompt object in the collection references a prompt in the Transformer model.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the Query object.
<a href="#">Update Method</a>	Updates the Query object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">CurrentValueIndex Property</a>	Sets or returns the current prompt value index. This property is used to iterate through prompt values.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.

Property	Description
<a href="#">PromptValueType Property</a>	Sets or returns the type of prompt.
<a href="#">Type Property</a>	Sets or returns the data type of the prompt value or values.
<a href="#">Value Property</a>	Sets or returns the prompt value.
<a href="#">ValuesCount Property</a>	Returns the number of values set for the prompt.

## Examples

```
new_report = model.Reports.Add()
new_query = new_report.Queries.Add()
new_prompt = new_query.Prompts.Add()
```

## Query Object

The Query object represents a query that is part of a report or package.

### Discussion

A Query collection is returned by the Queries property of the Package or Report object. Each Package and Report object contains query objects. These query objects contain column objects that reference columns in the underlying package or report data source. To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the Query object.
<a href="#">Update Method</a>	Updates the Query object.
<a href="#">Verify Method</a>	Verifies the Query object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">AutoSummary Property</a>	Sets or returns the auto-summary option for a query.
<a href="#">Columns Property</a>	Returns a Columns collection that contains all the Column objects in a data source.
<a href="#">Filters Property</a>	Returns a filter object.
<a href="#">GenerateCategories Property</a>	Sets or returns whether categories are generated for the data source.

Property	Description
<a href="#">GeneratePowerCube Property</a>	Sets or returns when a data source is referenced by a model.
<a href="#">IsAnyColumnMismatched Property</a>	Returns whether columns in the data source match the underlying data.
<a href="#">MaximizeSpeed Property</a>	Sets or returns whether category generation is optimized for speed.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Prompts Property</a>	Returns the collection of Prompt objects associated with a Query.
<a href="#">SetsCurrentPeriod Property</a>	Sets or returns whether Transformer searches a data source to find the current period date.
<a href="#">StreamExtractAllowed Property</a>	Sets or returns whether stream extraction is allowed. (SAP BW only).
<a href="#">StreamExtractSize Property</a>	Sets or returns the size of the buffer, in megabytes, used to transfer data from SAP BW when StreamExtract is set to true.
<a href="#">SuppressNull Property</a>	Sets or returns the null suppression option used for SAP BW data sources.

## Examples

```
new_package = model.Packages.Add()new_query
= new_package.Queries.Add()
```

## Report Object

The Report object represents a report data source in the Transformer model.

### Discussion

A Reports collection is returned by the Reports property (Model object). Each Report object in the collection references a report data source in the Transformer model.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the Report object.
<a href="#">Update Method</a>	Updates the Report object.

Method	Description
<a href="#">Verify Method</a>	Verifies the Query object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">PackagesDatasourceConnections Property</a>	Returns the collection of PackageDatasourceConnection objects associated with the report.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Path Property</a>	Returns the path to the report in IBM Cognos Connection.
<a href="#">Queries Property</a>	Returns the collection of Query objects associated with the report.
<a href="#">TimeStamp Property</a>	Sets or returns the timestamp of the report.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
new_report = model.Reports.Add()
```

## SecurityObject Object

The SecurityObject object represents a security object in the Transformer model.

### Discussion

A SecurityObjects collection is returned by the SecurityObjects property in a Namespace object. Each SecurityObject object in the collection references a security object in the Transformer model.

Namespaces are used to create SecurityObjects. SecurityObjects are added to CustomViews to provide user and group-level security in a Transformer PowerCube. The SecurityObject's Name property is set to a CAMID of a user, group, or role object. The Type property must be set to a xtrSecurityType constant.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">AddToCustomView Method</a>	Adds a security object to a custom view.
<a href="#">Delete Method</a>	Deletes the SecurityObject object.
<a href="#">Update Method</a>	Updates the SecurityObject object.

Method	Description
<a href="#">Verify Method</a>	Verifies the SecurityObject object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">DisplayName Property</a>	Sets or returns the name to display for the SecurityObject object.
<a href="#">Name Property</a>	Sets or returns the name (CAMID) of the SecurityObject object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
new_namespace = model.Namespaces.Add()
securityObject = new_namespace.SecurityObjects.Add()
```

## Signon Object

The Signon object defines the signon type which can be one of the following: data source or Cognos. The data source signon type, previously known as a database signon in IBM Cognos Series 7, applies to a database signon. The IBM Cognos signon is a security signon that is linked to a security namespace and is used exclusively in IBM Cognos.

There is only one Signon object. The SignonType property for a Signon object specifies whether you are using a DatasourceSignon type or a CognosSignon type.

## Discussion

- Data Source Signon

Transformer uses the Name, Password, and User ID properties of a Signon object to automatically connect to a database.

- Impromptu Files

Impromptu Query Definition files (.iqd) reference database sources. These files use a logical database name to represent the connection parameters to an underlying data source. When you add an Impromptu Query Definition file to your model, Transformer automatically adds a Signon object that contains the logical database name. It may also contain user ID and password information.

After you return a Signon object from the Signons Collection, use it to edit the user ID and password.

Signon objects added by Transformer cannot be removed from the collection.

- IBM Cognos Signon

Transformer uses the CognosSignon type for authentication to a security namespace.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the Signon object.
<a href="#">Update Method</a>	Updates the Signon object.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">AutoLogon Property</a>	Defines whether the application will automatically authenticate to the security namespaces associated with the signon. Applies only to CognosSignon type.
<a href="#">Description Property</a>	Sets or returns a description for the object.
<a href="#">Name Property</a>	Returns the name of the Signon object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Password Property</a>	Sets a case-sensitive password for the Signon object.
<a href="#">PromptForPassword Property</a>	Defines whether the user will always be prompted for a password when using the Transformer user interface. Only applies to the DatasourceSignon type.
<a href="#">SignOnNamespace Property</a>	Contains the security namespace associated with the CognosSignon type.
<a href="#">SignonType Property</a>	Defines the type of signon, which in this case will be xtrSignonType.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">UserID Property</a>	Sets or returns the user ID for the Signon object.

## Examples

```
signon = model.Signons.Add()
```

## SpecialCategory Object

The SpecialCategory object defines a category with a value that is not directly from a data source.

### Discussion

Use this object to create a category based on relative time; for example, year-to-date.

A collection of SpecialCategory objects is returned by a Dimension object. To create a SpecialCategory object, use the Add method of the Categories collection, and set the Type parameter to trSpecialCategory.

To save changes to the properties of this object, use the Update method.

## Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">ConnectWithCategory Method</a>	Moves a child category to a new parent category in the same level.
<a href="#">Delete Method</a>	Deletes the SpecialCategory object.
<a href="#">SetAllocation Method</a>	Changes the allocation type for the measure used by the object.
<a href="#">Update Method</a>	Updates the SpecialCategory object.

Property	Description
<a href="#">Aggregate Property</a>	Sets or returns the type of relative time category.
<a href="#">AllocationMeasure Property</a>	Returns the Measure object used as a weighting factor.
<a href="#">AllocationType Property</a>	Returns how an object allocates a measure.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">AutoLogon Property</a>	Defines whether the application will automatically authenticate to the security namespaces associated with the signon. Applies only to the CognosSignon type.
<a href="#">CanAllocate Property</a>	Returns whether you can allocate any measure values to descendant levels and categories.
<a href="#">CanAllocateByMeasure Property</a>	Returns whether you can use the specified measure as a weighting factor when you allocate by measure.
<a href="#">CanAllocateMeasure Property</a>	Returns whether you can allocate the specified measure to descendant levels and categories.
<a href="#">ChildCategories Property</a>	Returns a Categories collection.
<a href="#">Code Property</a>	Sets or returns a unique code for the special category within a dimension.
<a href="#">ContextLevel Property</a>	Sets or returns a date period.
<a href="#">ContextOffset Property</a>	Sets or returns the position of the context period relative to the current period.
<a href="#">Description Property</a>	Sets or returns the description of the SpecialCategory object.



Property	Description
<a href="#">Dimensions Property</a>	Returns the dimension for a SpecialCategory object.
<a href="#">ExpressionText Property</a>	Sets or returns the contents of an expression that defines a value for the SpecialCategory object.
<a href="#">Format Property</a>	Sets or returns how numeric values appear.
<a href="#">Inclusion Property</a>	Sets or returns when a category is included in a cube.
<a href="#">IsExpressionValid Property</a>	Returns whether an expression is valid.
<a href="#">Label Property</a>	Sets or returns the name of the object. In the Transformer user interface, the label is "Source Value", however, the default value is ""].
<a href="#">LastUseDate Property</a>	Returns the date the SpecialCategory object was last modified or used.
<a href="#">Level Property</a>	Returns the level for a SpecialCategory.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">ParentCategories Property</a>	Returns a ParentCategories collection.
<a href="#">ReverseSign Property</a>	Sets or returns whether PowerPlay reverses the sign of a measure.
<a href="#">Rollup Property</a>	Sets or returns whether measure values for the special category roll up into the parent category.
<a href="#">RunningPeriods Property</a>	Sets or returns the number of time periods used for running totals.
<a href="#">ShortName Property</a>	Sets or returns a short name for the SpecialCategory object.
<a href="#">TargetLevel Property</a>	Sets or returns the level of detail of a date period.
<a href="#">TargetOffset Property</a>	Sets or returns the position of the target period relative to the current period.
<a href="#">ToDateLevel Property</a>	Sets or returns the date period used for to-date totals.
<a href="#">Type Property</a>	Returns the object type.

## Examples

```
objSpecCategory =  
- objModel.Dimensions("Time").Categories.Add(xtrObjectType.trSpecialCategory)
```

## SuspendedModel Object

The `SuspendedModel` object represents an existing incomplete model.

### Discussion

Use this object to recover a model that does not close correctly. (For example, in the case of a system crash or power failure.)

Transformer looks for suspended models in the location specified by either the `ModelTemporaryFilesPath` property or the `ModelsPath` property. If both properties are set, Transformer only checks the location specified by `ModelTemporaryFilesPath`. If neither are set, Transformer checks the Temp location specified by the environment variable.

Use the `IsBad` property to test whether the suspended model is corrupt or recoverable. If it is corrupt, use the `RemoveSuspendedModel` method to delete it from the `SuspendedModels` collection. If it is recoverable, you can open it in Transformer and determine the status.

Transformer retains as much information as possible in a `SuspendedModel` object. Transformer adds checkpoint entries after each major stage in the model creation process. These checkpoints are written to a temporary file with a file extension beginning with 'qy?'. Because these temporary files are deleted when a model closes normally, the existence of a temporary model file indicates that Transformer terminated unexpectedly. (The file extension always includes a third character that varies.)

Transformer also writes messages to a log file (.log) that is stored in the same location as your models, or in a user-specified location. If Transformer is unable to automatically recover from the processing failure, or if you choose to ignore previous processing and begin again, you can read the log file to determine the cause of the failure.

### Related Topics

These tables list related collections, methods, and properties.

Property	Description
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">IsBad Property</a>	Returns whether the suspended model is corrupt or not.
<a href="#">ModelName Property</a>	Returns the name of the suspended model.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">QyPath Property</a>	Returns the path of a suspended model.
<a href="#">Type Property</a>	Returns the object type.

### Examples

```
objSuspendedModel = objTransApp.SuspendedModels(intX)
```

## View Object

The View object defines a partial view of a dimension.

### Discussion

Use a View object to present a subset of the information in a dimension to PowerPlay users.

There are dimension views and custom views. Both views offer all, some, or none of the categories and levels in a dimension.

- Dimension View

Each Dimension object automatically includes a collection of two View objects. By default, the DimensionView property is set to the first object in the collection, All Categories. To omit a dimension from a cube, set the DimensionView property to the second object in the collection, Omit Dimension.

To create a custom view for a dimension, first use the Add method to add a View object to the collection, and then set the ViewType property to trViewTypeCustom. You can then use the Apex property to create a new root category, or the SetViewStatus method to associate the View object with selected levels or categories. After you define the View object, use the DimensionView property to associate the View object with the cube.

- Custom View

When you include security in a model, each CustomView object automatically includes a collection of View objects: one for each dimension. The default view for each object in a collection is 'All Categories'. To change this view, you can use the DimensionInclude property to omit all categories or to specify a custom view.

If you use the DimensionInclude property to specify a custom view, you must then use the DimensionView property to return the View object associated with a Dimension. You can then use the Apex property to create a new root category, or the SetViewStatus method to associate the View object with selected levels or categories.

To complete a custom view, you must associate a custom view with a cube. To do this, use the Add method to add a CustomView object to the CubeCustomViews collection of the cube.

To save changes to the properties of this object, use the Update method.

### Related Topics

These tables list related collections, methods, and properties.

Method	Description
<a href="#">Delete Method</a>	Deletes the View object.
<a href="#">SetViewStatus Method</a>	Sets how a Category, SpecialCategory, Level or DateLevel object is viewed.
<a href="#">Update Method</a>	Updates the View object.

Property	Description
<a href="#">Apex Property</a>	Sets or returns the Category object that serves as the root for a dimension view.
<a href="#">Application Property</a>	Returns the Transformer Application object.

Property	Description
<a href="#">CustomViews Property</a>	Set or returns the custom view associated with a view.
<a href="#">Name Property</a>	Sets or returns the name of the object.
<a href="#">Parent Property</a>	Returns the parent object.
<a href="#">Type Property</a>	Returns the object type.
<a href="#">ViewType Property</a>	Sets or returns whether a view contains all, some, or none of the categories in a dimension.

## Examples

```
objDimension = objModel.Dimensions("Retailers")objView
= objDimension.Views.Add()
```

---

## Chapter 4. Methods

The following table lists all the Transformer OLE automation methods.

Method	Description
<a href="#">Add Method ()</a>	Adds an object to a collection.
<a href="#">Add Method (Categories)</a>	Adds a Category object or SpecialCategory object to a Categories collection.
<a href="#">Add Method (CustomViews)</a>	Adds a custom view to the list of users that can use a cube.
<a href="#">Add Method (DrillThroughTargets)</a>	Adds a new drill-through custom report to the Reports collection of a Cube, CubeGroup, or Measure object.
<a href="#">Add Method (Objects)</a>	Adds an object to a collection that contains objects of more than one type.
<a href="#">AddDeployLocation Method</a>	Adds a new deployment location for Copy and Activate.
<a href="#">AddToCustomView Method</a>	Adds a security object to a custom view.
<a href="#">AddToFolder Method</a>	Adds a measure to a measure folder.
<a href="#">AssociateWith Method</a>	Associates a source object with an object in the model and adds an Association object to the Associations collection.
<a href="#">CheckLocalPowerCubes Method</a>	Checks the cubes defined in the model against their associated PowerCube files (.mdc).
<a href="#">CheckModel Method</a>	Checks the current model to determine if the objects have any conflicts with the data source or with each other.
<a href="#">CleanHouse Method</a>	Removes inactive categories from a model or dimension.
<a href="#">ClearDeployLocations Method</a>	Removes all deployment locations.
<a href="#">Close Method</a>	Closes the active model.
<a href="#">ConnectWithCategory Method</a>	Moves a child category to a new parent category in the same level.
<a href="#">CreateAlternateDrillDown Method</a>	Creates an alternate drill-down path in a dimension and makes the level the convergence level.

Method	Description
<a href="#">CreateDateDimension Method</a>	Creates a DateDimension object based on the properties of the DateWizard object.
<a href="#">CreateMDCFile Method</a>	Creates PowerCube (.mdc) files for a single cube or all cubes in a cube group.
<a href="#">CreateMDCFiles Method</a>	Creates all PowerCube (.mdc) files in a model.
<a href="#">Delete Method</a>	Deletes an object.
<a href="#">DeleteAllCustomViews Method</a>	Deletes all custom views for a model.
<a href="#">DeleteAllSecurityObjects Method</a>	Removes security objects from a model.
<a href="#">DeployCube Method</a>	Deploys a PowerCube to all deployment locations.
<a href="#">DeployCubes Method</a>	Deploys all PowerCubes for a model to all deployment locations.
<a href="#">DimensionAssociateWith Method</a>	Associates a source object with the new DateDimension object and adds an Association object to the Associations collection.
<a href="#">DoAutoDesign Method</a>	Generates dimensions, levels, drill-down paths, cubes, and measures.
<a href="#">FindCategoryByCatCode Method</a>	Returns the category object that contains the specified category code string.
<a href="#">GenerateCategories Method</a>	Populates a model with categories.
<a href="#">GenerateDateCategories Method</a>	Populates a model with date categories.
<a href="#">GetDefaultCategory Method</a>	Returns the default category belonging to a dimension.
<a href="#">GetViewStatus Method</a>	Returns the view status from Category, SpecialCategory, Level or DateLevel objects.
<a href="#">IsExcludeDateDimension Method</a>	Returns whether a given Date Dimension is excluded.
<a href="#">IsExcludeDateLevel Method</a>	Returns whether the given Date Level is excluded.
<a href="#">IsExcludeDimension Method</a>	Returns whether a Dimension is restricted or not.
<a href="#">IsExcludeLevel Method</a>	Returns whether a Level is restricted or not.
<a href="#">Item Method</a>	Returns a specific object in a collection.
<a href="#">Item Method ()</a>	Returns a specific object in a collection.
<a href="#">Item Method ()</a>	Returns a specific object in a collection.

Method	Description
<a href="#">LoadCurrencyTable Method</a>	Loads a currency table into the model.
<a href="#">Logoff Method</a>	Logs off from all namespaces.
<a href="#">Logon Method</a>	Logs onto a namespace.
<a href="#">Move Method</a>	Rearranges objects in a collection.
<a href="#">MoveToCategory Method</a>	Moves a child category to a different parent category.
<a href="#">MoveToLevel Method</a>	Moves a child category to a new level under the current parent category.
<a href="#">NewModel Method</a>	Creates a new model.
<a href="#">OpenModel Method</a>	Opens an existing model.
<a href="#">OpenSuspendedModel Method</a>	Opens a suspended model.
<a href="#">PublishDatasource Method</a>	Publishes the datasource connection for a PowerCube.
<a href="#">PublishPackage Method</a>	Publishes both the datasource and package for a PowerCube.
<a href="#">RemoveCubeCustomView Method</a>	Removes the custom view from the cube or child cube.
<a href="#">Remove Method</a>	Removes a specific object from a collection.
<a href="#">RemoveFromFolder Method</a>	Removes a child measure from the current measure folder.
<a href="#">RemoveSuspendedModel Method</a>	Removes a suspended model.
<a href="#">ResetPartitions Method</a>	Removes current cube partitions.
<a href="#">Save Method</a>	Saves changes to the current model.
<a href="#">SaveAs Method</a>	Saves the current model to a different file name.
<a href="#">SetAllocation Method</a>	Changes the allocation type for a measure used by an object.
<a href="#">SetDefaultCategory Method</a>	Specifies a new default category for a dimension (other than the root).
<a href="#">SetDeployType Method</a>	Sets the deployment type for Copy and Activate.
<a href="#">SetExcludeDateDimension Method</a>	Sets the given date dimension to the isRestricted value.

Method	Description
<a href="#">SetExcludeDateLevel Method</a>	Sets the given date level to the isRestricted value.
<a href="#">SetExcludeDimension Method</a>	For a given Report and Dimension, sets the drill-through restriction to True if the isRestricted parameter is True.
<a href="#">SetExcludeLevel Method</a>	For a given Report and Level, sets the drill-through restriction to True if the isRestricted parameter is True.
<a href="#">SetViewStatus Method</a>	Returns the view status from Category, SpecialCategory, Level, or DateLevel objects.
<a href="#">TestBuild Method</a>	Creates a small test model or cube.
<a href="#">Update Method</a>	Updates the associated object.
<a href="#">Verify Method</a>	Verifies the associated object.

## Add Method ()

---

The Add method adds an object to a collection.

### Syntax

*collection* .Add()

### Applies To

[Associations Collection](#)

[CalculationDefinitions Collection](#)

[CategorySets Collection](#)

[Columns Collection](#)

[CurrencyRecords Collection](#)

[CurrencyTables Collection](#)

[CustomViews Collection](#)

[Filters Collection](#)

[PackageDatasourceConnections Collection](#)

[Packages Collection](#)

[Queries Collection](#)

[Measures Collection](#)

[Namespaces Collection](#)

[Reports Collection](#)

[SecurityObjects Collection](#)

[Signons Collection](#)



## Discussion

Use this method without parameters to add objects to their respective collections.

You can also use the `AssociateWith` and `DimensionAssociateWith` methods to add objects to an Associations collection.

## Return Type

Object

## Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()
```

## Add Method (Categories)

The Add method adds a Category object or SpecialCategory object to a Categories collection.

## Syntax

*Categories* .Add([Type] [, Level ])

## Applies To

[Categories Collection](#)

## Discussion

When you add an object to the Categories collection of a CategorySet object, the Type parameter must name a Category object or SpecialCategory object that already exists. For all other Categories collections, Type supplies a value of `xtrObjectType` or a variant that supplies an `xtrObjectType` value.

Use the optional Level parameter to specify which level the category belongs in.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Type	Optional. For most collections, specifies either the <code>trCategory</code> or <code>trSpecialCategory</code> constant of <code>xtrObjectType</code> . In the case of the Categories collection of a CategorySet, it names an existing object.  Type: Variant
Level	Optional. Specifies the name of the level where the category is to be added. If omitted, the category is added to the lowest level.  Type: Object

## Return Type

Object

## Examples

```
objSpecCategory =  
_ objModel.Dimensions("Time").Categories.Add(xtrObjectType.trSpecialCategory)
```

## Add Method (CustomViews)

---

The Add method adds a custom view to the list of users that can use a cube.

### Syntax

*CubeCustomViews* .**Add**(CustomView)

### Applies To

CubeCustomViews

### Discussion

Before using this property, create a CustomView object and set and update its properties.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
CustomView	Required. Specifies a custom view in the Transformer model.  Type: CustomView

## Return Type

CustomView

## Examples

```
cube = model.Cubes.Item("Sales and Marketing")cube.CubeCustomViews.Add(custom_view)
```

## Add Method (DrillThroughTargets)

---

The Add method (DrillThroughTargets) adds a new drill-through custom DrillThroughTarget to the DrillThroughTargets collection of a Cube, CubeGroup, ChildCube or Measure object.

### Syntax

*DrillThroughTargets* .**Add**(DrillThroughTargetName, Description)

### Applies To

[DrillThroughTargets Collection](#)

## Discussion

The DrillThroughTarget objects can represent Impromptu reports (.imr), Impromptu query definition files (.iqd), PowerPlay reports (.ppr), PowerCubes (.mdc), or macro script files (.mac).

When you add a DrillThroughTarget object to the DrillThroughTargets collection of a Measure object, you restrict the drill-through functionality to that measure alone. When you add a DrillThroughTarget object to the DrillThroughTargets collection of a Cube, ChildCube or CubeGroup object, the drill-through functionality is available at any point in a PowerPlay report. The DrillThroughTargetName parameter sets the DrillThroughTarget.Name property. A fully qualified file name must be given when setting the DrillThroughTarget.Name property. The same applies to the Description parameter. It gets set to the value passed as the Description parameter.

Parameter	Description
DrillThroughTargetName	Required. Specifies a fully qualified file name for the drill-through target. Type: String
Description	Required. Specifies a short textual description to explain the purpose of the drill-through target to the PowerPlay user. Type: String

## Return Type

Object - DrillThroughTarget

## Examples

```
objReport = objMeasure.DrillThroughTargets.Add(strReportPath,  
"Default Report")
```

## Add Method (Objects)

The Add method (Objects) adds an object to a collection that contains objects of more than one type.

## Syntax

*collection* .Add(Type)

## Applies To

[Cubes Collection](#)

[DataSources Collection](#)

[Dimensions Collection](#)

[Levels Collection](#)

## Discussion

Use when a collection contains objects of more than one type, such as in a DataSources collection. The Type parameter specifies a value from the xtrObjectType value list.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Type	Optional for DataSource objects. <b>Note:</b> If the DataSource type is not specified, then the default type is a FlatFileDataSource. Optional for other objects. Specifies a constant of the value list, xtrObjectType. Required when adding DateDimensions and when a type of trDateDimension must be used. Type: Variant

## Return Type

Object

## Examples

```
objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
```

## AddDeployLocation Method

---

The AddDeployLocation method adds a new deployment location for Copy and Activate.

## Syntax

*Cube* .**AddDeployLocation**(deployPath)

## Applies To

Cube Object

## Discussion

Use this method when setting up a PowerCube deployment. This method can be called multiple times to add multiple deployment locations to the Cube object.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
deployPath	Specifies a path to the directory where the PowerCube is to be deployed. Type: String

## Return Type

None

## Examples

```
cube = model.Cubes.Item("National")cube.AddDeployLocation("c:\NATIONAL\Deployment1")
```

## AddToCustomView Method

---

The AddToCustomView method adds a security object to a custom view.

### Syntax

*SecurityObject* .**AddToCustomView**

### Applies To

[SecurityObject Object](#)

### Discussion

Use when adding a SecurityObject object to a CustomView object. Use to add a user, group, or other security object to a custom view.

The CustomView parameter specifies the CustomView object that the security object is added to.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
CustomView	Required: Specifies the CustomView where the SecurityObject applies. Type: CustomView

### Return Type

None

## Examples

```
new_namespace = model.Namespaces.Add()securityObject  
= new_namespace.SecurityObjects.Add()securityObject.AddToCustomView(customView)
```

## AddToFolder Method

---

The AddToFolder method adds a measure to a measure folder.

### Syntax

*measure* .**AddToFolder**

### Applies To

[Measure Object](#)

## Discussion

You first set the IsFolder property to True to create a measure folder. Use the AddToFolder method to add a measure to that measure folder.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Measure	Required. Specifies the measure object to be added. Type: Object - Measure

## Return Type

None

## AssociateWith Method

---

The AssociateWith method associates a source object with an object in the model.

## Syntax

*object* .**AssociateWith Reference**, Role [, Context]

## Applies To

[CurrencyTable Object](#)

[DateDimension Object](#)

[DateLevel Object](#)

[Dimension Object](#)

[Level Object](#)

[Measure Object](#)

## Discussion

Use this method to associate an object in your model with an object in a data source. The Reference parameter performs the same function as the Reference property of an Association object. The Role parameter performs the same function as the AssociationRole property. The Context parameter performs the same function as the Context property.

When you use the AssociateWith method, it adds an Association object to the Associations collection of the object. For example, if the Associations collection of a Measure object already has two objects, the AssociateWith method adds a third object to the collection.

The AssociateWith method is an alternative to the Add method of the Associations collection.

When you use the AssociateWith method with an object, always follow with the Update method for the same object.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Reference	Required. Specifies the source of the association. Type: Object
Role	Required. Specifies how the reference is used. This parameter uses a constant of the value list xtrAssociationRole. Type: Constant
Context	Required for OrderBy Associations. Specifies the drill-down path in which a level is sorted. Applies to Level and DateLevel objects only. Type: Object

## Return Type

Object - Association

## Examples

```
objLocationsDim = objModel.Dimensions("Sales regions")objLevel
= objLocationsDim.DrillDowns(1).Levels("Employee")objAltDrill
= objLevel.CreateAlternateDrillDownobjNewLevel
= objAltDrill.Levels.Add(xtrObjectType.trLevel)objNewLevel.
AssociateWith(objRefSource, xtrAssociationRole.trAssociationSource)
```

## CheckLocalPowerCubes Method

The CheckLocalPowerCubes method checks the cubes defined in the model against their associated PowerCube files (.mdc).

## Syntax

*Model* .**CheckLocalPowerCubes**

## Applies To

Model Object

## Discussion

Use to check the status of cubes in the model and to change the status if required.

If the status is Warning, Invalid, Busy, or OK, this command checks for the existence of a valid PowerCube file for that cube. If Transformer cannot locate the PowerCube file, it sets the PowerCube status to Missing. If Transformer finds the cube but cannot open it, or if the file creation stamp does not agree with the one in the model (indicating that the cube was created from another model or from an obsolete version of the current model), Transformer sets the PowerCube status to Invalid.

## Return Type

None

## CheckModel Method

---

The CheckModel method checks the current model to determine if the objects have any conflicts with the data source or with each other.

### Syntax

*Model* .**CheckModel**

### Applies To

Model Object

### Discussion

Transformer always checks the model before generating categories or creating cubes, but you can check it at any time by using this method.

You should check your model at key stages in the design process, such as when new data sources or associations are added, and again after you create the cube definitions.

The CheckModel method returns a Names collection. Each Name object in this collection contains a complete message string. You can use the Name property to read the results of a CheckModel procedure.

### Return Type

Object - Names

### Examples

```
For intX = 1 To objModel.CheckModel.Count
```

## CleanHouse Method

---

The CleanHouse method removes inactive categories from a model or from a dimension.

### Syntax

*object* .**CleanHouse** Date

### Applies To

DateDimension Object

Dimension Object

Model Object

### Discussion

Use this method to check for inactive categories. As models are adapted to changes in your organization, they may retain categories that are no longer needed. A category is considered inactive if it was not created, updated, moved, or modified since a time specified by the Date parameter.

Transformer checks the LastUseDate property of each Category object during the cleaning operation and compares it to the value of the Date parameter. When the LastUseDate property contains a date that is older than the value of the Date parameter, Transformer removes the category.



When you use the CleanHouse method with a dimension, Transformer checks only the categories used by that dimension. When you use the CleanHouse method with a model, Transformer checks all the categories in the model.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Date	Required. Specifies the cutoff date for active categories in the form YYYYMMDD. Type: Long

### Return Type

None

## ClearDeployLocations Method

---

The ClearDeployLocations method removes all deployment locations.

### Syntax

*Cube*.ClearDeployLocations

### Applies To

[Cube Object](#)

### Discussion

Use this method to remove all deployment locations from the Cube object. This method can be used to prepare before or cleanup after a PowerCube deployment. In a cube group scenario, child cubes are also affected. This does not applies to time-partitioned cubes.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Return Type

None

### Examples

```
cube = model.Cubes.Item("National")cube.ClearDeployLocations()  
'Remove the deployment locations from the model'
```

## Close Method

---

The Close method closes the current model.

### Syntax

*Model*.Close

## Applies To

[Model Object](#)

## Discussion

Use this method to close the current model before you create or open another model.

If you close a model before you save it, you will lose all changes to the model including any updates to the objects you've made with the Update method.

## Return Type

None

## Examples

```
objModel.Close()
```

# ConnectWithCategory Method

---

The ConnectWithCategory method moves a child category to a new parent category in the same level.

## Syntax

*Category*.**ConnectWithCategory**(Parent [, DrillDown])

## Applies To

[Category Object](#)

## Discussion

You can also use the MoveToCategory and MoveToLevel methods to move Category and SpecialCategory objects to a new position or different collection.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Parent	Required. Specifies the parent category to connect to. Type: Object
DrillDown	Optional. Specifies a DateDrillDown or DrillDown object if the parent exists in more than one drill-down path. The value can be a numeric index position, a string giving an object name, or an object defined in the script. Type: Variant

## Return Type

None

## Examples

```
objDimension = objModel.Dimensions("Retailers")objCategories
= objDimension.DrillDowns(1).CategoriesobjParentCategory
= objCategories(parent_category_index)objCategories
= objDimension.DrillDowns(1).CategoriesobjCategory
= objCategories(5)objChildCategory
= objCategory.ChildCategories(1)objChildCategory.
ConnectWithCategory(objParentCategory)
```

## CreateAlternateDrillDown Method

---

The CreateAlternateDrillDown method creates an alternate drill-down path in a dimension and makes the level the convergence level.

### Syntax

*object* .**CreateAlternateDrillDown**

### Applies To

[DateLevel Object](#)

[Level Object](#)

### Discussion

Use this method to create an alternate drill-down path in a dimension. Apply it to the level that you want as the convergence level. Transformer then creates a drill-down path from the first level in the dimension to the convergence level. You then need to insert a level into the new drill-down path to connect to the convergence level and associate that new level with a source value. Because Transformer always adds objects to the end of a collection, you must move the new level to a position in the Levels collection before the convergence level.

The type of drill-down path created, either a DrillDown or DateDrillDown object, depends on the type of the level. The new drill-down path is added to the DrillDowns collection for that dimension.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Return Type

Object

## Examples

```
objLocationsDim = objModel.Dimensions("Sales regions")objLevel
= objLocationsDim.DrillDowns(1).Levels("Employee")objAltDrill
= objLevel.CreateAlternateDrillDown
```

## CreateDateDimension Method

---

The CreateDateDimension method creates a DateDimension object.

### Syntax

*DateWizard* .**CreateDateDimension**

## Applies To

[DateWizard Object](#)

## Discussion

After you set the applicable properties of the DateWizard object and assign a data source using the DimensionAssociateWith method, use this method to create the new DateDimension object.

This method returns a DateDimension object.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

Object - DateDimension

## Examples

```
objDateWizard = objModel.DateWizardobjDateWizard.DimensionAssociateWith  
(objColumn, xtrAssociationRole.trAssociationSource)
```

## CreateMDCFile Method

---

The CreateMDC File method creates PowerCube files(.mdc) for a single cube or for all cubes in a cube group.

## Syntax

*object* .CreateMDCFile

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

Use this method to create a PowerCube from a cube or cube group definition in your model.

Transformer automatically generates categories from columns in your data sources when you create a cube. Whether Transformer includes a category in the cube depends on the settings for various properties, such as Consolidate, Inclusion, and DuplicateRollup.

Before you use this method, you must ensure that the LocalPath and DataSourcePath properties are properly set.

Use the CreateMDCFiles method to create all cubes in a model.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

None

## Examples

```
objCube = objModel.Cubes.Item(1)objCube.CreateMDCFile()
```

## CreateMDCFiles Method

---

The CreateMDCFiles method creates all PowerCube files(.mdc) in a model.

### Syntax

*Model* .CreateMDCFiles

### Applies To

[Model Object](#)

### Discussion

Use this method to create a PowerCube for every cube and cube group in a model.

Transformer automatically generates categories from columns in your data sources when you create a cube. Whether Transformer includes a category in the cube depends on the settings for various properties, such as Consolidate, Inclusion, and DuplicateRollup.

Before you use this method, you must ensure that the LocalPath and DataSourcePath properties are properly set.

Use the CreateMDCFile method to create PowerCube files (.mdc) for a single cube or cube group.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Return Type

None

## Delete Method

---

The Delete method deletes an object from a model.

### Syntax

*object* .Delete()

### Applies To

The Delete method applies to the following objects:

[Association Object](#)

[CalculationDefinition Object](#)

[Category Object](#)

[CategorySet Object](#)

[Column Object](#)

[CrossTabDataSource Object](#)

[“Cube Object” on page 51](#)

[CubeGroup Object](#)

[CurrencyRate Object](#)

[CurrencyRecord Object](#)

[CurrencyTable Object](#)

[DateDimension Object](#)

[DateDrillDown Object](#)

[DataSource Object](#)

[DateLevel Object](#)

[DbDataSource Object](#)

[Dimension Object](#)

[DrillDown Object](#)

[DrillThroughTarget Object](#)

[Filter Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

[Level Object](#)

[Measure Object](#)

[Namespace Object](#)

[PackageDatasourceConnection Object](#)

[Package Object](#)

[Prompt Object](#)

[Query Object](#)

[Report Object](#)

[SecurityObject Object](#)

[Signon Object](#)

[SpecialCategory Object](#)

[View Object](#)

## **Discussion**

Use this method to remove an object from the model when the object exists outside of a collection. Use the Remove method to remove an object from the collection.

You cannot delete the default primary drill-down path created by Transformer for each dimension.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## **Return Type**

None

## Examples

```
objLevel  
= objTimeDimension.DrillDowns.Item(1).Levels.Item("Month")sobjLevel.Delete()
```

## DeleteAllCustomViews Method

---

The DeleteAllCustomViews method deletes all custom views for a model.

### Syntax

*Model* .DeleteAllCustomViews()

### Applies To

[Model Object](#)

### Discussion

The assigned security objects are also removed when custom views are removed.

### Return Type

None

## DeleteAllSecurityObjects Method

---

The DeleteAllSecurityObjects method removes the security objects from a model.

### Syntax

*Model* .DeleteAllSecurityObjects()

### Applies To

[Model Object](#)

### Discussion

When custom views are removed, the assigned security objects to the view are also removed.

### Return Type

None

## DeployCube Method

---

The DeployCube method deploys a PowerCube to all deployment locations.

### Syntax

*Cube* .DeployCube()

## Applies To

[Cube Object](#)

## Discussion

Use this method to deploy all PowerCube. Call PublishDatasource or PublishPackage after deploying the PowerCube.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

None

## Examples

```
cube = model.Cubes.Item("National")cube.DeployCube()
```

# DeployCubes Method

---

The DeployCubes method deploys all PowerCubes for a model to all deployment locations.

## Syntax

*Model* .**DeployCubes**()

## Applies To

[Model Object](#)

## Discussion

Use this method to deploy all PowerCubes for a model. Call PublishDatasource or PublishPackage after deploying the PowerCubes.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

Boolean value

# DimensionAssociateWith Method

---

The DimensionAssociateWith method associates a source object with a new DateDimension object and adds an Association object to the Associations collection.

## Syntax

*DateWizard* .**DimensionAssociateWith** Reference, Role [, Context]

## Applies To

[DateWizard Object](#)



## Discussion

Use this method to associate a DateDimension object with an object in a data source. The Reference parameter performs the same function as the Reference property of an Association object. The Role parameter performs the same function as the AssociationRole property. The Context parameter performs the same function as the Context property.

When you use the DimensionAssociateWith method, it adds an Association object to the Associations collection of the DateDimension object.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Reference	Required. Specifies the source of the association. Type: Object
Role	Required. Specifies how the reference is used. This parameter uses a constant of the value list xtrAssociationRole. Type: Constant - xtrAssociationRole
Context	Required for OrderBy Associations. Specifies the drill-down path in which a level is sorted. Applies to Level and DateLevel objects only. Type: Object

## Return Type

None

## Examples

```
objDateWizard = objModel.DateWizardobjDateWizard.DimensionAssociateWith  
(objColumn, xtrAssociationRole.trAssociationSource)
```

## DoAutoDesign Method

The DoAutoDesign method generates dimensions, levels, drill-down paths, cubes, and measures.

## Syntax

*Model* .DoAutoDesign

## Applies To

Model Object

## Discussion

Use this method to generate the principal collections of a new model or to add dimensions and levels to an existing model based on columns not used earlier.

This method analyzes the data types, column names, and structural framework of the data sources. It places dates in a time dimension, columns with numerical values in the Measures collection, and all remaining columns in Dimension objects. The new dimensions contain a Levels and DrillDowns collection.

## Return Type

None

## Examples

```
objModel = objTransApp.NewModel objModel.DoAutoDesign()
```

## FindCategoryByCatCode Method

---

The FindCategoryByCatCode method returns the category object that contains the specified category code string.

### Syntax

*Dimension* .**FindCategoryByCatCode** CatCode

### Applies To

[Dimension Object](#)

### Discussion

Use this method to return the category object that contains the specified category code.

If the method does not find a category with the given category code, it returns nothing. Using this method to return a category is faster than accessing the category through the category collection.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
CatCode	Required. Specifies the associated category. <b>Type:</b> String

## Return Type

Category

## GenerateCategories Method

---

The GenerateCategories method populates a model with categories.

### Syntax

*Model* .**GenerateCategories**

### Applies To

[Model Object](#)

### Discussion

Use this method to generate categories from your data source after you have used the DoAutoDesign method or have manually added dimensions and levels to your model. While it generates categories,

Transformer reads and analyzes your data source and builds the category hierarchy. Categories are automatically generated when you create a cube.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

None

## Examples

```
objModel.GenerateCategories()
```

# GenerateDateCategories Method

The GenerateDateCategories method populates a model with date categories specified by EarliestDate and LatestDate parameters.

## Syntax

*DateDimension* .**GenerateDateCategories()**

## Applies To

[DateDimension Object](#)

## Discussion

Use this method to generate date categories from your data source after creating the DateDimension object, and the associated DrillDown, with the EarliestDate and LatestDate representing the range of the generated date categories. Use this method after you have used the DoAutoDesign method or have manually added dimensions and levels to your model. While it generates categories, Transformer reads and analyzes your data source and builds the category hierarchy. Categories are automatically generated when you create a cube.

A COM exception is thrown in error situations. The message that is passed with the exception is TR0821.

## Return Type

None

Parameter	Description
EarliestDate	Required. Specifies the first date category. This value must be in the date format YYYYMMDD. If this value is not a date value, or is a date earlier than the Date Dimension EarliestDate, then the saved model will be invalid. Type: Long
LatestDate	Required. Specifies the last date category. This value must be in the date format YYYYMMDD. Type: Long

## Examples

```
objDateDim  
= objModel.Dimensions("Date")objDateDim.GenerateDateCategories(19990101, 20101231)
```

## GetDefaultCategory Method

---

The GetDefaultCategory method returns the default category belonging to a dimension. This category is used when the crosstab report is opened for the first time.

### Syntax

*DateDimension* .**GetDefaultCategory**

### Applies To

[DateDimension Object](#)

[Dimension Object](#)

### Discussion

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Return Type

A category object.

Parameter	Description
None	

## GetViewStatus Method

---

The GetViewStatus method returns the view status of a Category, SpecialCategory, DateLevel or Level object.

### Syntax

*View* .**GetViewStatus**

### Applies To

[View Object](#)

### Discussion

Once a View object is obtained, it can be used to retrieve the view status of an associated Category, SpecialCategory, DateLevel or Level object. The constant returned is one of the possible values of the xtrViewStatus Value List.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

Constant - xtrViewStatus

Parameter	Description
Object	Required. This method can only be applied to either a Category, SpecialCategory, DateLevel, or Level object.  <b>Type:</b> Object

## Examples

```
objDimension = objModel.Dimensions("Retailers")objView  
= objDimension.Views.Add()If objView.GetViewStatus(objCategory)  
<> _xtrViewStatus.trViewStatusSummaryMom Then
```

## IsExcludeDateDimension Method

---

The isExcludeDateLevel method returns whether a given Date Dimension is excluded.

### Syntax

*DrillThroughTarget* .**isExcludeDateDimension**

### Applies To

[DrillThroughTarget](#) Object

### Discussion

Use this property to retrieve whether the given Date Dimension is restricted from the drill-through target.

Parameter	Description
DateDimension	Required. Specifies the Date Dimension that the property applies to.

## Return Type

String

### Access

Read/Write

## IsExcludeDateLevel Method

---

The isExcludeDateLevel method returns whether a given Date Level is excluded.

### Syntax

*DrillThroughTarget* .**isExcludeDateLevel**

## Applies To

[DrillThroughTarget Object](#)

## Discussion

Use this property to retrieve whether the given Date Level is restricted from the drill-through target.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
DateLevel	Required. Specifies the Date Level that the property applies to. Type: DateLevel

## Return Type

Boolean value

## Access

Read/Write

## IsExcludeDimension Method

---

The isExcludeDimension method returns whether a Dimension is excluded.

## Syntax

*DrillThroughTarget* .**isExcludeDimension**

## Applies To

[DrillThroughTarget Object](#)

## Discussion

Use this property to retrieve whether the given Dimension is restricted from the drill-through target.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Level Object	Required. Specifies the Dimension that the property applies to. Type: Object - Level

## Type

String

## Access

Read/Write

## IsExcludeLevel Method

---

For a given Report, isExcludeLevel method specifies whether a Level is excluded.

### Syntax

*DrillThroughTarget* .**isExcludeLevel**

### Applies To

[DrillThroughTarget Object](#)

### Discussion

Use this property to retrieve if the given Level is restricted from the drill-through target.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Level	Required. Specifies the Level that the property applies to.

### Type

String

### Access

Read/Write

## Item Method

---

The Item method returns a specific object in a collection.

### Syntax

*collection* .**Item**(Key, [Type])

### Applies To

[Categories Collection](#)

[Cubes Collection](#)

[DataSources Collection](#)

[Dimensions Collection](#)

[DrillDowns Collection](#)

[Levels Collection](#)

[PackageDatasourceConnections Collection](#)

[Packages Collection](#)

[Queries Collection](#)

[Reports Collection](#)

## Discussion

The Type parameter modifies the Key parameter where a collection can contain more than one type of object. If you use Key only, the Item method counts from the start of the collection until it reaches the specified object or the end of the collection. When you add the optional Type parameter, the Item method counts that type of object only. This way, the Key and Type parameters let you select a specific object type in a specific position in the collection.

For example, the Dimensions collection can have both Dimension and DateDimension objects. To get the fifth object in the collection regardless of type, use a statement such as the one shown here. Note that this example applies to VB.NET, but not C#. In C#, both parameters must be provided regardless of whether the type is used.

```
Set objDim = objDimColl.Item(5)
```

```
Dimension dim = (Dimension)model.Dimensions.Item(5,null);
```

To get the third occurrence of a DateDimension object in a collection of Dimension and DateDimension objects, use a statement such as this:

```
Set objDim = objDimColl.Item(3, "trDateDimension")
```

Key can also be a string naming the object. Therefore, to get a DateDimension object named Order Dates, use a statement such as this:

```
Set objDim = objDimColl.Item("Order Dates")
```

The Type parameter applies only to objects in the Categories, Cubes, DataSources, and Dimensions collections. When iterating through all items in a collection that contains more than one type of object, the Type parameter must be set to null.

The Item method can be implied if the Type parameter is not used. For example, this statement, which applies only to VB.NET, has the same effect:

```
objModel.Dimensions("Dates").DrillDowns(1).Levels("Month")
```

**Note:** Beginning with version 7.0, the Item method is case-sensitive.

For example the following example will only match a dimension named 'Line', and not 'line':

```
objModel.Dimension("Line")
```

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Key	Required. Specifies a numeric index position starting at 1 or a string naming the object to retrieve. <b>Type:</b> Variant



Parameter	Description
Type	Optional. Specifies a constant of the value list xtrObjectType. Applies to Categories, Cubes, DataSources, Dimensions, DimensionLevels, DrillDowns, and Levels collections only. <b>Type:</b> Constant

## Return Type

Object

## Examples

```
objDimensions = objModel.DimensionsobjLocationsDim
= objDimensions.Item(3)objLocationsDim
= objDimensions.Item(3)
```

## Item Method ()

---

The Item method returns a specific object in a collection.

## Syntax

*collection* .**Item**(Key)

## Applies To

[Associations Collection](#)

[CalculationDefinitions Collection](#)

[CategorySets Collection](#)

[ChildCubes Collection](#)

[Columns Collection](#)

[CubeCustomViews Collection](#)

[Cubes Collection](#)

[CurrencyRates Collection](#)

[CurrencyRecords Collection](#)

[CurrencyTables Collection](#)

[CustomViews Collection](#)

[DrillThroughTargets Collection](#)

[Filters Collection](#)

[LevelDrillDowns Collection](#)

[Measures Collection](#)

[Names Collection](#)

[Namespaces Collection](#)

[Prompts Collection](#)

[SecurityObjects Collection](#)

[Signons Collection](#)

[SuspendedModels Collection](#)

[Views Collection](#)

## Discussion

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Key	Required. Specifies a numeric index position starting at 1 or a string naming the object to retrieve. A string naming the object to retrieve does not work for associations. <b>Type:</b> Variant

## Return Type

Object

## Item Method ()

---

The Item method returns a specific object in a collection.

## Syntax

*collection* .**Item**(Key, Type)

## Applies To

[DimensionLevels Collection](#)

[LevelCategories Collection](#)

## Discussion

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Key	Required. Specifies a numeric index position starting at 1 or a string naming the object to retrieve. A string naming the object to retrieve does not work for associations. <b>Type:</b> Variant

Parameter	Description
Type	Required. Specifies a constant of the value list xtrObjectType. Applies to Categories, Cubes, DataSources, Dimensions, DimensionLevels, DrillDowns, and Levels collections only. In some implementation languages, if this value is not needed it must be set to null as shown above.  <b>Type:</b> Constant

## Return Type

Object

## LoadCurrencyTable Method

---

The LoadCurrencyTable method loads a currency table into the model.

### Syntax

*Model* .LoadCurrencyTable

### Applies To

[Model Object](#)

### Discussion

Use this method to load a currency table with information from an external data source. To use this method, you must have at least one currency table in the CurrencyTables collection.

If you add a euro currency table to the CurrencyTables collection, you must use the LoadCurrencyTable method at separate stages in the process. Use this method after you associate each data source column with an association role in the euro table and after you add currency records to the euro table.

## Return Type

None

### Examples

```
objModel.LoadCurrencyTable()
```

## Logoff Method

---

The Logoff method logs off from all namespaces.

### Syntax

*Application* .Logoff()

### Applies To

[Application Object](#)

## Discussion

Use this property to log off from the current namespaces. Use with the Logoff method.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

None

## Logon Method

---

The Logon method logs onto a namespaces.

## Syntax

*Application* .**Logon**(Namespace, Username, Password)

## Applies To

[Application Object](#)

## Discussion

Use this property to log onto a namespace. Use with the Logoff method.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Namespace	Required. Specifies a namespace for logon. Type: String
Username	Required. Specifies a username for logon. Type: String
Password	Required. Specifies a password that corresponds to the user name. Type: String

## Return Type

None

## Examples

```
objTransApp.Logon("Cognos", "", "") 'Log into the Cognos namespace using Anonymous user.
```

## Move Method

---

The Move method rearranges objects in a collection.

## Syntax

*object* .**Move**(Before, After)

## Applies To

[“CrossTabDataSource Object” on page 50](#)

[DataSource Object](#)

[DateDimension Object](#)

[DateLevel Object](#)

[“DbDataSource Object” on page 72](#)

[Dimension Object](#)

[FlatFileDataSource Object](#)

[“IqdDataSource Object” on page 81](#)

[Level Object](#)

[Measure Object](#)

## Discussion

The Before and After parameters are mutually exclusive. You must use one parameter. If you omit both parameters, an error occurs and a COM exception is thrown. If both parameters are used, only the Before parameter is considered. If invalid objects are used, a COM exception is thrown.

Use the Item method to select the object in a collection.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Before	Required unless After is used. The value can be a numeric index position starting at 1 or an object defined in the script. Type: Variant
After	Required unless Before is used. The value can be a numeric index position starting at 1 or an object defined in the script. Type: Variant

## Return Type

None

## Examples

```
objMeasures = objModel.MeasuresobjMeasures.Item(2).Move(1)
```

## MoveToCategory Method

The MoveToCategory method moves a child category to a different parent category.

## Syntax

*Category* .**MoveToCategory**(Parent [, Sibling])

## Applies To

[Category Object](#)

## Discussion

You can move the category to any level below the new parent. The Sibling parameter gives you a way to position the moved category within the list of existing categories at the new position.

You can also use the `ConnectWithCategory` and `MoveToLevel` methods to move categories to a new position.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation. Here are some examples:

- "The server threw an exception. (Exception from HRESULT: 0x80010105 (RPC\_E\_SERVERFAULT))"
- "trCategory(Category) : Object is of an unexpected type" (if one of the parameter are of unexpected type)

Parameter	Description
Parent	Required. Specifies the parent category to connect to. Type: Object
Sibling	Optional. Specifies a child category in the same level to position the moved category after. If not specified, the category is moved to the end of the list of child categories. The value can be a numeric index position, a string giving an object name, or an object defined in the script. Type: Variant

## Return Type

None

## Examples

```
objCategory
= objModel.Dimensions("Retailers").DrillDowns(1).Categories(2)objCatToMove
= objCategory.ChildCategories(5)objCatToReceive
= objModel.Dimensions("Retailers").DrillDowns(1).Categories(1)objCatToMove.
MoveToCategory(objCatToReceive)
```

## MoveToLevel Method

The `MoveToLevel` method moves a child category to a different level under the current parent category.

## Syntax

*Category* .**MoveToLevel**(Level)

## Applies To

[Category Object](#)

## Discussion

You can move the category to any higher or lower level as long as the new position is below the original parent.

You can also use the `ConnectWithCategory` and `MoveToCategory` methods to move categories to a new position.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Level	Required. Specifies the level in which to move the category. Type: Object

## Return Type

None

## Example

No example is available.

## NewModel Method

---

The `NewModel` method creates a new model.

## Syntax

*Application* .**NewModel**

## Applies To

[Application Object](#)

## Discussion

Use this method to create a new model.

The model is essentially empty when created. You need to define the data sources, dimensions, levels, and measures before you can build cubes.

For new models, use the `SaveAs` method rather than the `Save` method to save your definitions.

Use the `OpenModel` method to modify an existing model.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

Model

## Examples

```
objModel = objTransApp.NewModel
```

## OpenModel Method

---

The OpenModel method opens an existing model.

### Syntax

*Application* .**OpenModel**(Name)

### Applies To

[Application Object](#)

### Discussion

Use this method to reference an existing model when you need to make modifications.

To create a new model, use the NewModel method. To save changes to a model, use the Save or SaveAs method. To close a model, use the Close method.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Name	Required. Specifies the fully qualified file name of the model. Type: String
Login	Required for secure models. Specifies the user ID of the person accessing the model. Type: String
Password	Required for secure models. Specifies the password of the person accessing the model. Type: String
UserClass	Required for secure models. Specifies the user class of the person accessing the model. Type: String

### Return Type

Object

### Examples

```
objModel = objTransApp.OpenModel(strModelPath)
```

## OpenSuspendedModel Method

---

The OpenSuspendedModel method opens an existing suspended model.



## Syntax

*Application* .**OpenSuspendedModel**(Name))

## Applies To

[Application Object](#)

## Discussion

Use this method to open an existing incomplete model.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
SuspendedModel Object.	Required. Specifies an existing incomplete model <b>Type:</b> Object - Suspended Model

## Return Type

Object - Model

## PublishDatasource Method

---

The PublishDatasource method publishes the datasource connection for a PowerCube.

## Syntax

*Cube* .**PublishDatasource**(overwrite)

## Applies To

[Cube Object](#)

## Discussion

Use this method to publish a datasource connection on the IBM Cognos server.

A Windows, or UNIX or Linux® location from where the IBM Cognos server accesses each cube must be specified. If user authentication is enabled, the configured namespace must also be specified. Returns true if successful.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
overwrite	Required. If true, republishes the datasource if it already exists. <b>Type:</b> Boolean

## Return Type

Boolean (True if publish is successful)

## Examples

```
cube = model.Cubes.Item("National")cube.PublishDatasource(True)
'Republish PowerCube if it already exists
```

## PublishPackage Method

The PublishPackage method publishes both the datasource and package for a PowerCube.

### Syntax

*Cube* .**PublishPackage**(datasourceOverwrite, packageOverwrite)

### Applies To

[Cube Object](#)

### Discussion

Use this method to publish a datasource connection and a package on the IBM Cognos Analytics server for a PowerCube.

A Windows, or UNIX or Linux location from where the IBM Cognos server accesses each cube must be specified. If user authentication is enabled, the configured namespace must also be specified. Returns true if successful.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
datasourceOverwrite	Required. If true, republishes the datasource if it already exists. <b>Type:</b> Boolean
packageOverwrite	Required. If true, republishes the package if it already exists. <b>Type:</b> Boolean

### Return Type

Boolean (True, if publish is successful.)

## Examples

```
cube = model.Cubes.Item("National")cube.PublishPackage(True,
True) 'Re-publish both Datasource & Package
```

## Remove Method

The Remove method removes a specific object from a collection.

### Syntax

*collection* .**Remove**(Item)

## **Applies To**

[Associations Collection](#)

[CalculationDefinitions Collection](#)

[Categories Collection](#)

[CategorySets Collection](#)

[Columns Collection](#)

[CubeCustomViews Collection](#)

[Cubes Collection](#)

[CurrencyRates Collection](#)

[CurrencyRecords Collection](#)

[CurrencyTables Collection](#)

[CustomViews Collection](#)

[DataSources Collection](#)

[Dimensions Collection](#)

[DrillDowns Collection](#)

[DrillThroughTargets Collection](#)

[Filters Collection](#)

[Levels Collection](#)

[Measures Collection](#)

[Namespaces Collection](#)

[PackageDatasourceConnections Collection](#)

[Packages Collection](#)

[Prompts Collection](#)

[Queries Collection](#)

[Reports Collection](#)

[SecurityObjects Collection](#)

[Signons Collection](#)

[Views Collection](#)

## **Discussion**

Use the Item parameter of this method to select the specific object to remove from the collection. Use the Delete method to remove objects that exist outside of a collection, such as those selected with the Item method.

You cannot remove the default primary drill-down path created by Transformer for each dimension.

When removing associations, an update on the parent object is called, therefore, all changes made to the parent are saved.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Item	Required. Specifies an object or position in the collection. The value can be a numeric index position (starting at 1), a string giving an object name, or an object defined in the script. Type: Variant

## Return Type

None

## Examples

```
packages = model.Packages packages.Remove(temp_package)
```

## RemoveCubeCustomView Method

The RemoveCubeCustomView method removes the custom view from the cube or child cube.

## Syntax

*Object* .**RemoveCubeCustomView**

## Applies To

- [ChildCube Object](#)
- [Cube Object](#)

## Discussion

Use this method to remove a cube custom view from a cube or child cube.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
CubeCustomView	Required. Specifies the custom view to be removed. Type: CustomView

## Return Type

None

## RemoveFromFolder Method

The RemoveFromFolder method removes a child measure from the current measure folder.

## Syntax

*Measure* .**RemoveFromFolder** Item

## Applies To

[Measure Object](#)

## Discussion

Use this method to remove child measures from the measure folder. If the child measure exists it will be moved to the same level as the measure folder. If the child measure does not exist beneath the measure folder an exception is returned.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Measure	Required. Specifies the measure object to be removed from the folder measure.  Type: Object - Measure

## Return Type

None

## RemoveSuspendedModel Method

---

The RemoveSuspendedModel method removes a suspended model from the SuspendedModels collection.

## Syntax

*Application* .**RemoveSuspendedModel** (*Model*)

## Applies To

[Application Object](#)

## Discussion

Use the IsBad property to test whether the suspended model is corrupt or recoverable. If it is corrupt, use the RemoveSuspendedModel method to delete it from the SuspendedModels collection.

Parameter	Description
Model	Required. Represents an existing incomplete model.  Type: SuspendedModel

## Return Type

Boolean

## Examples

```
objTransApp.RemoveSuspendedModel(objSuspendedModel)
```

## ResetPartitions Method

---

The ResetPartitions method removes current cube partitions.

### Syntax

*Model* .ResetPartitions

### Applies To

Model Object

### Discussion

If your cubes are very large or unusually structured, you may achieve faster build times and better runtime performance by manually defining your partitions. First, use this method to remove any automatic or manual partitions that were previously defined. Then use the Partition property to assign partition level numbers to categories in selected dimensions.

### Return Type

Object

### Examples

```
objModel.ResetPartitions()
```

## Save Method

---

The Save method saves changes to the current model.

### Syntax

*Model* .Save

### Applies To

Model Object

### Discussion

Use this method to write the changes that you made to the model to a model file (.mdl). Use the SaveAs method to save the changes to a different file name and when saving a new model for the first time. If you do not use the Save or SaveAs method before you close a model, you will lose all changes to the model, including those made with the Update method.

### Return Type

None

## SaveAs Method

---

The SaveAs method saves the current model to a different file name.

### Syntax

*Model* .SaveAs FileName

## Applies To

[Model Object](#)

## Discussion

Use this method to write the changes that you made to the model to a model file (.mdl) with a different file name. Always use SaveAs when saving a new model for the first time.

Use the Save method to save the changes to the same file name. If you do not use the Save or SaveAs method before you close a model, you will lose all changes to the model, including those made with the Update method.

Parameter	Description
FileName	Required. Provides a file name and path, if required.  Type: String  If no path is specified for the file name, Transformer uses the default directory.

## Return Type

None

## Examples

```
objModel.SaveAs("great outdoors salesX.mdl")
```

## SetAllocation Method

---

The SetAllocation method changes the allocation type for a measure used by an object.

## Syntax

*object*.**SetAllocation** Measure, AllocationType [, AllocationMeasure ]

## Applies To

[“Category Object” on page 43](#)

[DateDimension Object](#)

[DateLevel Object](#)

[Dimension Object](#)

[Level Object](#)

[SpecialCategory Object](#)

## Discussion

Allocation is possible only when your model uses multiple data sources. Use this method to specify how or if Transformer allocates summary data to a dimension, level, or category.

When you change the allocation type for a level, the new allocation type is applied from the categories in that level to all descendant categories.

When the source of a measure is a data source that does not reference a dimension in the model, Transformer automatically allocates the measure to the entire dimension and then allocates the measure value as a constant throughout the dimension. To roll back this allocation, use the SetAllocation method with the trAllocationNA constant.

The CanAllocate property must return a value of True for allocation to proceed.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Measure	Required. Specifies the measure to apply allocation to. Type: Object-Measure
AllocationType	Required. Sets the allocation option. Specifies a constant of the value list xtrAllocationType. Specifies the object to use as the source of allocation values. Type: Constant-xtrAllocationType
AllocationMeasure	Optional. Applies only when the AllocationType parameter is set to trAllocationAllocated. Type: Object

## Return Type

None

## Examples

```
objDimension.SetAllocation(objMeasure, xtrAllocationType.trAllocationNA)
```

## SetDefaultCategory Method

The SetDefaultCategory method specifies a new default category for a dimension, other than the root or parent of a group of scenario categories. There can only be one default category for each dimension.

## Syntax

*Dimension* .SetDefaultCategory

## Applies To

- [DateDimension Object](#)
- [Dimension Object](#)

## Discussion

We recommend that you use this method together with the HideValue property of the root or parent Category object to create a scenario dimension.

By setting a default level for opening the cube, you ensure that cubes containing budget values or other scenario-like data do not display zeros, non-applicable numbers, or meaningless numbers, when opened by report authors or consumers.



A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

None.

Parameter	Description
Category	Required. In the given dimension, specifies a valid Category object. Type: Object-Category

## SetDeployType Method

---

The SetDeployType method sets the deployment strategy for Copy and Activate.

### Syntax

*Cube* .**SetDeployType**(deployType

### Applies To

[Cube Object](#)

### Discussion

Use this method before deploying a PowerCube.

The deployType parameter allows a user to set the deployment strategy. trDeployType\_NONE specifies that a PowerCube cannot be deployed.

A value of trDeployType\_SWAPSINGLE specifies that if one or more of the deployment locations are unavailable, the deployment action is aborted for all specified locations.

A value of trDeployType\_SWAPTOGETHER specifies that if one or more locations specified in the deployment locations (see AddDeployLocation) is not available, the PowerCube does not deploy to these locations. However, the PowerCube does not deploy to all available locations.

For all values except trDeployType\_NONE, the Deployment location(s) needs to be set prior to calling DeployCube method.

If the cube is a child cube belonging to a cube group, the child cube can inherit the same deployment strategy as its parent cube. This does not apply for time-based partitioned cubes.

By default, the deployment strategy is 'Do not deploy'.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

None

Parameter	Description
deployType	Required. Specifies the deployment type. This parameter uses a constant of the value list xtrDeployType Type: Constant - xtrDeploytype

### Examples

```
cube
= model.Cubes.Item("National")cube.
  SetDeployType(xtrDeployType.trDeployType_SWAPSINGLE)
```

## SetExcludeDateDimension Method

For a given DrillThroughTarget and DateDimension, the SetExcludeDateDimension method sets the drill-through restriction to True if the isRestricted parameter is True. To remove the restrictions, set the isRestricted parameter to False.

### Syntax

*DrillThroughTarget* .**SetExcludeDateDimension**

### Applies To

DrillThroughTarget Object

### Discussion

Use this property to exclude the DateDimension from the drill-through target.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Date Dimension	Required. Specifies the Date Dimension object that is excluded from the drill-through action. Type: DateDimension
IsRestricted	Required. Specifies whether the Date Dimension object is excluded. Type: Boolean

### Type

Boolean

### Access

Write

## SetExcludeDateLevel Method

---

For a given DrillThroughTarget and DateLevel, the SetExcludeDateLevel method sets the drill-through restriction to True if the isRestricted parameter is True. To remove the restrictions, set the isRestricted parameter to False.

### Syntax

*DrillThroughTarget* .**SetExcludeDateLevel**

### Applies To

DrillThroughTarget Object

### Discussion

Use this property to exclude the DateLevel from the drill-through target.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Date Level	Required. Specifies the Date Level object that is excluded from the drill-through action. Type: Object-DateLevel
IsRestricted	Required. Specifies whether the Date Level object is excluded or not. Type: Boolean

### Type

Boolean

### Access

Write

## SetExcludeDimension Method

---

For a given DrillThroughTarget and Dimension, the SetExcludeDimension method sets the drill-through restriction to True if the isRestricted parameter is True. To remove the restrictions, set the isRestricted parameter to False.

### Syntax

*DrillThroughTarget* .**SetExcludeDimension**

### Applies To

DrillThroughTarget Object

## Discussion

Use this property to exclude the Dimension from the drill-through target.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Dimension	Required. Specifies the Dimension object that is excluded from the drill-through action. Type: Dimension
isRestricted	Required. Specifies whether the Dimension is excluded. Type: Boolean

## Return Type

None

## SetExcludeLevel Method

---

For a given DrillThroughTarget and Level, the SetExcludeLevel method sets the drill-through restriction to True if the isRestricted parameter is True. To remove the restriction, set the isRestricted parameter to False.

## Syntax

*DrillThroughTarget* .**SetExcludeLevel**

## Applies To

[DrillThroughTarget](#) Object

## Discussion

Use this property to exclude the level from the DrillThroughTarget.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Level	Required. Specifies the Level object that is excluded from the drill-through action. Type: Object
isRestricted	Required: Specifies whether the Level is excluded. Type: Boolean

## Return Type

Boolean

## SetViewStatus Method

---

The SetViewStatus method sets how a Category, SpecialCategory, Level or DateLevel object is viewed.

### Syntax

*View* .**SetViewStatus** Object, Status

### Applies To

View Object

### Discussion

Once a View object is obtained, it can be used to set the ViewStatus of an associated Category, SpecialCategory, DateLevel or Level object. Assign a constant from the xtrViewStatus value list to define the view.

To create a view of one category and child categories, use the Apex property.

To clear the current view status, retrieve the current status and call SetViewStatus with the result. This toggles the current status and clears it.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Object	Required. This method can only be applied to Category, SpecialCategory, DateLevel or Level objects.  Type: Object

### Return Type

Constant - xtrViewStatus

### Examples

```
objDimension = objModel.Dimensions("Retailers")objView  
= objDimension.Views.Add()objView.SetViewStatus(objCategory,  
xtrViewStatus.trViewStatusSummaryMom)
```

The following example shows how to clear the current view status.

```
xtrViewStatus currentStatus = objView.GetViewStatus(objCategory);  
objView.SetViewStatus(objCategory, currentStatus);
```

## TestBuild Method

---

The TestBuild method creates a small test model or cube.

### Syntax

*Model* .**TestBuild** Records, BuildCubes

## Applies To

[Model Object](#)

## Discussion

Use to generate categories or to create a cube for testing purposes. This can save considerable time if your source data is very large.

For example, you can specify a test build that uses the first 350 records from each data source to build a cube. Transformer quickly creates a small cube with categories you can evaluate in PowerPlay.

Parameter	Description
Records	Required. Specifies the number of records to test against. Type: Long
BuildCubes	Required. Specifies whether to add categories only or to add categories and build a cube. If set to True, Transformer builds a cube. Type: Boolean

## Return Type

None

## Examples

```
objModel = objTransApp.NewModel objModel.TestBuild(20, True)
```

## Update Method

---

The Update method updates the associated object.

## Syntax

*object*.**Update**

## Applies To

[Association Object](#)

[CalculationDefinition Object](#)

[Category Object](#)

[CategorySet Object](#)

[ChildCube Object](#)

[Column Object](#)

[CrossTabDataSource Object](#)

[Cube Object](#)

[CubeGroup Object](#)

[CurrencyRate Object](#)  
[CurrencyRecord Object](#)  
[CurrencyTable Object](#)  
[CustomView Object](#)  
[DataSource Object](#)  
[DateDimension Object](#)  
[DateDrillDown Object](#)  
[DateLevel Object](#)  
[DbDataSource Object](#)  
[Dimension Object](#)  
[DrillDown Object](#)  
[DrillThroughTarget Object](#)  
[Filter Object](#)  
[FlatFileDataSource Object](#)  
[IqdDataSource Object](#)  
[Level Object](#)  
[Measure Object](#)  
[Model Object](#)  
[Namespace Object](#)  
[Package Object](#)  
[PackageDatasourceConnection Object](#)  
[Prompt Object](#)  
[Query Object](#)  
[Report Object](#)  
[SecurityObject Object](#)  
[Signon Object](#)  
[SpecialCategory Object](#)  
[View Object](#)

## Discussion

Use this method to set the changes made to the properties of an object. Use the Save or SaveAs method to save changes to the model before you close it, or else all updates to objects are lost. If the Update method is not used, no changes made to the object are saved when the model is saved.

The Update method is not needed to set changes to the Application object or DateWizard object.

When you run Update, Transformer checks the changes to the object to determine if they are valid. For example, if you set an incorrect expression in the ExpressionText property of a Category object, the update fails and Transformer issues error messages.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

None

## Examples

```
objDimensions = objModel.DimensionsobjLocationsDim  
= objDimensions.Item(3)objLocationsDim.Update()
```

## Verify Method

---

The Verify method verifies the associated object.

### Syntax

*object* .**Verify**()

### Applies To

[Package Object](#)

[PackageDataSourceConnection Object](#)

[Query Object](#)

[Report Object](#)

[SecurityObject Object](#)

### Discussion

Use this method to verify any changes made to the properties of an object. The Update method also verifies the object so this method isn't needed if an Update is called on the object. Transformer checks the changes to the object to determine if they are valid.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Return Type

None



---

## Chapter 5. Properties

The following tables lists all the Transformer OLE automation properties.

Property	Description
<a href="#">ActivityMeasure Property</a>	Sets or returns the measure used by a category count measure.
<a href="#">Aggregate Property</a>	Sets or returns the basic type of relative time category.
<a href="#">AllocationMeasure Property</a>	Returns the Measure object used as a weighting factor.
<a href="#">AllocationType Property</a>	Returns how an object allocates a measure.
<a href="#">AllowCurrencyConversion Property</a>	Sets or returns whether you can change a currency.
<a href="#">AllowDrillThrough Property</a>	Sets or returns whether a cube or measure can drill through to a cube or report.
<a href="#">AlternateQueryPath Property</a>	Sets or returns an alternative data source path used by the cube.
<a href="#">AltMDCFile Property</a>	Specifies an alternate filename for the cube.
<a href="#">AlwaysUseTransformerSignon Property</a>	Specifies whether to use the Transformer Signon or the Content Manager Signon.
<a href="#">Apex Property</a>	Sets or returns the Category object that serves as the root for a dimension view.
<a href="#">Application Property</a>	Returns the Transformer Application object.
<a href="#">AssociationRole Property</a>	Sets or returns the role performed by the Association object.
<a href="#">Associations Property</a>	Returns an Associations collection.
<a href="#">AssociationType Property</a>	Sets or returns the type of data source related to an Association object.
<a href="#">AutoLogon Property</a>	When true, defines whether the application will automatically authenticate to the security namespaces associated with the signon.
<a href="#">AutoSummary Property</a>	Sets or returns the auto-summary option for a query.
<a href="#">BlankSubstitute Property</a>	Sets or returns the default label for blank categories generated in the date level.

<b>Property</b>	<b>Description</b>
<a href="#">BlockParentTotals Property</a>	Sets or returns whether parents of excluded categories display a denied value.
<a href="#">CacheCrossTabs Property</a>	Sets or returns whether summaries are cached for the initial PowerPlay crosstab in the cube.
<a href="#">CalculationDefinitions Property</a>	Returns a collection of CalculationDefinition objects.
<a href="#">CAMID Property</a>	Returns the CAMID of the namespace.
<a href="#">CanAllocate Property</a>	Returns whether you can allocate any measure values to descendant levels and categories.
<a href="#">CanAllocateByMeasure Property</a>	Returns whether you can use the specified measure as a weighting factor when you allocate by measure.
<a href="#">CanAllocateMeasure Property</a>	Returns whether you can allocate the specified measure to descendant levels and categories.
<a href="#">Categories Property</a>	Returns a Categories collection associated with an object.
<a href="#">Category Property</a>	Returns the applicable date category for a given currency rate.
<a href="#">CategoryCount Property</a>	Returns the number of regular categories in a dimension or level.
<a href="#">CategoryCountLevel Property</a>	Sets or returns the Level object to which a category count applies.
<a href="#">CategorySets Property</a>	Returns the CategorySets collection used by a CalculationDefinition object.
<a href="#">CharacterType Property</a>	Sets or returns the type of character set used by a data source.
<a href="#">ChildCategories Property</a>	Returns a Categories collection.
<a href="#">ChildCubes Property</a>	Returns a collection of ChildCube objects from a cube group.
<a href="#">ChildCustomViews Property</a>	Returns a collection of CustomView objects.
<a href="#">ChildMeasures Property</a>	Returns a collection of child measures from a measure folder.
<a href="#">Code Property</a>	Sets or returns a code that uniquely identifies the category within the entire dimension.
<a href="#">Columns Property</a>	Returns a Columns collection for a data source.

<b>Property</b>	<b>Description</b>
<a href="#">ColumnsLoaded Property</a>	Returns whether the columns in a data source have been used to build a model.
<a href="#">CompressMDC Property</a>	Sets or returns whether a cube is compressed for compact storage.
<a href="#">Connection Property</a>	Sets and gets the Content Manager connection.
<a href="#">Consolidate Property</a>	Sets or returns how a cube is consolidated.
<a href="#">Context Property</a>	Sets or returns the drill-down path used to order categories within a level.
<a href="#">ContextLevel Property</a>	Sets or returns a date period.
<a href="#">ContextOffset Property</a>	Sets or returns the position of the context period relative to the current period.
<a href="#">ConvergenceLevel Property</a>	Returns the convergence level for an alternate drill-down path.
<a href="#">Count Property</a>	Returns the number of objects in a collection.
<a href="#">CountryCode Property</a>	Sets or returns a code for the country or region to which a currency record applies.
<a href="#">CubeCodePage Property</a>	Sets or returns the cube code page setting for the model used to build the cube.
<a href="#">CubeCreation Property</a>	Sets or returns whether the cube is created.
<a href="#">CubeCustomViews Property</a>	Returns a collection of CustomView objects.
<a href="#">Cubes Property</a>	Returns a collection of Cube and CubeGroup objects.
<a href="#">CubeStamp Property</a>	Returns a cube creation time-stamp.
<a href="#">CurrencyCountryLabel Property</a>	Sets or returns the currency country or region label.
<a href="#">CurrencyDecimals Property</a>	Sets or returns the number of decimal places used in a currency.
<a href="#">CurrencyFormatOverride Property</a>	Sets or returns whether you can override the standard format for a currency.
<a href="#">CurrencyIsEMU Property</a>	Sets or returns whether the record is an EMU currency record.
<a href="#">CurrencyIsEuro Property</a>	Sets or returns whether the currency record is the base euro currency.

Property	Description
<a href="#">CurrencyRates Property</a>	Returns a CurrencyRates collection.
<a href="#">CurrencyRecord Property</a>	Returns the CurrencyRecord object to which the CurrencyRate object applies.
<a href="#">CurrencyRecords Property</a>	Returns a collection of CurrencyRecord objects.
<a href="#">CurrencySymbol Property</a>	Sets or returns the monetary symbol associated with a currency.
<a href="#">CurrencyTable Property</a>	Returns a CurrencyTable object related to a currency rate.
<a href="#">CurrencyTables Property</a>	Returns a collection of CurrencyTable objects.
<a href="#">CurrencyTableType Property</a>	Sets or returns the type of currency table.
<a href="#">CurrentModel Property</a>	Returns the currently active Model object.
<a href="#">CurrentValueIndex Property</a>	Sets or returns the current prompt value index.
<a href="#">CustomView Property</a>	Sets or returns whether a view contains all, some, or none of the categories in a dimension.
<a href="#">CustomViews Property</a>	Returns a collection of CustomView objects.
<a href="#">DataCharacterSet Property</a>	Sets or returns the default character set used by the application.
<a href="#">DataClass Property</a>	Sets or returns the data type of a source column.
<a href="#">DataRange Property</a>	Sets or returns the name of a database range in a data source.
<a href="#">DataSource Property</a>	Sets or returns the Content Manager data source.
<a href="#">DataSourcePath Property</a>	Sets or returns the location where Transformer searches for data source files.
<a href="#">DataSources Property</a>	Returns a collection of DataSource objects in a model.
<a href="#">DataSourceWindowsLocation Property</a>	Sets or returns the location of the cube, including the full path and cube name.
<a href="#">DataTemporaryFilesPath Property</a>	Sets or returns the name of the directory where Transformer creates temporary work files while generating cubes.
<a href="#">DateDegreeofDetail Property</a>	Sets or returns the date level at which reporting will occur.

Property	Description
<a href="#">DateDegreeofDetailLevelName Property</a>	Sets or returns the date level that applies to an externally rolled up measure.
<a href="#">DateFormat Property</a>	Sets or returns how dates appear.
<a href="#">DateFunction Property</a>	Sets or returns which date categories are generated in a level.
<a href="#">DateInputFormat Property</a>	Sets or returns date format order in the data source.
<a href="#">DateLevel Property</a>	Sets or returns the level in a time dimension to which currency rates apply.
<a href="#">DateWizard Property</a>	Returns the DateWizard object.
<a href="#">DecimalPoint Property</a>	Sets or returns the character used for a decimal point.
<a href="#">Decimals Property</a>	Returns the number of decimal places in a column, if defined in the source data.
<a href="#">DefaultCategoryOrderBy Property</a>	Sets the default sort order for all categories in the model.
<a href="#">DefaultDateFormat Property</a>	Sets or returns the default setting for the DateInputFormat property.
<a href="#">Description Property</a>	Sets or returns the description of the object.
<a href="#">DesiredPartitionSize Property</a>	Sets or returns the desired partition size.
<a href="#">DetachDataSource Property</a>	Sets or returns whether the connection to the data source is maintained or released.
<a href="#">DetailLevel Property</a>	Sets or returns the lowest detail level for cubes in a CubeGroup object.
<a href="#">Dimension Property</a>	Returns a dimension for a Category object.
<a href="#">DimensionInclude Property</a>	Sets or returns the type of view for a custom view.
<a href="#">DimensionLevels Property</a>	Returns a DimensionLevels collection.
<a href="#">DimensionName Property</a>	Sets the name for a new DateDimension object.
<a href="#">Dimensions Property</a>	Returns a collection of Dimension and DateDimension objects.
<a href="#">DimensionView Property</a>	Sets or returns the View object associated with a cube or custom view.

Property	Description
<a href="#">DimensionViewType Property</a>	Sets or returns which dimensions and views belong in a cube.
<a href="#">DisplayName Property</a>	Sets or returns the name to display for the SecurityObject.
<a href="#">DrillCode Property</a>	Sets or returns a code that uniquely identifies the drill-down category within the entire dimension.
<a href="#">DrillDowns Property</a>	Returns a DrillDowns collection.
<a href="#">DrillInclusion Property</a>	Sets or returns whether a drill-down path is included in a cube.
<a href="#">DrillThroughTargets Property</a>	Returns a collection of drill-through target objects associated with a Transformer model.
<a href="#">DuplicateRollup Property</a>	Sets or returns how duplicate measure values from consolidated records are rolled up.
<a href="#">DuplicateWeight Property</a>	Sets or returns the name of the measure that contains average weighting factors.
<a href="#">EarliestDate Property</a>	Sets the earliest date in a date range used to select categories.
<a href="#">EMUEntryDate Property</a>	Sets or returns the date on which euro triangulation calculations began for a currency.
<a href="#">EnableMessageLogging Property</a>	Sets or returns whether Transformer messages are written to a log file.
<a href="#">EnableTimePeriod Property</a>	Sets the level of detail for a time dimension.
<a href="#">EstimatedRows Property</a>	Sets or returns an estimate of the number of records that the cube contains before auto-partitioning.
<a href="#">ExcludeAutoPartition Property</a>	Sets or returns whether a dimension is excluded from the auto-partition process.
<a href="#">ExpressionText Property</a>	Sets or returns the contents of an expression that defines a value for an object.
<a href="#">External Property</a>	Sets or returns whether the data source contains presummarized values.
<a href="#">FieldSeparator Property</a>	Sets or returns the type of field delimiter used by the data source.
<a href="#">FileName Property</a>	Returns the name of a model file as it appears in a Windows folder or Windows Explorer.

Property	Description
<a href="#">Filters Property</a>	Returns the collection of Filter objects associated with a Query.
<a href="#">FindCategoryByCatCode Property</a>	Returns the category object that contains the specified category code string.
<a href="#">Format Property</a>	Sets or returns how numeric values appear.
<a href="#">FormatDecimals Property</a>	Sets or returns the number of decimal places PowerPlay displays for the measure.
<a href="#">FullName Property</a>	Returns the location of a model file.
<a href="#">GenerateCategories Property</a>	Sets or returns whether categories are generated for the data source.
<a href="#">GenerateDateCategories Property</a>	Sets or returns whether a date level generates date categories.
<a href="#">GenerateDates Property</a>	Sets whether the DateWizard object generates date categories.
<a href="#">GeneratePowerCube Property</a>	Sets or returns when a data source is referenced by a model.
<a href="#">GenerateTimePeriod Property</a>	Sets or returns category generation options for a time dimension.
<a href="#">Group Property</a>	Sets or returns whether a CalculationDefinition is grouped with the categories in the category set.
<a href="#">GroupDimension Property</a>	Sets or returns the dimension used to build the cube group.
<a href="#">GroupLevel Property</a>	Sets or returns the level whose categories become the individual cubes in a cube group.
<a href="#">HasSubdimension Property</a>	Returns whether a level object contains a subdimension.
<a href="#">HideValue Property</a>	Specifies whether to hide the value of a category object. Default: false.
<a href="#">ID Property</a>	Sets or returns the namespace ID.
<a href="#">IgnoreMissingValue Property</a>	Specifies whether to ignore null or missing values in a time state rollup. Default: false.
<a href="#">Inclusion Property</a>	Sets or returns the circumstances under which categories are included in a cube.
<a href="#">IncrementalUpdate Property</a>	Sets or returns whether a cube or cube group is incrementally updated from the data source.

Property	Description
<a href="#">InputScale Property</a>	Sets or returns a scale value used to convert column numbers from decimal values to integer values.
<a href="#">IsAnyColumnMismatched Property</a>	Returns whether columns in the data source match the underlying data.
<a href="#">IsBad Property</a>	Returns whether a suspended model is corrupt or recoverable.
<a href="#">IsExpressionValid Property</a>	Returns whether an expression is valid.
<a href="#">IsFolder Property</a>	Sets or returns whether a measure is a measure folder.
<a href="#">IsManual Property</a>	Returns whether a level is associated with a source value.
<a href="#">IsMDCInUse Property</a>	Returns whether a cube is in use or being rebuilt.
<a href="#">IsolationLevel Property</a>	Sets or returns the isolation level used to define permissible transactions.
<a href="#">IsPrimary Property</a>	Sets or returns whether the drill category or drill-down path is the primary one.
<a href="#">IsTimeBasedPartitionedCube Property</a>	Sets or returns whether a cube group is specified as a time-based partitioned cube.
<a href="#">KeyName Property</a>	Sets or returns the value that appears in the associated data source column.
<a href="#">Label Property</a>	Sets or returns a descriptive name that appears in PowerPlay.
<a href="#">LastUseDate Property</a>	Returns the date the category was last modified or used.
<a href="#">LatestDate Property</a>	Sets the latest date in a date range used to select categories.
<a href="#">Level Property</a>	Returns a level for a Category or SpecialCategory object.
<a href="#">LevelCategories Property</a>	Sets or returns the categories for a specific level.
<a href="#">LevelDrillDowns Property</a>	Returns a LevelDrillDowns collection.
<a href="#">Levels Property</a>	Returns a Levels collection.
<a href="#">LocalPath Property</a>	Sets or returns the location for a local data source.



Property	Description
<a href="#">LogErrorLevel Property</a>	Sets or returns the level of severity of error messages logged.
<a href="#">LogFileAppend Property</a>	Sets or returns whether Transformer appends messages to the log file or overwrites previous log messages.
<a href="#">LogFileName Property</a>	Sets or returns the name for the log file.
<a href="#">LogFilesPath Property</a>	Sets or returns the location where Transformer saves the log file.
<a href="#">Lunar Property</a>	Sets or returns whether the object is based on a lunar year.
<a href="#">ManualCurrentPeriod Property</a>	Sets or returns whether the current time period is set manually or automatically.
<a href="#">MaximizeSpeed Property</a>	Sets or returns whether category generation is optimized for speed.
<a href="#">MaxNumPartLevels Property</a>	Sets or returns the maximum number of times Transformer reads the data source when it partitions a cube.
<a href="#">MaxTransactionNumber Property</a>	Sets or returns the maximum number of records that Transformer processes before committing the changes to a cube.
<a href="#">MDCFile Property</a>	Sets or returns the name of a PowerCube file (.mdc).
<a href="#">MeasureInclude Property</a>	Sets or returns the name of a measure to include in a cube, cube group, or user class.
<a href="#">MeasureName Property</a>	Sets or returns a descriptive title that identifies a measure on the PowerPlay dimension line.
<a href="#">Measures Property</a>	Returns a Measures collection.
<a href="#">MeasureType Property</a>	Returns whether a measure is regular, calculated, or a category count.
<a href="#">MissingValue Property</a>	Sets or returns what appears in place of a blank or null value.
<a href="#">ModelName Property</a>	Returns the name of a suspended model.
<a href="#">ModelsPath Property</a>	Sets or returns the location where Transformer opens and saves model files.
<a href="#">ModelTemporaryFilesPath Property</a>	Sets or returns the location where Transformer creates temporary model files (.qy?).

Property	Description
<a href="#">ModelType Property</a>	Returns the file extension of a model file as it appears in a Windows folder or Windows Explorer.
<a href="#">MonthType Property</a>	Sets how to calculate the month level of a time dimension.
<a href="#">Name Property</a>	Sets or returns the name of an object.
<a href="#">Namespaces Property</a>	Returns a Namespaces collection.
<a href="#">NewCatsLocked Property</a>	Sets or returns whether you can add new categories.
<a href="#">ObjectCAMID Property</a>	Returns the CAMID of the object in the namespace set by the ObjectName property.
<a href="#">ObjectName Property</a>	Sets or returns the name of a namespace object.
<a href="#">Optimize Property</a>	Sets or returns the current cube optimization option.
<a href="#">OrderByDescending Property</a>	Sets or returns whether values appear in descending order.
<a href="#">OrderByStorageType Property</a>	Sets or returns how categories are sorted based on the storage type of a column.
<a href="#">Origin Property</a>	Returns the origin of the specified object.
<a href="#">OriginalName Property</a>	Sets or returns the name of the column in the data source.
<a href="#">Orphanage Property</a>	Sets or returns whether a category is an orphanage.
<a href="#">OutputScale Property</a>	Sets or a returns a scale value used to convert numbers from integer values to decimal values in PowerPlay.
<a href="#">Packages Property</a>	Returns the collection of Package objects associated with a Transformer model.
<a href="#">PackagesDatasourceConnections Property</a>	Returns the collection of PackageDatasourceConnection objects associated with a Package or Report object.
<a href="#">Parent Property</a>	Returns the name of an object's parent.
<a href="#">ParentCategories Property</a>	Returns a collection of parent categories for a category.
<a href="#">Partition Property</a>	Sets or returns a manual partition number.

Property	Description
<a href="#">Password Property</a>	Sets a case-sensitive password.
<a href="#">PatFile Property</a>	Sets or returns the location of the pattern file <code>cogtr_locale.pat</code> , such as <code>cogtr_en.pat</code> , for the associated product locale.
<a href="#">Path Property</a>	Returns the location of a model file.
<a href="#">PopulateByDataSource Property</a>	Sets or returns whether the currency rate is obtained through a data source or set within Transformer.
<a href="#">Position Property</a>	Sets or returns the ordinal or starting position of a column in the data source.
<a href="#">PowerCubesPath Property</a>	Sets or returns the location where Transformer creates PowerCube files (.mdc).
<a href="#">PowerPlayPath Property</a>	Sets or returns the location of the PowerPlay.exe executable.
<a href="#">Precision Property</a>	Sets or returns the number of decimal places for measures used in calculations.
<a href="#">PromptForPassword Property</a>	Specifies whether users are always prompted for a password when using Transformer in UI mode. Applies only to the DataSource signon type.
<a href="#">Prompts Property</a>	Returns the collection of Prompt objects associated with a Query.
<a href="#">PromptValueType Property</a>	Sets or returns the type of prompt.
<a href="#">QualifiedName Property</a>	Returns the fully qualified name of a level.
<a href="#">QuarterType Property</a>	Sets how to calculate the quarter level of a time dimension.
<a href="#">Queries Property</a>	Returns the collection of Query objects associated with a Package or Report object.
<a href="#">QyPath Property</a>	Returns the location of a suspended model.
<a href="#">Rate Property</a>	Sets or returns a currency exchange rate.
<a href="#">RefName Property</a>	Sets or returns the name of the filter in the data source.
<a href="#">RefreshDescription Property</a>	Sets or returns whether descriptions are updated.
<a href="#">RefreshLabel Property</a>	Sets or returns whether labels are updated.
<a href="#">RefreshShortName Property</a>	Sets or returns whether short names are updated.

Property	Description
<a href="#">RegularRollup Property</a>	Sets or returns the current regular rollup function for an object.
<a href="#">RegularWeight Property</a>	Sets or returns a measure name used in a weighted average calculation.
<a href="#">Reports Property</a>	Returns the collection of Report objects associated with a Transformer model.
<a href="#">ReverseSign Property</a>	Sets or returns whether PowerPlay reverses the sign of a measure.
<a href="#">Rollup Property</a>	Sets or returns whether measure values for a special category roll up into the parent category.
<a href="#">RollupTiming Property</a>	Sets or returns when to perform calculations for calculated measures.
<a href="#">RowsAsSample Property</a>	Sets or returns the number of rows that the DoAutoDesign method samples when creating a model.
<a href="#">RowsChecked Property</a>	Sets or returns the maximum number of rows that the DoAutoDesign method reads from the data source.
<a href="#">RunningPeriods Property</a>	Sets or returns the number of time periods used for running-totals.
<a href="#">SecurityObjects Property</a>	Returns a collection of SecurityObjects.
<a href="#">Server Property</a>	Sets or returns whether a cube is processed locally or on a server.
<a href="#">ServerModelPath Property</a>	Sets or returns the location of the server for a model.
<a href="#">ServerPath Property</a>	Sets or returns the location of the server for a data source.
<a href="#">ServerQuery Property</a>	Sets or returns whether data is processed locally or on a server.
<a href="#">ServicesBuildNumber Property</a>	Returns the version number of Transformer in numeric format.
<a href="#">ServicesVersionText Property</a>	Returns the version number of Transformer in text format.
<a href="#">SetsCurrentPeriod Property</a>	Sets or returns whether Transformer searches a data source to find the current period date.
<a href="#">ShortName Property</a>	Sets or returns a short name for the measure.

Property	Description
<a href="#">Signon Property</a>	Sets or returns the Signon object used by a cube or cube group.
<a href="#">SignOnNamespace Property</a>	Contains the security namespace associated with the IBM Cognos signon.
<a href="#">Signons Property</a>	Returns the Signons collection for a model.
<a href="#">SignonType Property</a>	Sets or returns the signon type. The signon type can be the Datasource signon or the IBM Cognos signon.
<a href="#">Size Property</a>	Sets or returns the size of a column or Model file (.mdl).
<a href="#">SortComparisonRule Property</a>	Sets or returns which comparison rule Transformer uses when sorting data.
<a href="#">SourceType Property</a>	Sets or returns the type of data file a data source uses.
<a href="#">SpecialCategoryCount Property</a>	Returns the number of drill, root, and special categories in a dimension.
<a href="#">SQLExpression Property</a>	Returns the SQL expression used to define an Impromptu query definition file (.iqd).
<a href="#">Status Property</a>	Returns a problem status associated with the cube the last time it was created.
<a href="#">StorageType Property</a>	Sets or returns the size of a numeric data type.
<a href="#">StreamExtractAllowed Property</a>	Sets or returns the flag that determines if stream extractions is allowed (applies only to SAP BW data source).
<a href="#">StreamExtractSize Property</a>	Sets or returns the size in megabytes of the buffer used to transfer data from SAP when StreamExtract is set to true.
<a href="#">SummaryLevel Property</a>	Sets or returns which level to use to summarize external categories in a cube group.
<a href="#">SuppressNull Property</a>	Sets or returns the null suppression option used for SAP BW data sources.
<a href="#">SuspendedModels Property</a>	Returns a collection of SuspendedModel objects.
<a href="#">TargetLevel Property</a>	Sets or returns the level of detail of a date period.
<a href="#">TargetOffset Property</a>	Sets or returns the position of the target period relative to the current period.

<b>Property</b>	<b>Description</b>
<a href="#">ThousandPoint Property</a>	Sets or returns the character used to separate numbers in thousands.
<a href="#">Time Property</a>	Returns the time stamp of a model as it appears in a Windows folder or Windows Explorer.
<a href="#">TimeArrayColumn Property</a>	Sets or returns the name of the first column in the array when the object represents a date array.
<a href="#">TimeArrayStartMonth Property</a>	Sets or returns the month in which a fiscal year begins when the object includes a date array.
<a href="#">TimeArrayType Property</a>	Sets or returns the type of array used for date values.
<a href="#">TimeRank Property</a>	Sets or returns the relative rank of date levels within a time dimension.
<a href="#">TimeStamp Property</a>	Sets or returns the time stamp of a Package or Report object.
<a href="#">TimeStateRollup Property</a>	Sets or returns the date period used for time state rollups.
<a href="#">TimeStateWeight Property</a>	Sets or returns a measure name used in a weighted average calculation.
<a href="#">ToDateLevel Property</a>	Sets or returns the date period used for to-date totals.
<a href="#">TransdaPath Property</a>	Sets or returns the location of the transda.exe executable.
<a href="#">TransformerSignon Property</a>	Sets or returns the Transformer signon object associated with a package data source connection (IBM Cognos signon).
<a href="#">Type Property</a>	Sets or returns the type of an object.
<a href="#">Unique Property</a>	Sets or returns whether Transformer can identify each category in the level by a unique source value.
<a href="#">UniqueMove Property</a>	Sets or returns how a unique level is treated when the related category is moved.
<a href="#">UseAltMDCFile Property</a>	Sets or returns whether a temporary filename may be used.
<a href="#">User Property</a>	Sets or returns a user name associated with the namespace.
<a href="#">UserCAMID Property</a>	Returns the CAMID of the user set by the User property.

Property	Description
<a href="#">UserID Property</a>	Sets or returns the signon user ID.
<a href="#">Value Property</a>	Sets or returns the prompt value.
<a href="#">ValuesCount Property</a>	Returns the number of values set for the prompt.
<a href="#">Version Property</a>	Returns the version number of Transformer.
<a href="#">Views Property</a>	Returns a Views collection.
<a href="#">ViewType Property</a>	Sets or returns whether a view contains all, some, or none of the categories in a dimension.
<a href="#">WeekAdd Property</a>	Sets or returns how many days are added to a lunar year.
<a href="#">WeekSpan Property</a>	Sets or returns how to treat a week that spans two years.
<a href="#">WeekStart Property</a>	Sets or returns the first day of the week.
<a href="#">WeekStartDay Property</a>	Sets the first day of the week.
<a href="#">WorkingDay Property</a>	Sets or returns whether a specific day is part of the working week.
<a href="#">WorkingDays Property</a>	Sets or returns which days are part of the working week.
<a href="#">YearStartDay Property</a>	Sets or returns the first day of a year.
<a href="#">YearType Property</a>	Sets how to calculate the year level of a time dimension.

## ActivityMeasure Property

The ActivityMeasure property sets or returns the measure used by a category count measure.

### Syntax

*Measure* .ActivityMeasure

### Applies To

[Measure Object](#)

### Discussion

You can define a measure that counts categories. For example, you can create a measure that shows how many different customers bought a specific product each month, quarter, or year. Based on a unique level (such as Customer No.), this count includes all non-missing, non-zero values, but does not double-count. For example, if the same customer buys a product twice in the same quarterly period, the quarterly rollup counts that customer only once.

Use the ActivityMeasure property to specify a particular activity measure for a category count. If it is not specified, all measures in the model that meet the criteria are used to generate results.

An activity measure cannot be a calculated measure, an after-rollup measure, or an externally rolled-up measure.

To create a category count measure, add a measure to the measures collection and set the CategoryCountLevel property to the level at which categories are to be counted. The level specified must be unique.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - Measure

## Access

Read/Write

## Examples

```
objMeasure.CategoryCountLevel = objLevelobjMeasure.ActivityMeasure  
= objActMeasure
```

# Aggregate Property

---

The Aggregate property sets or returns the type of relative time category.

## Syntax

*SpecialCategory*.**Aggregate**

## Applies To

SpecialCategory Object

## Discussion

Use this property to customize a category that you can then use to track measures for specific periods of time relative to the current time. Relative time categories can specify

- a single period, to show changes from one time period to another
- a sequential set of periods to show to-date totals that compare current performance to past performance
- a relative time period to show running totals, for example, a six-month period before the current date

The Aggregate property uses the values of xtrTimeAggregate.

The properties in the following example specify a running total category that spans the six months leading up to the current month, for this year and last year.

```
Aggregate = trAggregateRunningGrouped
```

```
ContextLevel = "Year"
```

```
ContextOffset = -1
```



```
RunningPeriods = 6
```

```
TargetLevel = "Month"
```

```
TargetOffset = 0
```

## Type

Constant - xtrTimeAggregate

## Access

Read/Write

## Examples

```
objSpecCategory  
= - objModel.Dimensions("Time").Categories.  
  Add(xtrObjectType.trSpecialCategory)objSpecCategory.Aggregate  
= xtrTimeAggregate.trAggregateRunning
```

## AllocationMeasure Property

The AllocationMeasure property returns the Measure object used as a weighting factor.

## Syntax

*object* .**AllocationMeasure**(Measure)

## Applies To

[Category Object](#)

[Dimension Object](#)

[DateLevel Object](#)

[DateDimension Object](#)

[Level Object](#)

[SpecialCategory Object](#)

## Discussion

A measure that is allocated proportionally to descendant categories requires that you use a second measure as a weighting factor. Use this property with the AllocationType property to determine current allocation settings.

For example, you can allocate the value of a fixed costs measure to various regions based on another measure, such as sales for each region.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Measure	Required. Specifies the measure that is proportionally allocated to descendant categories.  Type: Object

## Type

Object - Measure

## Access

Read

## AllocationType Property

---

The AllocationType property returns how an object allocates a measure.

## Syntax

*object* .AllocationType(Measure)

## Applies To

[Category Object](#)

[Dimension Object](#)

[DateLevel Object](#)

[DateDimension Object](#)

[Level Object](#)

[SpecialCategory Object](#)

## Discussion

You can allocate a measure to descendant categories proportionally, as a constant, or not at all. Use this property with the AllocationMeasure property to determine current allocation settings.

The Allocation property uses the values of xtrAllocationType.

Parameter	Description
Measure	Required. Specifies the measure used for, or suppressed from, allocation to descendant categories.  Type: Object - Measure

## Type

Constant - xtrAllocationType

## Access

Read

## Examples

```
objLevel
= objModel.Dimensions("Products").DimensionLevels(1)If
  objLevel.AllocationType(objMeasure)
<> _xtrAllocationType.trAllocationByAnotherMeasure Then
```

## AllowCurrencyConversion Property

---

The AllowCurrencyConversion property sets or returns whether a measure that represents currency can be switched to another currency in PowerPlay.

### Syntax

*Measure* .**AllowCurrencyConversion**

### Applies To

Measure Object

### Discussion

Set this property only after a currency table is set up. The currencies that a PowerPlay user can select are limited to those defined in the CurrencyRecords collection.

**Default:** False

### Type

Boolean

### Access

Read/Write

### Examples

```
objMeasure.CategoryCountLevel = objLevel.AllowCurrencyConversion  
= False
```

## AllowDrillThrough Property

---

The AllowDrillThrough property sets or returns whether a cube or measure can drill through to a cube or report.

### Syntax

*object* .**AllowDrillThrough**

### Applies To

ChildCube Object

Cube Object

CubeGroup Object

Measure Object

### Discussion

Use this property to determine if a cube or measure can drill through to an external file such as an Impromptu report or PowerCube.

First use a Report object to create a drill-through link, and then set the AllowDrillThrough property to True to permit drill-through capability.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objMeasure = objModel.Measures.Item("Quantity")objMeasure.AllowDrillThrough  
= True
```

## AlternateQueryPath Property

---

The AlternateQueryPath property sets or returns the name of an alternate data source for a cube.

## Syntax

*object* .**AlternateQueryPath**(DataSource)

## Applies To

Cube Object

CubeGroup Object

## Discussion

Use this property with the DataSource parameter to instruct Transformer to read data from alternate data sources when you create a cube. You must first use the access tools that come with your database software to set up alternate source files that are structurally identical to the source files on which the model is based. The alternate data source must have the same columns as the original source to be read into the model in Transformer.

For example, you decide to build cubes for different sales divisions from different source files; however, you can use the same sales analysis model, in all cases. Set up several Impromptu query definition files (.iqd), each one containing filters that result in the retrieval of data for only one sales division.

Parameter	Description
DataSource	Required to set the property. Specifies the object used to reference the alternate data source.  Type: Object

## Type

String

## Access

Read/Write

## AltMDCFile Property

---

The AltMDCFile property specifies an alternate filename for the cube.

### Syntax

*object* .AltMDCFile

### Applies To

[Cube Object](#)

[CubeGroup Object](#)

[ChildCube Object](#)

### Discussion

Use this property to specify a filename to use when the cube is in use by another application. For this property to be enabled, the UseAltMDCFile property must be set to true.

You can use the IsMDCInUse property to check if a cube is in use.

This is an optional property. When the UseAltMDCFile property is set to true, the alternate filename can be specified using this property. If this property is not set, then Transformer automatically uses a default alternate name. Transformer appends a digit to the end of the filename starting from one up to one hundred until a filename is accepted.

### Type

String

### Access

Read/Write

## AlwaysUseTransformerSignon Property

---

The AlwaysUseTransformerSignon property specifies whether to use the Transformer Signon or the Content Manager Signon.

### Syntax

*PackageDatasourceConnection* .AlwaysUseTransformerSignon

### Applies To

[PackageDatasourceConnection Object](#)

### Discussion

When true, the Transformer signon takes precedence over the Content Manager signon. If false, the Content Manager signon is used by default.

### Type

Boolean

## Access

Read/Write

## Examples

```
package = model.Packages.Add() connection.AlwaysUseTransformerSignon  
= True
```

## Apex Property

---

The Apex property sets or returns the Category object that is the root for a dimension view.

### Syntax

*View* .**Apex**

### Applies To

View Object

### Discussion

Use this property to limit the categories a user can view in PowerPlay to just the subset they need.

You can create a dimension view and apply the Apex property to a specified category. After you create a cube, it contains only the apex category and the immediate descendants. The ancestors, siblings, and descendants of these siblings are all omitted from the view.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object

### Access

Read/Write

## Examples

```
objProductsDim = objModel.Dimensions.Item("Products") objViewItem.Apex  
= objProductsDim.Drilldowns(1).Categories(1)
```

## Application Property

---

The Application property returns the Transformer Application object.

### Syntax

*object* .**Application**

## **Applies To**

[Associations Collection](#)

[Association Object](#)

[CalculationDefinitions Collection](#)

[CalculationDefinition Object](#)

[Categories Collection](#)

[Category Object](#)

[CategorySets Collection](#)

[CategorySet Object](#)

[Columns Collection](#)

[Column Object](#)

[CrossTabDataSource Object](#)

[Cubes Collection](#)

[Cube Object](#)

[CubeCustomViews Collection](#)

[CubeGroup Object](#)

[ChildCubes Collection](#)

[ChildCube Object](#)

[CurrencyRecords Collection](#)

[CurrencyRecord Object](#)

[CurrencyRates Collection](#)

[CurrencyRate Object](#)

[CurrencyTables Collection](#)

[CurrencyTable Object](#)

[CustomViews Collection](#)

[CustomView Object](#)

[DateDimension Object](#)

[DateLevel Object](#)

[DataSources Collection](#)

[DataSource Object](#)

[DateWizard Object](#)

[DbDataSource Object](#)

[DateDrillDown Object](#)

[Dimensions Collection](#)

[Dimension Object](#)

[DimensionLevels Collection](#)

[DrillDowns Collection](#)

[DrillDown Object](#)

[DrillThroughTargets Collection](#)  
[DrillThroughTarget Object](#)  
[FlatFileDataSource Object](#)  
[Filters Collection](#)  
[Filter Object](#)  
[IqdDataSource Object](#)  
[PackageDatasourceConnections Collection](#)  
[PackageDatasourceConnection Object](#)  
[Packages Collection](#)  
[Package Object](#)  
[Prompts Collection](#)  
[Prompt Object](#)  
[Queries Collection](#)  
[Query Object](#)  
[Levels Collection](#)  
[Level Object](#)  
[LevelDrillDowns Collection](#)  
[Measures Collection](#)  
[Measure Object](#)  
[Model Object](#)  
[Namespaces Collection](#)  
[Namespace Object](#)  
[Names Collection](#)  
[Name Object](#)  
[Reports Collection](#)  
[Report Object](#)  
[SecurityObjects Collection](#)  
[SecurityObject Object](#)  
[SuspendedModels Collection](#)  
[SuspendedModel Object](#)  
[Signons Collection](#)  
[Signon Object](#)  
[SpecialCategory Object](#)  
[Views Collection](#)  
[View Object](#)

## **Discussion**

Use this property to reference properties of the Transformer application object from other objects and collections in the model.



A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object

### Access

Read

## AssociationRole Property

---

The AssociationRole property sets or returns the role of an Association object.

### Syntax

*Association* .**AssociationRole**

### Applies To

Associations Collection

### Discussion

Currency tables, date dimensions, levels, date levels, dimensions, and measures maintain a relationship with their underlying source data through an Associations collection. Each Association object in the collection uses the AssociationRole property to determine the role.

For example, categories in a level may get their source values from one column, such as product\_code, and their label values from another column, such as product\_name. In this case, the collection contains two Association objects, each with a different role.

AssociationRole uses the values of xtrAssociationRole.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object - Associations

### Access

Read/Write

### Examples

```
objAssociation  
= objDrill.ConvergenceLevel.Associations.Add()objAssociation.AssociationRole  
= xtrAssociationRole.trAssociationOrderBy
```

## Associations Property

---

The Associations property returns a collection of Association objects.

## Syntax

*object* .**Associations**

## Applies To

[CurrencyTable Object](#)

[DateDimension Object](#)

[DateLevel Object](#)

[Dimension Object](#)

[Level Object](#)

[Measure Object](#)

## Discussion

Several types of objects in a model maintain a relationship with their underlying source column through an Associations collection. The association sets the source of the values for the object.

For example, categories in a level may get their source values from one column, such as product code, and their label values from another column, such as product name. In this case, the collection contains two Association objects, each with a different role.

For Measure objects, an association is valid only when the object has the MeasureType property set to trRegularMeasure. A measure that derives values from an association cannot use the ExpressionText and CategoryCountLevel properties to derive values. These are mutually exclusive.

## Type

Object

## Access

Read

## Examples

```
objAssociation = objDrill.ConvergenceLevel.Associations.Add()
```

## AssociationType Property

---

The AssociationType property sets or returns whether an Association object has an association with a data source.

## Syntax

*Association* .**AssociationType**

## Applies To

[Association Object](#)

## Discussion

The AssociationType property uses the values of xtrAssociationType.

## Type

Constant - xtrAssociationType

## Access

Read/Write

## Examples

```
objAssociation  
= objDrill.ConvergenceLevel.Associations.Add()objAssociation.AssociationType  
= xtrAssociationType.trAssociationQuery
```

## AutoLogon Property

---

The AutoLogon property, when true, defines whether the application will automatically authenticate to the security namespaces associated with the signon.

## Syntax

*Query*.AutoSummary

## Applies To

[Signon Object](#)

## Discussion

Setting this property to true allows the application to automatically authenticate to the security namespaces associated with the signon.

The AutoLogon property applies only to the CognosSignon type.

## Type

Boolean

## Access

Read/Write

## Examples

```
signon = model.Signons.Add()signon.AutoLogon = True
```

## AutoSummary Property

---

The AutoSummary property sets or returns the auto-summary option for a query.

## Syntax

*query*.AutoSummary

## Applies To

[Query Object](#)

## Discussion

This property should be set for a package. It should not be set for a report.

## Type

Boolean

## Access

Read/Write

## Examples

```
new_package = model.Packages.Add()new_query.AutoSummary  
= True
```

## BlankSubstitute Property

---

The BlankSubstitute property sets or returns the default label for blank categories generated in a level.

## Syntax

*object* .**BlankSubstitute**

## Applies To

[DateLevel Object](#)

[Level Object](#)

## Discussion

If no associated source value exists for a regular category, the string specified by this property is assigned as the default label. This way, you can deliberately include categories with blank values in a data source to preserve a particular category hierarchy.

## Type

String

## Access

Read/Write

## Examples

```
objLocationsDim = objModel.Dimensions("Sales regions")objLevel  
= objLocationsDim.DrillDowns(1).Levels("Employee")objAltDrill  
= objLevel.CreateAlternateDrillDownobjNewLevel  
= objAltDrill.Levels.Add(xtrObjectType.trLevel)objNewLevel.BlankSubstitute  
= "No Value"
```

## BlockParentTotals Property

---

The BlockParentTotals property sets or returns whether parents of excluded children display denied or the total of the non-excluded children.

## Syntax

*Cube* .**BlockParentTotals**

## Applies To

[Cube Object](#)

## Discussion

Use this optional property to ensure that parents of excluded children display a denied value rather than the totals of the non-excluded children. This prevents users from viewing data that is an inaccurate rollup of only the non-excluded categories. Missing values take precedence over denied values and will continue to show zero, N/A (not available), a blank (nothing in the cell), or missing values, depending on how the measures in the cube were designed to handle missing values. This allows the user to distinguish between missing and denied values.

**Default:** False

**Note:** Cubes built in 6.6 or earlier versions will continue to behave as previously regardless of what version they are opened in unless the user sets the BlockParentTotals property to be true after importing the cube into 7.0.

## Type

Boolean

## Access

Read/Write

## Examples

```
objCube = objModel.Cubes.Item(1)objCube.BlockParentTotals  
= True
```

# CacheCrossTabs Property

---

The CacheCrossTabs property sets or returns whether summaries are cached.

## Syntax

*object* .**CacheCrossTabs**

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

Use this property to store summaries in a cube for the initial PowerPlay crosstab. (The initial crosstab shows the first dimension as rows and the second as columns.) This helps optimize access time in PowerPlay for this initial crosstab only.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objCube = objModel.Cubes.Item(1)objCube.CacheCrossTabs  
= True
```

## CalculationDefinitions Property

---

The CalculationDefinitions property returns a collection of CalculationDefinition objects associated with a dimension.

## Syntax

*object* .**CalculationDefinitions**

## Applies To

[DateDimension Object](#)

[Dimension Object](#)

## Discussion

CalculationDefinition objects consist of complex expressions that use the values of one or more categories as part of the calculation.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - CalculationDefinitions

## Access

Read

## Examples

```
calcDef = dimension.CalculationDefinitions.Add()
```

## CAMID Property

---

The CAMID property returns the CAMID of the namespace.

## Syntax

*Namespace* .**CAMID**

## Applies To

[Namespace Object](#)

## Discussion

Use this property to get the CAMID of a namespace.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

String

## Access

Read

## Examples

```
CAMID_of_Namespace = new_namespace.CAMID
```

# CanAllocate Property

---

The CanAllocate property returns whether you can allocate measure values to descendant levels and categories.

## Syntax

*object*.**CanAllocate**

## Applies To

[Category Object](#)

[DateDimension Object](#)

[DateLevel Object](#)

[Dimension Object](#)

[Level Object](#)

[SpecialCategory Object](#)

## Discussion

Use this property to determine whether you can allocate measures from one data source to levels and categories associated with another data source. Allocation distributes data, specified at a summary level of a dimension, to lower levels. You can allocate over

- an entire dimension when the measure appears in a data source that does not reference the dimension
- levels within a dimension when the measure is already specified at a level in that dimension
- categories within levels when the measure is specified to the particular level

For example, sales revenue may be tracked daily, but sales revenue is forecast quarterly. Allocation by proportion may be useful for distributing quarterly forecasts to the month and day levels.

This property is True only when the model uses at least two data sources.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Boolean

## Access

Read

## Examples

```
objLevel = objModel.Dimensions("Products").DimensionLevels(1)If objLevel.CanAllocate  
= True Then
```

## CanAllocateByMeasure Property

---

The CanAllocateByMeasure property returns whether you can use the specified measure as a weighting factor when allocating by measure.

## Syntax

*object* .**CanAllocateByMeasure**(Measure)

## Applies To

[Category Object](#)

[DateDimension Object](#)

[DateLevel Object](#)

[Dimension Object](#)

[Level Object](#)

[SpecialCategory Object](#)

## Discussion

When a measure is allocated proportionally, it uses the values from another measure as a weighting factor. PowerPlay users see the apportioned value in all descendant categories. Use this property to determine if a specified measure can be used as a weighting factor.

Use the SetAllocation method to allocate the values of a measure to all descendant categories. The allocation feature distributes data, specified at a summary level of a dimension, to lower levels.

For example, you allocate the value of a fixed costs measure to various regions based on another measure, such as sales for each region.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Measure	Measure required. Specifies the Measure object used as a weighting factor. Type: Object - Measure



## Type

Boolean

## Access

Read

## Examples

```
objLevel  
= objModel.Dimensions("Products").DimensionLevels(1)If  
  objLevel.CanAllocateByMeasure(objByMeasure)  
= True Then
```

## CanAllocateMeasure Property

---

The CanAllocateMeasure property returns whether you can allocate the specified measure to descendant levels and categories.

## Syntax

*object* .**CanAllocateMeasure**(Measure)

## Applies To

[Category Object](#)

[DateDimension Object](#)

[DateLevel Object](#)

[Dimension Object](#)

[Level Object](#)

[SpecialCategory Object](#)

## Discussion

Use this property to determine if a specific measure is eligible for allocation. If you want to allocate the measure proportionally, you can use the CanAllocateByMeasure property to determine which measures you can use as weighting factors.

Use the SetAllocation method to allocate the values of a measure to all descendant categories. The allocation feature distributes data, specified at a summary level of a dimension, to lower levels.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Measure	Measure required. Specifies the Measure object used as a weighting factor.  Type: Object

## Type

Boolean

## Access

Read

## Examples

```
objLevel  
= objModel.Dimensions("Products").DimensionLevels(1)If  
objLevel.CanAllocateMeasure(objMeasure)  
= True Then
```

# Categories Property

---

## Description

The Categories property returns a Categories collection associated with an object.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Syntax

*object* .**Categories**

## Applies To

[CategorySet Object](#)

[DateDimension Object](#)

[Dimension Object](#)

[DateDrillDown Object](#)

[DrillDown Object](#)

## Discussion

For a CategorySet object, use this property to return the categories used in a calculation.

For a Dimension object, use this property to return the collection of special categories. It also returns calculated categories connected to special categories.

For a DrillDown object, use this property to return all the categories in a level. Use the ChildCategories property to return a collection of child categories.

## Type

Object

## Access

Read

## Examples

```
objCategory = objModel.Dimensions("Retailers").DrillDowns(1).Categories(2)
```

## Category Property

---

The Category property returns the applicable date category for a given currency rate.

### Syntax

*CurrencyRate* .**Category**

### Applies To

CurrencyRate Object

### Discussion

Currency rates must be tied to a date category (day, week, month, quarter, year) to be valid. For example, if rates are set for each month, this property returns the month of the year associated with a specific currency rate.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object

### Access

Read

### Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()objCurrencyRate  
= objCurrencyRecord.CurrencyRates(intX)Select Case objCurrencyRate.Category.KeyName
```

## CategoryCount Property

---

The CategoryCount property returns the number of regular categories in a dimension or level.

### Syntax

*object* .**CategoryCount**

### Applies To

DateDimension Object

DateLevel Object

Dimension Object

Level Object

### Discussion

This property only counts regular categories. Use the SpecialCategoryCount property to count root, drill, and special categories.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read

## Examples

```
objDateDrillDown = objModel.Dimensions("Years").DrillDowns(1)objLevel  
= objDateDrillDown.Levels("Year") For intX = 1 To objLevel.CategoryCount
```

# CategoryCountLevel Property

---

The CategoryCountLevel property sets or returns the Level object to which a category count applies.

## Syntax

*Measure* .**CategoryCountLevel**

## Applies To

Measure Object

## Discussion

You can define a measure that counts categories. For example, you can create a measure that shows how many customers of each type bought a specific product each month, quarter or year.

Use this property to specify which level a category count measure applies to. Transformer excludes non-missing or non-zero categories from the count. The category count becomes the numeric value of the measure.

After this property is set, the MeasureType property returns a value of trCountMeasure. A category count measure derives values from counting categories, and cannot use an association or expression to derive values.

Ensure that the Unique property for the level or date level is set to True before you set this property.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object

## Access

Read/Write

## Examples

```
objMeasure.CategoryCountLevel = objLevelobjMeasure.CategoryCountLevel  
= objLevel
```

## CategorySets Property

---

The CategorySets property returns a collection of CategorySet objects associated with a calculation definition.

### Syntax

*CalculationDefinition* .**CategorySets**

### Applies To

CalculationDefinition Object

### Discussion

A CategorySet object contains one or more categories that are used in a calculation.

When you create a calculation definition, some calculation functions allow one or more category sets to be specified as parameters. For example, the percentage growth function computes the percentage change of the second parameter compared to the first parameter.

### Type

Object - Category Sets

### Access

Read

### Examples

```
catSet = calcDef.CategorySets.Add()
```

## CharacterType Property

---

### Description

The CharacterType property sets or returns the type of character set used by the data source.

### Syntax

*FlatFileDataSource* .**CharacterType**

### Applies To

FlatFileDataSource Object

### Discussion

Use this property to set either the Windows ANSI (ISO 8859-1) or the DOS (OEM) character set.

We recommend that you use the same character set on both the client and the server while developing the client/server model to ensure correct results with extended characters (above 128).

The CharacterType property uses the values of xtrCharacterType.

## Type

Constant - xtrCharacterType

## Access

Read/Write

## Examples

```
objDataSource  
= objModel.DataSources.  
  Add(xtrObjectType.trFlatFileDataSource)objDataSource.CharacterType  
= xtrCharacterType.trCharAnsiSingleByte
```

## ChildCategories Property

---

The ChildCategories property returns a Categories collection.

## Syntax

*object* .**ChildCategories**

## Applies To

[Category Object](#)

[SpecialCategory Object](#)

## Discussion

Each Category object can return a Categories collection from a lower level by using the ChildCategories property.

## Type

Object - Categories

## Access

Read

## Examples

```
objCategory  
= objModel.Dimensions("Retailers").DrillDowns(1).Categories(2)objCatToMove  
= objCategory.ChildCategories(5)
```

## ChildCubes Property

---

The ChildCubes property returns a collection of ChildCube objects from a cube group.

## Syntax

*CubeGroup* .**ChildCubes**

## Applies To

[CubeGroup Object](#)

## Discussion

A cube group represents one level in a dimension. Each child cube in the collection reflects a single category in that level, and includes the descendant categories of that category.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - ChildCubes

## Access

Read

## Examples

```
objCubesByRegion = objModel.Cubes.Add(xtrObjectType.trCubeGroup)objChildCube =  
objCubesByRegion.ChildCubes("Central Europe")
```

# ChildCustomViews Property

---

The ChildCustomViews property returns a collection of CustomView objects.

## Syntax

*CustomView* .**ChildCustomViews**

## Applies To

[CustomView Object](#)

## Discussion

Use this property to create a custom view nested within a custom view. Use the Add method of the CustomViews collection to add a new a custom view. After adding the custom view to the collection, you can then update the properties of the new CustomView object.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

CustomViews

## Access

Read

## Examples

```
currentCustomView = model.CustomViews.Add()childCustomView  
= currentCustomView.ChildCustomViews.Add()
```

## ChildMeasures Property

---

The ChildMeasures property returns a collection of child measures from a measure folder.

### Syntax

*Measure* .ChildMeasures

### Applies To

[Measure Object](#)

### Discussion

Use this property to retrieve all child measures. The collection returned is a MeasureCollection object with the same properties and methods. Using this property with a regular measure will result in an exception.

An empty collection is returned if a measure folder does not contain any child measures. A COM exception is thrown with the message 'trMeasure (ChildMeasures) : The operation is not allowed for this object'.

The MeasureCollection obtained from the Model object will also contain the measure folder and its child measures. Accessing these folders and their children is possible in the same way as any other MeasureCollection; however, a MeasureCollection returned from a Model object will throw an exception if the Move method was applied to a measure folder or any of the child measures of the measure folder.

### Type

Object - Categories

### Access

Read

## Code Property

---

The Code property sets or returns a code that uniquely identifies the category within the entire dimension.

### Syntax

*object* .Code

### Applies To

[Category Object](#)

[SpecialCategory Object](#)



## Discussion

The code is an alphabetic or alphanumeric internal identifier, which is automatically set when Transformer generates categories. You can change the code to make it more descriptive. If you specify a code that already exists, Transformer adds a numerical suffix to create a unique code.

## Type

String

## Access

Read/Write

## Examples

```
objSpecCategory
= - objModel.Dimensions("Time").Categories
  .Add(xtrObjectType.trSpecialCategory)objSpecCategory.Code
= "Five Month Period"
```

# Columns Property

---

The Columns property returns a Columns collection that contains all the Column objects in a data source.

## Syntax

*object* .Columns

## Applies To

[CrossTabDataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

[Query Object](#)

## Discussion

Columns are the source of all data used by Transformer when you create a cube.

Use the ColumnsLoaded property to determine whether the column data has been used to build the dimensions, levels, and measures in a model.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object

## Access

Read

## Examples

```
objDataSource  
= objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)objColumn  
= objDataSource.Columns(1)
```

## ColumnsLoaded Property

---

The ColumnsLoaded property returns whether data source columns have been used to build the model.

### Syntax

*object* .ColumnsLoaded

### Applies To

[CrossTabDataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

### Discussion

A return value of True indicates that Column objects in the model are linked to columns in the underlying data source.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Boolean

### Access

Read

## CompressMDC Property

---

The CompressMDC property sets or returns whether a cube is compressed.

### Syntax

*object* .CompressMDC

### Applies To

[Cube Object](#)

[CubeGroup Object](#)

### Discussion

Use this property to optimize the distribution of a cube.

Transformer compresses the cube each time it is created. When you perform any Transformer operation on a compressed cube, such as an incremental update or checking of the cube status, Transformer decompresses the cube, completes the action, and then compresses it again.

The first time you open a compressed cube in PowerPlay it may take slightly longer to open. After that, the cube resumes its original size and opens normally.

**Default:** False

### Type

Boolean

### Access

Read/Write

### Examples

```
objCube = objModel.Cubes.Item(1)objCube.CompressMDC =  
False
```

## Connection Property

---

The Connection property sets or returns the Content Manager connection.

### Syntax

*PackageDatasourceConnection* .**Connection**

### Applies To

PackageDatasourceConnection Object

### Discussion

Use this property to retrieve the Content Manager connection.

### Type

String

### Access

Read/Write

### Examples

```
package = model.Packages.Add()connection  
= package.PackageDatasourceConnections.Add()connection.Connection  
= "great_outdoors_warehouse"
```

## Consolidate Property

---

The Consolidate property sets or returns how a cube is consolidated.

## Syntax

*object* .**Consolidate**

## Applies To

[ChildCube Object](#)

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

Use this property to reduce cube size and shorten access time in PowerPlay. A consolidated cube rolls up identical, non-measure values into a single record and summarizes measure values.

This property uses the values of `xtrCubeConsolidate`.

Data records may have identical non-measure values when

- the source contains transactions with identical, non-measure values; for example, a customer makes two purchases of the same product on the same day.
- the degree of detail permits it; for example, the associated `DateDegreeofDetail` property is set to `Month`, therefore day values in the source transactions are ignored while cubes are consolidated.
- a dimension is omitted from the cube; for example, two sales of the same product are made at different stores on the same day. If stores are omitted from the cube, the sales records have identical, non-measure values.
- categories in the cube are summarized or suppressed; for example, two sales of the same product are made to the same customer on the same day, but the colors differ. If colors are omitted from the cube by using either of these dimension view options, the sales records have identical non-measure values.

In the first three bulleted scenarios, consolidation uses the value of the `DuplicateRollup` property (if the value is other than none) to combine records with identical values in their non-measure columns. In the last scenario, unless the `TimeStateRollup` property is set to a value other than none for the measure, consolidation uses the value of the `RegularRollup` property to combine records with values made identical by dimension views.

## Type

Constant - `xtrCubeConsolidate`

## Access

Read/Write

## Examples

```
objCube.Consolidate = xtrCubeConsolidate.trConsolidateDefault
```

## Context Property

---

The Context property sets or returns which drill-down path to use for a category sort within a level.

## Syntax

*Association* .**Context**

## Applies To

[Association Object](#)

## Discussion

By default, categories appear in a level, and ultimately in a cube, in the order that they are encountered in the data source. To sort these objects, use an Association object and set the AssociationRole property to trAssociationOrderBy.

Use the Context property to name the drill-down path to which the sort applies. Where a level represents the convergence of two or more drill-down paths, you can apply a different sort order to each one.

Use the OrderByDescending property to specify how the sort is ordered. If you use a column of numeric data to define the sort, use the OrderByStorageType to specify the size of the data type.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object

## Access

Read/Write

## Examples

```
objAssociation = objDrill.ConvergenceLevel.Associations.Add()objAssociation.Context  
= objDrill
```

## ContextLevel Property

---

The ContextLevel property sets or returns a date period.

## Syntax

*SpecialCategory*.ContextLevel

## Applies To

[SpecialCategory Object](#)

## Discussion

Create a special category to track measures for a specific period of time relative to the current date period. Use the ContextLevel and ContextOffset properties to define relative time categories, such as prior year or prior quarter.

Use this property to specify a context for the TargetLevel property or ToDateLevel property. For example, If the target period is set to Month, the context period is either Year or Quarter.

## Type

String

## Access

Read/Write

## Examples

```
objSpecCategory  
= - objModel.Dimensions("Time").Categories  
  .Add(xtrObjectType.trSpecialCategory)objSpecCategory.ContextLevel  
= "Quarter"
```

## ContextOffset Property

---

The ContextOffset property sets or returns the position of the context period relative to the current period.

## Syntax

*SpecialCategory*.ContextOffset

## Applies To

[SpecialCategory Object](#)

## Discussion

Use this property to specify an offset value, relative to the current time period, for the ContextLevel property. For example, if the context period is Year, a context offset value of -1 corresponds to the first prior year.

For an aggregated or grouped time period, the ContextOffset property specifies the offset for the group of special categories.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Short

## Access

Read/Write

## Examples

```
objSpecCategory  
= - objModel.Dimensions("Time").Categories  
  .Add(xtrObjectType.trSpecialCategory)objSpecCategory.ContextOffset  
= -3
```

## ConvergenceLevel Property

---

The ConvergenceLevel property returns the level where a primary drill-down path and alternate drill-down path join or converge.

## Syntax

*object* .**ConvergenceLevel**

## Applies To

[DateDrillDown Object](#)

[DrillDown Object](#)

## Discussion

You can set up alternate drill-down path structures within a dimension to provide different perspectives on the data in PowerPlay. Each such path connects to the primary path at the convergence level. Each category in a convergence level must be unique and unambiguous.

Where a level represents the convergence of two or more drill-down paths, you can apply a different sort order to each one. Each sort requires an Association object with the Context property set to the drill-down path of your choice.

To create an alternate drill-down path, use the CreateAlternateDrillDown method.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object

## Access

Read

## Examples

```
objAssociation = objDrill.ConvergenceLevel.Associations.Add()
```

## Count Property

---

The Count property returns the number of objects in the collection.

## Syntax

*collection* .**Count**

## Applies To

[Associations Collection](#)

[CalculationDefinitions Collection](#)

[Categories Collection](#)

[CategorySets Collection](#)

[ChildCubes Collection](#)

[Columns Collection](#)

[CubeCustomViews Collection](#)

[Cubes Collection](#)

[CurrencyRates Collection](#)  
[CurrencyRecords Collection](#)  
[CurrencyTables Collection](#)  
[CustomViews Collection](#)  
[DataSources Collection](#)  
[DimensionLevels Collection](#)  
[Dimensions Collection](#)  
[DrillDowns Collection](#)  
[DrillThroughTargets Collection](#)  
[Filters Collection](#)  
[LevelCategories Collection](#)  
[LevelDrillDowns Collection](#)  
[Levels Collection](#)  
[Measures Collection](#)  
[Names Collection](#)  
[Namespaces Collection](#)  
[PackageDatasourceConnections Collection](#)  
[Packages Collection](#)  
[Prompts Collection](#)  
[Queries Collection](#)  
[Reports Collection](#)  
[SecurityObjects Collection](#)  
[Signons Collection](#)  
[SuspendedModels Collection](#)  
[Views Collection](#)

## Discussion

A value of zero indicates there are no objects in the collection.

## Type

Long

## Access

Read

## Examples

```
For intX = 1 To objModel.CheckModel.Count
```

## CountryCode Property

---



The `CountryCode` property sets or returns a country or region code.

## Syntax

*CurrencyRecord*.**CountryCode**

## Applies To

[CurrencyRecord Object](#)

## Discussion

Transformer uses the country or region specified in the regional settings of the Windows control panel to determine a default for currency attributes, such as the currency symbol and number of decimal places. Use the `CountryCode` property to override these defaults.

By default, the country or region code is an ISO-3166 three-character code that specifies the country or region where the currency is used, such as ITA for Italy. You can replace these codes if desired.

## Type

String

## Access

Read/Write

## Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()objCurrencyRecord.CountryCode  
= "AUS"
```

# CubeCodePage Property

---

The `CubeCodePage` property sets or returns the cube code page setting for the model used to build the cube. An empty string is returned if the model has the default setting which is the operating system code page.

## Syntax

*Model*.**CubeCodePage**

## Applies To

[Model Object](#)

## Discussion

Use this property to retrieve or change the processing cube code page setting for a model. If you change the Model Processing Code Page setting after the cubes are built, you must delete the cubes built with the original code page and recreate them using the new code page setting. Only valid strings are accepted. If an invalid value is assigned, it is reset to the default setting. The currently supported values are as follows:

- ANSI\_X3.4-1968
- Big5
- EUC-CN

- EUC-KR
- Extended\_UNIX\_Code\_Packed\_Format\_for\_Japanese
- GB\_2312
- GB\_2312-80
- GB18030
- GBK
- hp-roman8
- IBM850
- ISO\_8859-1
- ISO\_8859-1:1987
- ISO\_8859-2:1987
- ISO\_8859-3:1988
- ISO\_8859-4:1988
- ISO\_8859-5:1988
- ISO\_8859-6:1987
- ISO\_8859-7:1987
- ISO\_8859-8:1988
- ISO\_8859-9:1989
- ISO\_8859-10
- ISO\_8859-11
- ISO\_8859-13
- ISO\_8859-14
- ISO\_8859-15
- KS\_C\_5601-1987
- Shift\_JIS
- TIS-620
- UTF-8
- windows-1250
- windows-1251
- windows-1252
- windows-1253
- windows-1254
- windows-1255
- windows-1256
- windows-1257
- windows-874
- windows-936

## **Type**

String

## **Access**

Read/Write

## CubeCreation Property

---

The CubeCreation property sets or returns whether the cube is created.

### Syntax

*object* .CubeCreation

### Applies To

[ChildCube Object](#)

[Cube Object](#)

[CubeGroup Object](#)

### Discussion

Use this property in combination with the Status property as a filter to select a subset of available cubes for creation. For example, you can select all cubes for which warnings were issued.

Typically these two properties are used to select all cubes, cubes that were created previously without errors, or cubes with past creation problems.

The CubeCreation property uses the values of xtrCubeCreation.

### Type

Constant - xtrCubeCreation

### Access

Read/Write

### Examples

```
objCubesByRegion
= objModel.Cubes.Add(xtrObjectType.trCubeGroup)objChildCube.CubeCreation
= xtrCubeCreation.trCubeCreationON
```

## CubeCustomViews Property

---

The CubeCustomViews property returns a collection of CustomView objects.

### Syntax

*object* .CubeCustomViews

### Applies To

[ChildCube Object](#)

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

Use this property to associate a custom view with a cube. You must first use the properties of the CustomView object to define a custom view. Then use the Add method to add a CustomView object to the CubeCustomViews collection of the cube.

Each Cube, CubeGroup, or ChildCube object defines a CubeCustomViews collection.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - CubeCustomViews

## Access

Read

## Examples

```
cube = model.Cubes.Item("Sales and Marketing")cube.CubeCustomViews.Add(custom_view)
```

## Cubes Property

---

The Cubes property returns a collection of Cube and CubeGroup objects.

## Syntax

*Model*.Cubes

## Applies To

Model Object

## Discussion

The collection does not include child cubes. Use the ChildCubes property to return child cubes from a cube group.

## Type

Object - Cubes

## Access

Read

## Examples

```
objCube = objModel.Cubes.Item(1)
```

## CubeStamp Property

---

The CubeStamp property returns a cube creation time stamp.

## Syntax

*object* .**CubeStamp**

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

The CubeStamp value is set internally. If this value does not agree with the one in the model, Transformer sets the PowerCube status to Invalid.

An invalid result may indicate that the cube was created from another model or from an obsolete version of the current model.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read

# CurrencyCountryLabel Property

---

The CurrencyCountryLabel property sets or returns the currency country or region label.

## Syntax

*CurrencyRecord* .**CurrencyCountryLabel**

## Applies To

[CurrencyRecord Object](#)

## Discussion

This is a descriptive name to specify the name of the currency.

## Type

String

## Access

Read/Write

# CurrencyDecimals Property

---

The CurrencyDecimals property sets or returns the number of decimal places used for a currency.

## Syntax

*CurrencyRecord*.**CurrencyDecimals**

## Applies To

[CurrencyRecord Object](#)

## Discussion

Transformer uses the country or region specified in the regional settings of the Windows control panel to determine a default for currency decimal places. The decimal places value may change when you set the `CountryCode` property.

Use the `CurrencyDecimals` property to set the number of decimal places. This overrides the regional setting and the `CountryCode` property.

You must set the `CurrencyFormatOverride` property to `True` before you can change the `CurrencyDecimals` property.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read/Write

## Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()objCurrencyRecord.CurrencyDecimals  
= 2
```

## CurrencyFormatOverride Property

---

The `CurrencyFormatOverride` property sets or returns whether you can override currency attributes.

## Syntax

*CurrencyRecord*.**CurrencyFormatOverride**

## Applies To

[CurrencyRecord Object](#)

## Discussion

Transformer uses the country or region specified in the regional settings of the Windows control panel to determine a default for currency attributes, such as the currency symbol and decimal places. You can set these attributes by using the `CountryCode` property. To specifically override a currency attribute with the `CurrencyDecimals` or `CurrencySymbol` properties, you must first set the `CurrencyFormatOverride` property to `True`.

You can override currency attributes

- when the PowerPlay user computer does not have a country or region defined in the regional settings of the Windows control panel, or when a country or region uses more than one currency.
- to add a descriptive currency symbol. For example, you can prefix the dollar symbol with AUS, CDN, or US, to differentiate each currency as it is displayed in PowerPlay.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objCurrencyRecord
= objModel.CurrencyRecords.Add()objCurrencyRecord.CurrencyFormatOverride
= True
```

## CurrencyIsEMU Property

---

The CurrencyIsEMU property sets or returns whether the record is a European Monetary Union (EMU) currency record.

## Syntax

*CurrencyRecord* .**CurrencyIsEMU**

## Applies To

[CurrencyRecord Object](#)

## Discussion

EMU currencies require a euro table that is distinct from the base table. Use this property to add a currency record to the euro table. When you set this property, you must also set the EMUEntryDate property.

If currency conversion is from an EMU currency to a non-EMU currency, Transformer uses a process of triangulation. Transformer converts from

- the source currency to the base euro currency
- the base euro currency to the base currency
- the base currency to the target currency

The reverse is true when Transformer converts a non-EMU currency to an EMU currency.

When you create a euro table, you must use the LoadCurrencyTable method at separate stages in the process. Use this method after you

- associate each data source column with an association role in the euro table
- add currency records to the euro table

**Default:** False

## Type

Boolean value

## Access

Read/Write

## Examples

```
objCurrencyRec = objModel.CurrencyRecords(1)
objCurrencyRec.CurrencyIsEMU = True
```

## CurrencyIsEuro Property

---

The CurrencyIsEuro property sets or returns whether the currency record is the base euro currency.

### Syntax

*CurrencyRecord* .**CurrencyIsEuro**

### Applies To

[CurrencyRecord Object](#)

### Discussion

Use this property to identify a currency record as the base euro currency in the CurrencyRecords collection. Conversion to or from the euro requires a base value against which all other currency rates in the euro table are pegged.

When you add a euro table to the CurrencyTables collection, Transformer automatically sets the CurrencyIsEuro property of the <Base Euro> record to True. All other records return a value of false.

## Type

Boolean

## Access

Read/Write

## Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()
objCurrencyRecord.CurrencyIsEuro = False
```

## CurrencyRates Property

---

The CurrencyRates property returns a collection of CurrencyRate objects.

### Syntax

*CurrencyRecord* .**CurrencyRates**



## Applies To

[CurrencyRecord Object](#)

## Discussion

CurrencyRate objects represent currency conversion rates within a CurrencyRecord object.

Each object stores a currency rate set to a date detail level, such as Year, Quarter, Month, Week, or Day. Use the Category property of the CurrencyRate object to return the date category associated with a specific currency rate.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - CurrencyRates

## Access

Read

## Examples

```
objCurrencyRate = objCurrencyRecord.CurrencyRates(intX)
```

## CurrencyRecord Property

---

The CurrencyRecord property returns the CurrencyRecord object to which the CurrencyRate object applies.

## Syntax

*CurrencyRate* .**CurrencyRecord**

## Applies To

[CurrencyRate Object](#)

## Discussion

Each CurrencyRecord object represents a row of currency conversion information in a currency table. For example, use the CountryCode property of the CurrencyRecord object to return the country or region code to which the record and currency rate are associated.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - CurrencyRecord

## Access

Read

## CurrencyRecords Property

---

The CurrencyRecords property returns a collection of all CurrencyRecord objects in a model.

### Syntax

*Model* .**CurrencyRecords**

### Applies To

Model Object

### Discussion

Each CurrencyRecord object represents a row of currency conversion information in a CurrencyTable object.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object - CurrencyRecords

### Access

Read

### Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()
```

## CurrencySymbol Property

---

The CurrencySymbol property sets or returns the currency symbol used by a currency.

### Syntax

*CurrencyRecord* .**CurrencySymbol**

### Applies To

CurrencyRecord Object

### Discussion

Transformer uses the country or region specified in the regional settings of the Windows control panel to determine a default for the currency symbol. The symbol usually changes when you set the CountryCode property.

Use the CurrencySymbol property to specifically set the currency symbol, and to override the regional setting and the CountryCode property. For example, you can prefix the dollar symbol with AUS, CDN, or US, to differentiate each currency as it is displayed in PowerPlay.

You must set the CurrencyFormatOverride property to True before you can change the CurrencySymbol property.

## Type

String

## Access

Read/Write

## Examples

```
objCurrencyRecord.CurrencySymbol = "$"
```

## CurrencyTable Property

---

The CurrencyTable property returns a CurrencyTable object related to a currency rate.

### Syntax

*CurrencyRate* .**CurrencyTable**

### Applies To

CurrencyRate Object

### Discussion

The CurrencyTable object contains currency conversion records used for currency conversion. In PowerPlay, currency-specific measure values appear in any currency you define, at the date detail level for which conversion rates are available.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - CurrencyTable

## Access

Read

## CurrencyTables Property

---

The CurrencyTables property returns a collection of CurrencyTable objects.

### Syntax

*Model* .**CurrencyTables**

### Applies To

Model Object

## Discussion

Use this collection to add, select or remove a CurrencyTable object. Transformer uses the currency table information in a CurrencyTable object to make the correct currency rate conversions when users view cubes in PowerPlay.

For conversions to or from EMU currencies, you need two tables. One table must have the CurrencyTableType property set to trCurrencyTableBase and the other must be set to trCurrencyTableEuro. A CurrencyTables collection can only contain one of each type of table.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - CurrencyTables

## Access

Read

## Examples

```
objBaseTable = objModel.CurrencyTables.Add()
```

## CurrencyTableType Property

---

The CurrencyTableType property sets or returns the type of currency table.

## Syntax

*CurrencyTable* .**CurrencyTableType**

## Applies To

CurrencyTable Object

## Discussion

Use this property to specify whether the type of currency table is

- base, a currency table tied to your local currency as determined by regional settings of the Windows control panel
- euro, a table of European Monetary Union (EMU) currencies set to the euro as of January 1, 1999
- other, a customized currency table

For conversions to or from EMU currencies, you need two tables. One table must have the CurrencyTableType property set to trCurrencyTableBase and the other must be set to trCurrencyTableEuro. A CurrencyTables collection can only contain one of each type of table.

The CurrencyTableType property uses the values of xtrCurrencyTableType.

## Type

Constant - xtrCurrencyTableType

## Access

Read/Write

## Examples

```
objBaseTable = objModel.CurrencyTables.Add()objBaseTable.CurrencyTableType  
= xtrCurrencyTableType.trCurrencyTableBase
```

## CurrentModel Property

---

The CurrentModel property returns the Model object that is currently open in Transformer.

### Syntax

*Application* .CurrentModel

### Applies To

Application Object

### Discussion

Use this property if you need to reference the active model in your script.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object - Model

### Access

Read

## CurrentValueIndex Property

---

The CurrentValueIndex property sets or returns the current prompt value index.

### Syntax

*Prompt* .CurrentValueIndex

### Applies To

Prompt Object

### Discussion

Use this property to iterate through prompt values for multi-valued prompts. The index starts at 1.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Integer

## Access

Read/Write

## Examples

```
new_report = model.Reports.Add()new_query = new_report.Queries.Add()new_prompt  
= new_query.Prompts.Add()new_prompt.CurrentValueIndex = index
```

## CustomView Property

---

The CustomView property sets or returns whether a view contains all, some, or none of the categories in a dimension.

### Syntax

*View* .**CustomView**

### Applies To

[View Object](#)

### Discussion

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - CustomView

## Access

Read/Write

## CustomViews Property

---

The CustomViews property returns a collection of CustomView objects.

### Syntax

*model* .**CustomViews**

### Applies To

[View Object](#)

[Model Object](#)

## Discussion

Use this property to create a custom view within a model. Use the Add method of the CustomViews collection to add a new custom view. After adding the custom view to the collection, you can then update the properties of the new CustomView object.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - CustomViews

## Access

Read

## Examples

```
custom_view = model.CustomViews.Add()
```

## DataCharacterSet Property

---

The DataCharacterSet property sets or returns the default character set used by the application.

## Syntax

*Application* .DataCharacterSet

## Applies To

Application Object

## Discussion

Use this property to change the character set. Transformer supports both the Windows ANSI (ISO 8859-1) and DOS (OEM) character sets.

We recommend that you use the same character set on both the client and the server while developing the client/server model to ensure correct results with extended characters (above 128).

The DataCharacterSet property uses the values of xtrCharacterType.

## Type

Constant - xtrCharacterType

## Access

Read/Write

## Examples

```
objTransApp.DataCharacterSet = xtrCharacterType.trCharDefault
```

## DataClass Property

---

The DataClass property sets or returns the type of data in the source column.

### Syntax

*Column* .**DataClass**

### Applies To

Column Object

### Discussion

When you set a value for this property, you can speed up the actions of the DoAutoDesign method. Typically, text, date, and numeric data class types are used to describe structural, time, and performance data respectively.

When you reference a column in the ExpressionText property, that column must have the DataClass property set to a valid entry, otherwise a runtime error occurs.

The DataClass property uses the values of xtrDataClass.

### Type

Constant - xtrDataClass

### Access

Read/Write

### Examples

```
objDataSource
= objModel.DataSources.
  Add(xtrObjectType.trFlatFileDataSource)objDataSource.DataClass
= xtrDataClass.trDataClassDate
```

## DataRange Property

---

The DataRange property sets or returns the name of a database range.

### Syntax

*DbDataSource* .**DataRange**

### Applies To

DbDataSource Object

### Discussion

Use this property to specify the name of a table or block of data as defined in the database source file.



**Type**

String

**Access**

Read/Write

## DataSource Property

---

The DataSource property sets or returns data source in Content Manager.

**Syntax**

*PackageDataSourceConnection* .**DataSource**

**Applies To**

PackageDataSourceConnection Object

**Discussion**

Use this property to get and set the data source located in Content Manager.

**Type**

String

**Access**

Read/Write

## DataSourcePath Property

---

The DataSourcePath property sets or returns the location where Transformer searches for data source files.

**Syntax**

*Application* .**DataSourcePath**

**Applies To**

Application Object

**Discussion**

If no location is specified, Transformer searches the PowerPlay installation folder.

Use the LocalPath property to specify a path and file name for each data source. If the LocalPath property specifies a relative path, it is added to the value set in the DataSourcePath property. In such a case, the concatenated values must result in a properly defined location.

**Type**

String

## Access

Read/Write

## Examples

```
objTransApp.DataSourcePath = strStartLocation
```

## DataSources Property

---

The DataSources property returns a collection of DataSource objects in a model.

### Syntax

*Model* .**DataSources**

### Applies To

Model Object

### Discussion

Use this collection to group all data sources in a model. This group includes generic objects such as DataSource, and specific source objects such as IqdDataSource.

Data sources, except the DataSource object, contain Columns collections. Columns are the source for levels in a dimension.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object

### Access

Read

### Examples

```
objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
```

## DataSourceWindowsLocation Property

---

The DataSourceWindowsLocation property sets or returns the location of the cube, including the full path and cube name.

### Syntax

*Cube* .**DataSourceWindowsLocation**

### Applies To

Cube Object

## Discussion

Use this property to set the location of the cube from the frame of reference of the IBM Cognos Analytics server.

## Type

String

## Access

Read/Write

## Examples

```
cube = model.Cubes.Item("National")cube.DataSourceWindowsLocation
= "c:\NATIONAL\Deployment1\National.mdc"
```

## DataTemporaryFilePath Property

---

The DataTemporaryFilePath property sets or returns the location where Transformer creates temporary work files while it generates cubes.

## Syntax

*Application* .DataTemporaryFilePath

## Applies To

Application Object

## Discussion

Use this property to specify one or more locations in which to process very large cubes when a single location might not provide enough disk space for the files. Delimit the directories with a semi-colon (;).

If a location is not specified, Transformer searches the location (in order):

- specified by the ModelTemporaryFilePath property
- specified by the ModelsPath property
- specified by the PowerCubesPath property
- of the PowerPlay installation directory
- set by Temporary = in the [Services] section of the cs7g.ini file
- set by the TEMP environment variable
- set by the TMP environment variable

## Type

String

## Access

Read/Write

## Examples

```
objTransApp.DataTemporaryFilePath = strStartLocation
```

## DateDegreeofDetail Property

---

The DateDegreeofDetail property sets or returns the date level at which reporting can occur.

### Syntax

*Column* .**DateDegreeofDetail**

### Applies To

Column Object

### Discussion

Use this property to define the date level at which reporting occurs in Transformer. The date level can be day, week, month, quarter, or year.

When a level is set, reporting of measures in the dimension is restricted to that level of detail. The level cannot be lower than the date input set by the DateInputFormat property. For example, if the date input is set to quarter, you cannot set the degree of detail to month.

The DateDegreeofDetail property uses the values of xtrDateLevel. If you assign the constant trDateLevelUnspecified, Transformer sets the degree of detail based on other attributes of the source column.

### Type

Constant - xtrDateLevel

### Access

Read/Write

### Examples

```
objDataSource  
= objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)objDataSource  
  .Columns("EuroDate").DateDegreeofDetail  
= xtrDateLevel.trDateLevelMonth
```

## DateDegreeofDetailLevelName Property

---

The DateDegreeofDetailLevelName property sets or returns the date level that applies to an externally rolled up measure.

### Syntax

*Column* .**DateDegreeofDetailLevelName**

### Applies To

Column Object

## Discussion

If you maintain structural data and transactional data in separate files, you can specify rolled up measure values within the transactional data source files.

If a Column object represents an externally rolled up measure, use this property to specify the date level at which Transformer provides categories. You must specify a level of detail that corresponds to a level in the date dimension.

To use this property you must also set the External property of the DataSource object to True.

## Type

String

## Access

Read/Write

## Examples

```
.Columns("EuroDate").DateDegreeofDetail = xtrDateLevel.trDateLevelMonth
```

## DateFormat Property

---

The DateFormat property sets or returns the format code that Transformer uses to show dates in the level.

## Syntax

*DateLevel* .**DateFormat**

## Applies To

DateLevel Object

## Discussion

For levels in a time dimension, Transformer uses codes for portions of dates (such as, year, month, quarter, and day) to construct how date display formats appear. You can combine the codes to construct custom date formats.

By combining codes, you can create date formats such as

- "YYYY MMM DD", which appears as 1999 Jan 01
- "YY""Q""Q", which appears as 99 Q1

Code	Meaning	Sample (English)
YY	A 2-digit year	99
YYYY	A 4-digit year	1999
Q	A 1-digit quarter indicator	1
MM	A 2-digit month within the year	01

Code	Meaning	Sample (English)
MMM	The abbreviated month name	Jan
MMMM	The full month name	January
DD	A 2-digit day.	01
DDDD	A day of the week	Sunday
/, -, or space character	Separator characters for other date codes	1999/01/01, 1999-01-01
Any quoted string	The string	"AD", displays the abbreviation AD

For lunar years, quarters are labeled Q1-4, months are labeled 1-12/13, and days are labeled 1-28.

## Type

String

## Access

Read/Write

## Examples

```
objDateLevel = objDateDim.DimensionLevels("Month")objDateLevel.DateFormat
= "MMMM, yyyy"
```

# DateFunction Property

The DateFunction property sets or returns which date categories are generated in a level.

## Syntax

*DateLevel* .**DateFunction**

## Applies To

[DateLevel Object](#)

## Discussion

Use this property to create the appropriate date categories from a source column containing dates. For example, specify the trSpecialFunctionMonth constant to generate month categories.

The DateFunction property uses the values of xtrSpecialFunction.

## Type

Constant - xtrSpecialFunction

## Access

Read/Write

## Examples

```
objDateWizard = objModel.DateWizardobjDateDim  
= objDateWizard.CreateDateDimension()objDateLevel  
= objDateDim.DimensionLevels(1)objDateLevel.DateFunction  
= xtrSpecialFunction.trSpecialFunctionLunarQuarter
```

## DateInputFormat Property

---

The DateInputFormat property sets or returns the order of year, month, and day components of input date values.

### Syntax

*Column* .**DateInputFormat**

### Applies To

Column Object

### Discussion

This property uses the values of xtrDateFormat. Use the trPredefined constant when the format is defined in your data source, such as in a spreadsheet that has date-formatted cells.

### Type

Constant - xtrDateFormat

## Access

Read/Write

## Examples

```
objDataSource  
= objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)objDataSource  
  .Columns("EuroDate").DateInputFormat  
= xtrDateFormat.trMDY
```

## DateLevel Property

---

The DateLevel property sets or returns the DateLevel object related to a currency record.

### Syntax

*CurrencyRecord* .**DateLevel**

### Applies To

CurrencyRecord Object

## Discussion

Use this property to specify the level in the time dimension to which the rates in the currency record apply, such as Month, Week, or Day. Transformer creates a collection of CurrencyRate objects based on the date level specified and the range of dates available.

To edit the rates, use the Rate property.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - DateLevel

## Access

Read/Write

## Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()objCurrencyRecord.DateLevel =  
objLevel
```

# DateWizard Property

---

The DateWizard property returns the DateWizard object.

## Syntax

*Model* .DateWizard

## Applies To

[Model Object](#)

## Discussion

Use the DateWizard object to simplify the creation of a time dimension and date levels. Use the CreateDateDimension method to complete the process. This procedure automatically adds a standard set of special (relative time) categories, such as year-to-date.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - DateWizard

## Access

Read

## Examples

```
objDateWizard = objModel.DateWizard
```



## DecimalPoint Property

---

The DecimalPoint property sets or returns the separator character used for a decimal point.

### Syntax

*Object* .**DecimalPoint**

### Applies To

[CrossTabDataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

### Discussion

You cannot use a null character (no character).

### Type

String

### Access

Read/Write

### Examples

```
objDataSource.DecimalPoint = "."
```

## Decimals Property

---

The Decimal property returns the number of decimal places defined in the source metadata.

### Syntax

*Column* .**Decimals**

### Applies To

[Column Object](#)

### Discussion

If a data source, such as a flat file, does not contain this metadata information, use the InputScale and OutputScale properties to scale the measure values that PowerPlay shows at run time.

### Type

Long

## Access

Read

## DefaultCategoryOrderBy Property

---

The DefaultCategoryOrderBy property sets the default sort setting (ascending) for all categories in the model.

### Syntax

*Model* .DefaultCategoryOrderBy

### Applies To

Model Object

### Discussion

Use this property to specify that the model use the Preference setting for ordering all categories in the model; that is, labels are sorted in ascending alphanumeric order.

This property uses the trUsePreference value.

### Type

Constant - xtrDefaultOrderBy

## Access

Read/Write

## DefaultDateFormat Property

---

The DefaultDateFormat property sets or returns the default setting for the DateInputFormat property.

### Syntax

*Application* .DefaultDateFormat

### Applies To

Application Object

### Discussion

Use this property to specify that the DateInputFormat property uses either the date format predefined in the data source or the one defined in the regional settings of the Windows control panel.

When the data source does not define the date format, such as with text files, Transformer uses the YYYYMMDD format.

This property uses the values of xtrPreferences.

### Type

Constant - xtrPreferences

## Access

Read/Write

## Examples

```
objTransApp.DefaultDateFormat = xtrPreferences.trDateFormatFromControlPanel
```

# Description Property

---

The Description property sets or returns a description for the object.

## Syntax

*object* .**Description**

## Applies To

[Association Object](#)

[CalculationDefinition Object](#)

[Category Object](#)

[CategorySet Object](#)

[ChildCube Object](#)

[Column Object](#)

[CrossTabDataSource Object](#)

[Cube Object](#)

[CubeGroup Object](#)

[CustomView Object](#)

[DataSource Object](#)

[DateDimension Object](#)

[DateLevel Object](#)

[DbDataSource Object](#)

[Dimension Object](#)

[DrillThroughTarget Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

[Level Object](#)

[Measure Object](#)

[Model Object](#)

[Signon Object](#)

[SpecialCategory Object](#)

## Discussion

Use this property as internal documentation and to provide information to users through the Explain window in PowerPlay.

For example, when PowerPlay retrieves a description for a calculated measure, the Explain text can describe how the formula for the calculation is derived.

## Type

String

## Access

Read/Write

## Examples

```
objMeasure.Description = "Forcasted volume for product  
line."
```

# DesiredPartitionSize Property

---

The DesiredPartitionSize property sets or returns the preferred partition size for each partition in a cube.

## Syntax

*object* .**DesiredPartitionSize**

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

Use this property as a part of the auto-partition process to set the preferred maximum size for each partition in the cube. The value specified accurately limits the size of the partition that PowerPlay searches to find values when the user drills down or drills up. A smaller partition size indicates faster access to high-level summaries for affected categories in a cube.

To set this property, the Optimize property must specify the trOptimizeDefault or trOptimizeAutoPartition constant. You must also set the cube properties: EstimatedRows and MaxNumPartLevels.

To manually create partitions, use the Partition and MaxNumPartLevels properties.

**Default:** 5% of estimated number of consolidated records.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read/Write

## Examples

```
objCube = objModel.Cubes.Item(1)objCube.DesiredPartitionSize  
= 500000
```

## DetachDataSource Property

---

The DetachDataSource property sets or returns whether the connection to a data source is maintained or released.

### Syntax

*Application* .**DetachDataSource**

### Applies To

[Application Object](#)

### Discussion

If set to True, Transformer releases a data source when it has finished populating a model and creating cubes, and reconnects to the data source when necessary. If set to False, Transformer remains connected to the data source.

**Default:** False

### Type

Boolean

### Access

Read/Write

### Examples

```
objTransApp.DetachDataSource = True
```

## DetailLevel Property

---

The DetailLevel property sets or returns a Level object that defines the lowest detail level for cubes in a cube group.

### Syntax

*CubeGroup* .**DetailLevel**

### Applies To

[CubeGroup Object](#)

### Discussion

If not set, Transformer includes the category from the level specified by the GroupLevel property and all associated descendant categories.

Duplicate names are qualified by the name of the parent category.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object

## Access

Read/Write

## Examples

```
objCubesByRegion
= objModel.Cubes.Add(xtrObjectType.trCubeGroup)objCubesByRegion.DetailLevel
= objRegionsDrill.Levels("Branch")
```

## Dimension Property

---

The Dimension property returns a dimension for a Category or SpecialCategory object.

### Syntax

*Object* .**Dimension**

### Applies To

[Category Object](#)

[SpecialCategory Object](#)

### Discussion

A category or special category belongs to only one dimension.

### Type

Object - Dimension

### Access

Read

## DimensionInclude Property

---

The DimensionInclude property sets or returns the type of view for a custom view.

### Syntax

*CustomView* .**DimensionInclude**(Dimension)

### Applies To

[CustomView Object](#)

## Discussion

Each CustomView object automatically includes a collection of View objects: one for each dimension. The default view for each object in a collection is 'All Categories'. Use this property to omit all categories or create a custom view.

When you set this property, you must include a Dimension object as a parameter. This parameter specifies the dimension to which the view applies.

To define a view so that all categories in a dimension are omitted from a cube, set this property to trViewTypeOmitDimension.

Custom views provide a specific subset of cube information to PowerPlay users. A custom view may summarize an entire level or individual categories within the level. If you specify a custom view (trViewTypeCustom), you must also use the DimensionView property to return the View object associated with a Dimension. You can then use the Apex property to create a new root category, or the SetViewStatus method to associate the View object with selected levels or categories.

To complete a custom view, you must associate a custom view with a cube. Use the Add method to add a CustomView object to the CubeCustomViews collection of the cube.

The DimensionInclude property returns a constant from the xtrViewType value list.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Dimension	Required. Specifies the dimension to which the view applies.  Type: Dimension

## Type

Constant - xtrViewType

## Access

Read/Write

## Examples

```
custom_view = model.CustomViews.Add()custom_view.DimensionInclude(dimension) =  
xtrViewType.trViewTypeCustom
```

## DimensionLevels Property

---

The DimensionLevels property returns a DimensionLevels collection.

## Syntax

*object*.DimensionLevels

## Applies To

[DateDimension Object](#)

[Dimension Object](#)

## Discussion

The Levels collection contains a list of unique levels in the dimension. In cases where a dimension has alternate drill-down paths, an individual level may be included more than once in the dimension; however, this collection contains only one reference to that level, no matter how often it appears in the dimension.

When a level is in more than one drill-down path, use the Context property to specify the path.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - DimensionLevels

## Access

Read

## Examples

```
objLevel = objModel.Dimensions("Products").DimensionLevels(1)
```

## DimensionName Property

---

The DimensionName property sets the name of a new DateDimension object.

## Syntax

*DateWizard* .**DimensionName**

## Applies To

DateWizard Object

## Discussion

If you do not specify a name for the time dimension, Transformer generates an error.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

String

## Access

Write

## Examples

```
objDateWizard.DimensionName = "Dates"
```

## Dimensions Property

---



The Dimensions property returns a collection of all Dimension and DateDimension objects in a model.

## Syntax

*Model* .**Dimensions**

## Applies To

[Model Object](#)

## Discussion

After the collection is returned, you can use the Item method to return a specific dimension and then modify properties.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - Dimensions

## Access

Read

## Examples

```
objTimeDimension = objModel.Dimensions.Item("Time")
```

# DimensionView Property

---

The DimensionView property sets or returns the View object associated with a cube or custom view.

## Syntax

*object* .**DimensionView**(Dimension)

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

[CustomView Object](#)

## Discussion

When you set this property, you must include a Dimension object as a parameter. This parameter specifies the dimension to which the view applies.

Use the DimensionView property when you create customized views of a dimension. Custom views provide a specific subset of cube information to PowerPlay users. For example, a custom view may summarize an entire level or selected category hierarchies within the level.

Each Dimension object automatically includes a collection of two View objects. By default, the DimensionView property is set to the first object in the Views collection, 'All Categories'. To omit a dimension from a cube, set the DimensionView property to the second object in the collection, 'Omit

Dimension'. To create a custom view for a dimension, use the Add method to add a View object to the collection, and then set the ViewType property to trViewTypeCustom.

Use the Apex property to create a new root category, or the SetViewStatus method to associate the View object with selected levels or categories. You can then use the newly defined View object to set the DimensionView property.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Custom Views

If you use the DimensionInclude property to specify a custom view, you must then use the DimensionView property to return the View object associated with a Dimension. You can then use the Apex property to create a new root category, or the SetViewStatus method to associate the View object with selected levels or categories.

To complete a custom view, you must associate a custom view with a cube. To do this, use the Add method to add a CustomView object to the CubeCustomViews collection of the cube.

Parameter	Description
Dimension	Required. Specifies the dimension for which a custom view is defined.  Type: Object

## Type

Object

## Access

Read/Write (Cube)

Read (CustomView)

## Examples

```
objCube = objModel.Cubes("Sales and Marketing")objCube.DimensionView(objDimension)  
= objView
```

## DimensionViewType Property

The DimensionViewType property returns the type of view assigned to a dimension in a cube.

## Syntax

*object* .**DimensionViewType**(Dimension [, View])

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

Use this property to determine if some, none or all categories in the dimension are included.

The DimensionViewType property uses values of xtrViewType.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
Dimension	Required. Specifies a dimension. Type: Object
View	Optional. Specifies a view of type trViewTypeCustom. Type: Object

## Type

Constant - xtrViewType

## Access

Read/Write

## Examples

```
objCube  
= objModel.Cubes("Sales and Marketing")If objCube.  
  DimensionViewType(objDimension)  
= _xtrViewType.trViewTypeAllCategories Then
```

## DisplayName Property

---

The DisplayName property sets or returns the name to display for the SecurityObject object.

## Syntax

*SecurityObject* .**DisplayName**

## Applies To

[SecurityObject Object](#)

## Discussion

Use this property to set the display name shown in the Transformer user interface under Security Objects.

## Type

String

## Access

Read/Write

## Examples

```
new_namespace.DisplayName = Name_of_Object
```

## DrillCode Property

---

The DrillCode property sets or returns a code that uniquely identifies the drill-down category within the entire dimension.

### Syntax

*object* .DrillCode

### Applies To

[DateDrillDown Object](#)

[DrillDown Object](#)

### Discussion

The code can be any alphabetic or alphanumeric identifier (for example, By Product Line) and is handled by Transformer as a string data type.

### Type

String

### Access

Read/Write

### Examples

```
objDateDim = objDateWizard.CreateDateDimension()  
objDrillDown = objDateDim.DrillDowns(1) objDrillDown.DrillCode = "By Dates"
```

## DrillDowns Property

---

The DrillDowns property returns a collection of either DateDrillDown objects or DrillDown objects.

### Syntax

*object* .DrillDowns

### Applies To

[DateDimension Object](#)

[Dimension Object](#)

### Discussion

A DateDrillDown object and DrillDown object both define a drill-down path.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - DrillDowns

## Access

Read

## Examples

```
objDrill = objModel.Dimensions("Retailers").DrillDowns(2)
```

## DrillInclusion Property

---

The DrillInclusion property sets or returns whether a drill-down path is included in a cube.

## Syntax

*object*.**DrillInclusion**

## Applies To

[DateDrillDown Object](#)

[DrillDown Object](#)

## Discussion

In PowerPlay, an included drill-down category appears as a folder that contains the child categories associated with that drill-down path. A suppressed drill-down category does not appear in PowerPlay, but the child categories do appear.

The DrillInclusion property uses only three of the constants from the xtrInclusion value list:

- trInclusionDefault (shows the drill-down category)
- trInclusionSuppress (hides the drill-down category)
- trInclusionGenerate (shows the drill-down category)

## Type

Constant - xtrInclusion

## Access

Read/Write

## Examples

```
objDrillDown = objDateDim.DrillDowns(1)objDrillDown.DrillInclusion  
= xtrInclusion.trInclusionSuppress
```

## DrillThroughTargets Property

---

The DrillThroughTargets property returns a collection of drill-through target objects associated with a measure or cube (child cube or cube group).

## Syntax

*Object* .DrillThroughTargets

## Applies To

[ChildCube Object](#)

[Cube Object](#)

[CubeGroup Object](#)

[Measure Object](#)

## Discussion

Use this property to iterate through the drill-through targets for a Model object, add a new drill-through target, and change or remove a drill-through target.

The DrillThroughTargets property can represent the following:

- PowerCubes (.mdc)
- PowerPlay reports (.ppr, .ppx)
- Impromptu Web Query and Cognos Query files(.iwq, iqd)
- Impromptu reports (.imr)
- local third-party OLAP sources (MS SSOS .cub files, Essbase Linked Partitions or Reporting Objects)
- associated files (.mac, .doc, .xls, .ppt)

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - DrillThroughTargets

## Access

Read

## Examples

```
objReport = objMeasure.DrillThroughTargets.Add(strReportPath,
"Default Report")
```

## DuplicateRollup Property

---

The DuplicateRollup property sets or returns how measure values from consolidated records are rolled up.

## Syntax

*Measure* .DuplicateRollup

## Applies To

[Measure Object](#)

## Discussion

Use this property to specify the function that Transformer applies when it summarizes duplicate records in the source data. For example, when this property is set to `trDuplicateRollupSum`, the rollup for the consolidated record is the sum of the measure values in the duplicate records. Transformer performs the duplicate rollup before the regular rollup.

When set to a value other than `trDuplicateRollupNone`, this property overrides the Consolidate property setting of `trConsolidateNO` and forces Transformer to consolidate the data source file when generating cubes.

Ensure that you set the `DuplicateWeight` property when the `DuplicateRollup` property is set to `trDuplicateRollupAverage`.

This property uses the values of `xtrDuplicateRollup`.

## Type

Constant - `xtrDuplicateRollup`

## Access

Read/Write

## Examples

```
objMeasures = objModel.MeasurescurrentMeasure  
= objMeasures("Revenue")currentMeasure.DuplicateRollup  
= xtrDuplicateRollup.trDuplicateRollupAverage
```

## DuplicateWeight Property

---

The `DuplicateWeight` property sets or returns the name of the measure that contains weighting factors.

## Syntax

*Measure* .**DuplicateWeight**

## Applies To

[Measure Object](#)

## Discussion

Use this property to return a weighted average for the rollup measure instead of a true average. The weighted average of measure A, which uses measure B as a weight, is calculated using an equation similar to the following:

$$\sum(A_i * B_i) / \sum B_i$$

The measure that is rolled up must be set to `trDuplicateRollupAverage`.

## Type

String

## Access

Read/Write

## Examples

```
objMeasures = objModel.MeasurescurrentMeasure = objMeasures("Revenue")currentMeasure  
currentMeasure.DuplicateWeight = objModel.Measures("Quantity").Name
```

## EarliestDate Property

---

The EarliestDate property sets or returns the earliest date in a range of date categories.

### Syntax

*object* .**EarliestDate**

### Applies To

[DateDimension Object](#)

[DateWizard Object](#)

### Discussion

Use this property and the LatestDate property to limit the range of acceptable date categories included in a time dimension. You specify a range to eliminate categories, such as previous years, which are irrelevant to PowerPlay users.

To generate date categories only, use the GenerateTimePeriod property (DateDimension object) or GenerateDates property (DateWizard object).

If Transformer encounters date values outside of your specified range, it generates an Early Dates or a Late Dates category, or both, depending on when the out-of-range dates occur. If Transformer encounters dates that are neither early nor late, but cannot be placed within the specified range, it generates an Invalid Dates category.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Long

### Access

Write (DateWizard)

Read/Write (DateDimension)

### Examples

```
objDateWizard = objModel.DateWizardobjDateWizard.EarliestDate  
= 19930101
```

## EMUEntryDate Property

---



The EMUEntryDate property sets or returns the date on which euro triangulation calculations begin for a currency.

## Syntax

*CurrencyRecord* .**EMUEntryDate**

## Applies To

[CurrencyRecord Object](#)

## Discussion

The date format for this property is YYYYMMDD. To use this property you must set the CurrencyIsEMU property to True.

**Default:** 19990101

## Type

String

## Access

Read/Write

## Examples

```
objCurrencyRec = objModel.CurrencyRecords(1) objCurrencyRec.EMUEntryDate  
= "19990101"
```

# EnableMessageLogging Property

---

The EnableMessageLogging property sets or returns whether Transformer messages are written to a log file.

## Syntax

*Application* .**EnableMessageLogging**

## Applies To

[Application Object](#)

## Discussion

Use this property to create a log file. After this property is set to True, you can use the

- LogErrorLevel property to specify the severity of error messages logged
- LogFileName property to specify the name of the log file
- LogFilesPath property to set the location to which log files are written

**Default:** True

## Type

Boolean

## Access

Read/Write

## Examples

```
objTransApp.EnableMessageLogging = True
```

## EnableTimePeriod Property

---

The EnableTimePeriod property sets the level of detail for a time dimension.

## Syntax

*DateWizard* .**EnableTimePeriod**(TimeType)

## Applies To

DateWizard Object

## Discussion

Use this property to specify which date levels you add to a time dimension, for example, Year, Quarter, and Month. If you do not use this property, Transformer makes assumptions based on the data source.

The TimeType parameter indicates the level of detail for the date. For example, if TimeType is set to trTimeTypeMonth, the current period shows the year down to the month.

This property uses values of xtrTimeType.

Parameter	Description
TimeType	Required. Provides a value of xtrTimeType. Type: Constant

## Type

Boolean

## Access

Write

## Examples

```
objDateWizard  
= objModel.DateWizardobjDateWizard.EnableTimePeriod(xtrTimeType.trTimeTypeYear)  
= True
```

## EstimatedRows Property

---

The EstimatedRows property sets or returns an estimate of the number of records that the cube will contain.

## Syntax

*object* .**EstimatedRows**

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

Transformer uses the estimate you provide and then updates the value in the cube generation process to reflect the actual number of consolidated records added to the cube.

To set this property, the Optimize property must specify the `trOptimizeDefault` or `trOptimizeAutoPartition` constant. You must also set the cube properties: `DesiredPartitionSize` and `MaxNumPartLevels`.

To manually create partitions, use the `Partition` and `MaxNumPartLevels` properties.

**Default:** 10,000,000

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read/Write

## Examples

```
objCube = objModel.Cubes.Item(1)objCube.EstimatedRows  
= 100000000
```

## ExcludeAutoPartition Property

---

The `ExcludeAutoPartition` property sets or returns whether a specified dimension is excluded from the auto-partition process.

## Syntax

*object* .**ExcludeAutoPartition**

## Applies To

[DateDimension Object](#)

[Dimension Object](#)

## Discussion

If this property is set to `True`, no partition levels are created for the dimension when the cube is created.

Partitions summarize the cube data in several subordinate partitions for faster retrieval in PowerPlay. Because partitioning may increase the time needed to create the cube in Transformer, use the `ExcludeAutoPartition` property to exclude a dimension that does not require this type of optimization.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objDimensions = objModel.DimensionsobjLocationsDim  
= objDimensions.Item(3)objLocationsDim.ExcludeAutoPartition  
= True
```

## ExpressionText Property

---

The ExpressionText property sets or returns an expression used in a mathematical or string operation. In the current release, if-then-else conditional expressions are now supported.

## Syntax

*object* .**ExpressionText**

## Applies To

[CalculationDefinition Object](#)

[Category Object](#)

[Column Object](#)

[Measure Object](#)

[SpecialCategory Object](#)

## Discussion

Use this property to specify the elements used to build an expression. An expression can consist of a function and mathematical operators, and usually refers to measures, columns, or categories as a source of values.

Expressions can be text, date, or numeric, depending on the data type of the object referenced. Transformer evaluates numeric expressions according to standard mathematical rules.

When you reference a column in the ExpressionText property, that column must have the DataClass property set to a valid entry, otherwise a runtime error occurs. Invalid settings are trDataClassIgnore and trDataClassDefault where the default returns results in an unspecified data class.

After the expression is complete, you can use the IsExpressionValid property to determine if all components of the expression are valid.

When used with a Measure object, the ExpressionText property defines a measure as a calculated measure.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

String

## Access

Read/Write

## Examples

```
calcDef = dimension.CalculationDefinitions.Add()calcDef.ExpressionText
= "share ( catset ( "Set 1" ) , "" & _           parentCategory.Code
& "" )"
```

## External Property

---

The External property sets or returns whether the data source contains presummarized values.

## Syntax

*object* .**External**

## Applies To

[CrossTabDataSource Object](#)

[DataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

## Discussion

When you maintain structural data and transactional data in separate files, you can presummarize measure values within the transactional data source files.

Use the structural data sources to define your dimensions. For each transactional data source, set the External property to True and then add the measures to the Measures collection. Set the RegularRollup property of each Measure object to trRollupExternal. Then, use the AssociateWith method of each applicable dimension to specify a column from a transactional data source. The column must correspond to one in the structural data source.

If the model allocates measures, use the DateDegreeofDetailLevelName property to specify the degree of detail.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objDataSource  
= objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)objDataSource.External  
= False
```

## FieldSeparator Property

---

The FieldSeparator property sets or returns the type of field delimiter used in a data source.

### Syntax

*FlatFileDataSource* .**FieldSeparator**

### Applies To

FlatFileDataSource Object

### Discussion

Use this property to specify the character that acts as the field delimiter in a data source. Transformer examines only the first byte in double-byte and multi-byte characters to determine whether a character in the input stream matches the delimiter character you specify.

This property only applies to flat files with SourceType property set to trFlatFile or trFlatFileColumnNames.

Standard delimiters, such as the comma, semicolon, or space character, remain the same across character sets. If the source data is rendered in the DOS Code Page character set (OEM), Transformer converts the delimiter character to the OEM character set.

### Type

String

### Access

Read/Write

### Examples

```
objDataSource  
= objModel.DataSources.  
    Add(xtrObjectType.trFlatFileDataSource)objDataSource.FieldSeparator  
= ","
```

## FileName Property

---

The FileName property returns the name of a model file as it appears in a Windows folder or Windows Explorer.

### Syntax

*Model* .**FileName**

## Applies To

[Model Object](#)

## Discussion

Use this property to show the name of a model file or to write the name to a log file.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

String

## Access

Read

## Examples

```
MsgBox(objModel.FileName & " " & .Size &  
" " & .Time)
```

# Filters Property

---

The Filters property returns the collection of Filter objects associated with a Query.

## Syntax

*Query*.Filters

## Applies To

[Query Object](#)

## Discussion

Use this property to iterate through a Query object's filters, add a new filter, and change or remove a filter.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Filter

## Access

Read

## Examples

```
new_package = model.Packages.Add()  
new_query = new_package.Queries.Add()  
new_filter = new_query.Filters.Add()
```

## FindCategoryByCatCode Property

---

The FindCategoryByCatCode property returns the category object that contains the specified category code string.

### Syntax

*Dimension* .FindCategoryByCatCode

### Applies To

Dimension Object

### Discussion

Parameter	Description
CategoryCode	Required. The Category code name. Type: String

### Type

Object - Category

### Access

Read

## Format Property

---

The Format property sets or returns the format for numeric values.

### Syntax

*object* .Format

### Applies To

Category Object

Measure Object

SpecialCategory Object

### Discussion

Use this property to specify a format for numeric values. PowerPlay uses the formatted values in PowerPlay reports. You must enclose the format string in quotation marks ("").

To add decimal places to a measure value, use the FormatDecimals property.

The available formats are listed below. Several of the format features, such as the currency symbol and thousands separator, may change depending on the properties you set in the regional settings of the Windows control panel.



Format	Example
0	1000000
#,##0	1,000,000
\$0	\$1000000
\$#,##0	\$1,000,000
0%	100%
%0	%100
0E+00	1E+09
0K	1000K
#,##0K	1,000K
K0	K1000
K#,##0	K1,000
\$0K	\$1000K
\$#,##0K	\$1,000K
0M	1000M
#,##0M	1,000M
M0	M1000
M#,##0	M1,000
\$0M	\$1000M
\$#,##0M	\$1,000M

## Type

String

## Access

Read/Write

## Examples

```
objMeasures
= objModel.MeasurescurrentMeasure = objMeasures("Revenue")currentMeasure.Format
= "$#,##0"
```

## FormatDecimals Property

---

The FormatDecimals property sets or returns the number of decimal places PowerPlay shows for a measure or calculated categories.

### Syntax

*Object* .**FormatDecimals**

### Applies To

Category Object

Measure Object

### Discussion

Use this property as a format setting.

Range: 0 to 9.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Long (Measure Object), Int (Category Object)

### Access

Read/Write

### Examples

```
objMeasures = objModel.MeasurescurrentMeasure  
= objMeasures("Revenue")currentMeasure.FormatDecimals  
= 2
```

## FullName Property

---

The FullName property returns the location of a model file.

### Syntax

*Model* .**FullName**

### Applies To

Model Object

### Discussion

Use this property to show the name and location of a model file or to write the fully-qualified file name to a log file.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

String

## Access

Read

# GenerateCategories Property

---

The GenerateCategories property sets or returns whether categories are generated for the data source.

## Syntax

*object*.**GenerateCategories**

## Applies To

[CrossTabDataSource Object](#)

[DataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

[Query Object](#)

## Discussion

When this property is set to False, Transformer ignores the data source each time categories are generated. Set this property to False when you

- have multiple data sources and you want to exclude category generation for a data source
- are incrementally updating other data sources
- have a currency rates data source for which you do not require date category generation

You can also use the GeneratePowerCube property to control how Transformer references data sources.

**Default:** True

## Type

Boolean

## Access

Read/Write

## Examples

```
objDataSource  
= objModel.DataSources.  
  Add(xtrObjectType.trFlatFileDataSource)objDataSource.GenerateCategories  
= True
```

## GenerateDateCategories Property

---

The GenerateDateCategories property sets or returns whether a date level generates date categories.

### Syntax

*DateLevel* .**GenerateDateCategories**

### Applies To

DateLevel Object

### Discussion

To generate categories for a date level, set this property to trGenerateDatesAll. To suppress date category generation for a level, set this property to trGenerateDatesNone.

This property uses values of xtrDateCategoriesGeneration.

### Type

Constant - xtrDateCategoriesGeneration

### Access

Read/Write

### Examples

```
objDateDim = objModel.Dimensions("Date")objDateDim.GenerateDateCategories  
= xtrDateCategoriesGeneration.trGenerateDatesAll
```

## GenerateDates Property

---

The GenerateDates property sets whether the DateWizard object generates date categories.

### Syntax

*DateWizard* .**GenerateDates**

### Applies To

DateWizard Object

### Discussion

Use this property to generate date categories.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Boolean

## Access

Write

## GeneratePowerCube Property

---

The GeneratePowerCube property sets or returns when a data source is referenced by a model.

### Syntax

*object* .**GeneratePowerCube**

### Applies To

[CrossTabDataSource Object](#)

[DataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

[Query Object](#)

### Discussion

This property lets you determine if a specific data source is referenced during category generation, during cube creation, or both. In models with multiple data sources, you can limit access to some data sources to avoid unnecessary processing and to optimize model efficiency.

The GeneratePowerCube property uses the values of xtrPowerCubeGeneration.

You can also use the GenerateCategories property to control which data sources Transformer references during category creation.

### Type

Constant - xtrPowerCubeGeneration

## Access

Read/Write

### Examples

```
objDataSource  
= objModel.DataSources.Add(xtrObjectType  
    .trFlatFileDataSource)objDataSource.GeneratePowerCube  
= xtrPowerCubeGeneration.trGenerationDefault
```

## GenerateTimePeriod Property

---

The GenerateTimePeriod property sets or returns category generation options for a time dimension.

### Syntax

*DateDimension* .**GenerateTimePeriod**(TimeType)

## Applies To

[DateDimension Object](#)

## Discussion

Use this property to specify the degree of detail for dates before you generate categories for a time dimension. The `TimeType` parameter sets the level at which Transformer allows reporting to occur from time-related columns in a data source, which can be year, quarter, month, week, or day.

Use the `EarliestDate` and `LatestDate` properties to specify a range for the date categories generated.

The `TimeType` parameter must be one of the constants of `xtrTimeType`. The property sets or returns one of the constants of `xtrGenerateOptions`.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
<code>TimeType</code>	Required. Specifies a value of <code>xtrTimeType</code> . Type: Constant

## Type

Constant - `xtrTimeType`

## Access

Read/Write

## Examples

```
objDateDim  
= objModel.Dimensions("Date")objDateDim.  
GenerateTimePeriod(xtrTimeType.trTimeTypeMonth)  
= _      xtrGenerateOptions.trGenerateAll
```

## Group Property

---

The `Group` property sets or returns whether calculated categories are grouped together.

## Syntax

*CalculationDefinition* .**Group**

## Applies To

[CalculationDefinition Object](#)

## Discussion

When set to `True`, calculated categories at each level are grouped together in a PowerPlay report. Transformer groups calculated categories when the expression applies to all categories in a level.

When set to `False`, each calculated category in the report is adjacent to the category to which it refers.

**Default:** `True`

## Type

Boolean

## Access

Read/Write

## Examples

```
calcDef = dimension.CalculationDefinitions.Add() calcDef.Group  
= False
```

## GroupDimension Property

---

The GroupDimension property sets or returns the dimension used to build a cube group.

## Syntax

*CubeGroup* .GroupDimension

## Applies To

[CubeGroup Object](#)

## Discussion

A cube group refers to a single dimension. Use this property to specify what that dimension is, and then use the GroupLevel property to define the child cubes: one for each category in the level specified.

You then use the DetailLevel and SummaryLevel properties to define the degree of detail so that PowerPlay users see only the most relevant data.

For example, your organization has four regional sales offices. You create a cube group for each office manager, so that they can track their office sales performance in the last quarter. Because the managers do not need to see the details from the other three offices, you define the group so that it displays only summary data for these offices.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object

## Access

Read/Write

## Examples

```
objCubesByRegion  
= objModel.Cubes.Add(xtrObjectType.trCubeGroup)objCubesByRegion.GroupDimension  
= objModel.Dimensions("Sales region")
```

## GroupLevel Property

---

The GroupLevel property sets or returns a level used to define the individual cubes in a cube group.

## Syntax

*CubeGroup* .GroupLevel

## Applies To

[CubeGroup Object](#)

## Discussion

A cube group refers to a single dimension. Use the GroupDimension property to specify what that dimension is, and then use this property to define the child cubes: one for each category in the level specified.

You then use the DetailLevel and SummaryLevel properties to define the degree of detail so that PowerPlay users see only the most relevant data.

For example, your organization has four regional sales offices. You create a cube group for each office manager, so that they can track their office sales performance in the last quarter. Because the managers do not need to see the details from the other three offices, you define the group so that it displays only summary data for these offices.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object

## Access

Read/Write

## Examples

```
objCubesByRegion  
= objModel.Cubes.Add(xtrObjectType.trCubeGroup)objCubesByRegion.GroupLevel  
= objRegionsDrill.Levels("Sales region")
```

# HasSubdimension Property

---

The HasSubdimension property returns whether a level contains a subdimension.

## Syntax

*object* .HasSubdimension

## Applies To

[DateLevel Object](#)

[Level Object](#)

## Discussion

Subdimensions provide different levels of detail for specific categories, also known as unbalanced hierarchies. Categories in a subdimension are independent of levels in other parts of the dimension.



Changes made to levels outside the subdimension do not affect categories in a subdimension. The reverse is also true.

For example, some branch offices may report product sales down to the item level, whereas others may report only to the product level. You may have a subdimension for those branches that report to the item level.

### **Type**

Boolean

### **Access**

Read

## **HideValue Property**

---

The HideValue property specifies whether to hide the value of a category object. When the property is set to True then the value associated with that category is hidden.

We recommend that you set this option for the parent of categories in a scenario dimension, in conjunction with using the SetDefaultCategory method. The property is applied in the same way using OLE as it is on the user interface.

Default: False

### **Syntax**

*Category* .HideValue

### **Applies To**

Category Object

### **Access**

Read/Write

## **ID Property**

---

The ID property sets or returns the namespace ID.

### **Syntax**

*Namespace* .ID

### **Applies To**

Namespace Object

### **Discussion**

Use this property to set the namespace ID.

## Type

String

## Access

Read/Write

## Examples

```
new_namespace.ID = ID_of_Namespace
```

## IgnoreMissingValue Property

---

The IgnoreMissingValue property in Transformer Series 7 Version 4 and subsequent releases, specifies whether to ignore null or missing time state rollup values of type Average or Weighted Average. When the property is set to True then the value associated with that measure is not included in the rollup calculation.

Default: False

**Note:** This property cannot be set for time state rollups of type First Period, Last Period, or Current® Period. In those cases, null and missing values are always included in the rollup calculation. Missing (null) data values are always excluded from Min and Max calculations for rollups, whether they are set by Transformer to display as '0' or 'n/a' (the NA setting). The property is applied in the same way using OLE as it is on the user interface.

## Syntax

*Measure* .IgnoreMissingValue

## Applies To

Measure Object

## Type

Boolean

## Access

Read/Write

## Examples

```
objMeasures = objModel.MeasurescurrentMeasure  
= objMeasures("Revenue")currentMeasure.IgnoreMissingValue  
= False
```

## Inclusion Property

---

The Inclusion property sets or returns when a category is included in a cube.

## Syntax

*object* .Inclusion

## Applies To

[Category Object](#)

[DateLevel Object](#)

[Level Object](#)

[SpecialCategory Object](#)

## Discussion

Use this property to specify which categories are included in a cube. For example, you can exclude categories with a blank source value.

When you set the Inclusion property of a Level object, it applies to all categories in the level that are set to `trInclusionDefault`.

When you set the Inclusion property of a Category object, it applies to that category only.

The Inclusion property uses the values of `xtrInclusion`.

## Type

Constant - `xtrInclusion`

## Access

Read/Write

## Examples

```
objLocationsDim
= objModel.Dimensions("Sales regions")objLevel
= objLocationsDim.DrillDowns(1).Levels("Employee")objAltDrill
= objLevel.CreateAlternateDrillDownobjNewLevel
= objAltDrill.Levels.Add(xtrObjectType.trLevel)objNewLevel.Inclusion
= xtrInclusion.trInclusionGenerate
```

# IncrementalUpdate Property

---

The `IncrementalUpdate` property sets or returns whether a cube or cube group is incrementally updated from a data source.

## Syntax

*object*.**IncrementalUpdate**

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

Use this property to update cubes that are associated with data sources that contain incremental data. Older cube data is maintained.

You should periodically recreate the entire cube to optimize the partitioning scheme. For example, you may want to update a cube daily and recreate it weekly. Whenever you make structural changes to your model, you must recreate the cube before you perform another incremental update.

When this property is set to True for a cube group, all cubes within the group are incrementally updated.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objCube = objModel.Cubes.Item(1)objCube.IncrementalUpdate  
= False
```

# InputScale Property

---

The InputScale property sets or returns a scale value used to convert column numbers from decimal values to integer values.

## Syntax

*Column* .**InputScale**

## Applies To

Column Object

## Discussion

Use this property for data types other than Float to convert decimal values to integer values. The value you supply defines how source values are multiplied as they are read from your data source.

For example, if this property is set to 2, Transformer writes the source value, 94.5, to the cube as 9450; that is,  $94.5 \times 10^2$ . PowerPlay then displays the value 9450.

Use the OutputScale and FormatDecimals properties to change scaled values back to their original values. The OutputScale property specifies how the source values are divided when PowerPlay is run. In brief, the InputScale property removes decimal positions for use in calculations, and the OutputScale and FormatDecimals properties put them back.

For example, if OutputScale is set to 2, PowerPlay shows a value of 95 (rounded). If the FormatDecimals is set to 1, PowerPlay shows a value of 94.5.

The range for scaling values is from -16 to 16, which represents  $10^{-16}$  to  $10^{16}$ .

**Default:** 0

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation. For example, an exception occurs on Update() if outside the boundaries -16 to 16.

## Type

Long

## Access

Read/Write

## Examples

```
objDataSource  
= objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)objColumn.InputScale  
= 0
```

## IsAnyColumnMismatched Property

---

The IsAnyColumnMismatched property returns whether columns in the data source match the underlying data.

## Syntax

*object* .**IsAnyColumnMismatched**

## Applies To

[CrossTabDataSource Object](#)

[DataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

[Query Object](#)

## Discussion

Use this property to verify that columns defined in the model match the columns in the source files. The property returns False if there is no mismatch or True if there is a mismatch.

When you create a model, the columns in each data source are saved as part of the model definition. If you reorder, add, delete, or rename the columns in the source files, you must update the model.

## Type

Boolean

## Access

Read

## Examples

```
new_package = model.Packages.Add()new_query  
= new_package.Queries.Add()If new_query.IsAnyColumnMismatched  
= True Then
```

## IsBad Property

---

The IsBad property returns whether a suspended model is corrupt or recoverable.

## Syntax

*SuspendedModel* **.IsBad**

## Applies To

[SuspendedModel Object](#)

## Discussion

A suspended model is created when it is not closed properly, as in a system crash or power failure. Transformer retains as much information as it can as a *SuspendedModel* object in the *SuspendedModels* collection.

If the model is corrupt, use the *Remove* or *Delete* method to delete it from the collection. If it is recoverable, use the *OpenModel* method and set the *FileName* parameter to the return value of the *QyPath* property.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Boolean

## Access

Read

## Examples

```
objSuspendedModel = objTransApp.SuspendedModels(intX) If objSuspendedModel.IsBad  
= True Then
```

# IsExpressionValid Property

---

The *IsExpressionValid* property returns whether an expression is valid.

## Syntax

*object* **.IsExpressionValid**

## Applies To

[CalculationDefinition Object](#)

[Category Object](#)

[Column Object](#)

[Measure Object](#)

[SpecialCategory Object](#)

## Discussion

You can add information to a model that is not directly based on a data source by creating an expression with the *ExpressionText* property. Transformer evaluates all expressions when the parent

object is updated and at run time. To avoid unexpected errors, use `IsExpressionValid` to determine if all components of the expression are valid.

### Type

Boolean

### Access

Read

## IsFolder Property

---

The `IsFolder` property sets or returns whether a measure is a measure folder.

### Syntax

*Measure* .**IsFolder**

### Applies To

[Measure Object](#)

### Discussion

Use this property to change a measure into a measure folder, or the reverse.

### Type

Boolean

### Access

Read/Write

## IsManual Property

---

The `IsManual` property returns whether a level is associated with a source value.

### Syntax

*object* .**IsManual**

### Applies To

[DateLevel Object](#)

[Level Object](#)

### Discussion

A manual level is not associated with a source value. Instead, you create it manually as an intermediate level used to group large numbers of child categories, or to group categories based on an attribute that is different from any found in your data sources.

For example, if you have data source columns for state and city, you can create a manual level and the necessary categories to group the cities by geographic region rather than by state or province.

Manual levels have no directly associated source column and usually connect to child categories.

A value of True indicates a manual level. A value of False indicates a source level.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### **Type**

Boolean

### **Access**

Read

## **IsMDCInUse Property**

---

The IsMDCInUse property returns whether a cube is in use or being rebuilt.

### **Syntax**

*object* .**IsMDCInUse**

### **Applies To**

[Cube Object](#)

[CubeGroup Object](#)

[ChildCube Object](#)

### **Discussion**

A cube is locked when it is being used by a PowerPlay client, or being updated. Use IsMDCInUse to verify if the cube is open.

For example, if another PowerPlay client application has the cube open, or if someone is attempting to build the cube, IsMDCInUse will return True.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### **Type**

Boolean

### **Access**

Read

## **IsolationLevel Property**

---

The IsolationLevel property sets or returns the transaction types permitted when you reference an Impromptu query definition file(.iqd).



## Syntax

*IqdDataSource* .**IsolationLevel**

## Applies To

[IqdDataSource Object](#)

## Discussion

Set this property to one of the following transaction settings.

Long value	Description
0	Default. Uses the isolation level that was originally specified when the data source was created.
1	ReadUncommitted. Makes changes made by other transactions immediately available to a transaction.
2	ReadCommitted. Allows a transaction access to only those rows that have been committed by other transactions.
3	CursorStability. Prohibits other transactions from updating the row upon which a transaction is positioned.
4	ReproducibleRead. Ensures that rows selected or updated by a transaction will not be changed by another transaction until the first transaction is complete.
5	PhantomProtection. Prohibits access by a transaction to rows inserted or deleted since the start of the transaction.
6	Serializable. Ensures that a set of transactions executed concurrently produce the same result as if they were performed sequentially.

## Type

Long

## Access

Read/Write

## Examples

```
datasource  
= model.DataSources.Add(xtrObjectType.trIqdDataSource)datasource.IsolationLevel  
= 0
```

## IsPrimary Property

---

The IsPrimary property sets or returns whether an object is the primary drill category.

## Syntax

*object* .**IsPrimary**

## Applies To

[DateDrillDown Object](#)

[DrillDown Object](#)

[Category Object](#)

## Discussion

This property is True for the first drill-down path in a DrillDowns collection and False for any other. False indicates an alternate drill-down path. Alternate drill-down paths are created by using the CreateAlternateDrillDown method.

For example, two paths are available in the Channels dimension: the primary path is by Channel Type and the alternate path is by Region. If your drill-down path is the primary path, you drill from Channel Type to Customer. If your drill-down path is the alternate path, you drill from Region to Customer.

The IsPrimary property cannot be changed for a Category object. For a DateDrillDown and DrillDown object, it can be changed from False to True, but not True to False. When you set a drill-down path to True, it switches from an alternate drill-down path to a primary drill-down path, and the previous primary drill-down path becomes an alternate drill-down path.

In Transformer, allocations and partitioning are based on the primary drill-down path. In PowerPlay, category rollup values summarize to the top level of the primary drill-down path.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Boolean

## Access

Read (Category)

Read/Write (DateDrillDown and DrillDown)

## Examples

```
objLocationsDim = objModel.Dimensions("Sales regions")  
objLevel = objLocationsDim.DrillDowns(1).Levels("Employee")objAltDrill  
= objLevel.CreateAlternateDrillDownobjAltDrill.IsPrimary = True
```

## IsTimeBasedPartitionedCube Property

---

The IsTimeBasedPartitionCube property sets or returns whether a cube group is specified as a time-based partitioned cube.

## Syntax

*CubeGroup* .**IsTimeBasedPartitionedCube**

## Applies To

[CubeGroup Object](#)

## Discussion

A time-based partitioned cube is created by defining a CubeGroup object and then setting the IsTimeBasedPartitionedCube property to True. If this property has been set to True, setting it to False will result in an exception. The level and dimension may be set in the same way as for the CubeGroup object; however, they are accessed through the time-based partitioned cube.

**Default:** False

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Boolean

## Access

Read many/Write once

## KeyName Property

---

The KeyName property sets or returns the value that appears in the associated data source column.

## Syntax

*Category*.**KeyName**

## Applies To

[Category Object](#)

## Discussion

This value must be unique among all the categories owned by the immediate parent of the category. For convergence levels, this value must be unique among all categories in the level.

## Type

String

## Access

Read/Write

## Examples

```
objCategory = objModel.Dimensions("Sales region").Drilldowns(1).Categories(2)If  
objCategory.KeyName <> "Central Europe" Then
```

## Label Property

---

The Label property sets or returns a descriptive name that appears in PowerPlay.

## Syntax

*object* .Label

## Applies To

[Association Object](#)

[Category Object](#)

[CategorySet Object](#)

[CurrencyRecord Object](#)

[Measure Object](#)

[SpecialCategory Object](#)

## Discussion

The default label for a regular category is the Name property.

The default label for a currency record is the country or region code, as set by the CountryCode property.

The default label for a special category is the Name property. The default label for other objects is the Name property. The label of an Association is the name of the reference object.

Labels need not be unique.

## Type

String

## Access

Read/Write

## Examples

```
objAssociation = objDrill.ConvergenceLevel.Associations.Add()objAssociation.Label  
= objColumn.Name
```

## LastUseDate Property

---

The LastUseDate property returns the date when the category or special category was last active.

## Syntax

*object* .LastUseDate

## Applies To

[Category Object](#)

[SpecialCategory Object](#)

## Discussion

Use this property to show the date that Transformer generated a category or changed any properties of the category.

Use the return value of this property with the CleanHouse method to remove inactive categories. If the date in LastUseDate is older than the date in the Date parameter of CleanHouse, Transformer considers the category inactive and removes it.

### Type

Long

### Access

Read

## LatestDate Property

---

The LatestDate property sets the latest date in a range for selecting categories.

### Syntax

*object* .LatestDate

### Applies To

[DateDimension Object](#)

[DateWizard Object](#)

### Discussion

Use this property and the EarliestDate property to limit the range of acceptable date categories in a time dimension. By setting limits, you eliminate categories that are irrelevant to PowerPlay users, such as past years.

Transformer automatically updates all categories when cube categories are generated or a cube is created. To generate date categories only, use the GenerateTimePeriod property (DateDimension object). Transformer only generates a subset of records between the values of the EarliestDate and LatestDate properties.

If Transformer encounters date values outside of your specified range, it generates an Early Dates or a Late Dates category, or both, depending on when the out-of-range dates occur. If Transformer encounters dates that are neither early nor late, but cannot be placed within the specified range, it generates an Invalid Dates category.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Long

### Access

Read/Write (DateDimension)

Write (DateWizard)

### Examples

```
.LatestDate = 19941231
```

## Level Property

---

The Level property returns a level for a Category or SpecialCategory object.

### Syntax

*Category* .Level

*SpecialCategory* .Level

### Applies To

Category Object

SpecialCategory Object

### Discussion

Use this property to find the levels a category or special category belongs to. A category or special category belongs to only one level.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object - Level

### Access

Read

## LevelCategories Property

---

The LevelCategories property returns a collection of categories in a level.

### Syntax

*Object* .LevelCategories

### Applies To

DateLevel Object

Level Object

### Discussion

Use this property to access the collection of categories in a level. A category belongs to only one level.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object - LevelCategories

### Access

Read

## LevelDrillDowns Property

---

The LevelDrillDowns property returns the collection of drill-down path objects to which the level belongs.

### Syntax

*object* .LevelDrillDowns

### Applies To

[DateLevel Object](#)

[Level Object](#)

### Discussion

Most levels have a single drill-down path and therefore only one DrillDown or DateDrillDown object in the LevelDrillDowns collection. In the case of a drill-down path level that is the convergence of two or more drill-down paths, the collection has an equivalent number of entries.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object - LevelDrillDowns

### Access

Read

### Examples

```
objDateDrillDown = objModel.Dimensions("Years").DrillDowns(1)
```

## Levels Property

---

The Levels property returns a collection of Level or DateLevel objects.

### Syntax

*object* .Levels

### Applies To

[DateDrillDown Object](#)

[DrillDown Object](#)

### Discussion

Each dimension in a model contains one or more drill-down paths, each with a Levels collection.

The CreateDateDimension method automatically creates a collection of date levels. The DoAutoDesign method automatically creates a collection of levels for each drill-down path in the model.

A Levels collection contains Level objects or DateLevel objects but not both.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object - Levels

## Access

Read

## Examples

```
objTimeDimension = objModel.Dimensions.Item("Time")objLevel  
= objTimeDimension.DrillDowns.Item(1).Levels.Item("Month")
```

## LocalPath Property

---

The LocalPath property sets or returns the location of the data source.

## Syntax

*object* .LocalPath

## Applies To

[CrossTabDataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

## Discussion

This property can include a relative path and file name or a fully-qualified location. If the data source is on a server, use the ServerPath property instead.

If you specify a relative path, Transformer appends it to a start path value specified in the DataSource property. If the DataSource property is unspecified, Transformer appends the LocalPath value to the location where you installed PowerPlay. In either case, the concatenated values must result in a properly defined location.

## Type

String

## Access

Read/Write

## Examples

```
objDataSource  
= objModel.DataSources.  
Add(xtrObjectType.trFlatFileDataSource)objDataSource.LocalPath  
= strDataPath
```

## LogErrorLevel Property

---

The LogErrorLevel property sets or returns the level of severity of error messages logged.



## Syntax

*Application* .**LogErrorLevel**

## Applies To

[Application Object](#)

## Discussion

Use this property to specify which level of message or warning is recorded. For example, the constant `trLogInformationAndAbove` logs all messages to the file. To log messages, you must first set the `EnableMessageLogging` property to `True`.

This property uses the values of `xtrPreferences`.

## Type

Constant

## Access

Read/Write

## Examples

```
objTransApp.LogErrorLevel = xtrPreferences.trLogErrorsAndAbove
```

## LogFileAppend Property

---

The `LogFileAppend` property sets or returns whether Transformer appends messages to the log file or overwrites log messages.

## Syntax

*Application* .**LogFileAppend**

## Applies To

[Application Object](#)

## Discussion

When you set this property to `True`, Transformer appends information to the log file. If you set it to `False`, Transformer overwrites the log file.

Use the `LogFileName` and `LogFilesPath` properties to specify the name and location of a log file. If not specified, Transformer creates one in the folder where you installed the PowerPlay applications.

To log messages, you must set the `EnableMessageLogging` property to `True`.

**Default:** `False`

## Type

Boolean

## Access

Read/Write

## Examples

```
objTransApp.LogFileAppend = True
```

## LogFileName Property

---

The LogFileName property sets or returns a name for the log file.

### Syntax

*Application* .LogFileName

### Applies To

[Application Object](#)

### Discussion

If blank, Transformer uses the same name as the current model, but with a .log file extension. If there is no current model name, Transformer generates an arbitrary file name, such as Tfmr001e.log.

Use the LogFilesPath property to specify the location where log files are written. To log messages, you must first set the EnableMessageLogging property to True.

### Type

String

### Access

Read/Write

### Examples

```
objTransApp.LogFileName = "TrModelsLog.log"
```

## LogFilesPath Property

---

The LogFilesPath property sets or returns the location where Transformer saves the log file.

### Syntax

*Application* .LogFilesPath

### Applies To

[Application Object](#)

### Discussion

Use this property to direct where Transformer saves log files. If you do not specify a location, Transformer tries the location specified by the ModelsPath property. If ModelsPath is not specified, Transformer stores the log file in the PowerPlay installation directory.

Use the LogFileName property to specify a name for the log file. To log messages, you must first set the EnableMessageLogging property to True.

## Type

String

## Access

Read/Write

## Examples

```
objTransApp.LogFilePath = strStartLocation
```

## Lunar Property

---

The Lunar property sets or returns whether the DateDrillDown object is based on a lunar year.

### Syntax

*DateDrillDown* .**Lunar**

### Applies To

DateDrillDown Object

### Discussion

The type of calendar a DateDrillDown object represents, either regular or lunar, depends on the nature of the parent time dimension. False indicates a calendar year.

## Type

Boolean

## Access

Read/Write

## Examples

```
objDateWizard  
= objModel.DateWizardobjDateDim  
= objDateWizard.CreateDateDimension()objDrillDown  
= objDateDim.DrillDowns(1)objDrillDown.Lunar  
= True
```

## ManualCurrentPeriod Property

---

The ManualCurrentPeriod property sets or returns whether the current time period is set manually or by Transformer.

### Syntax

*DateDimension* .**ManualCurrentPeriod**

### Applies To

DateDimension Object

## Discussion

If set to True, you can set the current period to any category in the time dimension, typically one at the lowest level in the time dimension, for example, under a Week level.

Special categories use the current time period to create relative time periods such as Current Month, Last Month, and Quarter-to-Date.

When set to False, Transformer uses the category with the latest date to set the current time period. The `SetsCurrentPeriod` property indicates which data source provides the current period. If the only category found is Early Dates, Transformer sets no current period for the dimension.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objDateDim = objModel.Dimensions("Date")MsgBox("Current  
Period:" + objDateDim.ManualCurrentPeriod.Name)
```

# MaximizeSpeed Property

---

The `MaximizeSpeed` property sets or returns whether Transformer verifies category uniqueness.

## Syntax

*object* .**MaximizeSpeed**

## Applies To

[CrossTabDataSource Object](#)

[DataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

## Discussion

Use this property to optimize for speed. When set to True, Transformer does not check for category uniqueness as it populates a level in a dimension.

If set to False, Transformer attempts to detect uniqueness problems in the level definitions. Level uniqueness in Transformer means that the value of each category in a level is different.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objDataSource  
= objModel.DataSources.Add(xtrObjectType  
    .trFlatFileDataSource)objDataSource.MaximizeSpeed  
= True
```

## MaxNumPartLevels Property

---

The MaxNumPartLevels property sets or returns the maximum number of partition levels.

## Syntax

*object* .MaxNumPartLevels

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

Use this property to set the maximum number of times Transformer reads the source data. As the number of partition levels increases, Transformer requires more passes through the data.

If the Optimize property is set to auto-partition, you must also set the cube properties: DesiredPartitionSize and EstimatedRows.

To manually create partitions, use the Partition property of a level or category. Ensure that MaxNumPartLevels setting is at least as large as the number of partition levels added.

**Default:** 5 passes.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read/Write

## Examples

```
objCube = objModel.Cubes.Item(1)objCube.MaxNumPartLevels  
= 5
```

## MaxTransactionNumber Property

---

The MaxTransactionNumber property sets or returns the maximum number of records that Transformer processes before committing the changes to a cube.

## Syntax

*Application* .**MaxTransactionNumber**

## Applies To

[Application Object](#)

## Discussion

Use this property to determine when Transformer inserts checkpoints at various stages as it generates cubes. This setting limits the number of records held in temporary status before inserting a checkpoint. Should an error prevent the completion of a cube, you can restart the process at the last committed checkpoint.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

**Default:** 500,000.

## Type

Long

## Access

Read/Write

## Examples

```
objTransApp.MaxTransactionNumber = 500000
```

## MDCFile Property

---

The MDCFile property sets or returns the name of a PowerCube file (.mdc).

## Syntax

*object* .**MDCFile**

## Applies To

[ChildCube Object](#)

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

Use this property to specify a file name.

To specify a folder in which to save the PowerCube file, use the PowerCubesPath property. Transformer searches the PowerCubesPath property and then the ModelsPath property to find a location. If none are specified, Transformer saves the PowerCube file in the location where you installed PowerPlay.

## Type

String

## Access

Read/Write

## Examples

```
objCube = objModel.Cubes.Item(1)objCube.MDCFile = "GoCube"
```

## MeasureInclude Property

The MeasureInclude property sets or returns whether a measure is included in the specified object.

## Syntax

*object*.**MeasureInclude**(Measure)

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

[CustomView Object](#)

## Discussion

To include a specific measure, use the name as the parameter and set the property to True. To exclude a measure, set this property to False. To set the inclusion status of several measures, repeat the use of this property with a different measure named in the parameter each time.

When you exclude a measure that is used in an expression for a calculated measure that is itself included in the cube, Transformer includes the measure in the cube for the calculation, but the measure is not visible to PowerPlay users.

**Default:** True

Parameter	Description
Measure	Required. Specifies the Measure object that is included in or excluded from the specified object.  Type: Object

## Type

Boolean

## Access

Read/Write

## Examples

```
objCubesByRegion  
= objModel.Cubes.Add(xtrObjectType.trCubeGroup)objCubesByRegion  
  .MeasureInclude(objModel.Measures("Product cost"))  
= False
```

## MeasureName Property

---

The MeasureName property sets or returns a descriptive title that identifies a measure.

### Syntax

*object*.MeasureName

### Applies To

[ChildCube Object](#)

[Cube Object](#)

[CubeGroup Object](#)

### Discussion

Use this property to replace the default 'Measures' title on the PowerPlay dimension line.

### Type

String

### Access

Read/Write

### Examples

```
objCubesByRegion  
= objModel.Cubes.Add(xtrObjectType.trCubeGroup)objCubesByRegion.MeasureName  
= "Revenue Made"
```

## Measures Property

---

The Measures property returns a collection of Measure objects.

### Syntax

*Model*.Measures

### Applies To

[Model Object](#)

### Discussion

A Measures collection can contain regular, calculated, and category count measures.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object - Measures

### Access

Read



## Examples

```
objMeasures = objModel.Measures
```

## MeasureType Property

---

The MeasureType property returns the type of a Measure object.

### Syntax

*Measure*.MeasureType

### Applies To

[Measure Object](#)

### Discussion

Measures come in three types: regular, calculated, and category count measures. The type is automatically set to trRegularMeasure when you use the DoAutoDesign method. A regular measure is associated with a column or attribute.

You can use the ExpressionText property to define a calculated measure, and the CategoryCountLevel property to define a category count measure.

The MeasureType property uses the values of xtrMeasureType.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Constant - xtrMeasureType

### Access

Read

## MissingValue Property

---

The MissingValue property sets or returns a replacement value to use when the numeric value for a measure is missing in the data source.

### Syntax

*Measure*.MissingValue

### Applies To

[Measure Object](#)

### Discussion

By default, all missing values appear as zeros in PowerPlay. You can have PowerPlay show 'na' instead. That way, missing items are not misinterpreted as zero values.

The MissingValue property uses values of xtrMissingValue.

## Type

Constant - xtrMissingValue

## Access

Read/Write

## Examples

```
objMeasures  
= objModel.MeasurecurrentMeasure  
= objMeasures("Revenue").currentMeasure.MissingValue  
= xtrMissingValue.trMissingValueZERO
```

## ModelName Property

---

The ModelName property sets or returns the name of a model for which a temporary file (.qy?) exists due to an abnormal termination.

## Syntax

*SuspendedModel* .**ModelName**

## Applies To

SuspendedModel Object

## Discussion

If a client-based model was not saved due to abnormal termination, use the SuspendedModels property to return a collection of suspended models. Use the Item method to return each model, and the IsBad property to test whether a model is corrupt.

If you have not named the model prior to failure, this property returns the name New Model.

## Type

String

## Access

Read/Write

## Examples

```
objSuspendedModel = objTransApp.SuspendedModels(intX) MsgBox("Model:  
" + objSuspendedModel.ModelName + "_", located at " + objSuspendedModel.QyPath  
+ " is corrupt")
```

## ModelsPath Property

---

The ModelsPath property sets or returns the location where Transformer opens and saves model files.

## Syntax

*Application* .**ModelsPath**

## Applies To

[Application Object](#)

## Discussion

If a location is not specified, model files are stored in the PowerPlay installation directory.

## Type

String

## Access

Read/Write

## Examples

```
objTransApp.ModelsPath = strStartLocation
```

## ModelTemporaryFilePath Property

---

The ModelTemporaryFilePath property sets or returns the location where Transformer creates temporary model files (.qy?).

## Syntax

*Application* .**ModelTemporaryFilePath**

## Applies To

[Application Object](#)

## Discussion

Use temporary model files to recover a suspended model, should a fatal error occur. If this property is not specified, Transformer searches the location (in order):

- specified by the ModelsPath property
- where PowerPlay is installed
- set by Temporary = in the [Services] section of the cs7g.ini file
- set by the TEMP environment variable
- set by the TMP environment variable

## Type

String

## Access

Read/Write

## Examples

```
objTransApp.ModelTemporaryFilePath = strStartLocation
```

## ModelType Property

---

The ModelType property returns the file extension of a model file as it appears in a Windows folder or Windows Explorer.

### Syntax

*Model* .**ModelType**

### Applies To

Model Object

### Discussion

Use this property to check the type of a model file or to write the type to a log file. The model can be saved as a text-based file (.mdl) or a binary file (.py?). Transformer replaces the question mark with a number or letter.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

String

### Access

Read

## MonthType Property

---

The MonthType property sets how to calculate the month level of a time dimension.

### Syntax

*DateWizard* .**MonthType**

### Applies To

DateWizard Object

### Discussion

Use this property to determine if a month is defined as part of a calendar year or lunar year.

The MonthType property uses the value of xtrSpecialFunction.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Constant - xtrSpecialFunction

### Access

Write

## Examples

```
objModel.MonthType = xtrSpecialFunction.trSpecialFunctionMonth
```

## Name Property

---

The Name property sets or returns the name of an object.

### Syntax

*object* .Name

### Applies To

[“Application Object” on page 39](#)

[Association Object](#)

[CalculationDefinition Object](#)

[Category Object](#)

[CategorySet Object](#)

[ChildCube Object](#)

[Column Object](#)

[CrossTabDataSource Object](#)

[Cube Object](#)

[CubeGroup Object](#)

[CurrencyTable Object](#)

[CustomView Object](#)

[DataSource Object](#)

[DateDimension Object](#)

[DateDrillDown Object](#)

[DateLevel Object](#)

[DbDataSource Object](#)

[Dimension Object](#)

[DrillDown Object](#)

[DrillThroughTarget Object](#)

[Filter Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

[Level Object](#)

[Measure Object](#)

[Model Object](#)

[Name Object](#)

[Namespace Object](#)

[Package Object](#)

[Prompt Object](#)

[Query Object](#)

[“Report Object” on page 96](#)

[SecurityObject Object](#)

[Signon Object](#)

[SpecialCategory Object](#)

[View Object](#)

## Discussion

To create a dimension object, you must specify a name.

Some objects have Name and Label, or, Name and ShortName properties. If you do not specify a Label or ShortName value for an object, Transformer uses the Name property as a default source.

If the name of a newly created object does not have a source for name values, Transformer creates one and appends a unique identifier to it. For example, Products View1, Products View2.

For Category objects, Transformer uses the KeyName property as a default source for both the Name property and the Code property.

## Type

String

## Access

Read (Application, Name, and Report)

Read/Write (all other objects)

## Examples

```
objCube = objModel.Cubes.Item(1)objCube.Name = "Great  
Outdoors Sales (Optimized)"
```

# Namespaces Property

---

The Namespaces property returns a Namespaces collection.

## Syntax

*Model* .**Namespaces**

## Applies To

[Model Object](#)

## Discussion

Use this property to iterate through the namespaces for a Model object, add a new namespace, and change or remove a namespace.

## Type

Object - Namespaces

## Access

Read

## Examples

```
new_namespace = model.Namespaces.Add()
```

# NewCatsLocked Property

---

The NewCatsLocked property sets or returns whether you can add new categories to the object.

## Syntax

*object*.NewCatsLocked

## Applies To

[DateDimension Object](#)

[DateLevel Object](#)

[Dimension Object](#)

[Level Object](#)

## Discussion

Use this property to lock a level or dimension so that no new categories are added as a result of a category generation. When this property is set to True for a dimension, it is also set to True for all levels in that dimension.

If set to True, Transformer ignores source values in the data source that are unrelated to categories already in the model.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objLocationsDim = objModel.Dimensions("Sales regions")objLevel  
= objLocationsDim.DrillDowns(1).Levels("Employee")objLevel.NewCatsLocked  
= False
```

# ObjectCAMID Property

---

The ObjectCAMID property returns the CAMID of the object in the namespace set by the ObjectName property.

## Syntax

*Namespace*.**ObjectCAMID**

## Applies To

[Namespace Object](#)

## Discussion

Use this property to get the CAMID of an object in a namespace.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

String

## Access

Read

## Examples

```
new_namespace = model.Namespaces.Add() CAMID_of_Object  
= new_namespace.ObjectCAMID
```

# ObjectName Property

---

The ObjectName property sets or returns the name of a namespace object.

## Syntax

*Namespace*.**ObjectName**

## Applies To

[Namespace Object](#)

## Discussion

Use this property to set the name of a namespace object to get the CAMID of the object from the ObjectCAMID property.

## Type

String

## Access

Read/Write

## Examples

```
new_namespace = model.Namespaces.Add() new_namespace.ObjectName  
= Name_of_Object 'Authors is a group
```



## Optimize Property

---

The Optimize property sets or returns the current cube optimization option for your model and environment.

### Syntax

*object*.**Optimize**

### Applies To

[ChildCube Object](#)

[Cube Object](#)

[CubeGroup Object](#)

### Discussion

Cubes are optimized to increase performance in Transformer and PowerPlay. For example, you can specify the `trOptimizeDataPasses` constant to optimize the number of passes through the temporary working files during the creation of a cube.

If this property is set to `trOptimizeAutoPartition`, set the cube properties: `DesiredPartitionSize`, `EstimatedRows` and `MaxNumPartLevels`. You can use the `ExcludeAutoPartition` property to exclude a dimension from the auto-partitioning process.

The Optimize property uses the values of `xtrCubeOptimize`.

### Type

Constant - `xtrCubeOptimize`

### Access

Read/Write

### Examples

```
objCube = objModel.Cubes.Item(1)objCube.Optimize  
= xtrCubeOptimize.trOptimizeAutoPartition
```

## OrderByDescending Property

---

The OrderByDescending property sets or returns whether the categories are sorted in descending order.

### Syntax

*object*.**OrderByDescending**(DrillDown)

### Applies To

[DateLevel Object](#)

[Level Object](#)

### Discussion

Where a level represents the convergence of two or more drill-down paths, you can apply a different sort order to each one. Use the `DrillDown` parameter to specify the drill-down path to which the sort applies.

By default, categories are sorted in ascending order.

**Default:** False

Parameter	Description
DrillDown	Required. Specifies the drill-down path structure in which the level appears. Type: Variant

## Type

Boolean

## Access

Read/Write

## Examples

```
objDrill
= objModel.Dimensions("Retailers").DrillDowns(2)objDrill
  .Levels("Retailer site").OrderByDescending(objAssociation.Context)
= False
```

## OrderByStorageType Property

The OrderByStorageType property sets or returns how categories are sorted based on the storage type of a column.

## Syntax

*object*.**OrderByStorageType**(DrillDown)

## Applies To

[DateLevel Object](#)

[Level Object](#)

## Discussion

Use this property to define a category sort when the sort is based on a column with a numeric data class. For example, you base a product level sort on quantity sold, and the measure named Quantity, which is associated with the Quantity column, has a storage type of 16-bit integer.

The OrderByStorageType property uses the values of xtrStorage.

Parameter	Description
DrillDown	Required. Specifies the drill-down path structure (context) in which the level appears. Type: Variant

## Type

Constant - xtrStorage

## Access

Read/Write

## Origin Property

---

The Origin property returns the origin of the specified object.

### Syntax

*Column*.Origin

### Applies To

[Column Object](#)

### Discussion

An attribute or column object is created in one of several ways. For example, most column objects are generated from the data source by Transformer, while a calculated column is manually created.

The Origin property uses the values of xtrOrigin.

### Type

Constant - xtrOrigin

### Access

Read

## OriginalName Property

---

The OriginalName property sets or returns the name of the column in the data source.

### Syntax

*Column*.OriginalName

### Applies To

[Column Object](#)

### Discussion

Use this property to cross-reference the data in your model with the data in the original data source.

The return value depends on the format of the data source.

Source data	OriginalName
An Impromptu query definition file (.iqd)	The name defined in your query definition.
A delimited-field text file with column titles	The column title.
A delimited-field text file	A name constructed from the value in the corresponding field in the first record in the data source, prefixed by 'Things like'.

Source data	OriginalName
A fixed-field text file	No original name.
A dBase, Paradox, Lotus 1-2-3, Excel, Clipper, or FoxPro table	The column name defined in the table.
IBM Cognos package	The Name defined in your query definition.
IBM Cognos report	The Name defined in your query definition.
Lotus 1-2-3 or Excel Crosstab	The range name that represents a column in a crosstab data source.
PowerHouse Portable Subfile	The name defined in the subfile.

## Type

String

## Access

Read

## Examples

```
new_package
= model.Packages.Add()new_query
= new_package.Queries.Add()new_column1
= new_query.Columns.Add()new_column1.OriginalName
= "[Sales (query)].[Time dimension].[Date]"
```

## Orphanage Property

The Orphanage property sets or returns whether a category is an orphanage.

## Syntax

*Category*.Orphanage

## Applies To

[Category Object](#)

## Discussion

Use an orphanage with a dimension known to generate orphan categories.

An orphanage is a category created in a manual level. After an orphanage is created, any subsequently generated categories that do not have a position defined in the model become children of the orphanage. By using an orphanage, you keep new data separate until you can connect it to the appropriate category.

To create an orphanage, add a level to the Levels collection associated with that dimension. If you want the orphanage to collect new category values for the entire dimension, use the Move method to move it to the first position in the collection.

Use the Add (Categories) method to add a category to the Categories collection. Ensure that the Add method parameters specify the type of object (trCategory) and the manual level to which the category belongs. Finally, set the Orphanage property to True.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
.OutputScale = 0
```

# OutputScale Property

---

The OutputScale property sets or returns a scale value used to convert integer values to decimal values that appear in PowerPlay.

## Syntax

*object* .OutputScale

## Applies To

[Column Object](#)

[Measure Object](#)

## Discussion

This property sets the power of 10 by which the source values are divided. For example, with a setting of 2, PowerPlay displays the value 9450 as 95; that is,  $9450/10^2$  rounded to 0 decimal places.

To include decimal places in the output value, use the FormatDecimals property. In the above example, if FormatDecimals property is set to 2, PowerPlay shows a value of 94.50.

Use the InputScale property to specify how source values are multiplied as they are read from a data source.

The range for scaling values is from 0 to 16, which represents  $10^{**16}$ .

**Default:** 0

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation. For example, an exception occurs on Update() when the value is set outside the 0 to 16 range.

## Type

Long

## Access

Read/Write

## Examples

```
objMeasure.CategoryCountLevel = objLevelobjMeasure.OutputScale  
= 0
```

## Packages Property

---

The Packages property returns the collection of Package objects associated with a Transformer model.

### Syntax

*Model* .**Packages**

### Applies To

[Model Object](#)

### Discussion

Use this property to iterate through a Model object's packages, add a new package, and change or remove a package.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Packages

### Access

Read

### Examples

```
new_package = model.Packages.Add()
```

## PackagesDataSourceConnections Property

---

The PackageDataSourceConnections property returns the collection of PackageDataSourceConnection objects associated with a Package or Report object.

### Syntax

*object* .**PackageDataSourceConnections**

### Applies To

[Package Object](#)

### Discussion

In order to use this property, ensure that there is at least one query assigned to the Package or Report.

Use this property to iterate through the data source connections for a Package or Report, add new connections, change connection properties, or remove connections.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

PackageDatasourceConnections

## Access

Read

## Examples

```
package = model.Packages.Add()connection  
= package.PackageDatasourceConnections.Add()
```

## Parent Property

---

The Parent property returns the parent object.

## Syntax

*object* .Parent

## Applies To

[Association Object](#)

[Associations Collection](#)

[CalculationDefinition Object](#)

[CalculationDefinitions Collection](#)

[Categories Collection](#)

[Category Object](#)

[CategorySet Object](#)

[CategorySets Collection](#)

[ChildCube Object](#)

[ChildCubes Collection](#)

[Column Object](#)

[Columns Collection](#)

[CrossTabDataSource Object](#)

[Cube Object](#)

[CubeCustomViews Collection](#)

[CubeGroup Object](#)

[Cubes Collection](#)

[CurrencyRate Object](#)

[CurrencyRates Collection](#)

[CurrencyRecord Object](#)

[CurrencyRecords Collection](#)

[CurrencyTable Object](#)

[CurrencyTables Collection](#)

[CustomView Object](#)  
[CustomViews Collection](#)  
[DataSource Object](#)  
[DataSources Collection](#)  
[DateDimension Object](#)  
[DateDrillDown Object](#)  
[DateLevel Object](#)  
[DateWizard Object](#)  
[DbDataSource Object](#)  
[Dimension Object](#)  
[DimensionLevels Collection](#)  
[Dimensions Collection](#)  
[DrillDown Object](#)  
[DrillDowns Collection](#)  
[DrillThroughTarget Object](#)  
[DrillThroughTargets Collection](#)  
[Filter Object](#)  
[Filters Collection](#)  
[FlatFileDataSource Object](#)  
[IqdDataSource Object](#)  
[Level Object](#)  
[LevelCategories Collection](#)  
[LevelDrillDowns Collection](#)  
[Levels Collection](#)  
[Measure Object](#)  
[Measures Collection](#)  
[Model Object](#)  
[Name Object](#)  
[Names Collection](#)  
[Namespace Object](#)  
[Namespaces Collection](#)  
[Package Object](#)  
[PackageDatasourceConnections Collection](#)  
[PackageDatasourceConnection Object](#)  
[Packages Collection](#)  
[Prompt Object](#)  
[Prompts Collection](#)  
[Queries Collection](#)  
[Query Object](#)



[Report Object](#)

[Reports Collection](#)

[SecurityObject Object](#)

[SecurityObjects Collection](#)

[Signon Object](#)

[Signons Collection](#)

[SpecialCategory Object](#)

[SuspendedModel Object](#)

[SuspendedModels Collection](#)

[View Object](#)

[Views Collection](#)

## Discussion

Use this property to return the immediate ancestor of an object or collection. You can use dot notation to access and update the properties of a parent object.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object

## Access

Read

## Examples

```
objView = objDimension.Views.Add()objView.Parent.Update()
```

## ParentCategories Property

---

The ParentCategories property returns a collection of parent categories.

## Syntax

*object*.ParentCategories

## Applies To

[Category Object](#)

[SpecialCategory Object](#)

## Discussion

If a category is in more than one drill-down path it may have two or more parent categories. Each parent category is associated with a different level and different drill-down path.

## Type

Object

## Access

Read

# Partition Property

---

The Partition property sets or returns a manual partition number.

## Syntax

*object* .Partition

## Applies To

[Category Object](#)

[DateLevel Object](#)

[Level Object](#)

## Discussion

The Partition property specifies the partition level number and makes the category or the categories within the level the head of a partition in a cube. The best candidates for partitioning are categories that are several levels deep, with similar category-to-level ratios throughout.

If you are familiar with your source data and user requirements, you can manually define your partitions by assigning partition level numbers to the levels and categories in specific dimensions. If so, choose dimensions with large numbers of categories and levels, and similar category-to-level ratios. Avoid partitioning dimensions that contain alternate drill-down path structures, or levels that contain special categories. Do not specify partition numbers for leaf categories, drill categories, or the root category.

To create partitions, first, use the ResetPartitions method to remove any previous automatic or manual partitions. Then use the Partition property to create the partition levels. Use the MaxNumPartLevels property to specify how many passes Transformer makes through the source data.

Categories that are suppressed or filtered by values of xtrInclusion are not included in the partition.

If a PowerCube is incrementally updated, you cannot add new partition levels. To change partitioning for the model, you must repartition the model and rebuild your PowerCube from scratch, using all the data from all increments.

Range: 0 to 15.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read/Write

## Examples

```
objCategory
= objModel.Dimensions("Sales region").Drilldowns(1).
  Categories(2)objCategory.Partition
= 1
```

## Password Property

---

The Password property sets a case-sensitive password.

### Syntax

*object* .**Password**

### Applies To

[ChildCube Object](#)

[Cube Object](#)

[CubeGroup Object](#)

[Signon Object](#)

### Discussion

When a password is defined at the root node of a cube group, the same password applies to all cubes in the group. However, a password defined for a member of a cube group overrides the password defined at the root level for the group by changing the Password property for that individual child cube within the group. Note that the DataSource signon is imported with .iqd files or when a data source is configured to prompt for a password.

To use IBM Cognos to authenticate to an external namespace, the signon object must be configured with the userID, password, and associated namespace. A signon object must be created first.

For a Signon object, the Password property specifies the password required to automatically access a database. This access also requires that the UserID property is set.

Auto-logon to a database can be setup by storing the password in the signon object as well as the logical database name.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

String

### Access

Write

### Examples

```
signon = model.Signons.Add()signon.Password = "sa"
```

## PatFile Property

---

The PatFile property sets or returns the location of the pattern file `cogtr_locale.pat`, such as `cogtr_en.pat`, for the associated product locale.

This file is used to auto design models using pattern recognition. It is installed with IBM Cognos Transformer in the *installation\_location/bin* directory.

## Syntax

*Application* **.PatFile**

## Applies To

[Application Object](#)

## Discussion

Use this property to direct where Transformer can find the pattern file *cogtr\_locale.pat*, such as *cogtr\_en.pat*, for the associated product locale.

## Type

String

## Access

Read/Write

# Path Property

---

The Path property returns the location of a model file for a Model object. For a Report or Package object, it returns the path to the package or report in IBM Cognos Connection.

## Syntax

*object* **.Path**

## Applies To

[Model Object](#)

[Package Object](#)

[Report Object](#)

## Discussion

Use this property to return the location of a model file or to write this location information to a log file. To return the file name, use the *FileName* property. To return the complete path and file name, use the *FullName* property.

To specify a location in which to save models, use the *ModelsPath* property.

Use this property to get or set the path to a package or report data source. Here is an example of how the syntax of a package path appears.

```
/content/package[@name='GO Data Warehouse (analysis)']
```

## Type

String

## Access

Read for Model

Read/Write for Package or Report

## Examples

```
new_package = model.Packages.Add()new_package.Path = "/content/package[@name='GO  
Data Warehouse (analysis)']"
```

## PopulateByDataSource Property

---

The `PopulateByDataSource` property sets or returns whether the currency rate is obtained through a data source or set within Transformer.

### Syntax

*CurrencyRate* .**PopulateByDataSource**

### Applies To

[CurrencyRate Object](#)

### Discussion

True indicates that rates are defined in an external data source. False indicates that currency rates are set within Transformer.

**Default:** False

### Type

Boolean

### Access

Read/Write

## Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()objCurrencyRate  
= objCurrencyRecord.CurrencyRates(intX)If objCurrencyRate.PopulateByDataSource  
= False Then
```

## Position Property

---

The `Position` property sets or returns the position of the column in the data source.

### Syntax

*Column* .**Position**

### Applies To

[Column Object](#)

## Discussion

For all fields except those from fixed-field text data sources, this position corresponds to the ordinal position of a column in the data source. Source column positions start at 1.

If your data source is a fixed-field text file, the Position property corresponds to the starting byte for a column. The first byte in a record is byte number 1. A position value of zero (0) means that the column has missing values. If, for example, the first field holds 30 bytes (one byte per character), the offset position of the first byte of the second field is 31.

## Type

Long

## Access

Read/Write

## PowerCubesPath Property

---

The PowerCubesPath property sets or returns the location where Transformer creates PowerCube files (.mdc).

## Syntax

*Application* .**PowerCubesPath**

## Applies To

[Application Object](#)

## Discussion

Use this property to specify a location for a PowerCube. Use the MDCFile property to specify a file name.

If a directory is not specified, Transformer tries the following locations, in order:

- one specified by the ModelsPath property
- the PowerPlay installation directory

## Type

String

## Access

Read/Write

## Examples

```
objTransApp.PowerCubesPath = strStartLocation
```

## PowerPlayPath Property

---

The PowerPlayPath property sets or returns the location of the PowerPlay.exe executable. The executable is installed as part of the PowerPlay application installation.

## Syntax

*Application* .**PowerPlayPath**

## Applies To

Application Object

## Discussion

Use this property to direct where Transformer can find the PowerPlay.exe file.

## Type

String

## Access

Read/Write

# Precision Property

---

The Precision property sets or returns the number of decimal places for measures used in calculations.

## Syntax

*Measure* .**Precision**

## Applies To

Measure Object

## Discussion

Use this property to specify the number of decimal places that PowerPlay uses to calculate values for rollups and calculated categories. This property is only available when the *StorageType* property of the Measure object has a value of *trStorageBigFloat* or with calculated measures.

When the Precision setting is less than the *OutputScale* setting, PowerPlay rounds it up to the next highest decimal position.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read/Write

## Examples

```
objMeasure.CategoryCountLevel = objLevel objMeasure.Precision = 0
```

## PromptForPassword Property

---

The PromptForPassword property specifies whether users are always prompted for a password when using Transformer in UI mode.

### Syntax

*Signon* .**PromptForPassword**

### Applies To

Signon Object

### Discussion

Use this property to specify whether users are always prompted for a password when using Transformer in UI mode. This property only applies to the Datasource signon type.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Boolean

### Access

Read/Write

### Examples

```
signon = model.Signons.Add()signon.PromptForPassword = False
```

## Prompts Property

---

The Prompts property returns the collection of Prompt objects associated with a Query.

### Syntax

*Query* .**Prompts**

### Applies To

Query Object

### Discussion

Use this property to iterate through a Query object's prompts, add a new prompt, and change or remove a prompt.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Prompts



## Access

Read

## Examples

```
new_report = model.Reports.Add()new_query = new_report.Queries.Add()new_prompt  
= new_query.Prompts.Add()
```

## PromptValueType Property

---

The PromptValueType property sets or returns the type of property.

## Syntax

*Prompt* .**PromptValueType**

## Applies To

[Prompt Object](#)

## Discussion

This property needs to be assigned prior to setting the Value property of the Prompt object.

PromptValueType uses the values of xtrPrompValueType.

## Type

Constant - xtrPrompValueType

## Access

Read/Write

## Examples

```
new_report = model.Reports.Add()new_query  
= new_report.Queries.Add()new_prompt.PromptValueType  
= xtrPrompValueType.trSingleValuePrompt
```

## QualifiedName Property

---

The QualifiedName property returns the fully qualified name of the level.

## Syntax

*object* .**QualifiedName**

## Applies To

[DateLevel Object](#)

[Level Object](#)

## Discussion

Use this property to determine the dimension to which a level belongs.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

String

### Access

Read

## QuarterType Property

---

The QuarterType property sets how to calculate the quarter level of a time dimension.

### Syntax

*DateWizard* .**QuarterType**

### Applies To

[DateWizard Object](#)

### Discussion

Use this property to determine if a quarter is defined as part of a calendar year or a lunar year.

The QuarterType property uses value of xtrSpecialFunction.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Constant - xtrSpecialFunction

### Access

Write

### Examples

```
objModel.QuarterType = xtrSpecialFunction.trSpecialFunctionQuarter
```

## Queries Property

---

The Queries property returns the collection of Query objects associated with a Package or Report object.

### Syntax

*object* .**Queries**

### Applies To

[Package Object](#)

[Report Object](#)

## Discussion

Use this property to iterate through the queries for a Package or Report, add new queries, and change or remove queries.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Queries

## Access

Read

## Examples

```
new_package = model.Packages.Add()new_query = new_package.Queries.Add()
```

## QyPath Property

---

The QyPath property returns the location of a suspended model.

## Syntax

*SuspendedModel*.**QyPath**

## Applies To

SuspendedModel Object

## Discussion

As you work on a model, Transformer sets checkpoints in a temporary file that it creates. Temporary files are deleted if a model closes normally. The existence of one indicates that Transformer terminated unexpectedly.

When a model does not close correctly, as in the case of a system crash or a power failure, Transformer retains information about the suspended model up to the last checkpoint. The suspended model is saved with a .qy? file extension (the ? value depends on the version of Transformer).

Use the SuspendedModels property to return a collection of suspended models. Use the IsBad property to test if a model in the collection is corrupt or recoverable.

## Type

String

## Access

Read

## Examples

```
objSuspendedModel = objTransApp.SuspendedModels(intX)MsgBox("Model:  
" + objSuspendedModel.ModelName + "_", located at " + objSuspendedModel.QyPath  
+ " is corrupt")
```

## Rate Property

---

The Rate property sets or returns the currency conversion rate.

### Syntax

*CurrencyRate* .Rate

### Applies To

CurrencyRate Object

### Discussion

Transformer creates a collection of CurrencyRate objects for each record added to a currency table. The size of the collection depends on the level of detail and the date range. For example, a two-year time span and a level of detail set to month requires 24 currency rates.

If the record is created manually, use this property to set and maintain the rate within the model.

Conversely, if records are loaded from an external data source, the rates are set and maintained within the data source. In this case, use the Rate property to return the value specified in the data source.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Double

### Access

Read/Write

### Examples

```
objCurrencyRecord = objModel.CurrencyRecords.Add()objCurrencyRate  
= objCurrencyRecord.CurrencyRates(intX)objCurrencyRate.Rate = 1.54
```

## RefName Property

---

The RefName property sets or returns the name of the filter in the data source.

### Syntax

*Filter* .RefName

### Applies To

Filter Object

### Discussion

Use this property to cross-reference the filter in your model with the filter in the original data source.

## Type

String

## Access

Read/Write

## Examples

```
new_package = model.Packages.Add()new_query = new_package.Queries.Add()new_filter  
= new_query.Filters.Add()new_filter.RefName = "[go_data_warehouse].[2004]"
```

# RefreshDescription Property

---

The RefreshDescription property sets or returns whether descriptions are updated.

## Syntax

*object* .**RefreshDescription**

## Applies To

[DateLevel Object](#)

[Level Object](#)

## Discussion

Use this property to update the Description property of categories each time Transformer generates categories. The level must include an Association object with the AssociationRole property set to trAssociationDescription.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objLocationsDim = objModel.Dimensions("Sales regions")objLevel  
= objLocationsDim.DrillDowns(1).Levels("Employee") objAltDrill  
= objLevel.CreateAlternateDrillDownobjNewLevel  
= objAltDrill.Levels.Add(xtrObjectType.trLevel)objNewLevel.RefreshDescription  
= True
```

# RefreshLabel Property

---

The RefreshLabel property sets or returns whether labels are updated.

## Syntax

*object* .RefreshLabel

## Applies To

[DateLevel Object](#)

[Level Object](#)

## Discussion

Use this property to update the Label property of categories each time Transformer generates categories. The level must include an Association object with the AssociationRole property set to trAssociationLabel.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objLocationsDim
= objModel.Dimensions("Sales regions")objLevel
= objLocationsDim.DrillDowns(1).Levels("Employee")objAltDrill
= objLevel.CreateAlternateDrillDownobjNewLevel
= objAltDrill.Levels.Add(xtrObjectType.trLevel)objNewLevel.RefreshLabel
= True
```

# RefreshShortName Property

---

The RefreshShortName property sets or returns whether short names are updated.

## Syntax

*object* .RefreshShortName

## Applies To

[DateLevel Object](#)

[Level Object](#)

## Discussion

Use this property to update the ShortName property of categories each time Transformer generates categories. The level must include an Association object with the AssociationRole property set to trAssociationShortName.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objLocationsDim
= objModel.Dimensions("Sales_regions")objLevel
= objLocationsDim.DrillDowns(1).Levels("Employee")objAltDrill
= objLevel.CreateAlternateDrillDownobjNewLevel
= objAltDrill.Levels.Add(xtrObjectType.trLevel)objNewLevel.RefreshShortName
= True
```

# RegularRollup Property

---

The RegularRollup property sets or returns the type of regular rollup function in effect.

## Syntax

*Measure* .**RegularRollup**

## Applies To

Measure Object

## Discussion

Use this property to change the way values for a measure are rolled up.

A rollup summarizes values for each category in a level by performing a calculation on all of the child categories. By default, PowerPlay uses the Sum function in the roll up of categories. You can change this default to minimum, maximum, average, count, count all, any, or external by using the constants of xtrRollup.

Transformer consolidates records that contain duplicate non-measure values, that is, those with identical category names.

Ensure that you set the RegularWeight property when the RegularRollup property is set to trRollupAverage.

When you use RegularRollup and TimeStateRollup properties, PowerPlay performs the regular rollup first, and then the time state rollup.

The CanAllocateMeasure, Consolidate, DuplicateRollup, TimeStateRollup, and RollupTiming properties also play a part in how and when measures are rolled up.

This property uses the values of xtrRollup.

## Type

Constant - xtrRollup

## Access

Read/Write

## Examples

```
objMeasures
= objModel.MeasurescurrentMeasure
```

```
= objMeasures("Revenue").currentMeasure.RegularRollup  
= xtrRollup.trRollupAverage
```

## RegularWeight Property

---

The RegularWeight property sets or returns a measure name used in a weighted average calculation.

### Syntax

*Measure*.RegularWeight

### Applies To

[Measure Object](#)

### Discussion

Use the RegularWeight property to obtain a weighted average for the rollup measure instead of a true average.

The weighted average of measure A, which uses measure B as a weight, is calculated by using an equation that looks like this:

$$\sum(A_i * B_i) / \sum B_i$$

The measure that is rolled up must have an xtrRollup setting of trRollupAverage, and the measure specified as the weighting measure must have an xtrRollup setting of trRollupDefault or trRollupSum.

You cannot specify a weighted average for both regular and time-state rollups.

### Type

String

### Access

Read/Write

### Examples

```
objMeasures = objModel.MeasurescurrentMeasure  
= objMeasures("Revenue").currentMeasure.RegularWeight  
= objModel.Measures("Quantity").Name
```

## Reports Property

---

The Reports property returns a collection of Report objects associated with a Transformer model.

### Syntax

*Model*.Reports

### Applies To

[Model Object](#)



## Discussion

Use this property to iterate through the reports for a Model object, add a new report, and change or remove a report.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Objects - Reports

## Access

Read

## Examples

```
new_report = model.Reports.Add()
```

## ReverseSign Property

---

The ReverseSign property sets or returns whether PowerPlay reverses the sign of a measure.

## Syntax

*object*.ReverseSign

## Applies To

[Category Object](#)

[Measure Object](#)

[SpecialCategory Object](#)

## Discussion

When you set the ReverseSign property to True, you reverse the sign of values for a measure and categories based on that measure.

Use this property to reverse the sign in financial reporting systems, where measures used as expenses must show a positive sign in certain contexts and a negative sign in others.

ReverseSign must be True for both the measure and the associated category if PowerPlay is to show the values with the sign reversed.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objMeasure.CategoryCountLevel = objLevelobjMeasure.CategoryCountLevel  
= objLevel objMeasure.ReverseSign = False
```

## Rollup Property

---

The Rollup property sets or returns whether measure values for a special category roll up into the parent category.

### Syntax

*SpecialCategory* .**Rollup**

### Applies To

[SpecialCategory Object](#)

### Discussion

A category with a rolled up value shows a summary of child category values. When this property is set to False for all children of a special category, that category serves only as a place holder within the hierarchy of special categories.

**Default:** True

### Type

Boolean

### Access

Read/Write

### Examples

```
objSpecCategory  
= - objModel.Dimensions("Time").Categories.Add(xtrObjectType  
    .trSpecialCategory)objSpecCategory.Rollup  
= True
```

## RollupTiming Property

---

The RollupTiming property sets or returns when calculations are performed for calculated measures.

### Syntax

*Measure* .**RollupTiming**

### Applies To

[Measure Object](#)

## Discussion

Categories with rolled-up values show a summary of their respective child category values. Calculated categories may perform calculations on measure values before or after the rollup of these values. Use this property to specify the timing of these calculations.

If a calculated measure uses `trTimingBeforeRollup`, you can use neither auto-partitioning nor record consolidation. To include auto-partitioning and consolidation in a cube, use calculated columns instead of calculated measures. Calculated columns can perform the same calculations before rollup.

This property uses the values of `xtrRollupTiming`.

## Type

Constant - `xtrRollupTiming`

## Access

Read/Write

## Examples

```
objMeasures = objModel.MeasurescurrentMeasure  
= objMeasures("Revenue")currentMeasure.RollupTiming  
= xtrRollupTiming.trTimingDefault
```

# RowsAsSample Property

---

The `RowsAsSample` property sets or returns the number of rows that the `DoAutoDesign` method samples when it creates a model.

## Syntax

*Application* .**RowsAsSample**

## Applies To

[Application Object](#)

## Discussion

Use this property to specify how many rows the `DoAutoDesign` method samples as it determines possible hierarchical relationships for levels. The number should be no more than the number set by the `RowsChecked` property.

Minimum: 10 rows. Maximum: 1000 rows.

**Default:** 300 rows.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read/Write

## Examples

```
objTransApp.RowsAsSample = 600
```

## RowsChecked Property

---

The RowsChecked property sets or returns the maximum number of rows that the DoAutoDesign method reads from the data source.

### Syntax

*Application* .**RowsChecked**

### Applies To

[Application Object](#)

### Discussion

The DoAutoDesign method checks rows to determine which columns are sources for levels and measures.

Specify more rows to increase the accuracy of the DoAutoDesign method. Specify fewer rows to increase the speed at which Transformer creates the model.

If you specify zero, the DoAutoDesign method reads the entire data source.

**Default:** 600 rows.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Long

### Access

Read/Write

### Examples

```
objTransApp.RowsChecked = 600
```

## RunningPeriods Property

---

The RunningPeriods property sets or returns the number of time periods used for calculating running totals.

### Syntax

*SpecialCategory* .**RunningPeriods**

### Applies To

[SpecialCategory Object](#)

## Discussion

Use this property with the TargetLevel, ContextLevel, TargetOffset, and ContextOffset properties to define a special category that tracks measures for specific periods of time relative to the current time.

To use this property, the Aggregate property must be set to either trAggregateRunning or trAggregateRunningGrouped. For example, create a running total category that spans the six months leading up to the current month, for this year and last year.

```
Aggregate = trAggregateRunningGrouped
```

```
ContextLevel = "Year"
```

```
ContextOffset = -1
```

```
TargetLevel = "Month"
```

```
TargetOffset = 0
```

```
RunningPeriods = 6
```

## Type

Long

## Access

Read/Write

## Examples

```
objSpecCategory  
= - objModel.Dimensions("Time").Categories  
  .Add(xtrObjectType.trSpecialCategory)objSpecCategory.RunningPeriods  
= 5
```

## SecurityObjects Property

---

The SecurityObjects property returns a collection of SecurityObjects.

## Syntax

*Object* .**SecurityObjects**

## Applies To

[CustomView Object](#)

[Namespace Object](#)

## Discussion

Use this property to iterate through the security objects for a Namespace, add new security objects, and change or remove a security object.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

SecurityObjects

## Access

Read

## Examples

```
new_namespace = model.Namespaces.Add()securityObject  
= new_namespace.SecurityObjects.Add()
```

# Server Property

---

The Server property sets or returns whether a cube is processed locally or on a server.

## Syntax

*object* .**Server**

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

## Discussion

True means the cube is processed on a server.

Use the ServerModelPath to specify the name of the model file (.mdl) on the server. Use the ServerConnect property to specify the connection name of the server.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
securityObject = new_namespace.SecurityObjects.Add()
```

# ServerModelPath Property

---

The ServerModelPath property sets or returns the name of a model file (.mdl) on a server.

## Syntax

*Model* .**ServerModelPath**

## Applies To

[Model Object](#)

## Discussion

Use this property to specify the file name. Transformer (on UNIX) saves the model in the default directory. To save the model file to another directory, create a preference setting that specifies the full path and file name.

Use the ServerConnect property to specify the connection string.

## Type

String

## Access

Read/Write

## ServerPath Property

---

The ServerPath property sets or returns the name and location of a data source file.

## Syntax

*object* .ServerPath

## Applies To

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

## Discussion

You can specify the full path or the file name only. If you only specify the file name, Transformer checks the DataSourcePath property for a directory name. If the DataSourcePath is not specified, Transformer checks the PowerPlay installation directory.

If the Transformer model is processed locally, use the LocalPath property to specify the location of data sources.

## Type

String

## Access

Read/Write

## ServerQuery Property

---

The ServerQuery property sets or returns whether data is processed locally or on a server.

## Syntax

*object* .**ServerQuery**

## Applies To

[CrossTabDataSource Object](#)

[DataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

## Discussion

When set to True, data is processed on a server. In such a case, use the ServerPath property to specify the location of the data source.

**Default:** False

## Type

Boolean

## Access

Read/Write

# ServicesBuildNumber Property

---

The ServiceBuildNumber property returns the build number of Transformer.

## Syntax

*Application* .**ServicesBuildNumber**

## Applies To

[Application Object](#)

## Discussion

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read

# ServicesVersionText Property

---



The `ServicesVersionText` property returns the build version of Transformer.

## Syntax

*Application* .**ServicesVersionText**

## Applies To

[Application Object](#)

## Discussion

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

String

## Access

Read

# SetsCurrentPeriod Property

---

The `SetsCurrentPeriod` property sets or returns whether Transformer searches a data source to find the current period date.

## Syntax

*object* .**SetsCurrentPeriod**

## Applies To

[CrossTabDataSource Object](#)

[DataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

[Query Object](#)

## Discussion

When set to True, Transformer uses the associated data source to determine the current period date; that is, the latest date in the date column.

Transformer uses the current period date to evaluate relative time categories. For example, if the current date is 20010831, the Prior Month category shows a value of July.

If the model contains multiple data sources, choose which data sources Transformer examines to find the current period date. If this property is not specified, Transformer checks all data sources and selects the latest date.

Set this property to False in a data source used to populate a currency table. Alternatively, ensure that the name of the date column is different from the one used by the time dimension.

**Default:** True

## Type

Boolean

## Access

Read/Write

## Examples

```
objDataSource  
= objModel.DataSources  
  .Add(xtrObjectType.trFlatFileDataSource)objDataSource.SetsCurrentPeriod  
= True
```

## ShortName Property

---

The ShortName property sets or returns a short name for an object.

## Syntax

*object* .**ShortName**

## Applies To

[Category Object](#)

[DateDrillDown Object](#)

[DrillDown Object](#)

[Measure Object](#)

[SpecialCategory Object](#)

## Discussion

Use this property to show a more meaningful name for an object in PowerPlay.

## Type

String

## Access

Read/Write

## Examples

```
objMeasures = objModel.MeasurescurrentMeasure  
= objMeasures("Revenue")currentMeasure.ShortName  
= "Revenue"
```

## Signon Property

---

The Signon property sets or returns the Signon object associated with the cube, cube group, or a package datasource connection (IBM Cognos signon).

## Syntax

*object* .**Signon**

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

[PackageDatasourceConnection Object](#)

## Discussion

When you add an Impromptu query definition file (.iqd) to your model, Transformer automatically adds a Signon object. The Signon object contains the logical database name and may contain user ID and password information.

To use the PackageDatasourceConnection signon, a signon object needs to be added to provide authentication to an external namespace. This allows users to build cubes in batch mode. To enable Transformer to use the IBM Cognos signon automatically, the AutoLogon property of the Signon object needs to be enabled. The signon maintains the user ID, password, and associated namespace.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Signon (Cube and Cube Group objects)

String (PackageDatasourceConnection)

## Access

Read/Write

## Examples

```
package = model.Packages.Add()connection
= package.PackageDatasourceConnections.Add()connection.Signon
= "great_outdoors_warehouse"
```

## SignOnNamespace Property

---

The SignOnNamespace property contains the security namespace associated with the IBM Cognos signon.

## Syntax

*Signon* .**SignonNamespace**

## Applies To

[Signon Object](#)

## Discussion

The SignOnNamespace property applies only to the IBM Cognos signon type.

## Type

Boolean

## Access

Read/Write

## Examples

```
signon = model.Signons.Add() signon.SignOnNamespace =  
"Cognos"
```

## Signons Property

---

The Signons property returns the Signons collection for a model.

## Syntax

*Model* .**Signons**

## Applies To

Model Object

## Discussion

Before you can add or modify Signon objects, you must first return the Signons collection.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Object

## Access

Read

## Examples

```
signon = model.Signons.Add()
```

## SignonType Property

---

The SignonType property sets or returns the signon type, xtrSignonType.

## Syntax

*Signon* .**SignonType**

## Applies To

[Signon Object](#)

## Discussion

trDataSourceSignon is used for a data source signon and xtrSignonType. trCognosSignon is used for an IBM Cognos signon.

## Type

Constant - xtrSignonType

## Access

Read/Write

## Examples

```
signon = model.Signons.Add()signon.SignonType = xtrSignonType.trDataSourceSignon
```

# Size Property

---

The Size property sets or returns the size of a column or Model file (.mdl).

## Syntax

*object* .Size

## Applies To

[Column Object](#)

[Model Object](#)

## Discussion

For a Column object, this property sets or returns the size of the column in bytes for some data sources. Column size applies only when the SourceType property associated with the data source has a value of trPowerHousePortable, trFixedAscii, or trFixedAsciiNoCRLF.

For a Model object, this property returns the size of the file as it appears in a Windows folder or Windows Explorer.

## Type

Long (Column)

String (Model)

## Access

Read/Write (Column)

Read (Model)

## Examples

```
MsgBox(objModel.FileName & " " & .Size &  
" " & .Time)
```

## SortComparisonRule Property

---

The SortComparisonRule property sets or returns which text comparison rule Transformer uses to sort category names.

### Syntax

*Application* .SortComparisonRule

### Applies To

Application Object

### Discussion

You can choose either the comparison rule determined by regional settings in the Windows control panel, or the Transformer internal comparison rule, which is a byte-by-byte comparison of the binary representation of each string.

If you change the setting of this property for an existing model, the new rule applies only to new categories. Existing categories are not resorted. Therefore, existing categories may not be sorted consistently with new categories.

To have Transformer re-sort categories in a level, you must delete the Association object that contains the order-by information, and then create a new one.

To sort categories in a level, create an Association object for that level and set the AssociationRole property to trAssociationOrderBy.

This property uses values of xtrPreferences.

### Type

Constant - xtrPreferences

### Access

Read/Write

### Examples

```
objTransApp.SortComparisonRule = xtrPreferences.trSortIgnoreControlPanel
```

## SourceType Property

---

The SourceType property sets or returns the data file type.

### Syntax

*object* .SourceType

## Applies To

[CrossTabDataSource Object](#)

[DataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

## Discussion

Each data source object in a model references a type of data file. Use this property and the values of `xtrSourceType` to define the appropriate source.

When you add a data source to a collection, the `Type` parameter of the `Add` method uses a constant from the `xtrObjectType` value list. For example, the `Add` method uses the `trCrossTabDataSource` parameter to specify a spreadsheet file. You then use the `SourceType` property to specify how Transformer reads the file. In this example, the `SourceType` property determines if the spreadsheet file is a crosstab or a database file.

## Type

Constant - `xtrSourceType`

## Access

Read/Write

## Examples

```
objDataSource  
= objModel.DataSources  
  .Add(xtrObjectType.trFlatFileDataSource)objDataSource.SourceType  
= xtrSourceType.trFlatFileColumnNames
```

# SpecialCategoryCount Property

---

The `SpecialCategoryCount` property returns the number of drill, root, and special categories in a dimension.

## Syntax

*object* .**SpecialCategoryCount**

## Applies To

[DateDimension Object](#)

[Dimension Object](#)

## Discussion

This property only counts drill, root, and special categories. Use the `CategoryCount` property to count Category objects.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read

# SQLExpression Property

---

The SQLExpression property returns an SQL expression.

## Syntax

*IqdDataSource* .**SQLExpression**

## Applies To

IqdDataSource Object

## Discussion

Use this property to examine the SQL expression on which an Impromptu query definition file (.iqd) is based.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

String

## Access

Read/Write

# Status Property

---

The Status property returns the previous creation status of a cube.

## Syntax

*object* .**Status**

## Applies To

ChildCube Object

Cube Object

CubeGroup Object

## Discussion

Use status information to qualify a cube for production. For example, you can set the CubeCreation property to False for all cubes for which warnings were issued.

The Status property uses the values of xtrCubeStatus.



A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Constant - xtrCubeStatus

### Access

Read

## StorageType Property

---

The StorageType property sets or returns the size of a numeric data type.

### Syntax

*object*.StorageType

### Applies To

[Column Object](#)

[Measure Object](#)

### Discussion

Use StorageType to specify how Transformer stores measure data within work files used to build cubes. Specify a constant from the xtrStorage value list that is large enough to contain the source values.

Ensure that you specify a value large enough to handle rollup values and record consolidation.

### Type

Constant - xtrStorage

### Access

Read/Write

### Examples

```
objMeasure.StorageType = xtrStorage.trStorageDefault
```

## StreamExtractAllowed Property

---

The StreamExtractAllowed property sets or returns whether stream extraction is allowed. Stream extraction applies only to a SAP BW data source.

### Syntax

*Query*.StreamExtractAllowed

### Applies To

[Query Object](#)

## Discussion

This property is ignored for non-SAP BW data sources.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Boolean

## Access

Read/Write

## StreamExtractSize Property

---

The StreamExtractSize property sets or returns the size of the buffer, in megabytes, used to transfer data from SAP when StreamExtract is set to true.

## Syntax

*Query*.**StreamExtractSize**

## Applies To

[Query Object](#)

## Discussion

This property will be set to 0 (and ignored) for structure queries.

It will be set to 10 for a transaction query that is constructed for fetching the Measure data.

When this value is not zero (0) and all the Columns have the correct origin, Stream Extract is used to read the data.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Integer

## Access

Read/Write

## SummaryLevel Property

---

The SummaryLevel property sets or returns which level to use to summarize external categories in a cube group.

## Syntax

*CubeGroup*.**SummaryLevel**

## Applies To

[CubeGroup Object](#)

## Discussion

Use this property to summarize the categories of the other cubes in the cube group.

Child cubes in a cube group each represent one category. From the perspective of a single child cube, external categories are all categories in the dimension other than the category referenced by that child cube and the descendant categories of the category. In PowerPlay, users can see data from other cubes in the cube group down to the level specified by this property.

If no value is specified, Transformer excludes all external categories from each cube in the cube group.

## Type

Object

## Access

Read/Write

## Examples

```
objCubesByRegion  
= objModel.Cubes.Add(xtrObjectType.trCubeGroup)objCubesByRegion.SummaryLevel  
= objRegionsDrill.Levels("Sales region")
```

## SuppressNull Property

---

The SuppressNull Property sets or returns the null suppression option used for SAP BW data sources.

## Syntax

*Query*.**SuppressNull**

## Applies To

[Query Object](#)

## Discussion

This option applies to SAP BW data sources only.

Some queries can be very large because null values are not filtered out. Null suppression removes a row or column for which all of the values in the row or column are null (empty). Null Suppression is performed by SAP BW. This reduces the amount of data transferred to Transformer and improves performance.

Applying null suppression to a data source controls null suppression during data retrieval. This differs from applying null suppression when publishing packages, in that, the latter controls the display of null values only.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Integer

## Access

Read/Write

## SuspendedModels Property

---

The SuspendedModels property returns a collection of SuspendedModel objects.

### Syntax

*Application* .**SuspendedModels**

### Applies To

Application Object

### Discussion

If a model is closed abruptly, such as during a system outage, Transformer adds a SuspendedModel object to the SuspendedModels collection.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object - SuspendedModels

### Access

Read

### Examples

```
objSuspendedModel = objTransApp.SuspendedModels(intX)
```

## TargetLevel Property

---

The TargetLevel property sets or returns the level of detail of a date period.

### Syntax

*SpecialCategory* .**TargetLevel**

### Applies To

SpecialCategory Object

### Discussion

Special categories track measures for a specific period of time relative to the current date period.

Use this property to specify a reporting period. For example, If the target period is set to Month, PowerPlay shows a time period of month.

Use the TargetOffset, ContextLevel, ContextOffset and RunningPeriods properties to answer questions about the target period. For example, if the target period is Month, the properties answer the questions:

- Is it the current month (TargetOffset)?
- Is it set against a period of Quarter or Year (ContextLevel)?
- Is it the current Quarter or Year (ContextOffset)?

- How many target periods (RunningPeriods)?

### Type

String

### Access

Read/Write

### Examples

```
objSpecCategory
= objModel.Dimensions("Time").Categories.Add(xtrObjectType.
  trSpecialCategory)objSpecCategory.TargetLevel
= "Month"
```

## TargetOffset Property

---

The TargetOffset property sets or returns the position of the target period relative to the current period.

### Syntax

*SpecialCategory*.TargetOffset

### Applies To

[SpecialCategory Object](#)

### Discussion

Use this property to specify an offset value relative to the current time period for the Target Level property. For example, if the TargetLevel is set to Month and you want a special category for the previous month, set the TargetOffset to -1.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Short

### Access

Read/Write

### Examples

```
objSpecCategory
= objModel.Dimensions("Time").Categories
  .Add(xtrObjectType.trSpecialCategory)objSpecCategory.TargetOffset
= -1
```

## ThousandPoint Property

---

The ThousandPoint property sets or returns the character used to separate numbers in thousands.

## Syntax

*Object* .**ThousandPoint**

## Applies To

[CrossTabDataSource Object](#)

[DbDataSource Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

## Discussion

For example, use this property to

- specify a comma in a value, such as 1,000
- specify a space in a value, such as 1 000

You cannot use a null character, that is, no character for this property.

## Type

String

## Access

Read/Write

# Time Property

---

The Time property returns the time stamp of a model as it appears in a Windows folder or Windows Explorer.

## Syntax

*Model* .**Time**

## Applies To

[Model Object](#)

## Discussion

Use this property to check the date a model was last modified.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

String

## Access

Read

## Examples

```
MsgBox(objModel.FileName & " " & .Size &  
" " & .Time)
```

## TimeArrayColumn Property

---

The TimeArrayColumn property sets or returns the name of the first column in a date array.

### Syntax

*Column* .TimeArrayColumn

### Applies To

[Column Object](#)

### Discussion

A date array is a group of four or twelve contiguous columns that contain quarterly or monthly transaction values. If you store your transaction data monthly or quarterly, we recommend that you define the columns as members of a date array rather than as individual measures.

Each value in the date column represents the first month of a fiscal year. Each transaction column covers a time period such as month or quarter. For example, to report at the Quarterly level, your data source may include rows similar to this:

```
DATE, PRODUCT, Q1, Q2, Q3, Q4
```

```
1999Q1, Product1, 100, 200, 150, 400
```

```
1999Q1, Product2, 1110, 2265, 1995, 4200
```

```
2000Q1, Product1, 110, 210, 160, 420
```

After you create a time dimension, return the first column in the array and then use the Name property of the same column object to set TimeArrayColumn. You must also set the TimeArrayType and TimeArrayStartMonth properties. Transformer automatically adds the other columns to the array and sets their data class to trDataClassArrayMember. Finally, add only the first member of the array to the Measures collection. If you have more than one array, add the first member of each array to the Measures collection.

### Type

String

### Access

Read/Write

## TimeArrayStartMonth Property

---

The Time ArrayStartMonth property sets or returns the first month of the fiscal year in a date array.

## Syntax

*Column* .TimeArrayStartMonth

## Applies To

Column Object

## Discussion

A date array is a group of four or twelve contiguous columns that contain quarterly or monthly transaction values. If you store your transaction data monthly or quarterly, we recommend that you define the columns as members of a date array rather than as individual measures.

This property names the first month of the year regardless of the period the array spans.

You must also set the TimeArrayColumn and TimeArrayType properties.

## Type

Long

## Access

Read/Write

# TimeArrayType Property

---

The TimeArrayType property sets or returns the type of array used for date values.

## Syntax

*Column* .TimeArrayType

## Applies To

Column Object

## Discussion

A date array is a group of four or twelve contiguous columns that contain quarterly or monthly transaction values. If you store your transaction data monthly or quarterly, we recommend that you define the columns as members of a date array rather than as individual measures.

This property specifies an array of months or quarters. You must also set the TimeArrayColumn and TimeArrayStartMonth properties.

TimeArrayType uses the values of xtrTimeArrayType.

## Type

Constant - xtrTimeArrayType

## Access

Read/Write



## TimeRank Property

---

The TimeRank property sets or returns the relative rank of date levels within a time dimension.

### Syntax

*DateLevel* .TimeRank

### Applies To

DateLevel Object

### Discussion

The absolute value of a rank is not important; only the value relative to other date value ranks is important. When you add a DateLevel object to a time dimension, set this property to a value greater than the previous date level.

When Transformer assigns ranks, it sets a value 10 greater than the rank of the previous level, or half way between the previous and next levels, whichever is smaller.

Time dimensions with multiple drill-down paths may produce levels whose relative ranking is ambiguous. Transformer assumes that two levels in the same dimension are equal (that is, are the same type of period) if they hold the same rank. This assumption holds true even if the names or source columns for the levels are different.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Long

### Access

Read/Write

## TimeStamp Property

---

The TimeStamp property sets or returns the time stamp of a Package or Report object.

### Title

*object* .TimeStamp

### Applies To

Package Object

### Discussion

Use this property to provide or get the time stamp for a package or report. Here is an example of how the time stamp for a package may appear.

```
/content/folder/[@name='Oracle']/package[@name='oracle_gosales']  
/model[@name='2008-04-24T17:46:18.796Z']
```

## Type

String

## Access

Read/Write

## Examples

```
new_package = model.Packages.Add()new_package.TimeStamp
= "/content/package[@name='G0 Data Warehouse (analysis)']" _
& "/model[@name='" & timestamp & "']"
```

## TimeStateRollup Property

---

The TimeStateRollup property sets or returns the date period used for time state rollups.

### Syntax

*Measure*.TimeStateRollup

### Applies To

Measure Object

### Discussion

A rollup summarizes measure values used by categories. Time State Rollup is how Transformer represents the state of a measure at specific times.

First, set the DateDegreeofDetail property to the lowest level of detail. For example, if your source file contains daily transactions, your level of detail should be day. Then, use a constant from xtrTimeRollup value list to set TimeStateRollup.

When you set both the RegularRollup and TimeStateRollup properties, PowerPlay performs the regular rollup first, and then the time state rollup. However, regular rollup is not applied to the time dimension.

Ensure that you set the TimeStateWeight property when the TimeStateRollup property is set to trTimeRollupAverage.

The values of the CanAllocateMeasure, Consolidate, DuplicateRollup, RollupTiming, and RegularRollup properties also play a part in how and when measures are rolled up.

### Type

Constant - xtrTimeRollup

### Access

Read/Write

## TimeStateWeight Property

---

### Description

The TimeStateWeight property sets or returns a measure name used in a weighted average calculation.

## Syntax

*Measure* .TimeStateWeight

## Applies To

Measure Object

## Discussion

Use the TimeStateWeight property to obtain a weighted average for the rollup measure instead of a true average. For example, if measure A uses measure B for a weighted average, the equation used in the calculation looks like this:

$$\sum(A_i * B_i) / \sum B_i$$

The measure being rolled up must have the TimeStateRollup property set to trTimeRollupAverage, and the measure named as the weighting measure must have the RegularRollup property set to trRollupDefault or trRollupSum.

You cannot specify a weighted average for both regular and time-state rollups.

## Type

String

## Access

Read/Write

# ToDateLevel Property

---

The ToDateLevel property sets or returns the date period used for to-date totals.

## Syntax

*SpecialCategory* .ToDateLevel

## Applies To

SpecialCategory Object

## Discussion

Special categories track measures for a specific period of time relative to the current date period.

Use the ToDateLevel property to specify a reporting period. For example, if the ToDateLevel property is set to Quarter, PowerPlay shows a time period of quarter. To use this property, you must set the Aggregate property to trAggregateToDate or trAggregateToDateGrouped.

Use the ContextLevel and ContextOffset properties to answer questions about the to-date level value. From the previous example, if the to-date level value is Quarter, the properties answer these questions:

- Is it set against a period of Quarter or Year (ContextLevel)?
- Is it the current Quarter or Year (ContextOffset)?

## Type

String

## Access

Read/Write

## TransdaPath Property

---

The TransdaPath property sets or returns the location of the transda.exe executable. The executable is installed as part of the Transformer application installation. It is installed in the location *installation\_directory*/CS7Gateways/bin where *installation\_directory* can be

## Syntax

```
c:\Program Files\Cognos\
```

*Application* .**TransdaPath**

## Applies To

[Application Object](#)

## Discussion

Use this property to direct where Transformer can find the transda.exe file.

## Type

String

## Access

Read/Write

## TransformerSignon Property

---

The TransformerSignon property sets or returns the Transformer signon object associated with a package datasource connection (IBM Cognos signon).

## Syntax

*PackageDatasourceConnection* .**TransformerSignon**

## Applies To

[PackageDatasourceConnection Object](#)

## Discussion

To use the PackageDatasourceConnection signon, a signon object needs to be added to provide authentication to an external namespace. This allows users to build cubes in batch mode. To enable Transformer to use an IBM Cognos signon automatically, the AutoLogon property of the Signon object needs to be enabled. The signon maintains the user ID, password, and associated namespace. This value is used instead of the Signon Property if the AlwaysUseTransformerSignon Property is set to True.

## Type

String (PackageDatasourceConnection)

## Access

Read/Write

## Examples

```
package  
= model.Packages.Add()connection  
= package.PackageDataSourceConnections.Add()connection.TransformerSignon  
= "great_outdoors_warehouse"
```

## Type Property

---

The Type property returns an object type.

## Syntax

*object* .**Type**

## Applies To

[“Application Object” on page 39](#)

[Association Object](#)

[CalculationDefinition Object](#)

[Category Object](#)

[CategorySet Object](#)

[ChildCube Object](#)

[Column Object](#)

[CrossTabDataSource Object](#)

[Cube Object](#)

[CubeGroup Object](#)

[CurrencyRate Object](#)

[CurrencyRecord Object](#)

[CurrencyTable Object](#)

[CustomView Object](#)

[DataSource Object](#)

[DateDimension Object](#)

[DateDrillDown Object](#)

[DateLevel Object](#)

[DateWizard Object](#)

[DbDataSource Object](#)

[Dimension Object](#)

[DrillDown Object](#)

[DrillThroughTarget Object](#)

[FlatFileDataSource Object](#)

[IqdDataSource Object](#)

[Level Object](#)

[Measure Object](#)

[Model Object](#)

[Name Object](#)

[Package Object](#)

[Prompt Object](#)

[“Report Object” on page 96](#)

[SecurityObject Object](#)

[Signon Object](#)

[SpecialCategory Object](#)

[SuspendedModel Object](#)

[View Object](#)

## Discussion

Use this property to determine the type of object. This return value is useful when you retrieve an object from a collection that can contain more than one type of object. For example, an object returned from a Dimensions collection can be a Dimension object or a DateDimension object.

The value of the Type property for each object in Transformer corresponds to a constant of xtrObjectType, except for the DataSource object. In this case, the value of the Type property is set by the object in the DataSources collection that the DataSource object is currently representing.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Constant - xtrObjectType

## Access

Read

## Examples

```
new_namespace  
= model.Namespaces.Add()securityObject  
= new_namespace.SecurityObjects.Add()securityObject.Type  
= xtrSecurityType.trSecurityType_Role
```

## Unique Property

---

The Unique property sets or returns whether each category in the level can be identified by a unique source value.

## Syntax

*object*.Unique

## Applies To

[DateLevel Object](#)

[Level Object](#)

## Discussion

You specify a level as unique when it is the convergence level for alternate drill-down paths, or when the model contains multiple data sources.

When set to True, this property informs Transformer that categories in that level are identified by their source values alone, without reference to their ancestors. No two categories in that level can have the same value.

If a data source that contains the column for a level also contains all columns required for the ancestor levels, you do not have to specify uniqueness. In this case, Transformer can identify the categories in the level from the presence of the other columns in the data source. However, when the columns for the ancestor levels come from different data sources, the values for categories in that level must be specified as unique.

When you indicate that categories are unique, Transformer does not verify your assertion. If you identify a level as unique when it is not, Transformer generates incorrect dimensions.

When you move a category from a unique level, a uniqueness violation is reported during category generation because the moved category now appears in a different context. Use the UniqueMove property to control this problem.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objLocationsDim
= objModel.Dimensions("Sales regions")objLevel
= objLocationsDim.DrillDowns(1).Levels("Employee")objAltDrill
= objLevel.CreateAlternateDrillDownobjNewLevel
= objAltDrill.Levels.Add(xtrObjectType.trLevel)objNewLevel.Unique
= True
```

## UniqueMove Property

---

The UniqueMove property sets or returns how unique levels are treated when you move a category.

## Syntax

*object*.UniqueMove

## Applies To

[DateLevel Object](#)

[Level Object](#)

## Discussion

Use the UniqueMove property to avoid having to manually restructure the categories in a unique level to conform to changes in source data. A uniqueness violation can occur during category generation when you move a category from a unique level because the moved category now appears in a different context.

Set this property to True to specify that such changes are to be treated as unique moves. Measure values, even those accumulated under the old data structure, are thereafter rolled up the new path to the moved categories.

This property is available only for levels with their Unique property set to True.

**Default:** False

## Type

Boolean

## Access

Read/Write

## Examples

```
objLocationsDim = objModel.Dimensions("Sales regions")
objLevel = objLocationsDim.DrillDowns(1).Levels("Employee") objLevel.UniqueMove
= True
```

## UseAltMDCFile Property

---

The UseAltMDCFile property sets or returns whether a temporary filename may be used.

## Syntax

*object* .**UseAltMDCFile**

## Applies To

[Cube Object](#)

[CubeGroup Object](#)

[ChildCube Object](#)

## Discussion

Use the UseAltMDCFile property to check if the cube can be saved using an alternate filename.

If this property is set to true, then the object may be saved using the temporary filename. This property is available for Cube object, CubeGroup object and ChildCube object.

If the IsMDCInUse property returns true, then the cube is in use by another application. You can then set the UseAltMDCFile property to true and set the AltMDCFile property to the alternate filename.

**Default:** False

## Type

Boolean



## Access

Read/Write

## User Property

---

The User property sets or returns a user name associated with the namespace.

### Syntax

*Namespace*.**User**

### Applies To

[Namespace Object](#)

### Discussion

Use this property to set the user name of a user to get the CAMID of the user from the UserCAMID property.

### Type

String

### Access

Read/Write

### Examples

```
new_namespace = model.Namespaces.Add() new_namespace.User  
= Name_of_User
```

## UserCAMID Property

---

The UserCAMID property sets or returns the user ID for the Signon object.

### Syntax

*Namespace*.**UserCAMID**

### Applies To

[Namespace Object](#)

### Discussion

Use this property to get the CAMID of a user in a namespace.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

String

## Access

Read

## Examples

```
new_namespace = model.Namespaces.Add() CAMID_of_User = new_namespace.UserCAMID
```

## UserID Property

---

The UserID property sets or returns the user ID for a signon.

### Syntax

*Signon* .UserID

### Applies To

Signon Object

### Discussion

Transformer uses information from Signon objects to automatically access secure databases. You may not be able to change the value for this property if

- the Signon object is an IBM Cognos Signon object
- it references an Impromptu query definition file (.iqd) signon

## Type

String

## Access

Read/Write

## Examples

```
signon = model.Signons.Add() signon.UserID = "sa"
```

## Value Property

---

The Value property sets or returns the prompt value.

### Syntax

*Prompt* .Value

## Applies To

[Prompt Object](#)

## Discussion

This property is used to set or get the prompt value associated with the `CurrentValueIndex` of a Prompt object.

## Type

String

## Access

Read/Write

## Examples

```
new_report = model.Reports.Add()new_query = new_report.Queries.Add()new_prompt.Value  
= "2005-01-01"
```

# ValuesCount Property

---

The `ValuesCount` property returns the number of values set for the prompt.

## Syntax

*Prompt*.**ValuesCount**

## Applies To

[Prompt Object](#)

## Discussion

This property can be used to iterate through the prompt values in conjunction with the `CurrentValueIndex` property.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Integer

## Access

Read

## Examples

```
new_report = model.Reports.Add()new_query  
= new_report.Queries.Add()new_prompt.CurrentValueIndex  
= indexFor index = 1 To new_prompt.ValuesCount
```

## Version Property

---

The Version property returns the version number of Transformer.

### Syntax

*Application* .**Version**

### Applies To

[Application Object](#)

### Discussion

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

String

### Access

Read

## Views Property

---

The Views property returns a collection of View objects associated with a dimension, time dimension, or view.

### Syntax

*object* .**Views**

### Applies To

[CustomView Object](#)

[DateDimension Object](#)

[Dimension Object](#)

### Discussion

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Object

### Access

Read

## Examples

```
objView = objDimension.Views.Add()
```

## ViewType Property

---

The ViewType property sets or returns whether a view contains all, some, or none of the categories in a dimension.

### Syntax

*View* .**ViewType**

### Applies To

[View Object](#)

### Discussion

By default, the type of view for a dimension is trViewTypeAllCategories.

To define a custom view, set the ViewType property to trViewTypeCustom. You can then use the

- Apex property to create a view of one category and child categories
- SetViewStatus method to define a view for a category or level

The ViewType property uses the values of xtrViewType.

### Type

Constant - xtrViewType

### Access

Read/Write

## Examples

```
objProductsDim = objModel.Dimensions.Item("Products")objViewItem.ViewType  
= xtrViewType.trViewTypeCustom
```

## WeekAdd Property

---

The WeekAdd property sets or returns how many days are added to a lunar year.

### Syntax

*object* .**WeekAdd**

### Applies To

[DateDrillDown Object](#)

[DateWizard Object](#)

## Discussion

When you use lunar time periods, a year consists of 52 weeks of seven days each. A lunar year, therefore, contains 364 days, which is either one or two days less than a calendar or leap year, respectively.

Use this property to add a week to the lunar year. By doing so, you synchronize the lunar year with the calendar year. The extra week may contain four, five, six or seven days. Alternatively, you can specify None.

Transformer adds this extra week to the last month or quarter of the year. That way, each lunar year remains aligned with the specified Start-of-year day, while not falling too far out of alignment with the calendar year.

If the last lunar month in the year has five weeks in a 4-4-5 pattern as set by `xtrSpecialFunction`, the extra week in the quarter is added to the previous lunar month to make a 4-5-5 pattern, rather than 4-4-6, which is not a valid pattern in Transformer.

This property uses the values of `xtrWeekAdd`.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Constant - `xtrWeekAdd`

## Access

Read/Write (`DateDrillDown`)

Write (`DateWizard`)

## Examples

```
objDateWizard  
= objModel.DateWizardobjDateDim  
= objDateWizard.CreateDateDimension()objDrillDown  
= objDateDim.DrillDowns(1)objDrillDown.WeekAdd  
= xtrWeekAdd.trWeekAddDays4
```

## WeekSpan Property

---

The WeekSpan Property sets or returns how to treat a week that spans two years.

## Syntax

*object* .**WeekSpan**

## Applies To

[DateDrillDown Object](#)

[DateWizard Object](#)

## Discussion

If a time dimension contains a week level based on a calendar year, the last week will probably not end on the same day as the year ends. (A lunar year always ends at the end of the week.) WeekSpan specifies how to shift or split up the last week of a year.

This property uses the values of `xtrWeekSpan`.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Constant - xtrWeekSpan

## Access

Read/Write (DateDrillDown)

Write (DateWizard)

## Examples

```
objDateWizard = objModel.DateWizardobjDateWizard.WeekSpan  
= xtrWeekSpan.trWeekSpanNone
```

# WeekStart Property

---

The WeekStart property sets or returns the first day of the week.

## Syntax

*DateDrillDown* .**WeekStart**

## Applies To

DateDrillDown Object

## Discussion

When a model includes a week level, Transformer uses this property to determine which day is the first day of the week.

This property uses the values of xtrWeekDay.

When you create alternate drill-down paths in a lunar time dimension that includes quarters or lower levels of detail, both the WeekStart and the WeekAdd properties must align.

For information about alternate drill-down paths in time dimensions, see the Transformer online help.

## Type

Constant - xtrWeekDay

## Access

Read/Write

## Examples

```
objDateWizard  
= objModel.DateWizardobjDateDim  
= objDateWizard.CreateDateDimension()objDrillDown  
= objDateDim.DrillDowns(1)objDrillDown.WeekStart  
= xtrWeekDay.trMonday
```

## WeekStartDay Property

---

The WeekStartDay property sets the first day of the week.

### Syntax

*DateWizard* .**WeekStartDay**

### Applies To

DateWizard Object

### Discussion

If you use the DateWizard object to build a time dimension that includes a week level, Transformer uses the setting of this property to define week categories. For example, if this property is set to trSunday, Transformer uses Sunday as the first day of the week.

This property uses the values of xtrWeekDay.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

### Type

Constant - xtrWeekDay

### Access

Write

### Examples

```
objDateWizard = objModel.DateWizardobjDateWizard.WeekStartDay  
= xtrWeekDay.trMonday
```

## WorkingDay Property

---

The WorkingDay property sets or returns whether a day is part of the working week.

### Syntax

*object* .**WorkingDay**(WeekDay)

### Applies To

DateDimension Object

DateWizard Object

### Discussion

Use the WorkingDay property to determine whether an individual day is part of a working week. In comparison, you can use the WorkingDays property to set several working days at once. You can also use the WorkingDays property and WorkingDay property in tandem to define a working week.

If the time dimension includes a week level, any days specified by the WorkingDay and WorkingDays properties set the days of the week for which Transformer generates categories.



This property uses the values of `xtrWeekDay`.

**Default:** False

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
WeekDay	Required. Specifies a value of <code>xtrWeekDay</code> . Type: Constant

## Type

Boolean

## Access

Read/Write (DateDimension)

Write (DateWizard)

## Examples

```
objDateWizard = objModel.DateWizardobjDateWizard.WorkingDays(127)
```

# WorkingDays Property

---

The `WorkingDays` property sets or returns which days are part of the working week.

## Syntax

*object*.**WorkingDays**(WeekDays)

## Applies To

[DateDimension Object](#)

[DateWizard Object](#)

## Discussion

Use the `WorkingDays` property to specify in one statement all the days that make up a working week. In comparison, you can use the `WorkingDay` property to determine whether an individual day is part of a working week.

If the time dimension includes a week level, any days specified by the `WorkingDays` and `WorkingDay` properties set the days of the week for which Transformer generates categories.

The value you specify for the `WeekDays` parameter can be values of `xtrWeekday` joined by plus signs (+), one of the numbers below, or the sum of two or more of those numbers.

- Sunday = 1
- Monday = 2
- Tuesday = 4
- Wednesday = 8
- Thursday = 16

- Friday = 32
- Saturday = 64

For example, to specify a working week of Monday to Friday, enter either 62 (the sum of 2, 4, 8, 16, and 32), or `trMonday+trTuesday+trWednesday+trThursday+trFriday`.

For all seven days, enter 127. You can use the `WorkingDays` property and `WorkingDay` property in tandem to define a working week.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

Parameter	Description
WeekDays	<p>Required. Specifies either values of <code>xtrWeekday</code> joined by plus signs (+), or a number that is the sum of one or more numeric values for days of the week.</p> <p><b>Note:</b> The way to add the different <code>xtrWeekday</code> joined by a plus sign depends on the implementation language used. For example, in C#, you must do the following:</p> <pre>(int) xtrWeekDay.trMonday + (int) xtrWeekDay.trWednesday + (int) xtrWeekDay.trFriday;</pre> <p>Type: Long</p>

## Type

Long

## Access

Read/Write (DateDimension)

Write (DateWizard)

## Examples

```
objDateWizard = objModel.DateWizardobjDateWizard.WorkingDays(127)
```

## YearStartDay Property

The `YearStartDay` property sets or returns the first day of a year.

## Syntax

*object*.**YearStartDay**

## Applies To

[DateDrillDown Object](#)

[DateWizard Object](#)

## Discussion

Use this property to specify the date on which a year begins when years do not begin on January 1, such as fiscal or lunar years.

Since lunar years contain 52 weeks (not 365 days), you must ensure that the YearStartDay and WeekStart properties coincide. For example, if YearStartDay is set to 20000104, then WeekStart must have a value of trMonday.

The date must be in YYYYMMDD format.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Long

## Access

Read/Write (DateDrillDown)

Write (DateWizard)

## Examples

```
objDateWizard = objModel.DateWizardobjDateDim  
= objDateWizard.CreateDateDimension()objDrillDown  
= objDateDim.DrillDowns(1)objDrillDown.YearStartDay = 19900101
```

# YearType Property

---

The YearType property sets how to calculate the year level of a time dimension.

## Syntax

*DateWizard* .**YearType**

## Applies To

DateWizard Object

## Discussion

Use this property to define a year as a calendar year or a lunar year.

The YearType property uses value of xtrSpecialFunction.

A COM exception is thrown in error situations. The message that is passed with the exception varies depending on the error situation.

## Type

Constant - xtrSpecialFunction

## Access

Write

## Examples

```
objDateWizard = objModel.DateWizardobjDateWizard.YearType  
= xtrSpecialFunction.trSpecialFunctionYear
```



---

# Chapter 6. Constants

A constant is a fixed value that you can use in an expression.

## xtrAllocationType Value List

---

Determines how measures are allocated in a model.

### Applies To

[AllocationType Property](#)

[SetAllocation Method](#)

### Discussion

Use the constants of this value list with the AllocationType property and the SetAllocation method of the DateDimension, Dimension, DateLevel, Level, Category, and SpecialCategory objects.

If a constant of xtrAllocationType is not assigned to a Category or SpecialCategory object, the object inherits the setting of the related Level or DateLevel object, which in turn inherits the default setting from the related Dimension or DateDimension object.

If the model contains multiple data sources, Transformer automatically allocates a measure from one data source to levels and categories associated with another data source. In such a case, a measure value is allocated as a constant throughout the dimension. This is the same as using trAllocationConstant. You can use trAllocationNA to reverse any default allocation.

You can have Transformer proportionally allocate measures, based on values in another measure, by setting the type to trAllocationByAnotherMeasure and by identifying the other measure in the SetAllocation method.

When you set the allocation type for a level, the new allocation type is applied from the categories in that level to all descendant categories.

Constant	Description
trAllocationByAnotherMeasure	Bases the allocation on the value of another object. You can proportionally allocate the measure value to descendant categories using values from another measure.
trAllocationConstant	Uses the measure value associated with the current category as a constant value for all descendant categories.
trAllocationDefault	Uses the allocation setting of the higher level object.
trAllocationNA	Suppresses allocation for a measure.

## xtrAssociationRole Value List

---

Determines what role an Association object takes.

## Applies To

[“AssociateWith Method” on page 114](#)

[“AssociationRole Property” on page 181](#)

[“DimensionAssociateWith Method” on page 124](#)

## Discussion

Use the constants of this value list with the AssociationRole property of the Association object, and the AssociateWith and DimensionAssociateWith methods.

An Association object defines the relationship between model objects and their underlying data source. Each association has a role. Often the role is as the source of data, as indicated by the constant trAssociationSource. Other roles are possible. The role that the association takes is limited by the nature of the data. Not all constants of xtrAssociationRole apply to all source items.

Constant	Description
trAssociationCategoryCode	Acts as the source for a unique category code.
trAssociationCountryCode	Acts as the source for a currency country or region code.
trAssociationCurrencyDate	Acts as the source for a currency date.
trAssociationDescription	Acts as the source for a description.
trAssociationDrillThrough	Acts as the source for drill-through information.
trAssociationLabel	Acts as the source for a label.
trAssociationOrderBy	Acts as the source for sort information.
trAssociationRate	Acts as the source for a currency rate.
trAssociationRoleNone (Default)	No association is set.
trAssociationShortName	Acts as the source for a short name.
trAssociationSource	Acts as the source for data.

## xtrAssociationType Value List

---

Determines the type of Association object.

## Applies To

[AssociationType Property](#)

## Discussion

Use the constants of this value list with the AssociationType property of the Association object to determine the type of Association object.

Constant	Description
trAssociationQuery	Represents an association with a data source.
trAssociationTypeNone (Default)	There is no association.

## xtrCharacterType Value List

Determines the character set used by an object.

### Applies To

[CharacterType Property](#)

[DataCharacterSet Property](#)

### Discussion

Use the constants of this value list with the Application object to specify which character set is used by Transformer, or with the FlatFileDataSource object to specify which character set is used by the data source.

Constant	Description
trCharAnsiDoubleByte	Specifies that the originating Windows application uses multibyte characters.
trCharAnsiSingleByte	Specifies that the originating Windows application does not use multibyte characters.
trCharDefault	Specifies that the default setting trCharAnsiDoubleByte is in effect.
trCharOEM	Specifies that the originating DOS or OS/2 application uses the standard IBM PC character set.
trUnicode	Specifies that the originating Windows application uses multibyte characters UTF-8.

## xtrCubeConsolidate Value List

Determines the current cube consolidation setting.

### Applies To

[Consolidate Property](#)

### Discussion

Use the constants of this value list with the Consolidate property of the Cube, ChildCube, and CubeGroup objects to determine if and how consolidation occurs. The consolidation process rolls up identical non-measure values into a single record and summarizes measure values.

For a cube group, the Consolidate property sets the default consolidation for each of the cubes in the group. Even when the constant trConsolidateNO is in effect, consolidation still occurs if the DuplicateRollup property is set to a value other than trDuplicateRollupNone.

Constant	Description
trConsolidateDefault	Consolidates source records if deemed useful, or if data is sorted for other reasons.
trConsolidateNO	Suppresses consolidation.
trConsolidatePresorted	Consolidates duplicate data, but does not sort the source file.
trConsolidateYES	Sorts the source file, and consolidates duplicate data.

## xtrCubeCreation Value List

---

Determines how cubes are selected for the cube creation process.

### Applies To

CubeCreation Property

### Discussion

Use the constants of this value list with the CubeCreation property of the Cube, CubeGroup, and ChildCube objects to determine which cubes to create when you use the CreateMDCFile or CreateMDCFiles methods. The constants determine which PowerCubes in the current model are selected for cube creation. For example, to optimize cube creation, use trCubeCreationOFF to limit the cubes created to just those that have changed.

The xtrCubeStatus value list also limits the cubes selected for creation.

Constant	Description
trCubeCreationDefault	Uses either the setting of the parent cube or ON for the highest -level cube.
trCubeCreationOFF	Doesn't select the cube if the data is unchanged since the last update.
trCubeCreationON	Selects the cube. Not valid for a ChildCube object.

## xtrCubeOptimize Value List

---

Determines optimization settings for cubes.

### Applies To

Optimize Property



## Discussion

Use the constants of this value list with the Optimize property of the Cube, CubeGroup, and ChildCube objects to increase the performance of PowerCubes and Transformer. The constants represent optimization settings.

Cube size, processing time in Transformer, and access time in PowerPlay are all affected by the optimization setting chosen. Select the best one for your model and data.

Constant	Description
trOptimizeAutoPartition	Specifies that the auto-partitioning feature is available. This is the default setting for models created in Transformer versions 6.0 and later.
trOptimizeCategories	Minimizes the number of categories in a cube. Only categories referenced in the data source or specifically designated to be included are added. There is an extra data pass for each cube to find the categories needed for that cube. This is the default for models created in Transformer versions before 6.0.
trOptimizeDataPasses	Optimizes the number of passes through the temporary working files during cube creation. All categories are included in the resulting cube, though categories not directly referenced or indirectly referenced via an ancestor will not be visible in PowerPlay.
trOptimizeDefault	Sets the default applicable to the version of Transformer the model was created with.
trOptimizeDirectCreate	Adds all categories in the model to the cube before the data sources are processed. It is best used with models that generate few new categories, and where all categories are expected to be added to the cube. Not applicable for individual cubes within a cube group.

## xtrCubeStatus Value List

Determines the last creation status for a cube.

### Applies To

Status Property

## Discussion

Use the constants of this value list with the Status property of the Cube, CubeGroup, and ChildCube objects to limit which cubes to include next time you execute the CreateMDCFile or CreateMDCFiles method. Generally, these constants filter cubes that are not functioning properly or select cubes for re-creation that failed in the past.

The xtrCubeCreation value list also filters cube lists.

Constant	Description
trCubeStatusBUSY	Shows that the cube is currently being updated.
trCubeStatusFAILED	Shows that the cube did not update correctly during the last cube update.
trCubeStatusINVALID	Shows that the cube is inconsistent with the model.
trCubeStatusMISSING	Shows that the cube existed but cannot be found or opened.
trCubeStatusNEW	Shows that the cube is defined in the model but has not been created.
trCubeStatusOK	Shows that the cube exists and has no errors.
trCubeStatusWARNINGS	Shows that the cube exists but warnings were issued during creation or during the last update.

## xtrCurrencyTableType Value List

Determines whether a currency table applies to a base currency or the euro.

### Applies To

[CurrencyTableType Property](#)

### Discussion

Use the constants of this value list with the CurrencyTableType property of the CurrencyTable object to determine the type of currency table used.

Constant	Description
trCurrencyTableBase	Specifies a table that includes a base currency rate against which other currencies are converted.
trCurrencyTableEuro	Specifies a table that includes a euro base currency rate against which other currencies are converted.
trCurrencyTableOther	Specifies that the currency table in use is other than the base or euro.

## xtrDataClass Value List

Determines the data classification for a column.

### Applies To

[DataClass Property](#)

## Discussion

Use the constants of this value list with the `DataClass` property of the `Column` object to determine the type of data value assigned to their data source.

Constant	Description
<code>trDataClassArrayMember</code>	Specifies that the data class contains data from individual elements in an array.
<code>trDataClassDate</code>	Specifies that the data class contains date values in one of the formats defined by the <code>xtrDateFormat</code> value list.
<code>trDataClassDefault</code>	The data class is unspecified in the model. If a data type definition is available in the data source, Transformer uses it.
<code>trDataClassDescription</code>	Specifies that the data class contains text, such as labels, or alphanumeric values, such as codes.
<code>trDataClassIgnore</code>	Specifies that the content of the source is ignored and not processed.
<code>trDataClassQuantity</code>	Specifies that the data class contains numbers that represent quantities or counts used as performance indicators.

## xtrDateCategoriesGeneration Value List

Determines which categories are generated for a date level.

### Applies To

[GenerateCategories Property](#)

## Discussion

Use the constants of this value list with the `GenerateDateCategories` property of the `DateLevel` object.

Constant	Description
<code>trGenerateDatesAll</code>	Specifies that all categories are generated.
<code>trGenerateDatesDefault</code>	Specifies that the default for the level is used: <code>trGenerateDatesAll</code> is assumed for all but the highest level in each drill-down path; <code>trGenerateDatesNone</code> is assumed for the highest level.
<code>trGenerateDatesNeed</code>	Specifies that only categories required by the cube are generated from the data source.
<code>trGenerateDatesNone</code>	Specifies that no categories are generated.

## xtrDateFormat Value List

Determines the date input format for a column.

### Applies To

[DateInputFormat Property](#)

### Discussion

Use the constants of this value list with the DateInputFormat property of the Column object to determine the default date format used by Transformer when the date format is not predefined in the data source. The Month portion of dates with a month component can be a two digit number or three letter abbreviation, depending on the data source.

Constant	Description
trPredefined (Default)	Specifies that the date format is predefined in the data source.
trD	Specifies that the date is in day format: DD.
trDMY	Specifies that the date is in day-month-year format: DDMMYYYY
trM	Specifies that the date is in month format: MM
trMDY	Specifies that the date is in month-day-year format: MMDDYYYY
trMY	Specifies that the date is in month-year format: MMYYYY
trQ	Specifies that the date is a single digit from 1 to 4 giving a quarter: Q
trW	Specifies that the date is in week format.
trY	Specifies that the date is in year format: YYYY
trYM	Specifies that the date is in year-month format: YYYYMM
trYMD	Specifies that the date is in year-month-day format: YYYYMMDD

## xtrDateLevel Value List

Determines the degree of detail for date levels.

### Applies To

[DateDegreeofDetail Property](#)

## Discussion

Use the constants of this value list with the DateDegreeofDetail property of the Column object.

The constants define the lowest date level (degree of detail) where measures can be reported. When a level is set, reporting of measures in the dimension is restricted to that level of detail.

Constant	Description
trDateLevelDay	Specifies that the degree of detail is by day.
trDateLevelMonth	Specifies that the degree of detail is by month.
trDateLevelQuarter	Specifies that the degree of detail is by quarter.
trDateLevelUnspecified	Specifies that the degree of detail is unspecified. Transformer sets the degree of detail based on other attributes of the column.
trDateLevelWeek	Specifies that the degree of detail is by week.
trDateLevelYear	Specifies that the degree of detail is by year.

## xtrDeployType Value List

Sets the type of deployment.

### Applies To

[SetDeployType Method](#)

## Discussion

Use the constants of this value list with the SetDeployType Method of the Cube object to set the deployment type. These types correspond to the PowerCube deployment strategy. If you use trDeployType\_SWAPSINGLE, a failure when deploying to a location does not impact the deployment to the other locations. This action corresponds to the user-interface command **Copy to available locations, then activate**. If you use trDeployType\_SWAPTOGETHER, any deployment failure to a location causes a rollback on the other locations, even if these deployments were successful. This action corresponds to the user-interface command **Copy to all locations, then activate**.

Constant	Description
trDeployType_DEFAULT	Specifies the default deployment type.
trDeployType_NONE	Specifies no deployment.
trDeployType_NULL	Specifies the null deployment type.
trDeployType_SWAPSINGLE	Specifies to copy to available locations, then activate.
trDeployType_SWAPTOGETHER	Specifies to copy to all locations, then activate.

## xtrDuplicateRollup Value List

Determines how Transformer summarizes duplicate records retrieved from the data source.

### Applies To

[DuplicateRollup Property](#)

### Discussion

Use the constants of this value list with the DuplicateRollup property of the Measure object to set rollup options.

Constant	Description
trDuplicateRollupAverage	Specifies that the rollup uses the average of all values for the measure found in the duplicate records.  If the RegularRollup property is set to trRollupAverage, the duplicate rollup is performed first.
trDuplicateRollupFirst	Specifies that the rollup uses the first value of all values for the measure found in the duplicate records, using the order of the records in the data source.
trDuplicateRollupLast	Specifies that the rollup uses the last value of all values for the measure found in the duplicate records, using the order of the records in the data source.
trDuplicateRollupMaximum	Specifies that the rollup uses the largest of all values for the measure found in the duplicate records.
trDuplicateRollupMinimum	Specifies that the rollup uses the smallest of all values for the measure found in the duplicate records.
trDuplicateRollupNone	Specifies that no duplicate rollup is performed. Uses the constant of xtrRollup specified for the RegularRollup property instead.
trDuplicateRollupSum	Specifies that the rollup sums the measure values found in the duplicate records.

## xtrGenerateOptions Value List

Determines the generate option for categories in a time dimension.

### Applies To

[GenerateTimePeriod Property](#)

## Discussion

Use the constants of this value list with the `GenerateTimePeriod` property of the `DateDimension` object to determine which categories to generate in the dimension.

Constant	Description
<code>trGenerateAll</code>	Specifies that all categories are generated.
<code>trGenerateDefault</code>	Specifies that the default for the level is used: <code>trGenerateAll</code> is assumed for all but the highest level in each drill-down path; and, <code>trGenerateNone</code> is assumed for the highest level.
<code>trGenerateNeed</code>	Specifies that only categories that are needed by the cube are generated from the data source.
<code>trGenerateNone</code>	Specifies that no categories are generated.

## xtrInclusion Value List

Determines if a category is included in a model or a cube.

### Applies To

[DrillInclusion Property](#)

[Inclusion Property](#)

## Discussion

Use the constants of this value list with the `DrillInclusion` property of the `DrillDown` and `DateDrillDown` objects, and the `Inclusion` property of the `Category`, `SpecialCategory`, `DateLevel`, and `Level` objects. The constants determine under what circumstances categories are included in a model or cube.

If a constant of `xtrInclusion` is not assigned to a `Category` or `SpecialCategory`, the category inherits the setting of the related `Level` or `DateLevel` object.

When it is necessary to complete a cube, Transformer includes ancestors of a category, regardless of the inclusion setting.

Constant	Description
<code>trInclusionDefault</code>	Level: <code>trInclusionWhenNeeded</code> . Category: uses the setting of the level in which the category resides.
<code>trInclusionExclude</code>	Excludes the category, the descendants, and all related data from the cube.  To use this constant: the category cannot be the share category of another category or link to a special category.
<code>trInclusionGenerate</code>	Always includes the category.

Constant	Description
trInclusionSuppress	<p>Level: Retains categories in the model but excludes from cubes those categories with a blank source value.</p> <p>Categories: Excludes the category from the cube but includes all the descendants and their values.</p> <p>To use this constant: the category cannot be the share category of another category or link to a special category.</p>
trInclusionWhenNeeded	Includes the category in the model but omits it from the cube when the category has a zero value.

## xtrMeasureType Value List

Determines the type and use of a Measure object.

### Applies To

[MeasureType Property](#)

### Discussion

Use the constants of this value list with the MeasureType property of the Measure object to determine what type of measure was created. The types of Measure objects include: regular, calculated, and category count.

Constant	Description
trCalculatedMeasure	Specifies that the measure is a measure that derives values from the ExpressionText property.
trCountMeasure	Specifies that the measure is a measure that derives values from the CategoryCountLevel property.
trRegularMeasure (Default)	Specifies that the measure is a regular measure that derives values from an association.

## xtrMissingValue Value List

Determines how PowerPlay displays missing values.

### Applies To

[MissingValue Property](#)

### Discussion

Use the constants of this value list with the MissingValue property of the Measure object to determine what should appear in PowerPlay when the numeric value for a measure is missing in the data source.



Constant	Description
trMissingValueNA	Specifies that the abbreviation 'na' is used. This way missing items aren't misinterpreted as zero values.
trMissingValueZERO (Default)	Specifies that zeros are used when values are missing.

## xtrObjectType Value List

Determines the object type.

### Applies To

[Add Method \(Objects\)](#)

### Discussion

Use the constants of this value list with the Type property of Transformer objects, and the Type parameter of the Add and Item methods. For example, you can use constants of xtrObjectType with the Add method to add new objects to a collection.

Constant	Description
trApplication	Specifies that the object is the Transformer application.
trAssociation	Specifies that the object is an Association.
trCalculationDefinition	Specifies that the object is a CalculationDefinition.
trCategory	Specifies that the object is a Category.
trCategorySet	Specifies that the object is a CategorySet.
trChildCube	Specifies that the object is a ChildCube.
trColumn	Specifies that the object is a Column.
trCrossTabDataSource	Specifies that the object is a CrossTabDataSource.
trCube	Specifies that the object is a Cube.
trCubeGroup	Specifies that the object is a CubeGroup.
trCurrencyRate	Specifies that the object is a CurrencyRate.
trCurrencyRecord	Specifies that the object is a CurrencyRecord.
trCurrencyTable	Specifies that the object is a CurrencyTable.
trCustomView	Specifies that the object is a CustomView.

<b>Constant</b>	<b>Description</b>
trDateDimension	Specifies that the object is a DateDimension.
trDateDrillDown	Specifies that the object is a DateDrillDown.
trDateLevel	Specifies that the object is a DateLevel.
trDateWizard	Specifies that the object is a DateWizard.
trDbDataSource	Specifies that the object is a DbDataSource.
trDimension	Specifies that the object is a Dimension.
trDrillDown	Specifies that the object is a DrillDown.
trDrillThroughTarget	Specifies that the object is a DrillThroughTarget.
trFlatFileDataSource	Specifies that the object is a FlatFileDataSource.
trFilter	Specifies that the object is a filter.
trIqdDataSource	Specifies that the object is a IqdDataSource.
trLevel	Specifies that the object is a Level.
trMeasure	Specifies that the object is a Measure.
trModel	Specifies that the object is a Model.
trName	Specifies that the object is a Name.
trNamespace	Specifies that the object is a Namespace.
trNoType	Specifies that the object type is unidentified.
trPackage	Specifies that the object is a Package.
trPrompt	Specifies that the object is a Prompt.
trReport	Specifies that the object is a Report.
trSecurityObject	Specifies that the object is a SecurityObject.
trSignon	Specifies that the object is a Signon.
trSpecialCategory	Specifies that the object is a SpecialCategory.
trSuspendedModel	Specifies that the object is a SuspendedModel.
trView	Specifies that the object is a View.

## xtrOrigin Value List

Determines the origin of a column.

### Applies To

[Origin Property](#)

### Discussion

Use the constants of this value list with the Origin property of the Column object to determine the origin of the column, such as if it is derived from a data source or a calculation.

Constant	Description
trOriginCalculated	Specifies that the column is based on a calculation.
trOriginGenerated	Specifies that the column originates from generated categories.
trOriginManual	Specifies that the column originated through manual creation in Transformer.
trOriginSource	Specifies that the column originates from an IQD data source.
trOriginUnspecified	Specifies that the origin is unknown.

## xtrPowerCubeGeneration Value List

Determines when a data source is referenced by a model.

### Applies To

[GeneratePowerCube Property](#)

### Discussion

Use the constants of this value list with the GeneratePowerCube property of the CrossTabDataSource, DataSource, DbDataSource, FlatFileDataSource, IqdDataSource objects and Query objects.

Use the constants of xtrPowerCubeGeneration to determine if a data source is referenced during category generation, cube creation, both category generation and cube creation, or not at all. Use constants of this value list in models with multiple data sources to avoid unnecessary processing and to optimize model efficiency.

Constant	Description
trGenerationCreatePowerCubes	<p>Specifies that Transformer reads the data source to generate categories, create cubes, or both as required even for purely structural data sources.</p> <p>Use this option in a design and development environment or to build cubes with measures based on record counts rather than on the measure values in the records themselves.</p>

Constant	Description
trGenerationDefault	Specifies that Transformer reads any columns in the data source that relate to levels in the model to see if they are associated with measures. If so, the cube is created or updated using the measure values. If the data source is purely structural, Transformer only generates categories.  Use this option in a production environment, for transactional data sources and structural data sources that contain non-static data.
trGenerationGenerateCategories	Specifies that Transformer reads the data source only for structural information and to generate categories.
trGenerationNoCreatePowerCubes	Specifies that Transformer does not access the data source when it creates categories and cubes.

## xtrPreferences Value List

Determines date format, error logging, and sorting preferences for an application.

### Applies To

[DefaultDateFormat Property](#)

[LogErrorLevel Property](#)

[SortComparisonRule Property](#)

### Discussion

Use the constants of this value list with the DefaultDateFormat, LogErrorLevel, and SortComparisonRule properties of the Application object.

Not all the constants of this value list apply to all three properties.

Constant	Description
trDateFormatFromControlPanel	Specifies that the date format is defined in the regional settings of your Windows control panel. Applies to the DefaultDateFormat property.
trDateFormatPredefined	Specifies that the date format is defined in the data source used by the model. Applies to the DefaultDateFormat property.
trLogErrorsAndAbove	Specifies whether to log severe and error messages to the log file. Errors occur at the transaction level and cause cubes to be marked as invalid. Applies to the LogErrorLevel property.
trLogInformationAndAbove	Specifies whether to log all messages to the log file. Applies to the LogErrorLevel property.

Constant	Description
trLogSevereErrors	Specifies whether to log only severe error messages to the log file. Severe errors are caused by a system administration limit on CPU, disk, file, or transaction resources, or by a corrupted model, cube, or temporary file. Applies to the LogErrorLevel property.
trLogWarningsAndAbove	Specifies whether to log, severe, error, and warning messages to the log file. Warnings don't impede processing, but indicate a potential problem. Applies to the LogErrorLevel property.
trSortFromControlPanel	Specifies the application uses the comparison rule associated with the regional settings of your Windows control panel. Applies to the SortComparisonRule property.
trSortIgnoreControlPanel	Specifies the application uses the Transformer internal comparison rule, a byte-by-byte comparison of the binary representation of each string. Applies to the SortComparisonRule property.

## xtrPromptValueType Value List

Determines the type of prompt.

### Applies To

[PromptValueType Property](#)

### Discussion

Use the constants of this value list with the PromptValueType property of the Prompt object to set the type of prompt.

Constant	Description
trSingleValuePrompt	Specifies that the prompt is a single-value prompt.
trMultiValuePrompt	Specifies that the prompt is a multi-value prompt.
trBoundedRangePrompt	Specifies that the prompt is a bounded prompt.
trUnboundedRangePrompt	Specifies that the prompt is an unbounded prompt.

## xtrRollup Value List

Determines how Transformer rolls up measure values from lower levels to higher category levels.

### Applies To

[RegularRollup Property](#)

## Discussion

Use the constants of this value list with the RegularRollup property of the Measure object to set rollup options.

Constant	Description
trRollupAny	Determines whether category values exist.  In PowerPlay, shows 1, if any records for a descendant category contain values; or 0 if no records exist for this measure or all records that do exist have null values for this measure.
trRollupAverage	Generates an average of the values of all records of descendant categories of the current category.
trRollupCount	Generates the number of records that contain non-null values in all descendant categories of the current category.
trRollupCount All	Generates the number of records, including those containing null values for this measure, for all descendant categories of the current category.
trRollupDefault	Specifies that trRollupSum is used.
trRollupExternal	Generates source values that have been directly assigned to specific data records.
trRollupMaximum	Generates the maximum data value among all descendant categories of the current category.
trRollupMinimum	Generates the minimum data value among all descendant categories of the current category.
trRollupSum	Generates the sum of the values of all descendant categories of the current category.

## xtrRollupTiming Value List

Determines when calculations are performed for calculated measures.

### Applies To

[RollupTiming Property](#)

## Discussion

Use the constants of this value list with the RollupTiming property of the Measure object. The constants determine when rollups for measures occur in relation to calculations involving those measures. The MeasureType property of the Measure object must be set to trCalculatedMeasure.

Constant	Description
trTimingAfterRollup	Specifies that calculated measures are based on the rolled up values. The sequence is: 1 Regular rollup takes place on all measures. 2 Time state rollup takes place. 3 Allocated measures are calculated. 4 Calculations are performed on calculated measures.
trTimingBeforeRollup	Specifies that calculated measures are based on original source values. The sequence is: 1 Calculations are performed as the cube is created. 2 Regular rollup takes place on all measures. 3 Time state rollup takes place. 4 Allocated measures are calculated.
trTimingDefault	Specifies that trTimingAfterRollup is used.

## xtrSecurityType Value List

Determines the type of security object.

### Applies To

[Type Property](#)

### Discussion

Use the constants of this value list with the Type property of the SecurityObject object to set the type of security object.

Constant	Description
trSecurityType_User	Specifies that the security object is a user.
trSecurityType_Group	Specifies that the security object is a group.
trSecurityType_Role	Specifies that the security object is a role.

## xtrSourceType Value List

Determines the type of file represented by a data source.

### Applies To

[SourceType Property](#)

## Discussion

Use the constants of this value list with the `SourceType` property of the `CrossTabDataSource`, `DataSource`, `DbDataSource`, `FlatFileDataSource`, and `IqdDataSource` objects to determine the external source of the data and data definitions.

Constant	Description
<code>trAccess</code>	Specifies that the source is an Access database file.
<code>trAccessQuery</code>	Specifies that the data source table is described in an Access Query file(.mdb). Input values are retrieved from an ODBC-supported database by executing an SQL query that is stored in the Access Query file.
<code>trClipper</code>	Specifies that the source is a Clipper file.
<code>trDBase</code>	Specifies that the source is a dBase database file.
<code>trDictionary</code>	Specifies that the source is a data dictionary.
<code>trExcelCrossTab</code>	Specifies that the source is an Excel spreadsheet crosstab file.
<code>trExcelDatabase</code>	Specifies that the source is an Excel spreadsheet database file.
<code>trFixedAscii</code>	Specifies that the source is a text file with one record per line. Each field occupies a specified number of bytes. Each record ends with a text line delimiter.
<code>trFixedAsciiNoCRLF</code>	Specifies that the source is a text file with one record per line. Each field occupies a specified number of bytes. Each record end is not marked by a text line delimiter.
<code>trFlatFile</code>	Specifies that the source is a text file with one record per line. The fields are delimited by the character specified by the <code>FieldSeparator</code> property.
<code>trFlatFileColumnNames</code>	Specifies that the source is a text file with one record per line, where the first line represents column names. The fields (column values) are delimited by the character specified by the <code>FieldSeparator</code> property.
<code>trFoxPro</code>	Specifies that the source is a FoxPro database file.
<code>trLotus123CrossTab</code>	Specifies that the source is a Lotus 1-2-3 spreadsheet crosstab file.
<code>trLotus123Database</code>	Specifies that the source is a Lotus 1-2-3 spreadsheet database file.



Constant	Description
trParadox	Specifies that the source is a Paradox database file.
trPowerHousePortable	Specifies that the source is a PowerHouse portable subfile.
trQuery	<p>Specifies that the source is an Impromptu query definition file(.iqd). Input values are retrieved from a supported Impromptu database by executing an SQL query defined in Impromptu.</p> <p>The contents of the Impromptu query definition file are stored in the model and the embedded contents are refreshed whenever you generate categories or create cubes.</p>

## xtrSpecialFunction Value List

Determines the type of calendar used for dates.

### Applies To

[DateFunction Property](#)

[MonthType Property](#)

[QuarterType Property](#)

[YearType Property](#)

### Discussion

Use the constants of this value list with the DateFunction property of the DateLevel object, and the MonthType, QuarterType, and YearType properties of the DateWizard object. The constants determine if a date level uses a standard or lunar calendar, and also determine the format the date uses.

Because lunar calendar segments are shorter than standard calendar segments, lunar periods leave unassigned days at the end of a year. You can adjust these by using the WeekAdd property and constants of the xtrWeekAdd value list.

Constant	Description
trSpecialFunctionDay	Specifies that this is a standard day in the form YYYY/MMM/DD or YY/MMM/DD.
trSpecialFunctionLunarMonth	Specifies that the month is based on a Lunar month, which contains exactly 4 weeks.
trSpecialFunctionLunarMonth445	Specifies that the month is based on repeating 3 lunar month sequences containing 4 weeks, 4 weeks, and 5 weeks.
trSpecialFunctionLunarMonth454	Specifies that the month is based on repeating 3 lunar month sequences containing 4 weeks, 5 weeks, and 4 weeks.

Constant	Description
trSpecialFunctionLunarMonth544	Specifies that the month is based on repeating 3 lunar month sequences containing 5 weeks, 4 weeks, and 4 weeks.
trSpecialFunctionLunarQuarter	Specifies that the quarter is based on Lunar quarters, which contain exactly 13 weeks.
trSpecialFunctionLunarYear	Specifies that the year is based on a Lunar calendar, which contains exactly 52 weeks.
trSpecialFunctionMonth	Specifies that the month is based on the standard calendar, in the format YYYY/MMM or YY/MMM.
trSpecialFunctionNone	Specifies that the date is based on the level's source column.
trSpecialFunctionNull	Specifies that there is nothing set for the date.
trSpecialFunctionQuarter	Specifies that the quarter is based on the standard calendar in the form YYYY Q, or YY Q, where Q is the Quarter number (1, 2, 3, or 4).
trSpecialFunctionWeek	Specifies that the week is a standard calendar week, in the format YYYY/MMM/DD or YY/MMM/DD.
trSpecialFunctionYear	Specifies that the year is based on the standard calendar in the format YYYY or YY.

## xtrStorage Value List

Determines how numeric drill-down paths in levels are sorted and how measure data is stored.

### Applies To

[OrderByStorageType Property](#)

[StorageType Property](#)

### Discussion

Use the constants of this value list with the StorageType property of Measure object to determine how Transformer stores measure values within the work files used to build cubes.

You can also use one of these constants with the OrderByStorageType property to define a category sort when the sort is based on a column with a data class of type numeric. For example, you base a product sort on quantity sold, and the measure named 'Quantity', which is associated with the Quantity column, has a storage type of 16-bit integer.

Constant	Description
trStorageBigFloat	Provides 64-bit floating point storage for numbers with a range of 1.7E +/- 308.

Constant	Description
trStorageBigInt	Provides 32-bit integer storage for numbers between -2,147,483,648 and +2,147,483,647.
trStorageDefault	Provides a default storage depending on the data type. For numeric data, the default is trStorageBigInt.
trStorageSmallInt	Provides 16-bit integer storage for numbers between -32,768 and +32,767.

## xtrTimeAggregate Value List

Determines the type of relative time period.

### Applies To

[Aggregate Property](#)

### Discussion

Use the constants of this value list with the Aggregate property of the SpecialCategory object to customize special categories for relative time. Such categories track measures for specific periods of time relative to the current time. These constants specify the type of relative time period on which to base the special category.

A custom relative time SpecialCategory object also requires settings for other properties, such as ContextOffset and TargetOffset, to supply a time range and context in which the aggregate operates.

Constant	Description
trAggregateNone	Specifies that no aggregate time period is defined.
trAggregateRunning	Represents a number of time periods ending at a specific N-period indicated by properties such as ContextOffset and TargetOffset.
trAggregateRunningGrouped	Does the same as trAggregateRunning except that a range of N-period categories are created.
trAggregateSingle	Represents a single time period defined by properties such as ContextOffset and TargetOffset.
trAggregateToDate	Represents a sequential set of periods starting at the beginning of a period and ending at another specific period. The time periods are defined by properties such as ContextOffset and TargetOffset.
trAggregateToDateGrouped	Does the same as trAggregateToDate, except that a range of to-date categories are created.

## xtrTimeArrayType Value List

Determines date array settings.

## Applies To

[TimeArrayType Property](#)

## Discussion

Use the constants of this value list with the TimeArrayType property of the Column object to determine whether a date array contains quarterly or monthly values.

Constant	Description
trTimeArrayMonth	Specifies that it is the first of 12 adjacent columns that together make up an array of months.
trTimeArrayNA	Specifies that the object is not associated with a time array.
trTimeArrayQuarter	Specifies that it is the first of four adjacent columns that together make up an array of quarters.

## xtrTimeRollup Value List

Determines how Transformer rolls up measures at specific time periods.

## Applies To

[TimeStateRollup Property](#)

## Discussion

Use the constants of this value list with the TimeStateRollup property of the Measure object to set rollup options for date values.

Constant	Description
trTimeRollupAverage	Uses the average of the measure values from all categories in the time period being examined.
trTimeRollupCurrentPeriod	<p>Uses the measure value from the category that is designated the 'current period' in the time dimension. If the time period being examined does not include the current period, the result is identical to Last Period.</p> <p>For example, a time dimension contains years, quarters, and months, where quarter 1 starts in January. The current period is set to April 2000.</p> <p>At the year level, PowerPlay shows the measure value for April 2000. At the quarter level, it shows the measure value for April in quarter 2 (because April is the current period), but shows the value of the last active month in every other quarter (that is, March for quarter 1; September for quarter 3; December for quarter 4.</p>

Constant	Description
trTimeRollupEarlyPeriod	<p>Uses the measure value from the first subordinate period in the time period being examined.</p> <p>For example, if a time dimension contains years, quarters, and months, and you are examining data at the quarter level, PowerPlay shows for each quarter the measure value from the first month of each quarter.</p> <p>When you examine data at the Year level, PowerPlay shows the first value from the first month in the first quarter of each year.</p>
trTimeRollupLatePeriod	<p>Uses the measure value from the last subordinate period in the time period being examined.</p> <p>For example, if a time dimension contains years, quarters, and months, and you are examining data at the quarter level, PowerPlay shows for each quarter the measure value from the last month of each quarter.</p> <p>When you examine data at the Year level, PowerPlay shows the value from the last month in the last quarter of each year.</p>
trTimeRollupMaximum	Uses the largest measure value from all categories in the time period being examined.
trTimeRollupMinimum	Uses the smallest measure value from all categories in the time period being examined.
trTimeRollupNone	<p>Specifies that no time state rollup is performed.</p> <p>Uses the value set for the RegularRollup property instead.</p>

## xtrTimeType Value List

Determines the degree of detail setting for dates.

### Applies To

[EnableTimePeriod Property](#)

[GenerateTimePeriod Property](#)

### Discussion

Use the constants of this value list with the EnableTimePeriod property of the DateWizard object and GenerateTimePeriod property of the DateDimension object. These constants indicate the level at which Transformer allows reporting to occur from date-related columns in a data source. For example, a date column with values for Year, Quarter, and Month can be restricted to reporting only to the quarter level. You cannot specify a reporting period that is lower than the lowest level set by the DateInputFormat property.

Constant	Description
trTimeTypeDay	Specifies that the lowest reporting period is by day.
trTimeTypeMonth	Specifies that the lowest reporting period is by month.
trTimeTypeQuarter	Specifies that the lowest reporting period is by quarter.
trTimeTypeWeek	Specifies that the lowest reporting period is by week.
trTimeTypeYear	Specifies that the lowest reporting period is by year.

## xtrViewStatus Value List

Determines the categories to include in a view.

### Applies To

[GetViewStatus Method](#)

[SetViewStatus Method](#)

### Discussion

Use the constants of this value list with the SetViewStatus method of the Category and SpecialCategory objects and the GetViewStatus method of the View object. These constants describe which categories are included when a PowerCube is created from the view.

Constant	Description
trViewStatusApexAncestor	Contains only the apex category and the immediate descendants.
trViewStatusClear	Specifies that no restrictions are set.
trViewStatusCloakMom	Specifies that the category and all the descendants are omitted but their values are retained for rollup into higher-level categories.
trViewStatusClipMom	Specifies that the category and all the descendants are excluded.
trViewStatusInvisibleKid	Specifies that the category is excluded from a Cloaked or Summary ancestor.
trViewStatusRemoveKid	Specifies that the category is excluded from an Excluded ancestor.
trViewStatusSummaryMom	Includes a category with summarized data for all the descendants.

Constant	Description
trViewStatusSuppressed	Specifies that the category is not included, but the parent and child categories are.

## xtrViewType Value List

---

Determines the type of view.

### Applies To

[DimensionInclude Property](#)

[DimensionViewType Property](#)

[ViewType Property](#)

### Discussion

Use the constants of this value list with the DimensionViewType property of the Cube and CubeGroup object, the ViewType property of the View object, and the DimensionInclude property.

When you create a dimension, the related cube contains a view of the relevant categories in that dimension. The view can contain all, some, or none of the categories.

Constant	Description
trViewTypeAllCategories	Specifies that the view contains the dimension and all the categories.
trViewTypeCustom	Specifies that the view contains only the categories selected for the view.
trViewTypeNone	Specifies the object is not associated with a view.
trViewTypeOmitDimension	Specifies that the entire dimension is omitted.

## xtrWeekAdd Value List

---

Determines how many days are added to a lunar year to equal a calendar year.

### Applies To

[WeekAdd Property](#)

### Discussion

Use the constants of this value list with the WeekAdd property of the DateWizard or DateDrillDown object to synchronize lunar years with calendar years.

A lunar year contains 364 days, which is either one or two days less than a calendar or leap year, respectively. In Transformer, you can add an extra week of four to seven days to the lunar year. Transformer adds this extra week to the end of the last month or quarter of the year.

Constant	Description
trWeekAddNone (Default.)	Specifies that extra weeks or partial weeks are never added.
trWeekAddDays7	Adds 7 days.
trWeekAddDays6	Adds 6 days.
trWeekAddDays5	Adds 5 days.
trWeekAddDays4	Adds 4 days.

## xtrWeekDay Value List

Determines whether a day is part of the working week.

### Applies To

[WeekStart Property](#)

[WorkingDay Property](#)

[WorkingDays Property](#)

### Discussion

Use the constants of this value list with the WorkingDay and WorkingDays properties of the DateWizard or DateDimension object, and the WeekStart property of the DateDrillDown object.

For a time dimension, these constants indicate if a day is part of a working week and are useful for creating non-standard work weeks. The constants also specify the days of the week for which Transformer generates categories in the dimension.

For a date drill-down path, these constants determine which day marks the starting day of week categories within higher-level time periods.

Constant	Description
trSunday	Specifies that Sunday is part of the working week.
trMonday	Specifies that Monday is part of the working week.
trTuesday	Specifies that Tuesday is part of the working week.
trWednesday	Specifies that Wednesday is part of the working week.
trThursday	Specifies that Thursday is part of the working week.
trFriday	Specifies that Friday is part of the working week.
trSaturday	Specifies that Saturday is part of the working week.



## xtrWeekSpan Value List

---

Determines how to define a week that spans two years.

### Applies To

WeekSpan Property

### Discussion

Use the constants of this value list with the WeekSpan property of the DateWizard or DateDrillDown object to determine where to fit the last week of a year.

If a time dimension contains a week level in a calendar year, the last week often will not end on the same day as the year ends. (A lunar year always ends at the end of the week.) Transformer can shift it or split it up based on the setting of the WeekSpan property.

Constant	Description
trWeekSpanFirstPeriod	Shifts the week into the year in which it begins.
trWeekSpanLargerPeriod	Shifts the week into the year in which the greatest number of days in that week fall.
trWeekSpanLastPeriod	Shifts the week into the year in which it ends.
trWeekSpanNone (Default.)	Specifies that a week spanning a year is ignored.
trWeekSpanSplitAll	Splits the week into two distinct partial weeks and applies one part to each year the week spanned.
trWeekSpanSplitMost	Splits the spanning week between the two years, provided that each split portion contains at least two days; otherwise, places the week into the year in which the greatest number of days in that week fall.



---

## Chapter 7. UI Equivalents

IBM Cognos Transformer OLE automation uses a program language interface to provide an alternative to the Transformer user interface. This document includes information about the UI equivalents of most Transformer OLE methods and properties, to help you get started. Be advised, however, that elements added in Version 7.3 and subsequent releases do not have their UI equivalents listed.

OLE automation presents a Transformer model as a set of collections and objects that are modified by properties and acted upon by methods. Use automation to create and manage dimensions, levels, data sources, measures, categories, drill-down paths, and other model objects, and to create PowerCubes.

When you create a model in OLE, you must create your objects and assign values to them in a hierarchical sequence. For example, you cannot create a level until you have created the dimension in which the level resides. For more information about object hierarchies, see [Transformer Object Hierarchy Map](#).

### Collections

---

The following table shows a summary of user interface equivalents for collections.

Collections	User interface equivalent
<a href="#">Associations Collection</a>	Dimension property sheet, General tab, External Rollup Column box (when the Contains Externally Rolled Up Measure Values check box is selected in the General tab of the Data Source property sheet)  Level property sheet, Source tab  Measure property sheet, Type tab (when Type option is set to Column)  Currency Table property sheet, Base Table Columns box (when Use An External Currency Data Source check box is selected)
<a href="#">CalculationDefinitions Collection</a>	Dimension property sheet, Calculation tab
<a href="#">Categories Collection</a>	Dimension diagram, category viewer
<a href="#">CategorySets Collection</a>	Dimension Category Calculation dialog box (accessed through Dimension property sheet, Calculation tab)
<a href="#">ChildCubes Collection</a>	PowerCubes list
<a href="#">Columns Collection</a>	Data Sources list
<a href="#">Cubes Collection</a>	PowerCubes list
<a href="#">CurrencyRates Collection</a>	Currency Table property sheet
<a href="#">CurrencyRecords Collection</a>	Currency Table property sheet
<a href="#">CurrencyTables Collection</a>	Currency Table property sheet
<a href="#">DataSources Collection</a>	Data Sources list

<b>Collections</b>	<b>User interface equivalent</b>
<a href="#">DimensionLevels Collection</a>	Dimension map
<a href="#">Dimensions Collection</a>	Dimension map
<a href="#">DrillDowns Collection</a>	Dimension diagram, category viewer
<a href="#">LevelDrillDowns Collection</a>	Dimension map
<a href="#">Levels Collection</a>	Dimension map
<a href="#">Measures Collection</a>	Measures list
<a href="#">Names Collection</a>	Any Transformer window which contains objects
<a href="#">Signons Collection</a>	View menu, Signons command, Signons list
<a href="#">SuspendedModels Collection</a>	File menu, View Suspended Models command, Select Suspended Model box
<a href="#">Views Collection</a>	Dimension diagram, Dimension tab

## Objects

The following table shows a summary of user interface equivalents for objects.

<b>Objects</b>	<b>UI equivalent</b>
Application	IBM Cognos Transformer window
Association	Dimension property sheet, General tab, External Rollup Column box (when the Contains Externally Rolled Up Measure Values check box is selected in the General tab of the Data Source property sheet)  Level property sheet, Source tab  Currency Table property sheet (when Use an External Currency Data Source check box is selected)  Measure property sheet, Type tab (when Type is set to Column)
CalculationDefinition	Dimension property sheet, Calculation tab  Measure property sheet, Type tab, Calculation button, Measure Calculation dialog box
Category	Category viewer
CategorySet	Dimension property sheet, Calculation tab, Add or Modify button, Calculation button, Dimension Category Calculation dialog box, Available Components pane

<b>Objects</b>	<b>UI equivalent</b>
ChildCube	PowerCubes list (cube group)
Column	Data Sources list (when the data source object is expanded)
CrossTabDataSource	Data Sources list
Cube	PowerCubes list
CubeGroup	PowerCubes list
CurrencyRate	Currency Table property sheet
CurrencyRecord	Currency Table property sheet
CurrencyTable	Currency Table property sheet
CustomView	Custom View list
DataSource	Data Sources list
DateDimension	Dimension map
DateDrillDown	Dimension diagram
DateLevel	Dimension map
DateWizard	Tools menu, Date Wizard command
DbDataSource	Data Sources list
Dimension	Dimension map
DrillDown	Dimension diagram
Filters	Data Sources list
FlatFileDataSource	Data Sources list
IqdDataSource	Data Sources list
Level	Dimension map
Measure	Measures list
Model	Help menu, About <model file name> command
Name	<Object> property sheet, Name box
Package	Data Sources list
Prompts	Data Sources list

Objects	UI equivalent
Query	Data Sources list
Report	Data Sources list
DrillThroughTarget	DrillThrough Target
SecurityObject	Security Object Management that is opened from the Security menu, Show Security Objects menu item.
Signon	View menu, Signons command, Signons list
SpecialCategory	Dimension diagram
SuspendedModel	File menu, View Suspended Models command, Select Suspended Model box
View	Dimension diagram, Dimension tab

## Methods

The following table shows a summary of user interface equivalents for methods.

Methods	UI equivalent
Add	Edit menu, Insert <i>object</i> command
AssociateWith	Dimension property sheet, General tab, External Rollup Column box (when the Contains Externally Rolled Up Measure Values check box is selected in the General tab of the Data Source property sheet) Level property sheet, Source tab or Order By tab Measure property sheet, Type tab (when Type set to Column) Currency Table property sheet, Base Table Columns box (when Use An External Currency Data Source check box is selected)
CheckLocalPowerCubes	Tools menu, Check Local PowerCubes command
CheckModel	Tools menu, Check Model command
CleanHouse	Tools menu, Clean House command
Close	File menu, Close command
ConnectWithCategory	Dimension diagram (when dragging a category to a new parent of the same level)
CreateAlternateDrillDown	Edit menu, Create Drill-Down command (when a level is selected)

Methods	UI equivalent
CreateDateDimension	Date Wizard, Finish button
CreateMDCFile	Run menu, Create Selected PowerCube command
CreateMDCFiles	Run menu, Create PowerCubes command
Delete	Edit menu, Delete <object> command
DimensionAssociateWith	Date Wizard
DoAutoDesign	New Model wizard, Run AutoDesign check box
GenerateCategories	Run menu, Generate Categories command
Item	A selected category, cube, data source, or dimension
LoadCurrencyTable	Currency Table property sheet, Load Table button (when Use An External Currency Data Source check box is selected)
Logon	Security menu, Log On command
Move	Any qualified use of the drag and drop method
MoveToCategory	Dimension diagram (when dragging a category to a new position in the hierarchy)
MoveToLevel	Dimension diagram (when dragging a category to a new position in the hierarchy)
NewModel	File menu, New command
OpenModel	File menu, Open command
Remove	Edit menu, Delete <object> command
ResetPartitions	Tools menu, Reset Partitions command
Save	File menu, Save command
SaveAs	File menu, Save As command
SetAllocation	Object property sheet, Allocation tab, OK button (when there are multiple data sources)
TestBuild	Run menu, Test Build command
Update	Object property sheet, OK button

## Properties

The following table shows a summary of user interface equivalents for properties.

Properties	UI equivalent
ActivityMeasure	Measure property sheet, Rollup tab, Activity Measure box (when measure type is Category Count) Access: Read/Write
Aggregate	Special Category property sheet, Relative Time tab, Basic Approach box Access: Read/Write
AllocationMeasure	Dimension, Level, or Category property sheet, Allocation tab Access: Read
AllocationType	Dimension, Level, or Category property sheet, Allocation tab Access: Read
AllowCurrencyConversion	Measure property sheet, General tab, Allow Currency Conversion check box Access: Read/Write
AllowDrillThrough	Object property sheet, Drill Through tab, Allow Drill Through For This Measure check box Access: Read/Write
AlternateQueryPath	PowerCube property sheet, General tab, Source File box Access: Read/Write
Apex	Diagram menu, Apex command (when a category in a view is selected) Access: Read/Write
Application	Transformer user interface Access: Read



Properties	UI equivalent
AssociationRole	<p>Dimension property sheet, General tab, External Rollup Column box (when the Contains Externally Rolled Up Measure Values check box is selected in the General tab of the Data Source property sheet)</p> <p>Level property sheet, Source tab</p> <p>Measure property sheet, Type tab (when Type option is set to Column)</p> <p>Currency Table property sheet, Base Table Columns box (when Use An External Currency Data Source check box is selected)</p> <p>Access: Read/Write</p>
Associations	<p>Dimension property sheet, General tab, External Rollup Column box (when the Contains Externally Rolled Up Measure Values check box is selected in the General tab of the Data Source property sheet)</p> <p>Level property sheet, Source tab</p> <p>Measure property sheet, Type tab (when Type set to Column)</p> <p>Currency Table property sheet, Base Table Columns box (when Use An External Currency Data Source check box is selected)</p> <p>Access: Read</p>
AssociationType	<p>Dimension property sheet, General tab, External Rollup Column box (when the Contains Externally Rolled Up Measure Values check box is selected in the General tab of the Data Source property sheet)</p> <p>Level property sheet, Source tab</p> <p>Measure property sheet, Type tab (when Type set to Column)</p> <p>Currency Table property sheet, Base Table Columns box (when Use An External Currency Data Source check box is selected)</p> <p>Access: Read/Write</p>
BlankSubstitute	<p>Level property sheet, General tab, Blank Substitution box</p> <p>Access: Read/Write</p>
BlockParentTotals	<p>PowerCube property sheet, Processing tab, Block totals for parents with excluded children check box</p> <p>Access: Read/Write</p>

<b>Properties</b>	<b>UI equivalent</b>
CacheCrossTabs	PowerCube property sheet, Processing tab, Enable Crosstab Caching check box Access: Read/Write
CalculationDefinitions	Dimension property sheet, Calculation tab Access: Read
CanAllocate	Dimension, Level, Category property sheets (when Allocation tab is visible) Access: Read
CanAllocateByMeasure	Object property sheet, Allocation tab, Select a Measure box Access: Read
CanAllocateMeasure	Dimension, Level, Category property sheets, Allocation tab Access: Read
Categories	Dimension diagram, category viewer Access: Read
Category	Dimension diagram, category viewer Access: Read/Write
CategoryCount	Edit menu, Show Counts command Access: Read
CategoryCountLevel	Measure property sheet, Type tab, Dimension box and Level box (when Category Count button is selected) Access: Read/Write
CategorySets	Dimension Category Calculation dialog box (accessed through Dimension property sheet, Calculation tab) Access: Read
CharacterType	Data Source property sheet, Source tab, Character Set box Access: Read/Write
ChildCategories	dimension diagram, category viewer Access: Read
ChildCubes	PowerCubes list Access: Read

Properties	UI equivalent
ChildCustomViews	Custom Views list and Categories Viewer, Custom View tab
Code	Category property sheet, General tab, Category Code box Access: Read/Write
Columns	Data Sources list Access: Read
ColumnsLoaded	Data Sources list Access: Read
Consolidate	PowerCube property sheet, General tab, Consolidate box Access: Read/Write
Context	Level property sheet, Order By tab, Drill-Down list Access: Read/Write
ContextLevel	Special Category property sheet, Relative Time tab, Context Period box Access: Read/Write
ContextOffset	Special Category property sheet, Relative Time tab, Context Offset box Access: Read/Write
ConvergenceLevel	In the dimension map, the level at which two or more alternate drill-down paths meet Access: Read
Count	None Access: Read
CountryCode	Currency property sheet, Country Code role, or Currency Record dialog box, Country Code box Access: Read/Write
CubeCreation	PowerCube property sheet, Processing tab, Cube Creation box, one of: Enabled, Disabled Access: Read/Write
Cubes	PowerCubes list Access: Read
CubeStamp	None

<b>Properties</b>	<b>UI equivalent</b>
CurrencyDecimals	Currency property sheet, Currency Record dialog box, Decimals box Access: Read/Write
CurrencyFormatOverride	Currency property sheet, Currency Record dialog box, Override the Country Code check box Access: Read/Write
CurrencyIsEMU	Currency property sheet, Currency Record dialog box, Member of the Economic and Monetary Union (EMU) check box Access: Read/Write
CurrencyIsEuro	None Access: Read/Write
CurrencyRates	Currency Table property sheet Access: Read
CurrencyRecord	Currency property sheet, Currency Record dialog box Access: Read
CurrencyRecords	Currency Table property sheet Access: Read
CurrencySymbol	Currency property sheet, Currency Record dialog box, Currency Symbol box Access: Read/Write
CurrencyTable	File menu, Currency Table command, Currency Table property sheet Access: Read
CurrencyTables	File menu, Currency Table command, Currency Table property sheet Access: Read
CurrencyTableType	Currency Table property sheet, Base Table or Euro Table Access: Read/Write
CurrentModel	Help menu, About <model file name> command, File Name Access: Read
CustomViews	CustomViews list and Categories Viewer, Custom View tab

<b>Properties</b>	<b>UI equivalent</b>
DataCharacterSet	Preferences property sheet, General tab, Data Character Set box Access: Read/Write
DataClass	Object property sheet, General tab, Data Class box Access: Read/Write
DataRange	New Data Source wizard or New Model wizard, Table or Range box Data Source property sheet, Source tab, Table or Range box Access: Read/Write
DataSourcePath	Preferences property sheet, Directories tab, Data Source box Access: Read/Write
DataSources	Data Sources list Access: Read
DataTemporaryFilePath	Preferences property sheet, Directories tab, Data Temporary Files (Dir1;Dir2) box Access: Read/Write
DateDegreeofDetail	Column property sheet, Time tab, Degree of Detail box Access: Read/Write
DateDegreeofDetailLevelName	Column property sheet, General tab, Degree of Detail box (when the Contains Externally Rolled Up Measure Values check box is selected in the General tab of the Data Source property sheet) Access: Read/Write
DateFormat	Level property sheet (Date), Time tab, Date Format box Access: Read/Write
DateFunction	Level property sheet (Date), Time tab, Date Function box Access: Read/Write
DateInputFormat	Column property sheet, Time tab, Date Input Format box Access: Read/Write

Properties	UI equivalent
DateLevel	Currency Table property sheet, Currency Record dialog box, Date Level box Access: Read/Write
DateWizard	Tools menu, Date Wizard command Access: Read
DecimalPoint	Data Source property sheet (flat file data source type), General tab, Decimal Separator box Access: Read/Write
Decimals	Column Object: None Access: Read
DefaultDateFormat	Preferences property sheet, General tab, Default Date Format box Access: Read/Write
Description	Object property sheet, Description tab Access: Read/Write
DesiredPartitionSize	PowerCube property sheet, Auto-Partition tab, Desired Partition Size box Access: Read/Write
DetachDataSource	Preferences property sheet, Files tab, Detach the Data Source after Generating Categories check box Access: Read/Write
DetailLevel	PowerCube property sheet (Cube Group), Cube Group tab, Lowest Detail of Categories in the Level box Access: Read/Write
DimensionInclude	Dimension diagram, left pane, User Class tab Access: Read/Write
DimensionLevels	Dimension Map Access: Read
DimensionName	Date Wizard Access: Write
Dimensions	Dimension Map Access: Read

<b>Properties</b>	<b>UI equivalent</b>
DimensionView	Dimension diagram, User Class tab Access: Read
DimensionViewType	PowerCube property sheet, Dimensions tab (when you right-click a dimension) Access: Read/Write
DrillCode	Drill Category property sheet, Category Code box Access: Read/Write
DrillDowns	Dimension Map Access: Read
DrillInclusion	Drill Category property sheet, General tab, Inclusion box Access: Read/Write
DrillThroughTargets	PowerCube property sheet, Drill Through tab, Custom Reports box Access: Read
DuplicateRollup	Measure property sheet, Rollup tab, Duplicates Rollup box Access: Read/Write
DuplicateWeight	Measure property sheet, Rollup tab, Duplicate Weight box Access: Read/Write
EarliestDate	Date Wizard Dimension property sheet (Date), Time tab, Earliest Date box Run menu, Generate Date Categories command, Starting box Access: Write (Date Wizard) Read/Write (DateDimension)
EMUEntryDate	Currency Table property sheet, Currency Record dialog box, Entry Date in the EMU (YYYYMMDD) box Access: Read/Write
EnableMessageLogging	Preferences property sheet, Logging tab, Enable Message Logging check box Access: Read/Write

Properties	UI equivalent
EnableTimePeriod	Date Wizard Access: Write
EstimatedRows	PowerCube property sheet, Auto-Partition tab, Estimated Number of Consolidated Records box Access: Read/Write
ExcludeAutoPartition	Dimension property sheet, General tab, Exclude the Dimension from Auto-Partitioning check box Access: Read/Write
ExpressionText	Column property sheet, General tab, Calculated option, Calculation button, Column Calculation box Dimension property sheet, Calculation tab, Add button, Calculation button, Dimension Category Calculation box Measure property sheet, Type tab, Calculated option, Calculation button, Measure Calculation box Access: Read/Write
External	Data Source property sheet, General tab, Contains Externally Rolled Up Measure Values check box Access: Read/Write
FieldSeparator	Data Source property sheet (flat file data source type), Source tab, Field Delimiter box Access: Read/Write
FileName	Windows Explorer Access: Read/Write
Format	Object property sheet, Format tab Access: Read/Write
FormatDecimals	Measure property sheet, Format tab, Decimal Places box Access: Read/Write
FullName	Windows Explorer Access: Read
GenerateCategories	Data Source property sheet, General tab, Generate Categories check box Access: Read/Write



Properties	UI equivalent
GenerateDateCategories	Level property sheet (Date), Time tab, Generate All Categories in the Period check box (enabled when set to trGenerateDatesAll) Access: Read/Write
GeneratePowerCube	Data Source property sheet, General tab, PowerCube Creation check box Access: Read/Write
GenerateTimePeriod	Run menu, Generate Date Categories (when a time dimension is selected) Access: Read/Write
Group	Calculation Definition box, Group Calculated Categories Together check box Access: Read/Write
GroupLevel	PowerCube property sheet, Cube Group tab, Level box Access: Read/Write
HasSubdimension	In the dimension map, a level that contains a subdimension has an ellipsis (...) next to its name Access: Read
Inclusion	Object property sheet, General tab, Inclusion box Access: Read/Write
IncrementalUpdate	PowerCube property sheet, Processing tab, This Cube is Incrementally Updated check box Access: Read/Write
InputScale	Column property sheet, General tab, Input Scale box Access: Read/Write
IsAnyColumnMismatched	Tools menu, Check Columns command Access: Read
IsBad	File menu, View Suspended Models command Access: Read

Properties	UI equivalent
IsExpressionValid	<p>OK button for</p> <p>Dimension property sheet, Calculation tab, Add or Modify button, Dimension Category Calculation dialog box</p> <p>Column property sheet, General tab, Calculation button, Column Calculation dialog box (when Column Type is set to Calculated)</p> <p>Measure property sheet, Type tab, Calculation button, Measure Calculation dialog box (when Measure Type is set to Calculated)</p> <p>Access: Read</p>
IsManual	<p>In the dimension map, manual levels appear with a hand icon</p> <p>Access: Read</p>
IsolationLevel	<p>Data Source property sheet, Source tab, Isolation Level box (when the data source is an IQD)</p> <p>New Model wizard</p> <p>Access: Read/Write</p>
IsPrimary	<p>Object property sheet, General tab, Primary Drill Category check box</p> <p>Access: Read/Write (DateDrillDown and DrillDown) Read (Category)</p>
KeyName	<p>Category property sheet, General tab, Source Value box</p> <p>Access: Read/Write</p>
Label	<p>Object property sheet, General tab, &lt;object&gt; Label box</p> <p>Access: Read/Write</p>
LastUseDate	<p>Object property sheet, General tab, Last Used On box</p> <p>Access: Read</p>
LatestDate	<p>Date Wizard</p> <p>Date Dimension property sheet, Time tab, Latest Date box</p> <p>Run menu, Generate Date Categories command, Ending box</p> <p>Access: Write (DateWizard) Read/Write (DateDimension)</p>

<b>Properties</b>	<b>UI equivalent</b>
LevelDrillDowns	Dimension Map, view of alternate drill-down paths Access: Read
Levels	Dimension Map, view of levels in a dimension Access: Read
LocalPath	Data Source property sheet, Source tab, Local Data File box Access: Read/Write
LogErrorLevel	Preferences property sheet, Logging tab, one of four options: Severe Errors, Errors and Above, Warnings and Above, or Informational and Above Access: Read/Write
LogFileAppend	Preferences property sheet, Logging tab, Append Logging Information check box Access: Read/Write
LogFileName	Preferences property sheet, Logging tab, Log File Name box Access: Read/Write
LogFilesPath	Preferences property sheet, Directories tab, Log Files box Access: Read/Write
Lunar	none (use the Level property sheet, Time tab, Date Function box to determine the value) Access: Read/Write
ManualCurrentPeriod	Dimension property sheet, Time tab, Automatically Set the Current Time Period check box Access: Read/Write
MaximizeSpeed	Data Source property sheet, General Tab, Uniqueness Verification box Access: Read/Write
MaxNumPartLevels	PowerCube property sheet, Auto-Partition tab, Maximum Number of Passes box Access: Read/Write
MaxTransactionNumber	Preferences property sheet, General tab, Maximum Number of Transactions per Commit box Access: Read/Write

Properties	UI equivalent
MDCFile	PowerCube property sheet, Output tab, PowerCube File Name box Access: Read/Write
MeasureInclude	PowerCube property sheet, Measures tab Access: Read/Write
MeasureName	PowerCube property sheet, General tab, Measure Name box Access: Read/Write
Measures	Measures list Access: Read
MeasureType	Measure property sheet, Type tab, one of Column, Calculated, or Category Count Access: Read
MissingValue	Measure property sheet, General tab, Missing Value box Access: Read/Write
ModelName	File menu, View Suspended Models command, Select Suspended Model box Access: Read/Write
ModelsPath	Preferences property sheet, Directories tab, Models box Access: Read/Write
ModelTemporaryFilesPath	Preferences property sheet, Directories tab, Model Temporary Files box Access: Read/Write
ModelType	Windows Explorer Access: Read
MonthType	Date Wizard Level property sheet (Date), Time tab, Date Function box Access: Write
Name	Object property sheet, <object> Name box Object property sheet, <object> Label box Access: Read/Write Read (Application, Name, Report)

<b>Properties</b>	<b>UI equivalent</b>
NewCatsLocked	Object property sheet, General tab, Prohibit Automatic Creation of New Categories check box Access: Read/Write
Optimize	PowerCube property sheet, Processing tab, Optimization box Access: Read/Write
OrderByDescending	Level property sheet, Order By tab, Sort Order box Access: Read/Write
OrderByStorageType	Level property sheet, Order By tab, Sort As box Access: Read/Write
Origin	None Access: Read
OriginalName	Object property sheet, General tab, Original Name box Access: Read
Orphanage	Category property sheet, Orphanage check box Access: Read/Write
OutputScale	Measure property sheet, General tab, Output Scale box Access: Read/Write
Parent	Any hierarchical view in Transformer Access: Read
ParentCategories	Dimension diagram, category viewer Access: Read
Partition	Object property sheet, General tab, Partition Number box Access: Read/Write
Password	PowerCube property sheet, Output tab, Password box Signon property sheet, General tab, Password box Access: Write
Path	Windows Explorer Access: Read

Properties	UI equivalent
PopulateByDataSource	None Access: Read/Write
Position	Column property sheet, General tab, Position box Access: Read/Write
PowerCubesPath	Preferences property sheet, Directories tab, PowerCubes box Access: Read/Write
Precision	Measure property sheet, Format tab, Precision box Access: Read/Write
QualifiedName	Dimension map Access: Read
QuarterType	Date Wizard Level property sheet (Date), Time tab, Date Function box Access: Write
QyPath	Preferences property sheet, Directories tab, Model Temporary Files box Access: Read
Rate	Currency Table property sheet Access: Read/Write
RefreshDescription	Level property sheet, Source tab, Description Refresh check box Access: Read/Write
RefreshLabel	Level property sheet, Source tab, Label Refresh check box Access: Read/Write
RefreshShortName	Level property sheet, Source tab, Tag Refresh check box Access: Read/Write
RegularRollup	Measure property sheet, Rollup tab Access: Read/Write
RegularWeight	Measure property sheet, Rollup tab, Regular Weight box Access: Read/Write

<b>Properties</b>	<b>UI equivalent</b>
ReverseSign	Object property sheet, General tab, Reverse the Sign check box Access: Read/Write
Rollup	Special Category property sheet, General tab, Category Rollup check box Access: Read/Write
RollupTiming	Measure property sheet, Rollup tab, Regular Timing box (for a calculated measure) Access: Read/Write
RowsAsSample	Preferences property sheet, AutoDesign tab, Number of Rows Used as a Sample box Access: Read/Write
RowsChecked	Preferences property sheet, AutoDesign tab, Number of Rows Checked box Access: Read/Write
RunningPeriods	Special Category property sheet, Relative Time tab, Number of Periods box (when Relative Time is set to Custom and Basic Approach is set to a running total) Access: Read/Write
Server	PowerCube property sheet, Processing tab, Processed box Access: Read/Write
ServerModelPath	Model property sheet, Server tab, Model Path box Access: Read/Write
ServerPath	Data Source property sheet, Source tab, Server Data File box (when Data Source Location is set to Server) Access: Read/Write
ServerQuery	Data Source property sheet, Source tab, Data Source Location box Access: Read/Write
ServicesBuildNumber	About box Access: Read
ServicesVersionText	About box Access: Read

<b>Properties</b>	<b>UI equivalent</b>
SetsCurrentPeriod	Data Sources property sheet, General tab, Sets the Current Period check box Access: Read/Write
ShortName	Object property sheet, General tab, Short Name box Access: Read/Write
Signon	PowerCube property sheet, Output tab, Signon box (when database type is set to a value other than Local) Access: Read/Write
Signons	Signons list Access: Read
Size	Column property sheet, General tab, Size box Access: Read/Write
Size (Model)	Windows Explorer Access: Read
SortComparisonRule	Preferences property sheet, General tab, Sort Comparison Rules box Access: Read/Write
SourceType	Data Source property sheet, Source tab, Source Type box Access: Read/Write
SpecialCategoryCount	Dimension map (second figure in brackets under the dimension name) Access: Read
SQLExpression	Data Source property sheet, SQL View tab Access: Read
Status	PowerCube property sheet, Processing tab, Status box Access: Read
StorageType	Measure property sheet, General tab, Storage Type box Access: Read/Write



<b>Properties</b>	<b>UI equivalent</b>
SummaryLevel	PowerCube property sheet, Cube Group tab, Summarize All External Categories in the Level box Access: Read/Write
SuspendedModels	File menu, View Suspended Models command, Select Suspended Model box Access: Read
TargetLevel	Special Category property sheet, Relative Time tab, Target Period box (when Relative Time is set to Custom) Access: Read/Write
TargetOffset	Special Category property sheet, Relative Time tab, Target Offset box (when Relative Time is set to Custom) Access: Read/Write
ThousandPoint	Data Source property sheet, General tab, 1000 Separator box (for flat file data sources) Access: Read/Write
Time	Windows Explorer Access: Read
TimeArrayColumn	Column property sheet, Column Name box Access: Read/Write
TimeArrayStartMonth	Column property sheet, Array tab, Start Month box Access: Read/Write
TimeArrayType	Column property sheet, Array tab, Array Type box Access: Read/Write
TimeRank	Level property sheet (Date), Time tab, Time Level Ranking box Access: Read/Write
TimeStateRollup	Measure property sheet, Rollup tab, Time State Rollup box Access: Read/Write
TimeStateWeight	Measure property sheet, Rollup tab, Time State Weight box Access: Read/Write

Properties	UI equivalent
ToDateLevel	Special Category property sheet, Relative Time tab, To-date Period box (when Relative Time is set to Custom) Access: Read/Write
Type	None Access: Read
Unique	Level property sheet, Source tab, Unique check box Access: Read/Write
UniqueMove	Level property sheet, Source tab, Move check box Access: Read/Write
UserID	Signon property sheet, General tab, User ID box Access: Read/Write
Version	Help menu, About IBM Cognos PowerPlay Transformer command, About Transformer box Access: Read
Views	dimension diagram, Dimension tab or User Class tab Access: Read
ViewStatus	Diagram menu, one of; Exclude, Cloak, Suppress, Summarize or Apex command (when a category in a view is selected) Access: Read/Write
ViewType	PowerCube property sheet, Dimensions tab (right-click a selection) Access: Read/Write
WeekAdd	Date Wizard Drill Category property sheet (Date), Time tab, Add an Extra Week box Access: Write (DateWizard) Read/Write (DateDrillDown)
WeekSpan	Date Wizard Drill Category property sheet (Date), Time tab, Partial Weeks box Access: Write (DateWizard) Read/Write (DateDrillDown)

<b>Properties</b>	<b>UI equivalent</b>
WeekStart	Drill Category property sheet (Date), Time tab, Week Begins On box Access: Read/Write
WeekStartDay	Date Wizard Access: Write
WorkingDay	Date Wizard Dimension property sheet (Date), Time tab, Days in Week box Access: Write (DateWizard) Read/Write (DateDimension)
WorkingDays	Date Wizard Dimension property sheet (Date), Time tab, Days in Week box Access: Write (DateWizard) Read/Write (DateDimension)
YearStartDay	Date Wizard Drill Category property sheet (Date), Time tab, Year Begins box Access: Write (DateWizard) Read/Write (DateDrillDown)
YearType	Date Wizard Level property sheet (Date), Time tab, Date Function box Access: Write

## Value Lists and Constants

The following table shows a summary of user interface equivalents for value lists and constants.

<b>Constants</b>	<b>UI equivalent</b>
xtrAllocationType	Level property sheet, Allocation tab, Shortcut menu (right-click a selected measure)

Constants	UI equivalent
xtrAssociationRole	<p>Dimension property sheet, General tab, External Rollup Column box (when the Contains Externally Rolled Up Measure Values check box is selected in the General tab of the Data Source property sheet)</p> <p>Level property sheet, Source tab or Order By tab</p> <p>Measure property sheet, Type tab (when Type set to Column)</p> <p>Currency Table property sheet, Base Table Columns box (when Use An External Currency Data Source check box is selected)</p>
xtrAssociationType	<p>Dimension property sheet, General tab, External Rollup Column box (when the Contains Externally Rolled Up Measure Values check box is selected in the General tab of the Data Source property sheet)</p> <p>Level property sheet, Source tab, or</p> <p>Measure property sheet, Type tab (when Type set to Column), or</p> <p>Currency Table property sheet, Base Table Columns box (when Use An External Currency Data Source check box is selected)</p>
xtrCharacterType	Data Source property sheet (flat file data source), Character Set box
xtrCubeConsolidate	PowerCube property sheet, General tab, Consolidate box
xtrCubeCreation	PowerCube property sheet, Processing tab, Cube Creation box
xtrCubeOptimize	PowerCube property sheet, Processing tab, Optimization box
xtrCubeStatus	PowerCube property sheet, Processing tab, Status box
xtrCurrencyTableType	Currency Table property sheet
xtrDataClass	Object property sheet, General tab, Data Class box
xtrDateCategoriesGeneration	Level property sheet (Date), Time tab, Generate All Categories in the Period check box
xtrDateFormat	Column property sheet, Time tab, Date Input Format box
xtrDateLevel	Column property sheet, Time tab, Degree of Detail box

<b>Constants</b>	<b>UI equivalent</b>
xtrDuplicateRollup	Measure property sheet, Rollup tab, Duplicates Rollup box
xtrGenerateOptions	Level property sheet (Date), General tab, Inclusion box
xtrInclusion	Object property sheet, General tab, Inclusion box
xtrMeasureType	Measure property sheet, Type tab
xtrMissingValue	Measure property sheet, General tab, Missing Value box
xtrObjectType	An item in the UI that references a Transformer OLE object
xtrOrigin	None
xtrPowerCubeGeneration	Data Source property sheet, General tab, Timing box, PowerCube Creation check box (option buttons)
xtrPreferences	File menu, Preferences command, Preferences property sheet
xtrRollup	Measure property sheet, Rollup tab, Regular Rollup box
xtrRollupTiming	Measure property sheet, Rollup tab, Regular Timing box (for a calculated measure)
xtrSourceType	Data Source property sheet, Source tab, Source Type box
xtrSpecialFunction	Date Level property sheet, Time tab, Date Function box
xtrStorage	Measure property sheet, General tab, Storage Type box
xtrTimeAggregate	Special Category property sheet, Relative Time tab, Basic Approach box (when Relative Time set to Custom)
xtrTimeArrayType	Column property sheet, Array tab, Array Type box
xtrTimeRollup	Measure property sheet, Rollup tab, Time State Rollup box
xtrTimeType	Date Wizard

<b>Constants</b>	<b>UI equivalent</b>
xtrViewStatus	Diagram menu, choose one of the following commands: Exclude, Cloak, Suppress, Summarize or Apex (when a category in a view is selected)
xtrViewType	PowerCube property sheet, Dimension tab, shortcut menu (right-click a selected dimension)
xtrWeekAdd	Drill Category property sheet (Date), Time tab, Add an Extra Week box
xtrWeekDay	Dimension property sheet (Date), Time tab, Days in Week box  Drill Category property sheet (Date), Time tab, Week Begins On box
xtrWeekSpan	Drill Category property sheet (Date), Time tab, Partial Weeks box

---

## Chapter 8. Samples and Examples

This section provides a variety of Visual Basic code examples to illustrate certain OLE automation concepts. The code examples were tested in Visual Basic 2008.

**Note:** You may want to reuse the examples provided in this section. Be aware that reusing the examples may involve more than copying and pasting the example code. For example, your development environment may handle line breaks differently.

You can also use the samples that are included with the product installation. The sample GO Data Warehouse (analysis) and GO Data Warehouse (query) packages are contained in the IBM\_Cognos\_Samples.zip file, which is located in the *installation\_location*\webcontent\samples\content\ directory.

For information about installing and setting up IBM Cognos samples, see the *Installation and Configuration Guide*.

---

### Open a Model and Specify an Order by Association Example

This example opens a model and specifies an order by association for one of the drill-down paths at a convergence level.

```
Sub Example1()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDrill As Object
    Dim objAssociation As Object
    Dim objColumn As Object
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "Great outdoors 8.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objDrill = objModel.Dimensions("Retailers").DrillDowns(2)
    'Create an OrderBy association for the convergence level.
    objAssociation = objDrill.ConvergenceLevel.Associations.Add()
    'Select the drill-down path to which the sort applies.
    objAssociation.Context = objDrill
    'Select the data source column, RetailerName, on which to base
    the sort.
    objColumn = objModel.DataSources("Retailer site (csv)").Columns("Retailer name")
    With objAssociation
        .AssociationRole = xtrAssociationRole.trAssociationOrderBy
        .AssociationType = xtrAssociationType.trAssociationQuery
        .Label = objColumn.Name
        .Update()
    End With
    objDrill.Levels("Retailer site").OrderByDescending(objAssociation.Context) =
False
    With objModel
        .GenerateCategories()
        .Update()
        .SaveAs("Great outdoors 8X.mdl")
        .Close()
    End With
    objColumn = Nothing
    objAssociation = Nothing
    objDrill = Nothing
    objModel = Nothing
End Sub
```

```

objTransApp = Nothing
End Sub

```

## Open a Model and Add a Calculation Example

This example opens a model and adds a calculation definition to the Products dimension. The expression uses the share function and includes a category set to create two calculated categories.

**Note:** In some cases, you cannot use a macro to set the properties of a category object. Your macro may appear valid but when you run the 'parentCategory =' or 'childCategory =' portions, you receive a message: 'This collection is Read Only in this context.' We recommend using the user interface instead.

```

Sub Example2()
    Dim objTransApp As Object
    Dim model As Model
    Dim dimension As Dimension
    Dim calcDef As CalculationDefinition
    Dim catSet As CategorySet
    Dim drillDown As DrillDown
    Dim parentCategory As Category
    Dim childCategory As Category
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "great outdoors sales.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    model = objTransApp.OpenModel(strModelPath)
    dimension = model.Dimensions.Item("Products")
    calcDef = dimension.CalculationDefinitions.Add()
    catSet = calcDef.CategorySets.Add()

    drillDown = dimension.DrillDowns.Item(1)
    parentCategory = drillDown.Categories.Item(1) 'select "Camping
Equipment"
    For child_category_index = 1 To parentCategory.ChildCategories.Count
        childCategory = parentCategory.ChildCategories.Item(child_category_index)
        If childCategory.Name = "Cooking Gear" Or childCategory.Name
= "Tents" Then
            catSet.Categories.Add(childCategory)
        End If
    Next
    catSet.Label = "Set 1"
    With calcDef
        .ExpressionText = "share ( catset ( ""Set 1"" ) , "" &
-
        parentCategory.Code & """)"
        .Group = False
        .Name = "share(""Set 1"", ""Camping Equipment")"
        .Update()
    End With
    With model
        .SaveAs("great outdoors salesX.mdl")
        .Close()
    End With
End Sub

```

## Create a Relative Time Category Example

This example uses the SpecialCategory object to create a relative time category and then sets the applicable properties.

```

Sub Example3()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objSpecCategory As Object

```



```

Dim strIBMCognos10Location As String
Dim strModelPath As String
Dim strModelSource As String
'Change these paths to match your installation
strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
strModelSource = "Sales and Marketing.mdl"
strModelPath = strIBMCognos10Location & _
    "webcontent\samples\models\Transformer8\EN\" & strModelSource
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
With objTransApp
    .DataSourcePath = strIBMCognos10Location & "bin"
    .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
End With
objModel = objTransApp.OpenModel(strModelPath)
'Create a relative time special category in a time dimension.
objSpecCategory = _
    objModel.Dimensions("Time").Categories.Add(xtrObjectType.trSpecialCategory)
With objSpecCategory
    .Aggregate = xtrTimeAggregate.trAggregateRunning
    .Code = "Five Month Period"
    .ContextLevel = "Quarter"
    .ContextOffset = -3
    .Name = "Five Month Period"
    .Rollup = True
    .RunningPeriods = 5
    .TargetLevel = "Month"
    .TargetOffset = -1
    .Update()
End With
With objModel
    .SaveAs("Sales and MarketingX.mdl")
    .Close()
End With
objSpecCategory = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Add a Cube Group Example

This macro adds a cube group to an existing model using the 'Sales region' dimension as the basis for the group. One child cube in the group is not created.

```

Sub Example4()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDimRegion As Object
    Dim objCubesByRegion As Object
    Dim strCategoryCode As String
    Dim objChildCube As Object
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "great outdoors sales.mdl"

    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objDimRegion = objModel.Dimensions("Sales region")
    objCubesByRegion = objModel.Cubes.Add(xtrObjectType.trCubeGroup)
    With objCubesByRegion
        'Specify the category levels on which to base the cubes
        in the group.
        .GroupDimension = objDimRegion
        .GroupLevel = objDimRegion.DrillDowns(1).Levels("Sales region")
        'Specify the level of detail and summary level.
        .DetailLevel = objDimRegion.DrillDowns(1).Levels("Country")
        .SummaryLevel = objDimRegion.DrillDowns(1).Levels("Sales
region")
        .Name = "Regions"
    End With
End Sub

```

```

        .MDCFile = ""
        .CubeCreation = xtrCubeCreation.trCubeCreationON
        .Optimize = xtrCubeOptimize.trOptimizeDefault
        .CompressMDC = False
        .CacheCrossTabs = False
        .MeasureInclude(objModel.Measures("Unit cost")) = False
        .MeasureName = "Revenue Made"
        .Update()
    End With
    'Name each cube in the group after its category name
    For index = 1 To objCubesByRegion.GroupLevel.CategoryCount
        strCategoryCode = objCubesByRegion.GroupLevel.LevelCategories(index).Code
        objChildCube = objCubesByRegion.ChildCubes(strCategoryCode)
        With objChildCube
            .Name = objCubesByRegion.GroupLevel.LevelCategories(index).Name
            .Update()
        End With
    Next
    'Do not generate a child cube for Central Europe.
    objChildCube = objCubesByRegion.ChildCubes("Central Europe")
    With objChildCube
        .CubeCreation = xtrCubeCreation.trCubeCreationOFF
        .Update()
    End With
    'Generate the cubes (this may take a few minutes)
    objCubesByRegion.CreateMDCFile()
    With objModel
        .SaveAs("great outdoors salesX.mdl")
        .Close()
    End With
    objChildCube = Nothing
    objDimRegion = Nothing
    objCubesByRegion = Nothing
    objModel = Nothing
    objTransApp = Nothing
End Sub

```

## Add an Additional Data Source to a Model Example

This example adds another data source to a model and then sets allocation using the new data source. It creates the Forecast measure and allocates by proportion using the Revenue measure as a weighting factor.

```

Sub Example5()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDataSource As Object
    Dim objDimension As Object
    Dim objColumn As Object
    Dim objLevel As Object
    Dim objMeasure As Object
    Dim objByMeasure As Object
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    Dim strDataSource As String
    Dim strDataPath As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "great outdoors sales.mdl"
    strDataSource = "prod_plan.csv"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    strDataPath = strIBMCognos10Location & _
        "webcontent\samples\datasources\cubes\PowerCubes\EN\great
    outdoors sales\" & _
        strDataSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    'Add and define a second datasource.
    objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
    With objDataSource
        .CharacterType = xtrCharacterType.trCharAnsiSingleByte
    End With
End Sub

```

```

.External = False
.FieldSeparator = ","

.LocalPath = strDataPath
.MaximizeSpeed = True
.Name = "Product Plan (csv)"
.SourceType = xtrSourceType.trFlatFileColumnNames
.Update()
End With
'Specify data classes and rename columns.
objColumn = objDataSource.Columns(1)
With objColumn
.DataClass = xtrDataClass.trDataClassDate
.Name = "Order Date"
.Update()
End With
objColumn = objDataSource.Columns(3)
With objColumn
.DataClass = xtrDataClass.trDataClassDescription
.Name = "Product line code"
.Update()
End With
'Add a measure.
objColumn = objDataSource.Columns("EXPECTED_VOLUME")
With objColumn
.DataClass = xtrDataClass.trDataClassQuantity
.InputScale = 0
.Name = "Forecast"
.Update()
End With
objMeasure = objModel.Measures.Add()
With objMeasure
.AssociateWith(objColumn, xtrAssociationRole.trAssociationSource)
.Description = "Forcasted volume for product line."
.AllowDrillThrough = False
.Update()
End With
'Test whether allocation is possible.
objLevel = objModel.Dimensions("Products").DimensionLevels(1)
If objLevel.CanAllocate = True Then
objMeasure = objModel.Measures("Forecast")
'Test whether the specified measure can be allocated.
If objLevel.CanAllocateMeasure(objMeasure) = True Then
'Test whether allocation by proportion is already set.
If objLevel.AllocationType(objMeasure) <> _
xtrAllocationType.trAllocationByAnotherMeasure Then
objByMeasure = objModel.Measures("Revenue")
'Test whether the specified measure can be used
as a weighting factor.
If objLevel.CanAllocateByMeasure(objByMeasure) = True
Then
objLevel.SetAllocation(objMeasure, _
xtrAllocationType.trAllocationByAnotherMeasure,
objByMeasure)
End If
End If
End If
'Suppress allocation to the Margin range dimension.
objDimension = objModel.Dimensions("Margin range")
If objDimension.CanAllocate = True Then
objDimension.SetAllocation(objMeasure, xtrAllocationType.trAllocationNA)
End If
With objModel
.SaveAs("great outdoors salesX.mdl")
.Close()
End With
objByMeasure = Nothing
objMeasure = Nothing
objLevel = Nothing
objColumn = Nothing

```

## Open a Model and Modify the Cube Properties Example

This example opens a model and modifies the cube properties.

```

Sub Example6()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objCube As Object
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "great outdoors sales.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objCube = objModel.Cubes.Item(1)
    With objCube
        .BlockParentTotals = True
        .CacheCrossTabs = True
        .CompressMDC = False
        .Consolidate = xtrCubeConsolidate.trConsolidateDefault
        .CubeCreation = xtrCubeCreation.trCubeCreationON
        .DesiredPartitionSize = 500000
        .EstimatedRows = 10000000
        .IncrementalUpdate = False

        .MaxNumPartLevels = 5
        .MDCFile = "GoCube"
        .Optimize = xtrCubeOptimize.trOptimizeAutoPartition
        .Name = "Great Outdoors Sales (Optimized)"
        .Update()
        .CreateMDCFile()
    End With
    objModel.SaveAs("great outdoors salesX.mdl")
    objModel.Close()
    objCube = Nothing
    objModel = Nothing
    objTransApp = Nothing
End Sub

```

## Create a Custom View Example

This example creates a custom view. It then associates the custom view with the cube.

**Note:** In some cases, you cannot use a macro to set the properties of a category object. Your macro may appear valid but when you run the 'objCategory =' portion, you receive a message: 'This collection is Read Only in this context.' We recommend using the user interface instead.

```

Sub Example7()
    Dim objTransApp As Object
    Dim model As Model
    Dim dimension As Dimension
    Dim measure As Measure
    Dim custom_view As CustomView
    Dim view As TransformerSDKLib.View
    Dim category As Category
    Dim cube As Cube
    Dim intX As Integer
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "Sales and Marketing.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    model = objTransApp.OpenModel(strModelPath)
    'Set a custom view for a dimension.

```

```

dimension = model.Dimensions.Item("Retailers")
'Exclude the "Planned revenue" measure
measure = model.Measures.Item("Planned revenue")
'Create custom view
custom_view = model.CustomViews.Add()
With custom_view
    .Name = "Central Europe"
    .DimensionInclude(dimension) = xtrViewType.trViewTypeCustom
    .MeasureInclude(measure) = False
    .Update()
End With
'Get associated View object
view = custom_view.DimensionView(dimension)
'Set a view for each category in the top level of the collection.
For intX = 1 To dimension.DrillDowns.Item(1).Categories.Count
    category = dimension.DrillDowns.Item(1).Categories(intX)
    If category.Name <> "Central Europe" Then
        view.SetViewStatus(category, xtrViewStatus.trViewStatusSummaryMom)
    End If
Next intX
'Associate a custom view with a cube
cube = model.Cubes.Item("Sales and Marketing")
cube.CubeCustomViews.Add(custom_view)
With model
    .SaveAs("Sales and MarketingX.mdl")
    .Close()
End With
objTransApp = Nothing
End Sub

```

## Open a Model and Add a Currency Record Example

This example opens a model and adds a currency record to an existing currency table. It also sets the currency rates for the new record.

```

Sub Example8()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objBaseTable As Object
    Dim objCurrencyRecord As Object
    Dim objCurrencyRate As Object
    Dim objDateDrillDown As Object
    Dim objLevel As Object
    Dim intX As Integer
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "Great outdoors 8.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objBaseTable = objModel.CurrencyTables.Add()
    With objBaseTable
        .CurrencyTableType = xtrCurrencyTableType.trCurrencyTableBase
        .Update()
    End With
    objDateDrillDown = objModel.Dimensions("Years").DrillDowns(1)
    objCurrencyRecord = objModel.CurrencyRecords.Add()
    objLevel = objDateDrillDown.Levels("Year")
    objCurrencyRecord.DateLevel = objLevel
    With objCurrencyRecord
        .CountryCode = "AUS"
        .Label = "Australian Dollar"
        .CurrencyDecimals = 2
        .CurrencyFormatOverride = True
        .CurrencyIsEMU = False
        .CurrencyIsEuro = False
        .CurrencySymbol = "$"
        .Update()
    End With
End Sub

```

```

End With
For intX = 1 To objLevel.CategoryCount
    objCurrencyRate = objCurrencyRecord.CurrencyRates(intX)
    If objCurrencyRate.PopulateByDataSource = False Then
        Select Case objLevel.LevelDrillDowns(1).Categories(intX).KeyName
            Case "2004"
                objCurrencyRate.Rate = 1.54
            Case "2005"
                objCurrencyRate.Rate = 1.55
            Case Else
                '
        End Select
        Select Case objCurrencyRate.Category.KeyName
            Case "2006"
                objCurrencyRate.Rate = 1.56

            Case "2007"
                objCurrencyRate.Rate = 1.57
            Case Else
                '
        End Select
        objCurrencyRate.Update()
    End If
Next intX
With objModel
    .SaveAs("Great outdoors 8X.mdl")
    .Close()
End With
objLevel = Nothing
objDateDrillDown = Nothing
objCurrencyRate = Nothing
objCurrencyRecord = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Create a Cube Using DoAutoDesign and TestBuild Methods

### Example

This example creates a model and adds a data source. It then uses the DoAutoDesign and TestBuild methods to create a small cube.

```

Sub Example9()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDataSource As Object
    Dim strIBMCognos10Location As String
    Dim strDataSource As String
    Dim strDataPath As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strDataSource = "NATIONAL.ASC"
    strDataPath = strIBMCognos10Location & _
        "webcontent\samples\datasources\cubes\PowerCubes\EN\National\"
    & _
        strDataSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    objModel = objTransApp.NewModel
    objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
    With objDataSource
        .GenerateCategories = True
        .GeneratePowerCube = xtrPowerCubeGeneration.trGenerationDefault
        .CharacterType = xtrCharacterType.trCharAnsiSingleByte
        .FieldSeparator = ","
        .LocalPath = strDataPath
        .MaximizeSpeed = True
        .Name = "National"
        .SetsCurrentPeriod = True
        .SourceType = xtrSourceType.trFlatFileColumnNames
        .Update()
    End With
    With objModel
        .Name = "National"
        .DoAutoDesign()
        .TestBuild(20, True)
        .Update()
    End With
End Sub

```

```

        .SaveAs("NationalX.mdl")
        .Close()
    End With
    objDataSource = Nothing
    objModel = Nothing
    objTransApp = Nothing
End Sub

```

## Select, Change, and Update a Dimension Example

This example selects a Dimension object from the Dimensions collection, changes one property, and updates the dimension.

```

Sub Example10()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDimensions As Object
    Dim objLocationsDim As Object
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "great outdoors sales.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objDimensions = objModel.Dimensions
    objLocationsDim = objDimensions.Item(3)
    With objLocationsDim
        .ExcludeAutoPartition = True
        .Update()
    End With
    objModel.SaveAs("great outdoors salesX.mdl")
    objModel.Close()
    objLocationsDim = Nothing
    objDimensions = Nothing
    objModel = Nothing
    objTransApp = Nothing
End Sub

```

## Delete a Level from a Level Collection Example

This example deletes one level from a Levels collection.

```

Sub Example11()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objTimeDimension As Object
    Dim objLevel As Object
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "Sales and Marketing.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objTimeDimension = objModel.Dimensions.Item("Time")
    objLevel = objTimeDimension.DrillDowns.Item(1).Levels.Item("Month")
    objLevel.Delete()
    objTimeDimension.Update()
    objModel.SaveAs("Sales and MarketingX.mdl")
    objModel.Close()
    objLevel = Nothing

```

```

objTimeDimension = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Move a Measure Object and Change the Revenue Measure Rollup Example

This example moves a Measure object to the first position in the Measures collection. It changes the rollup of the Revenue measure to duplicate rollup. The Revenue measure uses the Quantity measure to create a weighted average.

```

Sub Example12()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objMeasures As Object
    Dim currentMeasure As Measure
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "great outdoors sales.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objMeasures = objModel.Measures
    objMeasures.Item(2).Move(1)
    currentMeasure = objMeasures("Revenue")
    With currentMeasure
        .RegularRollup = xtrRollup.trRollupAverage
        .RegularWeight = objModel.Measures("Quantity").Name
        .RollupTiming = xtrRollupTiming.trTimingDefault
        .DuplicateRollup = xtrDuplicateRollup.trDuplicateRollupAverage
        .DuplicateWeight = objModel.Measures("Quantity").Name
        .Format = "$#,##0"
        .FormatDecimals = 2
        .IgnoreMissingValue = False
        .MissingValue = xtrMissingValue.trMissingValueZERO
        .ReverseSign = False
        .ShortName = "Revenue"
        .Update()
    End With
    objModel.SaveAs("great outdoors salesX.mdl")
    objModel.Close()
    objMeasures = Nothing
    objModel = Nothing
End Sub

```

## Create a Partition and Check the Model Example

This example opens a model, creates a partition for Central Europe and the descendant categories, and checks the model.

**Note:** In some cases, you cannot use a macro to set the properties of a category object. Your macro may appear valid but when you run the 'objCategory =' portion, you receive a message: 'This collection is Read Only in this context.' We recommend using the user interface instead.

```

Sub Example13()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objCategory As Object
    Dim intX As Integer
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"

```



```

strModelSource = "great outdoors sales.mdl"
strModelPath = strIBMCognos10Location & _
    "webcontent\samples\models\Transformer8\EN\" & strModelSource
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
With objTransApp
    .DataSourcePath = strIBMCognos10Location & "bin"
    .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
End With
objModel = objTransApp.OpenModel(strModelPath)
objModel.ResetPartitions()
objCategory = objModel.Dimensions("Sales region").Drilldowns(1).Categories(2)
objCategory.Partition = 1
objCategory.Update()
For intX = 1 To objModel.CheckModel.Count
    MsgBox(objModel.CheckModel(intX).Name)
Next intX
With objModel
    .SaveAs("great outdoors salesX.mdl")
    .Close()
End With
objCategory = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Open a Model and Drill Through to a PowerCube Example

This example opens a model and uses the Quantity measure to set a drill through to a PowerCube.

```

Sub Example14()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objMeasure As Object
    Dim objReport As Object
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    Dim strReportPath As String
    Dim strReportSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "great outdoors sales.mdl"
    strReportSource = "drill_through_to_cube_7.mdc"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    strReportPath = strIBMCognos10Location & _
        "webcontent\samples\datasources\cubes\PowerCubes\EN\" &
strReportSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objMeasure = objModel.Measures.Item("Quantity")
    With objMeasure
        .AllowDrillThrough = True
        .Update()
    End With
    objReport = objMeasure.DrillThroughTargets.Add(strReportPath,
"Default Report")
    objModel.SaveAs("great outdoors salesX.mdl")
    objModel.Close()
    objReport = Nothing
    objMeasure = Nothing
    objModel = Nothing
End Sub

```

## Add the Authors Role to a Custom View Example

This example uses the **Cognos** namespace to add the Authors role to a Custom View.

```

Sub Example15()
    Dim objTransApp As Object
    Dim model As Model
    Dim new_namespace As TransformerSDKLib.Namespace
    Dim securityObject As SecurityObject

```

```

Dim customView As CustomView
Dim CAMID_of_Namespace As String
Dim CAMID_of_User As String
Dim CAMID_of_Object As String
Dim Name_of_Namespace As String
Dim Name_of_User As String
Dim Name_of_Object As String
Dim ID_of_Namespace As String
Dim strIBMCognos10Location As String
Dim strModelPath As String
Dim strModelSource As String
'Change these paths to match your installation
strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
strModelSource = "great outdoors sales.mdl"
strModelPath = strIBMCognos10Location & _
    "webcontent\samples\models\Transformer8\EN\" & strModelSource
Name_of_Namespace = "Cognos"
ID_of_Namespace = ""
Name_of_User = ""
Name_of_Object = "Authors"
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
With objTransApp

    .DataSourcePath = strIBMCognos10Location &
"bin"
    .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
End With
model = objTransApp.OpenModel(strModelPath)
'Provide namespace, username, and password in that order
objTransApp.Logon("Cognos", "", "") 'Log into the Cognos namespace
using Anonymous user
new_namespace = model.Namespaces.Add()
'Provide: Namespace and Object names
'Receive: Both Namespace's and Object's CAMIDs
With new_namespace
    .Name = Name_of_Namespace
    .ObjectName = Name_of_Object 'Authors is a group
    .Update()
    CAMID_of_Namespace = .CAMID
    CAMID_of_Object = .ObjectCAMID
    CAMID_of_User = ""
End With
'Another way to get CAMIDs below (Commented Out)
'The code below can be adapted to use any Namespace and User
'Provide: Namespace ID and user
'Receive: Namespace and User CAMIDs and Namespace name
'With new_namespace
'    .ID = ID_of_Namespace
'    .User = Name_of_User
'    .Update()
'    CAMID_of_Namespace = .CAMID
'    CAMID_of_User = .UserCAMID
'    Name_of_Namespace = .Name
'End With
'Create custom view
customView = model.CustomViews.Add()
With customView
    .Name = Name_of_Object
    .Update()
End With
'Create and add security object to custom view
securityObject = new_namespace.SecurityObjects.Add()
With securityObject
    .Name = CAMID_of_Object 'provide the User or Object CAMID
here
    .DisplayName = Name_of_Object
    .Type = xtrSecurityType.trSecurityType_Role
    .AddToCustomView(customView)
    .Update()
End With
With model
    .SaveAs("great outdoors salesX.mdl")
    .Close()
End With
objTransApp.Logoff()
objTransApp = Nothing
End Sub

```

## Check for a Suspended Model Example

This example checks for a suspended model and, if one is found, displays a message if the model is corrupt. It then uses the RemoveSuspendedModel method to delete it from the SuspendedModels collection.

```
Sub Example16()  
    Dim objTransApp As Object  
    Dim objSuspendedModel As Object  
    Dim intX As Integer  
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")  
    If objTransApp.SuspendedModels.Count = 0 Then  
        MsgBox("There are no suspended models")  
    Else  
        For intX = objTransApp.SuspendedModels.Count To 1 Step -1  
            objSuspendedModel = objTransApp.SuspendedModels(intX)  
            If objSuspendedModel.IsBad = True Then  
                MsgBox("Model: " + objSuspendedModel.ModelName +  
-                ", located at " + objSuspendedModel.QyPath + " is  
corrupt")  
                End If  
                objTransApp.RemoveSuspendedModel(objSuspendedModel)  
            Next intX  
        End If  
        objSuspendedModel = Nothing  
        objTransApp = Nothing  
    End Sub
```

## Open a Model and Create a Dimension View Example

This example opens a model and creates a dimension view for the Retailers dimension. The view provides a full drill-down path for Central Europe, but only summary-level detail for other regions. It then associates the dimension view with a cube.

**Note:** In some cases, you cannot use a macro to set the properties of a category object. Your macro may appear valid but when you run the 'objCategory =' portion, you receive a message: 'This collection is Read Only in this context.' We recommend using the user interface instead.

```
Sub Example17()  
    Dim objTransApp As Object  
    Dim objModel As Object  
    Dim objDimension As Object  
    Dim objView As Object  
    Dim objCategory As Object  
    Dim objCube As Object  
    Dim intX As Integer  
    Dim strIBMCognos10Location As String  
    Dim strModelPath As String  
    Dim strModelSource As String  
    'Change these paths to match your installation  
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\  
    strModelSource = "Sales and Marketing.mdl"  
    strModelPath = strIBMCognos10Location &  
-    "webcontent\samples\models\Transformer8\EN\" & strModelSource  
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")  
    With objTransApp  
        .DataSourcePath = strIBMCognos10Location & "bin"  
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"  
    End With  
    objModel = objTransApp.OpenModel(strModelPath)  
    objDimension = objModel.Dimensions("Retailers")  
    objCube = objModel.Cubes("Sales and Marketing")  
    If objCube.DimensionViewType(objDimension) = -  
xtrViewType.trViewTypeAllCategories Then  
        objView = objDimension.Views.Add()  
        With objView  
            .Name = "Central Europe"  
            .ViewType = xtrViewType.trViewTypeCustom  
            .Update()  
            .Parent.Update()  
        End With  
        For intX = 1 To objDimension.DrillDowns(1).Categories.Count  
            objCategory = objDimension.DrillDowns(1).Categories(intX)
```

```

        If objCategory.Name <> "Central Europe" Then
            objView.SetViewStatus(objCategory,
                xtrViewStatus.trViewStatusSummaryMom)
            If objView.GetViewStatus(objCategory) <> _
                xtrViewStatus.trViewStatusSummaryMom Then
                MsgBox("Failed to set ViewStatus!")
            End If
        End If
    End If
Next intX
objCube.DimensionView(objDimension) = objView
objCube.Update()

```

```

End If
With objModel
    .SaveAs("Sales and MarketingX.mdl")
    .Close()
End With
objCube = Nothing
objCategory = Nothing
objView = Nothing
objDimension = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Add a Cube Group to a Model Example

This macro adds a cube group to an existing model. The 'Sales region' dimension becomes the basis for the group.

```

Sub Example18()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDataSource As Object
    Dim objRegionsDrill As Object
    Dim objCubesByRegion As Object
    Dim objChildCube As Object
    Dim strCategoryCode As String
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "great outdoors sales.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objRegionsDrill = objModel.Dimensions("Sales region").DrillDowns(1)
    objCubesByRegion = objModel.Cubes.Add(xtrObjectType.trCubeGroup)
    With objCubesByRegion
        .GroupDimension = objModel.Dimensions("Sales region")
        .GroupLevel = objRegionsDrill.Levels("Sales region")
        .DetailLevel = objRegionsDrill.Levels("Branch")
        .SummaryLevel = objRegionsDrill.Levels("Sales region")
        .CacheCrossTabs = False
        .CompressMDC = False
        .CubeCreation = xtrCubeCreation.trCubeCreationON
        .MeasureInclude(objModel.Measures("Product cost")) = False
        .Name = "Sales Regions"
        .Optimize = xtrCubeOptimize.trOptimizeDefault
        .Update()
    End With
    'Name each cube in the group after its category name
    For index = 1 To objCubesByRegion.GroupLevel.CategoryCount
        strCategoryCode = objCubesByRegion.GroupLevel.LevelCategories(index).Code
        objChildCube = objCubesByRegion.ChildCubes(strCategoryCode)
        With objChildCube
            .Name = objCubesByRegion.GroupLevel.LevelCategories(index).Name
            .Update()
        End With
    Next
End Sub

```

```

        .SaveAs("great outdoors salesX.mdl")
    .Close()
End With
objChildCube = Nothing
objCubesByRegion = Nothing
objRegionsDrill = Nothing
objDataSource = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Create a Model and Update Properties for a Date Dimension Example

This macro example creates a model and updates several properties associated with the Date dimension.

```

Sub Example19()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDataSource As Object
    Dim objDateDim As Object
    Dim objDateLevel As Object
    Dim strIBMCognos10Location As String
    Dim strDataSource As String
    Dim strDataPath As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strDataSource = "NATIONAL.ASC"
    strDataPath = strIBMCognos10Location & _
        "webcontent\samples\datasources\cubes\PowerCubes\EN\National\"
    & _
        strDataSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    objModel = objTransApp.NewModel
    objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
    With objDataSource
        .LocalPath = strDataPath
        .Name = "National"
        .SetsCurrentPeriod = True
        .SourceType = xtrSourceType.trFlatFileColumnNames
        .Update()
    End With
    With objModel
        .Name = "National"
        .DoAutoDesign()
        .Update()
    End With
    objDateDim = objModel.Dimensions("Date")
    With objDateDim
        .EarliestDate = 19900101
        .GenerateTimePeriod(xtrTimeType.trTimeTypeMonth) = _
            xtrGenerateOptions.trGenerateAll
        .LatestDate = 20101231
        .GenerateDateCategories(19990101, 20101231)
        .WorkingDay(xtrWeekDay.trSunday) = False
        .ManualCurrentPeriod = False
        .Update()
    End With
    objDateLevel = objDateDim.DimensionLevels("Month")
    With objDateLevel
        .DateFormat = "MMM, yyyy"
        .GenerateDateCategories = xtrDateCategoriesGeneration.trGenerateDatesAll
        .Update()
    End With
    With objModel
        .TestBuild(20, True)
        .GenerateCategories()
        MsgBox("Current Period:" + objDateDim.CurrentPeriod.Name)
        .SaveAs("NationalX.mdl")
        .Close()
    End With
    objDateLevel = Nothing
    objDateDim = Nothing
    objDataSource = Nothing
    objModel = Nothing
End Sub

```

```

objTransApp = Nothing
End Sub

```

## Create an Alternate Drill-down Path Example

This example creates an alternate drill-down path in a dimension.

```

Sub Example20()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objLocationsDim As Object
    Dim objLevel As Object
    Dim objAltDrill As Object
    Dim objNewLevel As Object
    Dim objRefSource As Object
    Dim objRefLabel As Object
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "Great outdoors 8.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objLocationsDim = objModel.Dimensions("Sales regions")
    'Remove the Branch Code level and then use it in an alternate
    drill-down path.
    objLocationsDim.DrillDowns(1).Levels("Branch").Delete()
    'Obtain the convergence level and create the alternate drill-down
    path.
    objLevel = objLocationsDim.DrillDowns(1).Levels("Employee")
    With objLevel
        .Unique = True
        .UniqueMove = True
        .Update()
    End With
    objAltDrill = objLevel.CreateAlternateDrillDown
    'Add a new level to the Levels collection of the alternate drill-down
    path.
    objNewLevel = objAltDrill.Levels.Add(xtrObjectType.trLevel)
    'Associate the new level with columns in the data source.
    objRefSource = objModel.DataSources("Sales region (csv)").Columns("Branch code")
    objRefLabel = objModel.DataSources("Sales region (csv)").Columns("Branch city")
    With objNewLevel
        .AssociateWith(objRefSource, xtrAssociationRole.trAssociationSource)
        .AssociateWith(objRefLabel, xtrAssociationRole.trAssociationLabel)
        .BlankSubstitute = "No Value"
        .Inclusion = xtrInclusion.trInclusionGenerate
        .NewCatsLocked = False
        .Name = "Branch"
        .RefreshDescription = True
        .RefreshLabel = True
        .RefreshShortName = True
        .Update()
    End With
    'Move the level and then set the primary drill down.
    With objAltDrill
        .Levels("Branch").Move(1)
        .IsPrimary = True
        .Name = "By Branch"
        .Update()
    End With
    With objModel
        .GenerateCategories()

        .Update()
        .SaveAs("Great outdoors 8X.mdl")
        .Close()
    End With
    objRefLabel = Nothing
    objRefSource = Nothing
    objNewLevel = Nothing

```

```

objAltDrill = Nothing
objLevel = Nothing
objLocationsDim = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Create a Category Count Measure and Add to Model Example

This example creates a category count measure and adds it to an existing model.

```

Sub Example22()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objMeasure As Object
    Dim objLevel As Object
    Dim objActMeasure As Object
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String

    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "great outdoors sales.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objMeasure = objModel.Measures.Add()
    objLevel = objModel.Dimensions("Sales region").DrillDowns(1).Levels("Branch")
    With objLevel
        .Unique = True
        .Update()
    End With
    objActMeasure = objModel.Measures("Quantity")
    objMeasure.ActivityMeasure = objActMeasure
    objMeasure.CategoryCountLevel = objLevel
    With objMeasure
        .AllowCurrencyConversion = False
        .AllowDrillThrough = False
        .Name = "Sales branch count"
        .OutputScale = 0
        .Precision = 0
        .ReverseSign = False
        .StorageType = xtrStorage.trStorageDefault
        .Update()
    End With
    objModel.SaveAs("great outdoors salesX.mdl")
    objModel.Close()
    objActMeasure = Nothing
    objLevel = Nothing
    objMeasure = Nothing
    objModel = Nothing
    objTransApp = Nothing
End Sub

```

## Use the DateWizard to Create a Time Dimension Example

This example creates a new model, adds a data source, and uses the DateWizard object to create a time dimension.

```

Sub Example21()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDataSource As Object
    Dim objColumn As Object
    Dim objDateWizard As Object
    Dim objDateDim As Object
    Dim objDrillDown As Object
    Dim strIBMCognos10Location As String
    Dim strDataSource As String

```

```

Dim strDataPath As String
'Change these paths to match your installation
strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
strDataSource = "NATIONAL.ASC"
strDataPath = strIBMCognos10Location &
    "webcontent\samples\datasources\cubes\PowerCubes\EN\National\"
& _
    strDataSource
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
objModel = objTransApp.NewModel
objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
With objDataSource
    .DecimalPoint = "."
    .LocalPath = strDataPath
    .Name = "National"
    .SourceType = xtrSourceType.trFlatFileColumnNames
    .Update()
End With
objColumn = objDataSource.Columns.Item("Date")
objDateWizard = objModel.DateWizard
With objDateWizard
    .DimensionName = "Dates"
    .DimensionAssociateWith(objColumn, xtrAssociationRole.trAssociationSource)
    .EarliestDate = 19930101
    .EnableTimePeriod(xtrTimeType.trTimeTypeYear) = True
    .EnableTimePeriod(xtrTimeType.trTimeTypeQuarter) = True
    .EnableTimePeriod(xtrTimeType.trTimeTypeMonth) = True
    .EnableTimePeriod(xtrTimeType.trTimeTypeWeek) = False
    .EnableTimePeriod(xtrTimeType.trTimeTypeDay) = False
    .GeneratedDates = False
    .LatestDate = 19941231
    .MonthType = xtrSpecialFunction.trSpecialFunctionMonth
    .QuarterType = xtrSpecialFunction.trSpecialFunctionQuarter
    .WeekAdd = xtrWeekAdd.trWeekAddNone
    .WeekSpan = xtrWeekSpan.trWeekSpanNone
    .WeekStartDay = xtrWeekDay.trMonday
    .WorkingDays(127)
    .WorkingDay(xtrWeekDay.trSunday) = False
    .YearType = xtrSpecialFunction.trSpecialFunctionYear
End With
objDateDim = objDateWizard.CreateDateDimension()
objDateDim.Update()
objDrillDown = objDateDim.DrillDowns(1)
With objDrillDown
    .DrillCode = "By Dates"
    .DrillInclusion = xtrInclusion.trInclusionSuppress

    .WeekSpan = xtrWeekSpan.trWeekSpanSplitMost
    .WeekStart = xtrWeekDay.trMonday
    .Update()
End With
With objModel
    .GenerateCategories()
    .SaveAs("NationalX.mdl")
    MsgBox(.FileName & " " & .Size & " " & .Time)
    .Close()
End With
objDrillDown = Nothing
objDateDim = Nothing
objDateWizard = Nothing
objColumn = Nothing
objDataSource = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Add a Dimension View to a Model Example

This example adds a dimension view to an existing model.

```

Sub Example23()
Dim objTransApp As Object
Dim objModel As Object
Dim objProductsDim As Object
Dim objViewItem As Object
Dim objCube As Object
Dim strIBMCognos10Location As String
Dim strModelPath As String

```



```

Dim strModelSource As String
'Change these paths to match your installation
strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
strModelSource = "great outdoors sales.mdl"
strModelPath = strIBMCognos10Location & _
    "webcontent\samples\models\Transformer8\EN\" & strModelSource
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
With objTransApp
    .DataSourcePath = strIBMCognos10Location & "bin"
    .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
End With
objModel = objTransApp.OpenModel(strModelPath)
objProductsDim = objModel.Dimensions.Item("Products")
objViewItem = objProductsDim.Views.Add()
objViewItem.Apex = objProductsDim.Drilldowns(1).Categories(1)
With objViewItem
    .Name = "OutDoor Products"
    .ViewType = xtrViewType.trViewTypeCustom
    .Update()
End With
objProductsDim.Update()
objCube = objModel.Cubes(1)
objCube.DimensionView(objProductsDim) = objViewItem
objCube.Update()
With objModel
    .SaveAs("great outdoors salesX.mdl")
    .Close()
End With
objCube = Nothing
objViewItem = Nothing
objProductsDim = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Move a Child Category to a Different Parent Example

This example moves a child category to a different parent to reassign one country or region to another region.

**Note:** In some cases, you cannot use a macro to set the properties of a category object. Your macro may appear valid but when you run the 'objCategory =' portion, you receive a message: 'This collection is Read Only in this context.' We recommend using the user interface instead.

```

Sub Example24()
Dim objTransApp As Object
Dim objModel As Object
Dim objCategory As Object
Dim objCatToMove As Object
Dim objCatToReceive As Object
Dim strIBMCognos10Location As String
Dim strModelPath As String
Dim strModelSource As String
'Change these paths to match your installation
strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
strModelSource = "Sales and Marketing.mdl"
strModelPath = strIBMCognos10Location & _
    "webcontent\samples\models\Transformer8\EN\" & strModelSource
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
With objTransApp
    .DataSourcePath = strIBMCognos10Location & "bin"
    .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
End With
objModel = objTransApp.OpenModel(strModelPath)
'Get the Asia Pacific region category.
objCategory = objModel.Dimensions("Retailers").DrillDowns(1).Categories(2)
'Get the Australia category.
objCatToMove = objCategory.ChildCategories(5)
'Get the Americas region category.
objCatToReceive = objModel.Dimensions("Retailers").DrillDowns(1).Categories(1)
'Move Australia to a new region.
objCatToMove.MoveToCategory(objCatToReceive)
objModel.SaveAs("Sales and MarketingX.mdl")
objModel.Close()
objCatToReceive = Nothing
objCatToMove = Nothing
objCategory = Nothing

```

```

objModel = Nothing
objTransApp = Nothing
End Sub

```

## Add a Table to a File and Load Data Example

This example creates a small sample file, EurosX.csv, that contains rates for several European Monetary Union currencies (EMU). It also includes base currency rates for the conversion between the base currency and the euro currency. It then adds a euro table and uses EurosX.csv to load the table with data.

```

Sub Example25()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDataSource As Object
    Dim objEuroTable As Object
    Dim objColumn As Object
    Dim objCurrencyRec As Object
    Dim intX As Integer
    Dim FileNumber As Integer
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    Dim strDataPath As String
    Dim strDataSource As String
    Dim strDataSourceName As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "great outdoors sales.mdl"
    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    strDataSource = "Euro CurrencyX.csv"
    strDataPath = System.IO.Directory.GetCurrentDirectory() &
-
        "\..\..\.." & strDataSource
    strDataSourceName = "Euro Currency"
    'Create Eurofile
    FileNumber = FreeFile()
    FileOpen(FileNumber, strDataPath, OpenMode.Output)
    PrintLine(FileNumber, "EuroDate,EuroCurrency,EuroCode,EuroConRate")
    PrintLine(FileNumber, "0,Austrian Schilling,AUT,13.7603")
    PrintLine(FileNumber, "0,French Franc,FRA,6.55957")
    PrintLine(FileNumber, "20040101,U.S. Dollar,USA,1.2597")
    PrintLine(FileNumber, "20040201,U.S. Dollar,USA,1.2452")
    FileClose(FileNumber)
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    objDataSource = objModel.DataSources.Add(xtrObjectType.trFlatFileDataSource)
    With objDataSource
        .GenerateCategories = False
        .GeneratePowerCube = xtrPowerCubeGeneration.trGenerationNoCreatePowerCubes
        .LocalPath = strDataPath
        .Name = strDataSourceName
        .SourceType = xtrSourceType.trFlatFileColumnNames
        .SetsCurrentPeriod = False
        .Update()
        .Columns("EuroDate").DataClass = xtrDataClass.trDataClassDate
        .Columns("EuroDate").DateDegreeofDetail = xtrDateLevel.trDateLevelMonth
        .Columns("EuroDate").DateInputFormat = xtrDateFormat.trMDY
        .Columns("EuroDate").Update()
    End With
    objEuroTable = objModel.CurrencyTables.Add()

    With objEuroTable
        .CurrencyTableType = xtrCurrencyTableType.trCurrencyTableEuro
        .Update()
    End With
    objColumn = _
        objModel.DataSources(strDataSourceName).Columns("EuroCurrency")
    objEuroTable.AssociateWith(objColumn, xtrAssociationRole.trAssociationLabel)
    objColumn = _
        objModel.DataSources(strDataSourceName).Columns("EuroConRate")
    objEuroTable.AssociateWith(objColumn, xtrAssociationRole.trAssociationRate)

```

```

objColumn = _
objModel.DataSources(strDataSourceName).Columns("EuroCode")
objEuroTable.AssociateWith(objColumn,
    xtrAssociationRole.trAssociationCountryCode)
objColumn = _
objModel.DataSources(strDataSourceName).Columns("EuroDate")
objEuroTable.AssociateWith(objColumn,
    xtrAssociationRole.trAssociationCurrencyDate)
objEuroTable.Update()
objModel.LoadCurrencyTable()
For intX = 1 To objModel.CurrencyRecords.Count
    objCurrencyRec = objModel.CurrencyRecords(intX)
    Select Case objCurrencyRec.CountryCode
        Case "AUT", "FIN", "FRA"
            With objCurrencyRec
                .CurrencyIsEMU = True
                .EMUEntryDate = "19990101"
                .Update()
            End With
        Case Else
            '
    End Select
Next intX
With objModel
    .LoadCurrencyTable()
    .SaveAs("great outdoors salesX.mdl")
    .Close()
End With
objCurrencyRec = Nothing
objColumn = Nothing
objEuroTable = Nothing
objDataSource = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Generate a Time Dimension Based On a Lunar Year Example

This example creates a new model, adds a data source, and generates a time dimension based on a lunar year.

```

Sub Example26()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDataSource As Object
    Dim objColumn As Object
    Dim objDateWizard As Object
    Dim objDateDim As Object
    Dim objDrillDown As Object
    Dim objDateLevel As Object
    Dim strIBMCognos10Location As String
    Dim strDataSource As String
    Dim strDataPath As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strDataSource = "NATIONAL.ASC"
    strDataPath = strIBMCognos10Location & _
        "webcontent\samples\datasources\cubes\PowerCubes\EN\National\"
    & _
        strDataSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    objModel = objTransApp.NewModel
    objDataSource = objModel.DataSources.Add()
    With objDataSource
        .LocalPath = strDataPath
        .Name = "National"
        .SourceType = xtrSourceType.trFlatFileColumnNames
        .Update()
    End With
    objColumn = objDataSource.Columns.Item("Date")
    objDateWizard = objModel.DateWizard
    With objDateWizard
        .DimensionName = "Dates"
        .DimensionAssociateWith(objColumn, xtrAssociationRole.trAssociationSource)
        .EarliestDate = 19900101
        .EnableTimePeriod(xtrTimeType.trTimeTypeMonth) = True
        .EnableTimePeriod(xtrTimeType.trTimeTypeQuarter) = True
        .EnableTimePeriod(xtrTimeType.trTimeTypeYear) = True
    End With
End Sub

```

```

        .LatestDate = 20101231
        .WorkingDays = xtrWeekDay.trMonday + xtrWeekDay.trTuesday
+ -
                xtrWeekDay.trWednesday + xtrWeekDay.trThursday
+ -
                xtrWeekDay.trFriday
    End With
    objDateDim = objDateWizard.CreateDateDimension()
    objDateDim.Update()
    objDrillDown = objDateDim.DrillDowns(1)
    With objDrillDown
        .DrillCode = "By Order Date"
        .DrillInclusion = xtrInclusion.trInclusionSuppress
        .Lunar = True
        .WeekAdd = xtrWeekAdd.trWeekAddDays4
        .WeekStart = xtrWeekDay.trMonday
        .YearStartDay = 19900101
        .Update()
    End With
    objDateLevel = objDateDim.DimensionLevels(1)
    objDateLevel.DateFunction = xtrSpecialFunction.trSpecialFunctionLunarYear
    objDateLevel.Update()
    objDateLevel = objModel.Dimensions("Dates").DimensionLevels(2)
    objDateLevel.DateFunction = xtrSpecialFunction.trSpecialFunctionLunarQuarter
    objDateLevel.Update()

    objDateLevel = objModel.Dimensions("Dates").DimensionLevels(3)
    objDateLevel.DateFunction = xtrSpecialFunction.trSpecialFunctionLunarMonth445
    objDateLevel.Update()
    With objModel
        .GenerateCategories()
        .SaveAs("NationalX.mdl")
        .Close()
    End With
    objDateLevel = Nothing
    objDrillDown = Nothing
    objDateDim = Nothing
    objDateWizard = Nothing
    objColumn = Nothing
    objDataSource = Nothing
    objModel = Nothing
    objTransApp = Nothing
End Sub

```

## Move a Child Category to a Different Parent Category Example

This example moves a child category to a different parent category in the same level.

**Note:** In some cases, you cannot use a macro to set the properties of a category object. Your macro may appear valid but when you run the 'objCategory =' portion, you receive a message: 'This collection is Read Only in this context.' We recommend using the user interface instead.

```

Sub Example27()
    Dim objTransApp As Object
    Dim objModel As Object
    Dim objDimension As Object
    Dim objCategories As Object
    Dim objCategory As Object
    Dim objChildCategory As Object
    Dim objParentCategory As Object
    Dim strIBMCognos10Location As String
    Dim strModelPath As String
    Dim strModelSource As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strModelSource = "Sales and Marketing.mdl"

    strModelPath = strIBMCognos10Location & _
        "webcontent\samples\models\Transformer8\EN\" & strModelSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataSourcePath = strIBMCognos10Location & "bin"
        .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
    End With
    objModel = objTransApp.OpenModel(strModelPath)
    'Get a dimension

```

```

objDimension = objModel.Dimensions("Retailers")
'Get a collection of categories
objCategories = objDimension.DrillDowns(1).Categories
'Get the 5th category
objCategory = objCategories(5)
'Find the Spain child category
For child_category_index = 1 To objCategory.ChildCategories.Count
    objChildCategory = objCategory.ChildCategories(child_category_index)
    If objChildCategory.Name = "Spain" Then
        'Find the Americas category and move the Spain child
category to it
        For parent_category_index = 1 To objCategories.Count
            objParentCategory = objCategories(parent_category_index)
            If objParentCategory.Name = "Americas" Then
                objChildCategory.ConnectWithCategory(objParentCategory)
                Exit For
            End If
        Next
    Exit For
End If
Next
With objModel
    .SaveAs("Sales and MarketingX.mdl")
    .Close()
End With
objParentCategory = Nothing
objChildCategory = Nothing
objCategory = Nothing
objCategories = Nothing
objDimension = Nothing
objModel = Nothing
objTransApp = Nothing
End Sub

```

## Set Attributes for an Application Example

This example demonstrates how to set various attributes of the Application object.

```

Sub Example28()
    Dim objTransApp As Object
    Dim strStartLocation As String
    strStartLocation = System.IO.Directory.GetCurrentDirectory()

    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    With objTransApp
        .DataCharacterSet = xtrCharacterType.trCharDefault
        .DataSourcePath = strStartLocation
        .DataTemporaryFilePath = strStartLocation
        .DefaultDateFormat = xtrPreferences.trDateFormatFromControlPanel
        .DetachDataSource = True
        .EnableMessageLogging = True
        .LogErrorLevel = xtrPreferences.trLogErrorsAndAbove
        .LogFileAppend = True
        .LogFileName = "TrModelsLog.log"
        .LogFilePath = strStartLocation
        .MaxTransactionNumber = 500000
        .ModelsPath = strStartLocation
        .ModelTemporaryFilePath = strStartLocation
        .PowerCubesPath = strStartLocation
        .RowsAsSample = 600
        .RowsChecked = 600
        .SortComparisonRule = xtrPreferences.trSortIgnoreControlPanel
    End With
    objTransApp = Nothing
End Sub

```

## Add and Delete a Package Example

This example opens an existing model, adds a couple of packages, displays the information about each package, deletes one of the packages, and saves the model to a new file.

```

Sub Example29()
    Dim objTransApp As Object
    Dim model As Model
    Dim packages As Packages

```

```

Dim package As Package
Dim new_package As Package
Dim temp_package As Package
Dim path As String
Dim timestamp As String
Dim name As String
Dim index As Integer
Dim strIBMCognos10Location As String
Dim strModelPath As String
Dim strModelSource As String
'Change these paths to match your installation
strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
strModelSource = "Sales and Marketing.mdl"
strModelPath = strIBMCognos10Location & _
    "webcontent\samples\models\Transformer8\EN\" & strModelSource
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
With objTransApp
    .DataSourcePath = strIBMCognos10Location & "bin"
    .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
End With
'Open a package based model
model = objTransApp.OpenModel(strModelPath)
'Create a new package and add it to the model
timestamp = Format(Now, "yyyy-MM-ddTHH:mm:ss.fffZ")

```

```

new_package = model.Packages.Add()
new_package.Name = "GO Data Warehouse (analysis)"
new_package.Path = "/content/package[@name='GO Data Warehouse (analysis)']"
new_package.TimeStamp = "/content/package[@name='GO Data Warehouse (analysis)']"

-
    & "/model[@name='" & timestamp & "']"
new_package.Update()
'Create a second package and add it to the model
timestamp = Format(Now, "yyyy-MM-ddTHH:mm:ss.fffZ")
temp_package = model.Packages.Add()
temp_package.Name = "Temporary Package"
temp_package.Path = "/content/package[@name='Temporary Package']"
temp_package.TimeStamp = "/content/package[@name='Temporary Package']"
Package']"
    & "/model[@name='" & timestamp & "']"
temp_package.Update()
'Iterate through all packages and display information
packages = model.Packages
For index = 1 To packages.Count
    package = packages.Item(index)
    name = package.Name
    path = package.Path
    timestamp = package.TimeStamp
    MsgBox("Package name: " & name & Chr(13) & _
        "Package path: " & path & Chr(13) & _
        "Package time stamp: " & timestamp)
Next
'Delete second package
packages.Remove(temp_package)
model.Update()
'Save the model under a different location
model.SaveAs("Sales and MarketingX.mdl")
model.Close()
objTransApp = Nothing
End Sub

```

## Add and Delete a Report Example

This example opens an existing model, adds a couple of reports, displays the information about each report, deletes one of the reports, and saves the model to a new file.

```

Sub Example30()
Dim objTransApp As Object
Dim model As Model
Dim reports As Reports
Dim report As Report
Dim new_report As Report
Dim temp_report As Report
Dim path As String
Dim timestamp As String
Dim name As String

```

```

Dim index As Integer
Dim strIBMCognos10Location As String
Dim strModelPath As String
Dim strModelSource As String
'Change these paths to match your installation
strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
strModelSource = "Employee expenses.mdl"
strModelPath = strIBMCognos10Location & _
    "webcontent\samples\models\Transformer8\EN\" & strModelSource
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
With objTransApp
    .DataSourcePath = strIBMCognos10Location & "bin"
    .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
End With
'Open a report based model
model = objTransApp.OpenModel(strModelPath)
'Create a new report and add it to the model
timestamp = Format(Now, "yyyy-MM-ddTHH:mm:ss.fffZ")
new_report = model.Reports.Add()
new_report.Name = "TOC report"
new_report.Path = "/content/package[@name='GO Data Warehouse
(query)']"
    & "/folder[@name='Reporting Report Samples']" _
    & "/report[@name='TOC report']"
new_report.TimeStamp = timestamp
new_report.Update()
'Create a second report and add it to the model
timestamp = Format(Now, "yyyy-MM-ddTHH:mm:ss.fffZ")
temp_report = model.Reports.Add()
temp_report.Name = "Temporary Report"
temp_report.Path = "/content/package[@name='Temporary Package']"
-
    & "/report[@name='Temporary Report']"
temp_report.TimeStamp = timestamp
temp_report.Update()
'Iterate through all reports and display information
reports = model.Reports
For index = 1 To reports.Count
    report = reports.Item(index)
    name = report.Name

    path = report.Path
    timestamp = report.TimeStamp
    MsgBox("Report name: " & name & Chr(13) & _
        "Report path: " & path & Chr(13) & _
        "Report time stamp: " & timestamp)
Next
'Delete second package
reports.Remove(temp_report)
model.Update()
'Save the model under a different location
model.SaveAs("Employee expensesX.mdl")
model.Close()
objTransApp = Nothing
End Sub

```

## Create a Query Example

This example demonstrates how to create a query in the context of creating a small model.

```

Sub Example31()
Dim objTransApp As Object
Dim model As Model
Dim new_package As Package
Dim new_query As Query
Dim new_column1 As Column
Dim new_column2 As Column
Dim new_column3 As Column
Dim measure As Measure
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
'Create a new model
model = objTransApp.NewModel
'Create a new package and add it to the model
new_package = model.Packages.Add()
With new_package
    .Name = "GO Data Warehouse (query)"
    .Path = "/content/package[@name='GO Data Warehouse (query)']"
    .Update()

```

```

End With
'Create a new query and add it to the package
new_query = new_package.Queries.Add()

With new_query
    .Name = "Sales Revenue"
    .AutoSummary = True
    .MaximizeSpeed = True
    .SetsCurrentPeriod = True
    .Update()
End With
'Create 3 new columns and add them to the query
new_column1 = new_query.Columns.Add()
With new_column1
    .Name = "Date"
    .OriginalName = "[Sales (query)].[Time dimension].[Date]"
    .Update()
End With
new_column2 = new_query.Columns.Add()
With new_column2
    .Name = "Region"
    .OriginalName = "[Sales (query)].[Retailer site].[Region]"
    .Update()
End With
new_column3 = new_query.Columns.Add()
With new_column3
    .Name = "Revenue"
    .OriginalName = "[Sales (query)].[Sales fact].[Revenue]"
    .Update()
End With
'Add Revenue column as a measure
measure = model.Measures.Add()
With measure
    .AssociateWith(new_column3, xtrAssociationRole.trAssociationSource)
    .Update()
End With
model.DoAutoDesign()
' See if there is any columns that are mismatched
If new_query.IsAnyColumnMismatched = True Then
    MsgBox("There is at least one column that is mismatched.")
    For index = 1 To model.CheckModel.Count
        MsgBox(model.CheckModel.Item(index).Name)
    Next index
End If
'Save the model
With model
    .SaveAs("GO Data Warehouse (query).mdl")
    .Close()
End With
objTransApp = Nothing
End Sub

```

## Create and Delete Filters for a Model Example

This example demonstrates how to create and delete filters in the context of creating a small model.

```

Sub Example32()
    Dim objTransApp As Object
    Dim model As Model
    Dim new_package As Package
    Dim new_query As Query
    Dim new_column As Column
    Dim new_filter As Filter
    Dim filters As Filters
    Dim current_filter As Filter
    Dim measure As Measure
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    'Create a new model
    model = objTransApp.NewModel
    'Create a new package and add it to the model
    new_package = model.Packages.Add()
    With new_package
        .Name = "GO Data Warehouse (query)"
        .Path = "/content/package[@name='GO Data Warehouse (query)']"
        .Update()
    End With
    'Create a new query and add it to the package
    new_query = new_package.Queries.Add()

```



```

With new_query
    .Name = "Sales Revenue"
    .AutoSummary = True
    .MaximizeSpeed = True
    .SetsCurrentPeriod = True
    .Update()
End With
'Create 3 new columns and add them to the query
new_column = new_query.Columns.Add()
With new_column
    .Name = "Date"
    .OriginalName = "[Sales (query)].[Time dimension].[Date]"
    .Update()
End With
new_column = new_query.Columns.Add()
With new_column
    .Name = "Region"
    .OriginalName = "[Sales (query)].[Retailer site].[Region]"
    .Update()
End With
new_column = new_query.Columns.Add()
With new_column
    .Name = "Revenue"
    .OriginalName = "[Sales (query)].[Sales fact].[Revenue]"
    .Update()
End With
'Create 3 new filters
new_filter = new_query.Filters.Add()
With new_filter
    .Name = "2004"
    .RefName = "[go_data_warehouse].[2004]"
    .Update()
End With
new_filter = new_query.Filters.Add()
With new_filter
    .Name = "2005"
    .RefName = "[go_data_warehouse].[2005]"
    .Update()
End With
new_filter = new_query.Filters.Add()
With new_filter
    .Name = "GO Asia Pacific"
    .RefName = "[go_data_warehouse].[GO Asia Pacific]"
    .Update()
End With
'Add Revenue column as a measure
measure = model.Measures.Add()
With measure
    .AssociateWith(new_column, xtrAssociationRole.trAssociationSource)
    .Update()
End With
'Delete the 2005 filter
filters = new_query.Filters
For index = 1 To filters.Count
    current_filter = filters.Item(index)
    If current_filter.Name = "2005" Then
        current_filter.Delete()
    Exit For
End If
Next
model.DoAutoDesign()
'Save the model
With model
    .SaveAs("GO Data Warehouse (query).X.mdl")
    .Close()
End With
objTransApp = Nothing
End Sub

```

## Create a Single-valued Prompt Example

This example demonstrates how to create single-valued prompts in the context of creating a small model.

```

Sub Example33()
    Dim objTransApp As Object
    Dim model As Model
    Dim new_report As Report

```

```

Dim new_query As Query
Dim new_column As Column
Dim new_prompt As Prompt
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
'Create a new model
model = objTransApp.NewModel
'Create a new report and add it to the model
new_report = model.Reports.Add()
new_report.Name = "Date Between List"
new_report.Path = "/content/package[@name='GO Data Warehouse
(query)']"
    & "/folder[@name='SDK Report Samples']" _
    & "/report[@name='Date Between List']"
new_report.Update()
'Create a new query and add it to the report
new_query = new_report.Queries.Add()
With new_query
    .Name = "Date Between List (Has Prompt)"
    .AutoSummary = False
    .MaximizeSpeed = True
    .SetsCurrentPeriod = True
    .Update()
End With
'Create 3 new columns and add them to the query
new_column = new_query.Columns.Add()
With new_column
    .Name = "Date"
    .OriginalName = "[Report].[Query1.0].[Date]"
    .Update()
End With
new_column = new_query.Columns.Add()
With new_column
    .Name = "Order number"
    .OriginalName = "[Report].[Query1.0].[Order number]"
    .Update()
End With
new_column = new_query.Columns.Add()
With new_column
    .Name = "Order method"
    .OriginalName = "[Report].[Query1.0].[Order method]"
    .Update()
End With
'Create 2 new Single-valued prompts and add them to the query
new_prompt = new_query.Prompts.Add()
With new_prompt
    .PromptValueType = xtrPromptValueType.trSingleValuePrompt
    .Name = "StartDate"
    .Type = "Date Time (1999-01-31 18:30:00)"
    .Value = "2005-01-01"
    .Update()
End With
new_prompt = new_query.Prompts.Add()
With new_prompt
    .PromptValueType = xtrPromptValueType.trSingleValuePrompt
    .Name = "EndDate"
    .Type = "Date Time (1999-01-31 18:30:00)"
    .Value = "2006-12-31"
    .Update()
End With
'Save the model
With model
    .SaveAs("GO Data Warehouse (query)X.mdl")
    .Close()
End With
objTransApp = Nothing
End Sub

```

## Create a Multi-valued Prompt Example

This example demonstrates how to create a multi-valued prompt in the context of creating a small model.

```

Sub Example34()
Dim objTransApp As Object
Dim model As Model

```

```

Dim new_report As Report
Dim new_query As Query
Dim new_column As Column
Dim new_prompt As Prompt
Dim list_of_values As String = ""
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
'Create a new model
model = objTransApp.NewModel
'Create a new report and add it to the model
new_report = model.Reports.Add()
new_report.Name = "Product Method Date List"
new_report.Path = "/content/package[@name='GO Data Warehouse
(query)']"
    & "/folder[@name='SDK Report Samples']" _
    & "/report[@name='Product Method Date List']"
new_report.Update()
'Create a new query and add it to the report
new_query = new_report.Queries.Add()
With new_query
    .Name = "Product Method Date List (Has Prompt)"
    .AutoSummary = False
    .MaximizeSpeed = True
    .SetsCurrentPeriod = True
    .Update()
End With

```

```

'Create 3 new columns and add them to the query
new_column = new_query.Columns.Add()
With new_column
    .Name = "Date"
    .OriginalName = "[Report].[Query1.0].[Date]"
    .Update()
End With
new_column = new_query.Columns.Add()
With new_column
    .Name = "Product Name"
    .OriginalName = "[Report].[Query1.0].[Product name]"
    .Update()
End With
new_column = new_query.Columns.Add()
With new_column
    .Name = "Order method"
    .OriginalName = "[Report].[Query1.0].[Order method]"
    .Update()
End With
'Create a new Multi-valued prompt and add it to the query
new_prompt = new_query.Prompts.Add()
With new_prompt
    .PromptValueType = xtrPromptValueType.trMultiValuePrompt
    .Name = "MP"
    .Type = "String"
    .Value = "Fax"
    .Value = "Mail"
    .Value = "Telephone"
    .Value = "Web"
    .Update()
End With
'List all the prompt values
For index = 1 To new_prompt.ValuesCount
    new_prompt.CurrentValueIndex = index
    If list_of_values <> "" Then
        list_of_values = list_of_values & ", " & new_prompt.Value
    Else
        list_of_values = new_prompt.Value
    End If
Next

```

```

MsgBox("These are the prompt values: " & list_of_values)
'Save the model
With model
    .SaveAs("GO Data Warehouse (query)X.mdl")
    .Close()
End With
objTransApp = Nothing
End Sub

```

## Create a New Model and Publish a PowerCube Example

This example uses the NATIONAL.ASC data source to create a new model and publish a PowerCube to the server.

```
Sub Example35()
    Dim objTransApp As Object
    Dim model As Model
    Dim cube As Cube
    Dim dataSource As FlatFileDataSource
    Dim strIBMCognos10Location As String
    Dim strDataSource As String
    Dim strDataPath As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strDataSource = "NATIONAL.ASC"
    strDataPath = strIBMCognos10Location & _
        "webcontent\samples\datasources\cubes\PowerCubes\EN\National\"
    & _
        strDataSource
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    objTransApp.Logon("Cognos", "", "")
    'Create new model and add flat file data source
    model = objTransApp.NewModel
    dataSource = model.DataSources.Add(xtrObjectType.trFlatFileDataSource)
    With dataSource
        .GenerateCategories = True
        .GeneratePowerCube = xtrPowerCubeGeneration.trGenerationDefault
        .CharacterType = xtrCharacterType.trCharAnsiSingleByte
        .External = False
        .FieldSeparator = ","
        .LocalPath = strDataPath
        .MaximizeSpeed = True
        .Name = "National"
        .SourceType = xtrSourceType.trFlatFileColumnNames
        .SetsCurrentPeriod = True
        .Update()
    End With
    'Do Autodesign and make National cube
    With model
        .Name = "National"
        .DoAutoDesign()
        .TestBuild(20, True)
        .Update()
    End With
    'Publish the National Cube
    cube = model.Cubes.Item("National")
    With cube
        .MDCFile = "c:\National.mdc"
        .Update()
        .CreateMDCFile()
        .PublishPackage(True, True) 'Re-publish both Datasource
    & Package
    End With
    'Save model
    With model
        .SaveAs("NationalX.mdl")
        .Close()
    End With
    objTransApp.Logoff()
    objTransApp = Nothing
End Sub
```

## Copy and Activate a PowerCube Example

This example demonstrates how to use the copy and activate feature.

```
Sub Example36()
    Dim objTransApp As Object
    Dim model As Model
    Dim cube As Cube
    Dim dataSource As FlatFileDataSource
    Dim strIBMCognos10Location As String
    Dim strDataSource As String
    Dim strDataPath As String
```

```

'Change these paths to match your installation
strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
strDataSource = "NATIONAL.ASC"
strDataPath = strIBMCognos10Location & _
    "webcontent\samples\datasources\cubes\PowerCubes\EN\National\"
& _
    strDataSource
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
objTransApp.Logon("Cognos", "", "")
'Create new model and add flat file data source
model = objTransApp.NewModel
dataSource = model.DataSources.Add(xtrObjectType.trFlatFileDataSource)
With dataSource
    .GenerateCategories = True
    .GeneratePowerCube = xtrPowerCubeGeneration.trGenerationDefault
    .CharacterType = xtrCharacterType.trCharAnsiSingleByte
    .External = False
    .FieldSeparator = ","
    .LocalPath = strDataPath
    .MaximizeSpeed = True
    .Name = "National"
    .SourceType = xtrSourceType.trFlatFileColumnNames
    .SetsCurrentPeriod = True
    .Update()
End With

```

```

'Do Autodesign and make National cube
With model
    .Name = "National"
    .DoAutoDesign()
    .TestBuild(20, True)
    .Update()
End With
'Copy on Activate the National cube
cube = model.Cubes.Item("National")
With cube
    .MDCFile = "c:\National.mdc"
    .Update()
    .CreateMDCFile()
    .SetDeployType(xtrDeployType.trDeployType_SWAPSINGLE)
    .AddDeployLocation("c:\NATIONAL\Deployment1")
    .AddDeployLocation("c:\NATIONAL\Deployment2")
    .AddDeployLocation("c:\NATIONAL\Deployment3")
    .DataSourceWindowsLocation = "c:\NATIONAL\Deployment1\National.mdc"
    .Update()
    .DeployCube()
    .PublishDatasource(True) 'Republish PowerCube if it already
exists
    .ClearDeployLocations() 'Remove the deployment locations
from the model
    .Update()
End With
'Save model
With model
    .SaveAs("NationalX.mdl")
    .Close()
End With
objTransApp.Logoff()
objTransApp = Nothing
End Sub

```

## Create a Model Using a Signon and an IQD Data Source Example

This example demonstrates how to create a model with a signon and an IQD data source.

```

Sub Example37()
    Dim objTransApp As Object
    Dim model As Model
    Dim datasource As IqdDataSource
    Dim signon As Signon
    Dim strIBMCognos10Location As String
    Dim strDataSource As String
    Dim strDataPath As String
    'Change these paths to match your installation
    strIBMCognos10Location = "C:\Program Files\IBM\Cognos\c10\"
    strDataSource = "prod.iqd"
    strDataPath = strIBMCognos10Location & _
        "webcontent\samples\datasources\cubes\PowerCubes\EN\great

```

```

outdoors sales\" & _
    strDataSource
objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
With objTransApp
    .DataSourcePath = strIBMCognos10Location & "bin"
    .TransdaPath = strIBMCognos10Location & "CS7Gateways\bin\TransDa.exe"
End With
model = objTransApp.NewModel
'Create signon before adding the data source
signon = model.Signons.Add()
With signon
    .Name = "great_outdoors_warehouse"
    .Description = "Signon used for prod.iqd data source"
    .UserID = "sa"
    .Password = "sa"
    .PromptForPassword = False
    .SignonType = xtrSignonType.trDataSourceSignon
    .Update()
End With

```

```

'Add IQD data source
datasource = model.DataSources.Add(xtrObjectType.trIqdDataSource)
With datasource
    .GenerateCategories = True
    .GeneratePowerCube = xtrPowerCubeGeneration.trGenerationDefault
    .IsolationLevel = 0
    .LocalPath = strDataPath
    .Name = "Products"
    .SetsCurrentPeriod = True
    .SourceType = xtrSourceType.trQuery
    .Update()
End With
'Auto Design, build cube, and save model
With model
    .Name = "Products"
    .DoAutoDesign()
    .TestBuild(20, True)
    .Update()
    .SaveAs("ProductsX.mdl")
    .Close()
End With
objTransApp = Nothing
End Sub

```

## Create a Model Using a Signon and Package Data Source Example

This example demonstrates how to create a model with a signon and a package data source.

```

Sub Example38()
    Dim objTransApp As Object
    Dim model As Model
    Dim package As Package
    Dim query As Query
    Dim column As Column
    Dim signon As Signon
    Dim connection As PackageDatasourceConnection
    Dim measure As Measure
    objTransApp = CreateObject("IBMCognosTransformer.ApplicationCtrl.1")
    'Create a new model
    model = objTransApp.NewModel
    'Create a new package and add it to the model
    package = model.Packages.Add()
    With package
        .Name = "GO Data Warehouse (query)"
        .Path = "/content/package[@name='GO Data Warehouse (query)']"
        .Update()
    End With

    'Create signon before adding the PackageDatasourceConnection
    signon = model.Signons.Add()
    With signon
        .Name = "great_outdoors_warehouse"
        .Description = "Signon used for package data source"
        .SignOnNamespace = "Cognos"
        .UserID = "sa"
        .Password = "sa"
        .SignonType = xtrSignonType.trCognosSignon
    End With

```

```

        .AutoLogon = True
        .Update()
    End With
    'Add PackageDataSourceConnection
    connection = package.PackageDataSourceConnections.Add()
    With connection
        .DataSource = "great_outdoors_warehouse"
        .Connection = "great_outdoors_warehouse"
        .Signon = "great_outdoors_warehouse"
        .TransformerSignon = "great_outdoors_warehouse"
        .AlwaysUseTransformerSignon = True
        .Update()
    End With
    'Create a new query and add it to the package
    query = package.Queries.Add()
    With query
        .Name = "Sales Revenue"
        .AutoSummary = True
        .MaximizeSpeed = True
        .SetsCurrentPeriod = True
        .Update()
    End With

```

```

    'Create 3 new columns and add them to the query
    column = query.Columns.Add()
    With column
        .Name = "Date"
        .OriginalName = "[Sales (query)].[Time dimension].[Date]"
        .Update()
    End With
    column = query.Columns.Add()
    With column
        .Name = "Region"
        .OriginalName = "[Sales (query)].[Retailer site].[Region]"
        .Update()
    End With
    column = query.Columns.Add()
    With column
        .Name = "Revenue"
        .OriginalName = "[Sales (query)].[Sales fact].[Revenue]"
        .Update()
    End With
    'Add Revenue column as a measure
    measure = model.Measures.Add()
    With Measure
        .AssociateWith(column, xtrAssociationRole.trAssociationSource)
        .Update()
    End With
    'Auto Design, build cube, and save model
    With model
        .Name = "Sales Revenue"
        .DoAutoDesign()
        .TestBuild(20, True)
        .Update()
        .SaveAs("GO Data Warehouse (query)X.mdl")
        .Close()
    End With

```

```

    objTransApp = Nothing
End Sub

```





## Notices

---

This information was developed for products and services offered worldwide.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's

- name
- user name
- password

for purposes of

- session management
- authentication
- enhanced user usability
- single sign-on configuration
- usage tracking or functional purposes other than session management, authentication, enhanced user usability and single sign-on configuration

These cookies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at <https://www.ibm.com/privacy/us/en/>.

---

# Index

## A

Add method [108](#)

adding

collections [108](#)

AllocationMeasure property [173](#)

## C

collections

adding objects [108](#)





