Copyright Notice

Information in this document is subject to change without notice and does not represent a commitment on the part of *Data Dynamics, Ltd*. The software described in this document is furnished under a license agreement or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or nondisclosure agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of *Data Dynamics, Ltd.*

Copyright © 1995, 1996, 1997 *Data Dynamics, Ltd.*
All rights reserved.

All trade names referenced herein are either trademarks or registered trademarks of their respective companies.

Printed in the United States of America.

This manual was produced using Microsoft Word for Windows.

DynamiCube (Version 2.0)

Data Dynamics, Ltd.
2600 Tiller Lane
Columbus, OH 43231
Tel: 614.895.3142
Fax: 614.899.2943
http://www.datadynamics.com

What's New

- ***Release 2.0***

  Does not rely on any external DLLs; No MFC DLLs

  DynamiCube is now a lightweight control. As such, it now provides a dual interface and has a size of ~350K.

  Page Fields

  You can drag a dimension from a row or column and place in the page field area and it will not be displayed in the current view until it is dragged back to a row or column orientation.

  microCube file size is 4-6 times smaller due to compression; because of this compression, Version 1.x applications can not read Version 2.0 microCube files.

  Speed Improved 10-25%

  UNICODE international Character support

  Print Preview

  Has been significantly enhanced with zoom and better printer font handling.

  Note: The following Version 2.0 modifications may require changes to your existing applications:

  Attrib

  Attrib object, added for increased performance, is used instead of FetchAttributs' variables. To set or retrieve the attributes of a cell, you will simply modify or read the properties of the Attrib object.

  HitRecord

  HitRecord object has been added for increased functionality when detecting a user click event. HitRecord object is used instead of GetCellForPoint to return the cell address that contains a point address of (x,y) in row and column format.

- ***Release 1.41***

  Data Fields Ranking

  Sorting items based on a data field's total. Row or Column items can be ranked ascending or descending on any of the data field's values using RankOn and RankStyle Properties.

  Enhanced Print and Print Preview

  The print preview has been enhanced with interactive column width and margin sizing. Better font scaling algorithm is used to remove overlapping column headings.

About DynamiCube

The DynamiCube is a method of retrieving and displaying huge amounts of data stored in relational databases in a summarized and tabulated form that aids the end user in making accurate decisions. Numeric data can be pulled from multiple sources, tabulated, summarized and presented to the user in a multi-dimensional cube. The end user can manipulate the view, move dimensions, change the order, add fields, and change calculation parameters and aggregate functions to better answer their analysis needs.

The DynamiCube is n-dimensional and is only limited by the available resources of the operating environment. Each dimension consists of a finite number of data points. These data points are used to locate an individual data element.

Here is a simple example of data represented in a relational table view. In the following example, we have revenue and cost data for each division in each region. Suppose your company has five divisions (Software, Hardware, Consulting, Training, and Service) that operate in four regions (East, West, North, and South). Here is how your table view would look:

Division	**Region**	**Revenue**
Hardware	East	120
Hardware	West	150
Hardware	North	195
Hardware	South	165
Software	East	200
Software	West	185
Software	North	130
Software	South	134
Consulting	East	800
Consulting	West	536
Consulting	North	758
Consulting	South	436
Training	East	180
Training	West	152
Training	North	195
Training	South	135
Service	East	254
Service	West	354
Service	North	129
Service	South	284

The DynamiCube reformats the above table to a clearer two-dimensional view:

	East	West	North	South
Hardware	120	150	195	165
Software	200	185	130	134
Consulting	800	536	758	436
Training	180	152	195	135

Service	254	354	129	284

The revenue data is a two dimensional view, one dimension is product and the other is the region where the product is sold. The DynamiCube view provides a simpler and better focus on data than the relational representation. An expanded view would show row and column totals providing a quick summary of revenue by product and region.

	East	West	North	South	Total
Hardware	120	150	195	165	630
Software	200	185	130	134	649
Consulting	800	536	758	436	2530
Training	180	152	195	135	662
Service	254	354	129	284	1021
Total	1554	1377	1407	1154	5492

The DynamiCube can be expanded to include more dimensions and more summary information. It provides decision support information in a flexible format that end users can manipulate and modify according to their analytical needs.

DynamiCube Features

- High-speed, graphical navigation of multi-dimensional data.
- Optimized 32-bit Lightweight ActiveX Control.
- Works with Visual Basic 4.0 and 5.0 Professional and Enterprise Editions, as well as Visual C++, Delphi, Power Builder, Optima++, Microsoft IE 3.0 and Microsoft Access.
- Works with Microsoft® Windows 95 and Microsoft® Windows NT.
- Supports unbound or bound data through ODBC, DAO, RDO and BDE.
- microCube files can be saved and later retrieved for use in email or presentations
- Data Points limited only by the available resources of the operating environment.
- Time-series fields (Year, Quarter, Month) based on your date fields.
- 2D and 3D appearance.
- Variable properties for data area display such as alignment, format mask, etc.
- Developer control of availability of fields at run-time.
- Page Fields and Show/Hide detail items for concise summary views.
- Fetch attributes event to control the appearance of individual cells.
- Users can highlight data according to their own criteria (negative variances, etc.)
- Sample source code included to easily add graphing
- Support for multiple aggregate functions (Sum, Average, Count, …)
- Support for custom calculation at a single data point level.
- Design time heading text definition.
- Automatic size-to-fit of columns and rows.
- Runtime access to properties and layout definitions.
- Row and column data element filters.
- Full run-time view customization and data pivoting.
- Bitmap images can be embedded into row and column headings.
- Provides print/print preview/zoom functions

Before You Run Setup

Please take a few minutes before you install DynamiCube to do the following:

Read the README.TXT File

If there are any corrections or additions to this book, they will be listed in a file called README.TXT. This file can be displayed directly from the installation diskette using the Windows Notepad utility. After the installation, this file can be read by double-clicking the DynamiCube ReadMe icon in the DynamiCube program group.

Running Setup

When you run the Setup program (SETUP.EXE) to install DynamiCube on your computer, you will set a path for DynamiCube.

To start the setup program:

1. Place the DynamiCube installation diskette in your disk drive.
2. Click the Start button on the Task Bar.
3. Select the Run. Menu option.
4. Type <d>:SETUP.EXE, where <d> is the drive where the installation diskette is placed.
5. Follow the setup instructions displayed on your screen.

What was installed?

The install program will copy the control file DCUBE.OCX and DCUBEPP.DLL, an ODBC support DLL DYNAODBC.DLL, a BDE Support DLL DYNABDE.DLL and the license file DCUBE.LIC to your Windows System directory. All other files (Online Help and Samples) will be placed in the directory you selected during installation. For a description of these files see ‘Appendix B. Included Files.’

Including DynamiCube Control in Your Project

In Visual Basic, custom controls are included on a project basis. Once a custom control is included in a project and saved, it does not have to be included again. The control icon will appear in the Visual Basic toolbox.

To include the DynamiCube control in your project:

1. Start Visual Basic and open your project or start a new project.
2. Select Custom Controls from the Tools menu (Ctrl-T shortcut key)
3. Select the DynamiCube Control by checking the checkbox next to it in the custom control list.
4. If the DynamiCube is not in the list make sure that the Show | Controls checkbox is checked.
5. Once you click OK, you should see the DynamiCube icon in your toolbox.

Using the DynamiCube in Your Project

Once you have included the DynamiCube control in your project, its icon will appear in your Visual Basic toolbox. To use the control on one of your forms:

1. Click on the control icon in Visual Basic's toolbox.
2. Click and drag the mouse on the form to size the control on the form.

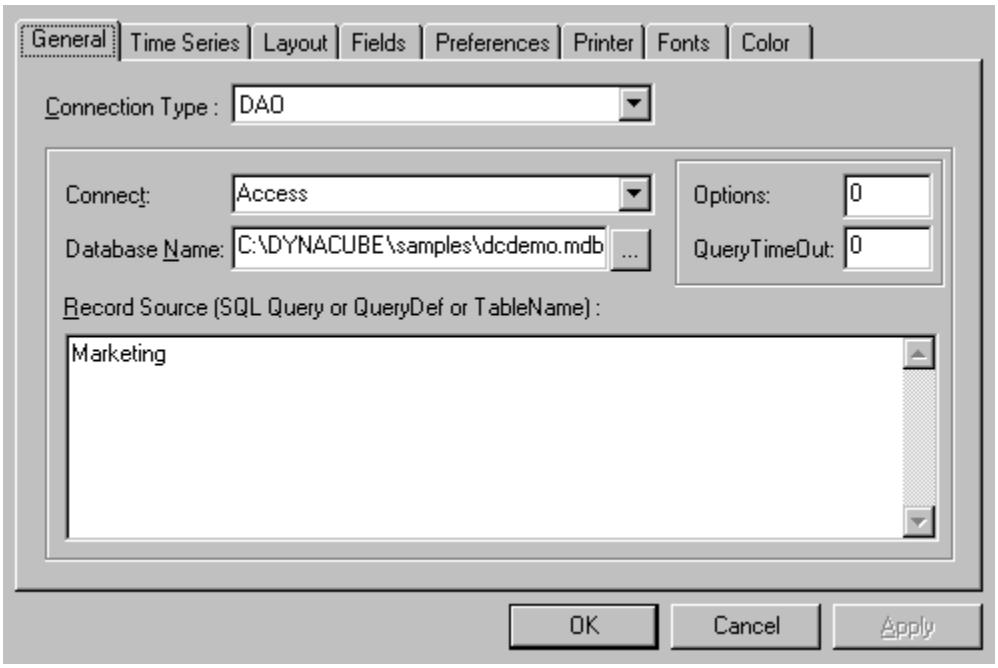
Or

1. Double-click on the control icon in Visual Basic's toolbox. The control will be placed on your form.
2. Size the control by dragging one of its borders.

Property Sheets

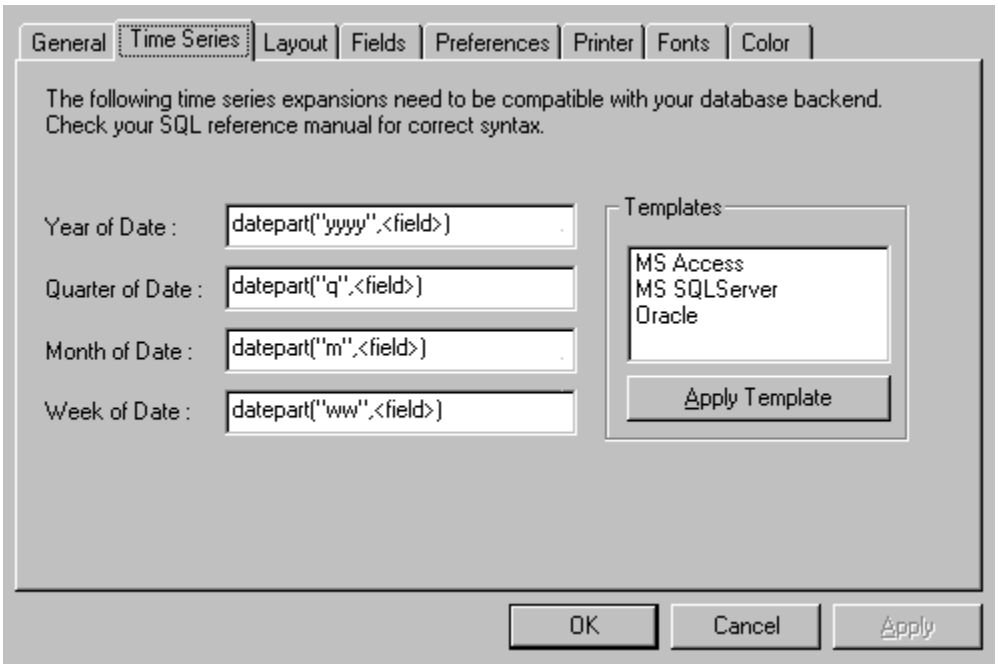
The DynamiCube property sheets offer an easy to use layout designer and access to DynamiCube properties.

General



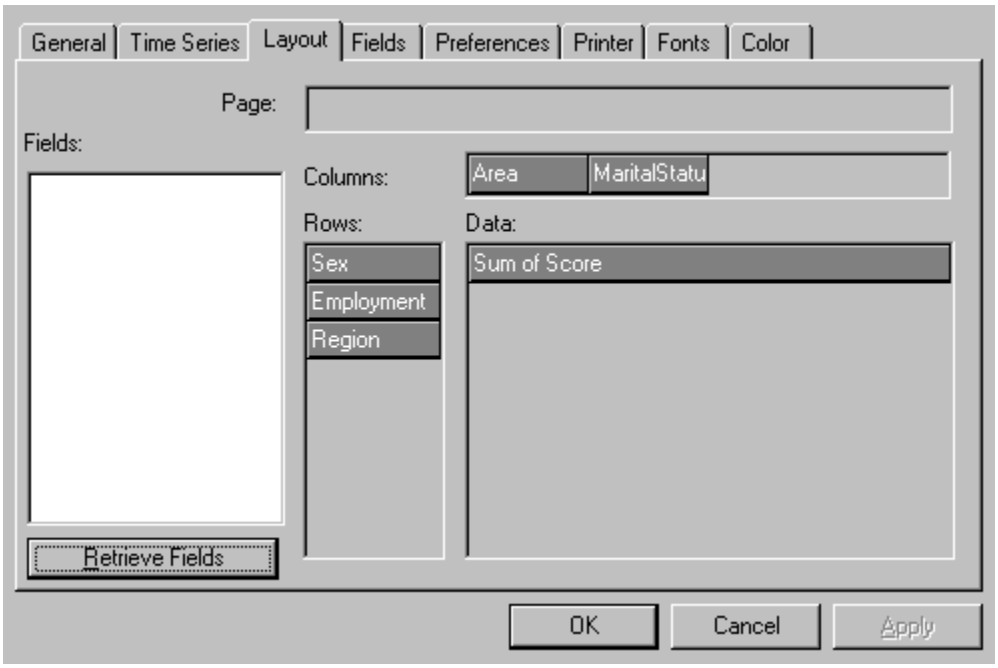
The General property sheet used to set properties relating to database connection and record source setup.

Time Series



The Time Series property sheet is used to customize the date part extraction function syntax. It defaults to the JET DatePart function syntax. If you need to set the syntax to a different template, it MUST be done before adding any fields in the Layout Property Sheet. The setting is used when placing the fields in the layout area and it cannot be reset, you will have to delete the fields and redo the layout.

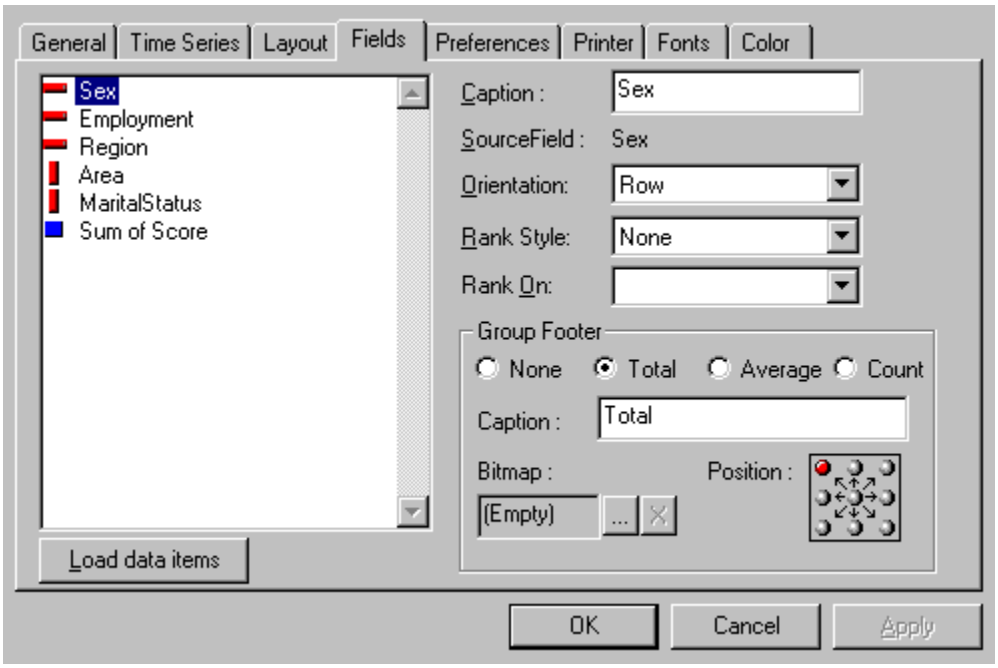
Layout



The layout design property sheet allows to set up the initial view of the DynamiCube. You can drag fields from the fields list and drop them in the area where they would appear in the view at run-time.

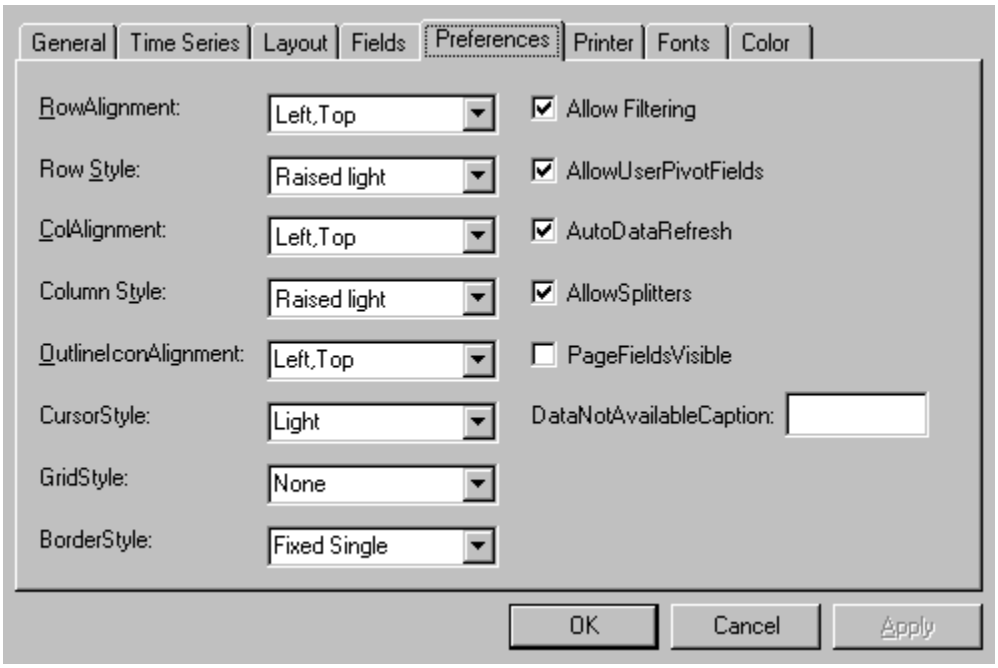
You can access data fields properties by right clicking on the field in the data area and selecting properties from the popup menu.

Fields



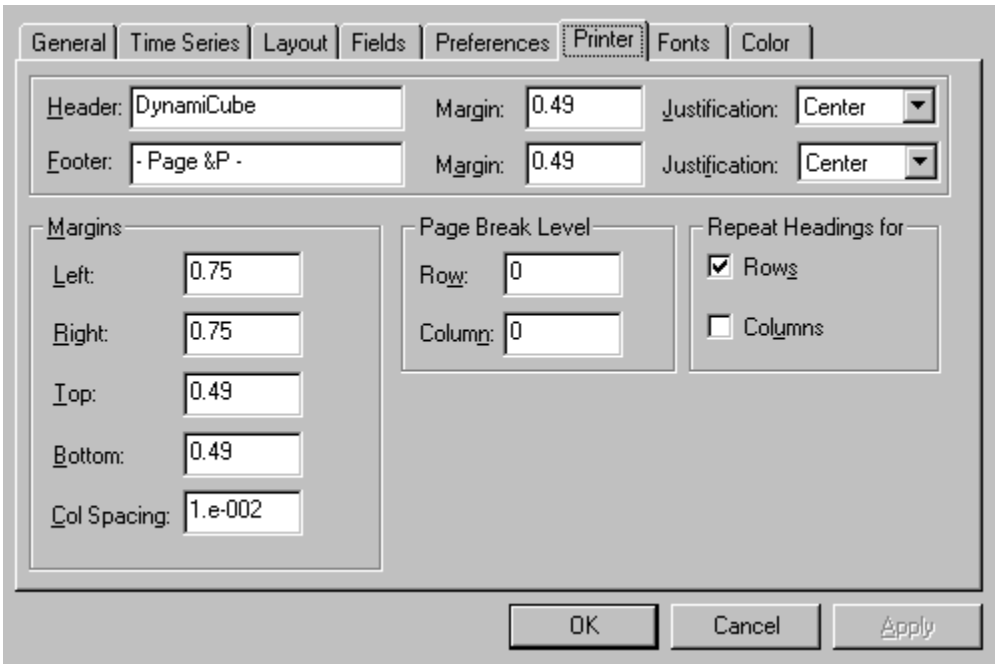
The fields property sheet allows you to set up properties of individual fields in the record source. You can setup caption, icons, and subtotal calculation. Fields data items can be loaded by clicking load data items button, which allows you to change individual items properties by clicking on the item in the list box.

Preferences



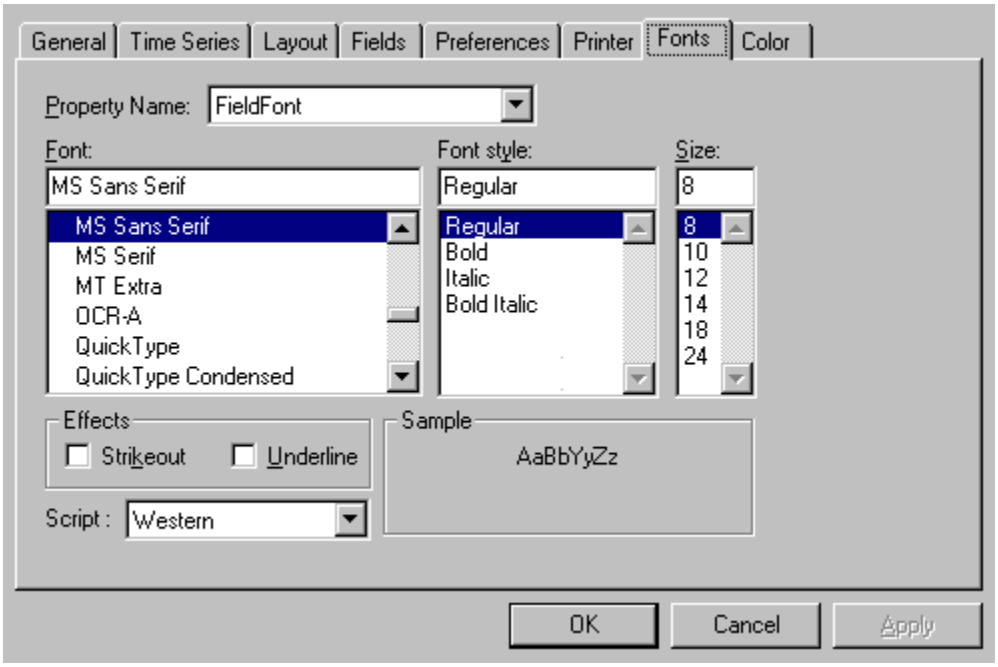
Preferences property sheet displays the style and alignment properties of row and column fields.

Print



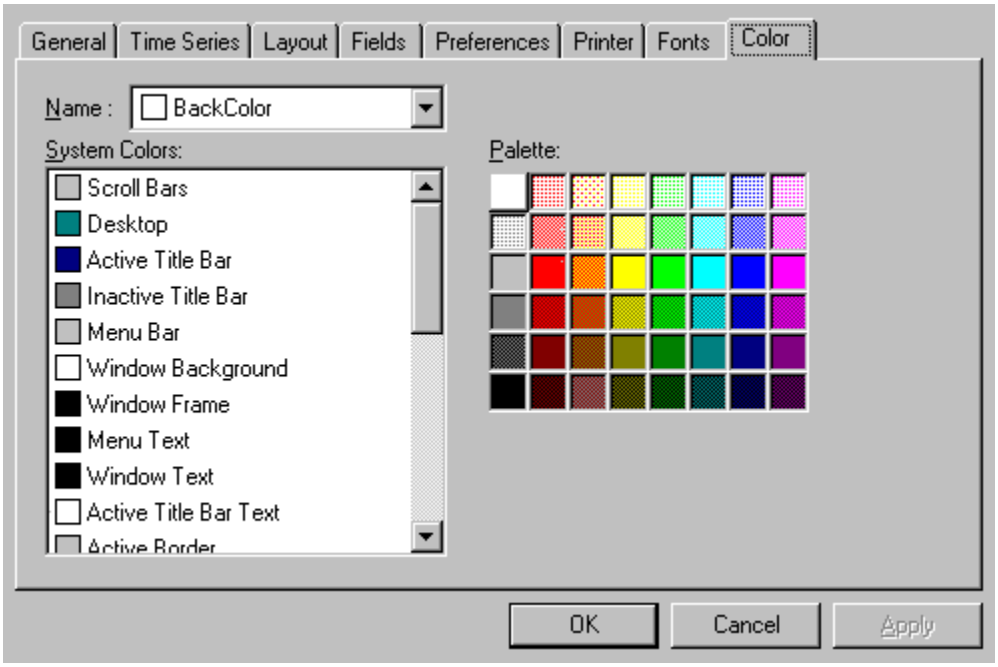
Print property sheet allows you to set page header, footer properties and page margins.

Fonts



Fonts property sheet allows you to set up fonts of various areas in the cube.

Colors



Colors property sheet allows you to setup colors of various areas in the cube.

Using Help

An electronic version of this manual is included in the installation directory of the DynamiCube. You can access it from the DynamiCube program group. Or by Pressing F1 when the DynamiCube Control is selected from the toolbox.

Online help has a comprehensive index of all keywords and topics in this manual. To save time, you can cut and paste sample code from the online help.

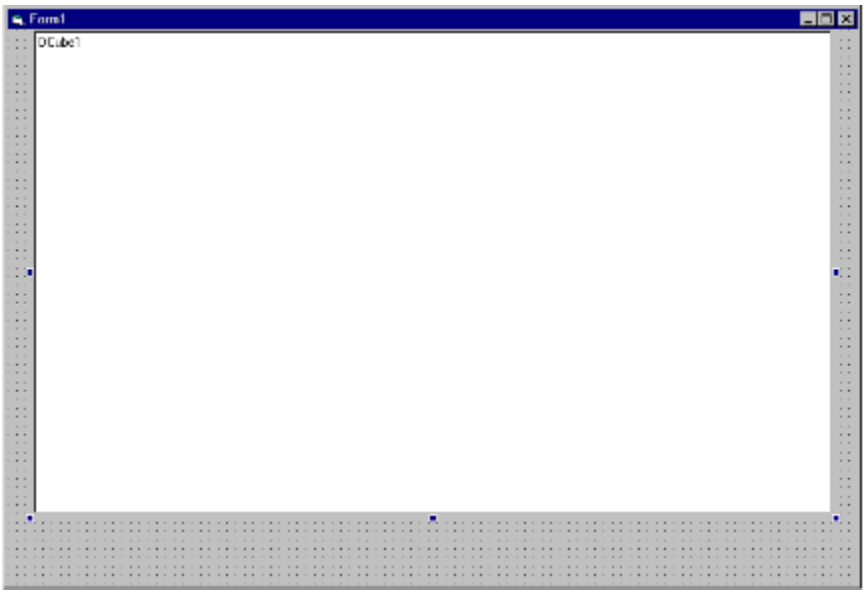
Guided Tour

This chapter guides you through creating your first DynamiCube view. We have included a Microsoft® Access™ database file, which contains a simple flat file of product sales data over a 3-year period across different regions. We will build a VB form that uses the DynamiCube to provide data analysis on this table. The form will use the built-in data control to connect to the database.

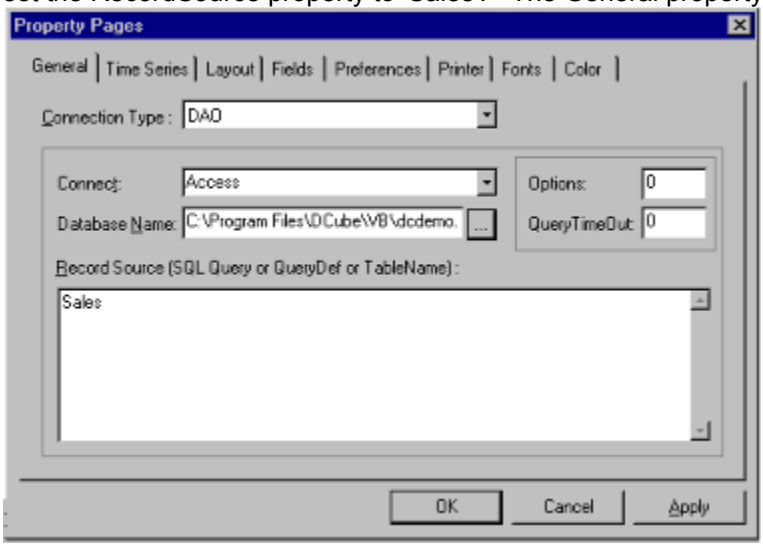
Creating Your First DynamiCube View

Start a new project in Visual Basic. Add the DynamiCube custom control ActiveX to your project.

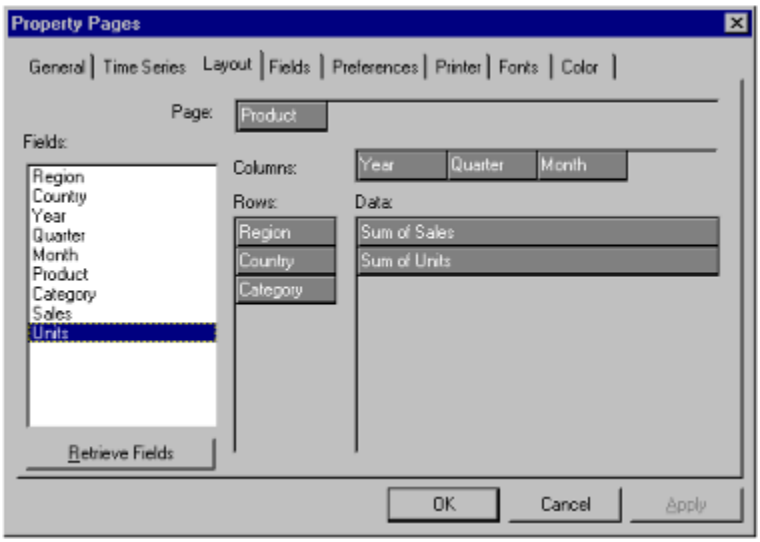
Add a DynamiCube control to the form. Now your form should look like this:



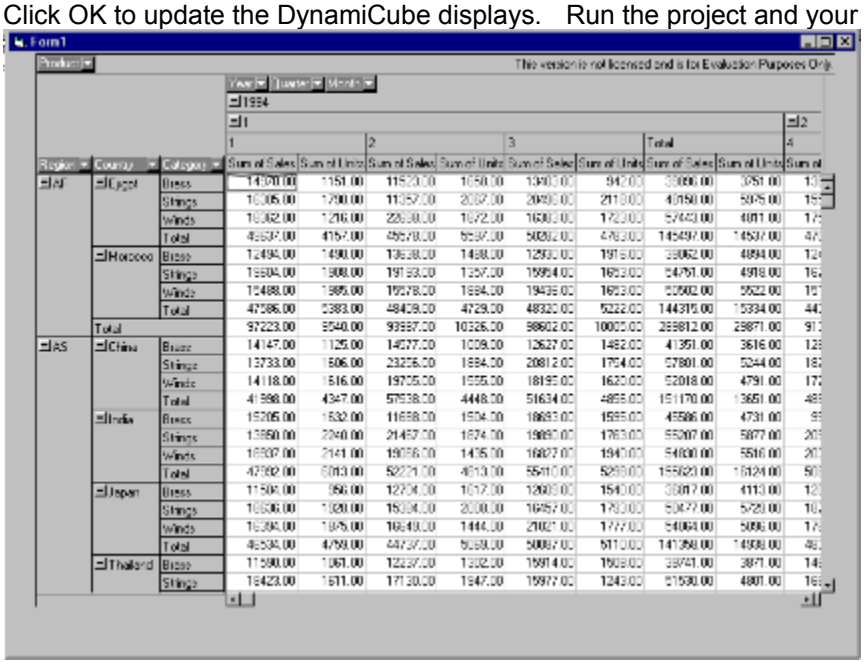
Call up DynamiCube's property sheets (either right click on the control and select "Properties…", or double-click the (Custom) property in the properties window.) On the General property sheet specify the 'Connection Type' as DAO. Select 'Access' for the connect combo box. Set the DatabaseName property to DCDEMO.MDB(be sure to qualify the filename with the path where the file is located). Then set the RecordSource property to 'Sales'. The General property sheet should now look like this:



Switch to the layout property sheet. Click the 'Retrieve Fields' button to update the fields list. Drag and drop fields from the row, column, page field, and data areas. You layout should look like this:



Click OK to update the DynamiCube displays. Run the project and your form should look like this:



Distributing DynamiCube Applications

Your application will need the DCube.ocx file to be included in the path. You should never distribute the DCube.lic file, it is a design-time license file that only you, the developer, can use.

What to Include

DCUBE.OCX	- DynamiCube ActiveX file
DynaODBC.DLL	- if you are using ODBC to connect to data sources.
DynaBDE.DLL	- if you are using Borland's BDE.

These files MUST be registered in Windows 95 or Windows NT registry using your setup program or manually using REGSVR32.EXE.

DynamiCube Object Properties

This is a complete listing of the DynamiCube object properties including Visual Basic standard control properties (noted with *). For more details on standard properties refer to your Visual Basic's Controls Reference.

AllowFiltering
AllowSplitters
AllowUserPivotFields
AutoDataRefresh
BackColor
BottomMargin
BorderStyle
ColAlignment
ColCount
ColHeadingsOnEveryPage
ColPageBreak
ColStyle
ColWidth
CursorStyle
DataPath
DataSource
DataNotAvailableCaption
dcConnect
dcConnectType
dcDataBaseName
dcOptions
dcQueryTimeout
dcRecordSource
FieldsFont
FieldsBackColor
FieldsForeColor
Font*
FooterCaption
FooterJustification
FooterMargin
ForeColor*
GridColor
GridStyle
HeaderCaption
HeaderJustification
HeaderMargin
HeadingsBackColor
HeadingsFont
HeadingsForeColor
LeftMargin
OutlineIconAlignment
PageFieldsVisible
PrinterColumnSpacing
PrinterFont
PrinterFontHeadings
PrinterFontTotals
PrinterGridColor
QueryByPass
RightMargin
RowAlignment

RowCount
RowHeadingsOnEveryPage
RowPageBreak
RowStyle
SelectionBackColor
SelectionForeColor
SelEndCol
SelEndRow
SelStartCol
SelStartRow
SQLMonthPart
SQLQuarterPart
SQLWeekPart
SQLYearPart
TopMargin
TotalsBackColor
TotalsFont
TotalsForeColor
Version
DragIcon*
Enabled*
MouseIcon*
Visible*

DynamiCube Field Object Properties

DynamiCube DataItem Object Properties

AllowFiltering

Determines if the end-user can filter data items from the DynamiCube view. Default is True.

Syntax

```
<object>.AllowFiltering = <bFilter>
```

Argument	Description
object	Valid reference to a DynamiCube Object
bFilter	True if end-user is allowed to filter data items

Example

```
DynamiCube1.AllowFiltering = False
DynamiCube1.Refresh
```

AllowSplitters

Defines or determines if the user is allowed to use splitter bars to create multiple view ports in the current DynamiCube view. If set to False, splitter handles will disappear.

Syntax

```
<object>.AllowSplitter = <bSplitters>
```

Argument	Description
object	Valid reference to a DynamiCube object.
bPivot	True or False setting

Example

```
DynamiCube1.AllowSplitters = False
```

AllowUserPivotFields

Defines or determines if the user can drag and drop fields between different areas. Setting this field to False locks the DynamiCube and makes it static, the fields cannot be moved around. Default is False.

Syntax

```
<object>.AllowUserPivotFields = <bPivot>
```

Argument	Description
object	Valid reference to a DynamiCube object.
bPivot	True or False setting

Example

```
DynamiCube1.AllowUserPivotFields = False
DynamiCube1.Reresh
```

AutoDataRefresh

Sets DynamiCube to refresh its data automatically as needed. DynamiCube refreshes the data set when a field is added or removed from the current view. If the AutoDataRefresh is set to False, DynamiCube needs to be refreshed manually by calling the RefreshData method.

Syntax

```
<object>.AutoDataRefresh = <bAuto>
```

Argument	Description
object	Valid reference to a DynamiCube object.
bAuto	True or False setting

Example

```
DynamiCube1.AutoDataRefresh = False
DynamiCube1.Fields(0).Orientation = 1
DynamiCube1.RefreshData
```

BackColor, ForeColor

Defines or determines the foreground and background color of the DynamiCube. BackColor and ForeColor are the colors of the grid cells. FieldsBackColor and FieldsForeColor are the colors of the field caption. GridColor is the color of the lines when the grid style is flat. HeadingsBackColor and HeadingsForeColor are the colors of the row and column data items area when the Row or Column Style is set to None. SelectionBackColor and SelectionForeColor are the colors of the selection. TotalsBackColor and TotalsForeColor are the colors of summary lines.

Syntax

```
<object>.BackColor [= <iColor>]
<object>.ForeColor [= <iColor>]
<object>.GridColor [= <iColor>]
<object>.HeadingsBackColor [= <iColor>]
<object>.HeadingsForeColor [= <iColor>]
<object>.SelectionBackColor [= <iColor>]
<object>.SelectionForeColor [= <iColor>]
<object>.TotalsBackColor [= <iColor>]
<object>.TotalsForeColor [= <iColor>]
```

Argument	Description
object	Valid reference to a DynamiCube object.
iColor	Value of the color defined with hexadecimal number, RGB function, or QBColor.

Example

```
DynamiCube1.BackColor = vbYellow
DynamiCube1.ForeColor = vbBlack
```

BottomMargin, FooterMargin, HeaderMargin, LeftMargin, RightMargin, TopMargin

Sets the value of page margins used when printing the DynamiCube view.

Syntax

```
<object>.BottomMargin = <iMargin>
<object>.LeftMargin = <iMargin>
<object>.RightMargin = <iMargin>
<object>.TopMargin = <iMargin>
```

Argument	Description
object	Valid reference to a DynamiCube object.
iMargin	Margin setting in inches.

Example

```
DynamiCube1.BottomMargin = 1
DynamiCube1.LeftMargin = 1
DynamiCube1.RightMargin = 1
DynamiCube1.TopMargin = 1
DynamiCube1.Print             ‘ Print it
```

BorderStyle

Sets or returns the Border style.

Syntax

```
<object>.BorderStyle = <iBorderStyle>
```

Argument	Description
object	Valid reference to a DynamiCube object
iBorderStyle	Valid setting for the grid style (as shown below)

Settings

Setting	Description
0-None	No border is displayed
1-FixedSingle	A single border line is displayed

Example

```
DynamiCube1.BorderStyle = 1  ‘ Display border line
```

ColAlignment, RowAlignment

Defines or determines the alignment of the Columns or Rows Headings in the DynamiCube view.

Syntax

```
<object>.ColAlignment = <iAlignment>
<object>.RowAlignment = <iAlignment>
```

Argument	Description
object	Valid reference to a DynamiCube object.
iAlignment	Valid alignment setting.

Settings

Setting	Description
0-Left Top	Align to left-top of the heading cell
1-Center Top	Align to the center-top of the heading cell
2-Right Top	Align to the right-top of the heading cell
3-Left Center	Align to the left-center of the heading cell.
4-Center Center	Align to the center-center of the heading cell.
5-Right Center	Align to the right-center of the heading cell.
6-Left Bottom	Align to the left-bottom of the heading cell.
7-Center Bottom	Align to the center-bottom of the heading cell.
8-Right Bottom	Align to the right-bottom of the heading cell.

Example

```
DynamiCube1.RowAlignment = 0  ‘ Left Top
DynamiCube1.ColAlignment = 4  ‘ Center
DynamiCube1.Refresh
```

ColCount, RowCount

Returns the current number of columns or rows in the DynamiCube view. Not available at design time. Read only at runtime.

Syntax

```
<iColumns> = <object>.ColCount
<iRows> = <object>.RowCount
```

Argument	Description
object	Valid reference to a DynamiCube object
iColumns	Number of columns in a Dynamic Cube view
iRows	Number of rows in a Dynamic Cube view

Example

```
‘ This example prints all values within a grid
For I = 0 to DynamiCube1.RowCount - 1
For J = 0 to DynamiCube1.ColCount - 1
      Debug.Print DynamiCube1.DataValue(I, J)
Next
```

ColHeadingsOnEveryPage, RowHeadingsOnEveryPage

Determines if the column and row headings of the dynamic cube view should be printed on every page when using the print method.

Syntax

```
<object>.ColHeadingOnEveryPage = <bColHeading>
<object>.RowHeadingOnEveryPage = <bRowHeading>
```

Argument	Description
object	Valid reference to a DynamiCube object
bColHeading	Boolean expression. Column headings will be printed on every page if the expression evaluates to true.
bRowHeading	Boolean expression. Row headings will be printed on every page if the expression evaluates to true.

Example

```
DynamiCube1.ColHeadingOnEveryPage = True  ‘ Columns
DynamiCube1.RowHeadingOnEveryPage = True  ‘ Rows
DynamiCube1.Print                         ‘ Print the cube
```

ColPageBreak, RowPageBreak

Sets the column or row level of page breaks when using the print method. A 0 setting sets no page breaks. A setting greater than 0 sets a page break after each data item value changes.

Syntax

```
<object>.ColPageBreak = <iPageBreak>
<object>.RowPageBreak = <iPageBreak>
```

Argument	Description
object	Valid reference to a DynamiCube object
iPageBreak	Sets the field level in rows or columns which to print on separate pages

Example

```
‘ if dynamic cube columns are: Year Quarter
‘                  and rows are: Region, Product
DynamiCube1.ColPageBreak = 1  ‘ Print each year on a page
DynamiCube1.RowPageBreak = 1  ‘ Print each region on a page
DynamiCube1.Print
```

ColStyle, RowStyle

Determines the appearance of the columns or rows headings cell.

Syntax

```
<object>.ColStyle = <iStyle>
<object>.RowStyle = <iStyle>
```

Argument	Description
object	Valid reference to a DynamiCube object.
iStyle	Valid appearance setting (as shown below)

Settings

Setting	Description
0-Flat	Flat with no 3D appearance.
1-Raised Light	Gives a light raised 3D look.
2-Raised Heavy	Gives a heavy raised 3D look.
3-Sunken Light	Gives a light inset 3D look.
4-Sunken Heavy	Gives a heavy inset 3D look.

Example

```
DynamiCube1.ColStyle = 1‘ Raised light
DynamiCube1.RowStyle = 3‘ Sunken light
DynamiCube1.Refresh
```

ColWidth

Sets or returns the column width.

Syntax

```
[<iWidth> = ] <object>.ColWidth(<iCol>)
```

Argument	Description
object	Valid reference to a DynamiCube object
iCol	Column index number (0 to number of columns -1)
iWidth	Width of the column in pixels

Remarks

If the view has multiple data items (sales, profit, …) sizing any of the sales columns will size all. For example, if the columns appeared as a sequence of Sales | Profit | Sales | Profit | Sales | Profit resizing column 0 would also resize columns 2 and 4.

Example

```
‘ Sets the second column width to 1200 pixels
DynamiCube1.ColWidth(1) = 1200
```

CursorStyle

Sets or returns the style of the single grid cell border

Syntax

```
<object>.CursorStyle = <iStyle>
```

Argument	Description
object	Valid reference to a DynamiCube object
iStyle	Valid cursor style setting (as shown below)

Settings

Setting	Description
0-Light	Thin border
1-Heavy	Thick border

Example

```
DynamiCube1.CursorStyle = 1
```

DataPath

Returns or sets a data path (a valid local dos file or a valid URL on a web server) of a microCube file to be loaded into the DynamiCube view asynchronously. When this property is set, all other data related properties (DataSource, DCRecordSource, …) are ignored and the DynamiCube retrieved a previously saved microCube instead.

Syntax

```
<object>.DataPath = <sDataPath>
```

Argument	Description
object	Valid reference to a DynamiCube Object
sDataPath	A valid local or URL path

Example

```
DynamiCube1.DataPath = App.Path & "\MYCUBE.CUB"
or
DynamiCube1.DataPath = "ftp://ftp.myweb.com/data/mycube.cub"
```

DataSource

Sets or returns the name of the data control that the DynamiCube is bound to.

Syntax

```
<object>.DataSource = <sDataSource>
```

Argument	Description
object	Valid reference to a DynamiCube Object
sDataSource	Set to name of data control access.

Example

```
DynamiCube1.dcConnectType = 0 ‘ Use the data control
DynamiCube1.DataSource = "Data1"
DynamiCube1.DCDataBaseName = App.Path & "\demo.mdb"
DynamiCube1.DCRecordSource = "SELECT * FROM Sample"
```

See Also

DCConnectType

DataNotAvailableCaption

Returns or sets text string to be displayed when there is not data available in any given data cell.

Syntax

```
<object>.DataNotAvailableCaption = <sCaption>
```

Argument	Description
object	Valid reference to a DynamiCube object.
sCaption	String displayed for unavailable data.

Example

```
‘ Display NA in each data cell that did not return any data
DynamiCube1.DataNotAvailableCaption = "NA"
```

dcConnect

Returns or sets a value that provides information about the source of an open database (see Visual Basic Data Control connect property for more information)

Syntax

```
<object>.dcConnect = <sConnect>
```

Argument	Description
object	Valid reference to a DynamiCube object.
sConnect	Valid connection string to connect the DynamiCube to its data source.

Example

```
DynamiCube1.dcConnectType = DCCT_DAO‘ Connect Using DAO
DynamiCube1.dcConnect = "Access"
DynamiCube1.DCDataBaseName = App.Path & "\demo.mdb"
DynamiCube1.DCRecordSource = "sales"
```

See Also

dcConnectType

dcConnectType

Returns or sets the database connection type. The DynamiCube can use either Remote Data Object (RDO), Data Access Object (DAO) or native ODBC to directly connect to the data source.

Syntax

```
<object>.dcConnectType = <iConnectType>
```

Argument	Description
object	Valid reference to a DynamiCube object.
IConnectType	Valid connection type setting (see below)

Settings

Setting	Description
0	Connect using a Data Control or Remote Data Control
1	Connect using Data Access Objects (DAO)
2	Connect using Remote Data Objects (RDO)
3	Connect using Native ODBC
4	Connect using Borland Database Engine (BDE)
5	Connect using unbound Mode, FetchData and AddRow

Example

```
‘ connect a cube through DAO
DynamiCube1.dcConnectType = DCCT_DAO
DynamiCube1.dcConnect = "Access"
DynamiCube1.DCDataBaseName = App.Path & "\demo.mdb"
DynamiCube1.DCRecordSource = "sales"

‘ Connect another cube through RDO
DynamiCube2.dcConnectType = DCCT_RDO
DynamiCube2.dcConnect = "Access"
DynamiCube2.DCDataBaseName = App.Path & "\demo.mdb"
DynamiCube2.DCOptions = 4     ‘ Read only Access
DynamiCube2.DCRecordSource = "SELECT * FROM sales"

‘ Connect a third using ODBC
DynamiCube3.dcConnectType = DCCT_ODBC
DynamiCube3.dcConnect = "DSN=Access;DBQ=demo.mdb"

‘ Connect a fourth using BDE
DynamiCube4.dcConnectType = DCCT_BDE
‘ For BDE SQLLinks Syntax is Alias;Password;Driver
DynamiCube4.dcConnect = "DynaDemo;MyPassword;Driver"
```

dcDataBaseName

Returns or sets the name and location of the source of data for a DynamiCube, or the DataSource name from ODBC when using RDO connection to the database.

Syntax

```
<object>.DCDataBaseName [= <sName>]
```

Argument	Description
object	Valid reference to a DynamiCube object.
sName	Proper path to the database location or data source name

Example

```
‘ connect a cube through DAO
DynamiCube1.dcConnectType = DCCT_DAO
DynamiCube1.dcConnect = "Access"
DynamiCube1.DCDataBaseName = App.Path & "\demo.mdb"
DynamiCube1.DCRecordSource = "sales"

‘ Connect another cube through RDO
DynamiCube2.dcConnectType = DCCT_RDO
DynamiCube2.dcConnect = "Access"
DynamiCube2.DCDataBaseName = App.Path & "\demo.mdb"
DynamiCube2.DCOptions = 4     ‘ Readonly Access
DynamiCube2.DCRecordSource = "SELECT * FROM sales"
```

See Also

dcConnectType

dcOptions

Returns or sets a value that specifies one or more characteristics of the Recordset object in the control's Recordset property. This property is valid only when the connection type (dcConnectType) is DAO, it should be set to 0 for all other connection types.

Syntax

```
<object>.dcOptions = <iOptions>
```

Argument	Description
object	Valid reference to a DynamiCube object.
iOptions	one or more setting value. Must be added for more than one setting.

Settings

Setting	Description
1	In a multi-user environment, other users can't make changes to records in the Recordset.
2	In a multi-user environment, other users can't read records (table-type Recordset only).
4	You can't make changes to records in the Recordset.
8	You can add new records to the Recordset, but you can't read existing records.
16	Updates can apply to all fields of the Recordset, even if they violate the join condition.
32	(Default) Updates apply only to those fields that don't violate the join condition.
64	When using Data controls with an SQL statement in the DCRecordSource property, sends the SQL statement to an ODBC database, such as a SQL Server or Oracle database, for processing.
256	The Recordset is a forward-only scrolling. The only move method allowed is MoveNext. This option cannot be used on Recordset objects manipulated with the Data control.
512	Generate a trappable error if another user is changing data you are editing.

Example

```
DynamiCube1.dcConnectType = DCCT_DAO
DynamiCube1.dcConnect = "Access"
DynamiCube1.DCDataBaseName = "C:\DCDEMO.MDB"
DynamiCube1.dcOptions = 256
DynamiCube1.RecordSource = "SELECT * FROM Sales"
```

dcQueryTimeout

Returns or sets the number of seconds to wait before a timeout error occurs when running the data extraction query against an ODBC source. A value of zero means no timeout.

Syntax

```
<object>.dcQueryTimeout = <iSeconds>
```

Argument	Description
object	Valid reference to a DynamiCube object.
iSeconds	Number of seconds.

Example

```
DynamiCube1.dcConnectType = DCCT_RDO      ‘ use RDO
DynamiCube1.dcConnect = "DynaDemo"
DynamiCube1.dcQueryTimeout = 60     ‘ wait 60 secs.
```

dcRecordSource

Returns or sets the underlying SQL record source for the DynamiCube.

Syntax

```
<object>.DCRecordSource [= <sRecordSource>]
```

Argument	Description
object	Valid reference to a DynamiCube object
sRecordSource	Table or QueryDef name or valid SQL statement that evaluates into a recordset.

Example

```
DynamiCube1.dcConnectType = DCCT_DAO
DynamiCube1.dcConnect = "Access"
DynamiCube1.DCDataBaseName = App.Path & "\demo.mdb"
DynamiCube1.DCRecordSource = "SELECT * FROM Sample"
```

FieldsFont, Font, HeadingsFont, TotalsFont

Returns a font object of DynamiCube areas (Headings, Grid, Fields and Totals).

Syntax

```
<object>.HeadingsFont.<FontProperty> = <Setting>
<object>.FieldFont.<FontProperty> = <Setting>
<object>.Font.<FontProperty> = <Setting>
<object>.TotalsFont.<FontProperty> = <Setting>
```

Argument	Description
object	Valid reference to a DynamiCube object
FontProperty	Valid font property name (Bold, Name, Italic, ..)
Setting	Valid setting for that font property

Example

```
DynamiCubes.TotalsFont.Name = "Sans Serif"
DynamiCube1.TotalsFont.Bold = True
DynamiCube1.Refresh
```

FooterCaption, HeaderCaption

Sets or retrieves the footer and header text on a Dynamic Cube printed page.

Syntax

```
<object>.FooterCaption = sFooter
<object>.HeaderCaption = sHeader
```

Argument	Description
object	Valid reference to a DynamiCube object
sFooter	Footer string to be printed on each page
sHeader	Header string to be printed on each page

Example

```
DynamiCube1.Header = "Sample Header"
DynamiCube1.Footer = "Sample Footer"
DynamiCube1.Print
```

FooterJustification, HeaderJustification

Sets the justification of a Dynamic Cube printed page header or footer.

Syntax

```
<object>.FooterJustification = iJustification
<object>.HeaderJustification = iJustification
```

Argument	Description
object	Valid reference to a DynamiCube object
iJustification	Valid justification setting (as shown below)

Settings

Setting	Description
0-Left	Print header or footer justified to the left margin of the page
1-Center	Print header or footer centered between left and right margins
2-Right	Print header or footer justified to the right margin of the page.

Example

```
DynamiCube1.FooterCaption = "Sample Footer"
DynamiCube1.HeaderCaption = "Sample Header"
DynamiCube1.FooterJustification = 1 ‘ Center
DynamiCube1.FooterJustification = 1 ‘ Center
DynamiCube1.Print
```

GridStyle

Sets or returns the gridlines style.

Syntax

```
<object>.GridStyle = <iGridStyle>
```

Argument	Description
object	Valid reference to a DynamiCube object
iGridStyle	Valid setting for the grid style (as shown below)

Settings

Setting	Description
0-None	No gridlines are displayed
1-Flat	Solid lines
2-Raised	Raised 3D lines
3-Sunken	Sunken 3D lines

Example

```
DynamiCube1.GridStyle = 2     ‘ Raised 3D lines
```

OutlineIconAlignment

Sets or returns the alignment of the outline icon [+/-] within the row or column cell.

Syntax

```
<object>.OutlineIconAlignment = <iIconAlign>
```

Argument	Description
object	Valid reference to a DynamiCube object
iIconAlign	Valid icon alignment setting (as shown below)

Settings

Setting	Description
0-Hidden	Hidden
1-Left Top	Align to left top of the data item cell
2-Left Center	Align to left center of the data item cell
3-Right Top	Align to the right top of the cell

Example

```
DynamiCube1.OutlineIconAlignment = 0 ‘ Hide the outline icon
```

PrinterColumnSpacing

Sets or returns the spacing between columns when printing or previewing a DynamiCube view.

Syntax

```
<object>.PrinterColumnSpacing = <iSpacing>
```

Argument	Description
object	Valid reference to a DynamiCube object
iSpacing	Value of spacing in current scale mode.

Example

```
DynamiCube1.PrinterColumnSpacing = 50
iRC = DynamiCube1.Print (True, 0, 0)
```

PageFieldsVisible

Defines or determines if the user is allowed to use Page Fields in the current DynamiCube view. If set to False, the page fields space will disappear.

Syntax

```
<object>.PageFieldsVisible = <bPagesVisible>
```

Argument	Description
object	Valid reference to a DynamiCube object.
bPagesVisible	True or False setting

Example

```
DynamiCube1.PageFieldsVisible = False
```

PrinterFont, PrinterFontHeadings, PrinterFontTotals

Sets or returns the printer fonts properties used in print and print preview. PrinterFont is used for all text except headings and totals. PrinterFontHeadings and PrinterFontTotals are used for those areas. You can set any of the standard font properties including name, size, ..

Syntax

```
<object>.PrinterFont.<Font Property> = <Property Value>
<object>.PrinterFontHeadings.<Font Property> = <Property Value>
<object>.PrinterFontTotals.<Font Property> = <Property Value>
```

Argument	Description
object	Valid reference to a DynamiCube object
Font Property	Any of the standard font properties, name, size, ...
Property Value	A valid font property value

Example

```
DynamiCube1.PrinterFont.Name = "Arial"
DynamiCube1.PrinterFont.Size = 10
DynamiCube1.PrinterFontTotals.Name = "Arial"
DynamiCube1.PrinterFontTotals.Size = 12
DynamiCube1.PrinterFontTotals.Bols = True
iRC = DynamiCube1.Print(True, 0, 0)
```

PrinterGridColor

Sets or returns the grid lines color used when printing or print previewing a DynamiCube view.

Syntax

```
<object>.PrinterGridColor = <Color Value>
```

Argument	Description
object	Valid reference to a DynamiCube object
Color Value	A valid VB color value

Example

```
DynamiCube1.PrinterGridColor = vbRED
iRC = DynamiCube1.Print(True, 0,0)
```

QueryByPass

Setting this property to True instructs the DynamiCube to use the Record Source SQL statement as is. DynamiCube will not generate its internal GROUP BY SQL statement to summarize the data, it will assume that the data is already summarized as a result of the Record Source SQL. This can be useful if the record source was a stored procedure call or if the tables were pre-summarized using a batch process.

Syntax

```
<object>.QueryByPass = <bByPass>
```

Argument	Description
object	Valid reference to a DynamiCube object
bByPass	On True - A summary GROUP BY SQL will not be generated to retrieve the data.

Example

```
DynamiCube1.RecordSource = "SELECT * FROM SummaryTable"
DynamiCube1.QueryByPass = True
…
```

SelEndCol, SelStartCol, SelEndRow, SelStartRow

Sets or returns the row and column coordinates of the current selection. Coordinates start at 0, 0 and end at DynamiCube.RowCount, DynamiCube.ColCount

Syntax

```
<object>.SelEndCol = <iEndCol>
<object>.SelStartCol = <iStartCol>
<object>.SelEndRow = <iEndRow>
<object>.SelStartRow = <iStartRow>
```

Argument	Description
object	Valid reference to a DynamiCube object
iEndCol	Number of last column in the current selection
iStartCol	Number of first column in the current selection
iEndRow	Number of last row in the current selection
iStartRow	Number of first row in the current selection

Example

```
‘ This example loops through each cell in the current
‘ selection and prints the value to the debug window
I = DynmiCube1.SelStartRow
Do While I <= DynamiCube1.SelEndRow
j = DynamiCube1.SelStartCol
Do While J <= DynamiCube1.SelEndCol
Debug.Print DynamiCube1.DataValue(I, J)
J = DynamiCube1.GetNextCol(J)
Loop
I = DynamiCube1.GetNextRow(I)
Loop
```

SQLMonthPart, SQLQuarterPart, SQLWeekPart, SQLYearPart

Sets or returns the date parsing function syntax. It defaults to JET's DatePart function syntax. It can be used to specify a different function name in the data grouping SQL query executed internally by DynamiCube when using other SQL data sources like MS SQL Server or Oracle.

Syntax

```
<object>.SQLMonthPart = <sSyntax>
<object>.SQLQuarterPart = <sSyntax>
<object>.SQLWeekPart = <sSyntax>
<object>.SQLYearPart = <sSyntax>
```

Argument	Description
object	Valid reference to a DynamiCube object
sSyntax	String to be used for extracting date part., use <date> to set the fieldname insertion point into the syntax string.

Example

```
‘ set Oracle data function syntax
‘ <date> is replaced with the field name of type date
DynamiCube1.SQLMonthPart = "to_char(<date>, "MM")
DynamiCube1.SQLQuarterPart = "to_char(<date>, "Q")
DynamiCube1.SQLWeekPart = "to_char(<date>, "WW")
DynamiCube1.SQLYearPart = "to_char(<date>, "YYYY")
```

Version

Returns a value that indicates the version of the object. Read Only.

Syntax

```
<object>.Version
```

Argument	Description
object	Valid reference to a DynamiCube Object

Return Values

The Version property return value is a string expression

AggregateFunc

Sets or returns the current aggregate function to be used in calculating the value of a data field. This property applies only to data fields i.e. Orientation = DCData. Default is 1-Sum.

Syntax

```
<object>.AggregateFunc [= <iAggregate>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
iAggregate	Valid aggregate function setting (as shown below)

Settings

Setting	Description
0-Sum	Calculate sum of a data field
1- Count	Calculate count of a data field
2-Average	Calculate average of a data field
3- Max	Calculate maximum value of a data field
4- Min	Calculate minimum value of a data field
5- StdDev	Calculate population sample standard deviation of a data field
6- StdDevP	Calculate population standard deviation of a data field
7- Var	Calculate population sample variance of a data field
8- VarP	Calculate population variance of a data field

Example

```
‘ Change aggregate to average
DynamiCube1.Fields("Sales").AggregateFunc = 3
```

Calculated

Sets a calculated field flag. This is used when the field SourceName (used in Add method) is an expression. Default value is False for all fields retrieved from the record set.

Syntax

```
<object>.Calculated = <bCalculated>
```

Argument	Description
object	Valid reference to a DynamiCube field object
bCalculated	True if the field is calculated from an expression

Example

```
‘ Record set has the following two fields
‘ Sales and Profit
Dim x As Object
‘ Add the sales field as the first data field
Set x = DynamiCube1.Fields.Add("Sales", "Sales", DCData)
x.AggregateFunc = DCSum
Set x = DynamiCube1.Fields.Add("Profit", "Profit", DCData)
x.AggregateFunc = DCSum
‘ Add Ratio As a new calculated data field
Set x = DynamiCube1.Fields.Add("Sales/Profit", "Ratio", DCData)
x.Calculated = True     ‘ Specify the Ratio as a calculated field
‘ There are two ways that the group footer can be calculated
‘     * Aggregate of the calculated field e.g. Sum(Ratio) or,
‘           x.GroupFooterType = DCFSum
‘     * Calculation of the expression’s source fields aggregation
‘           e.g. Sum(Sales) / Sum(Profit)
‘           x.GroupFooterType = DCFCalculated
‘ Here we are using the second method
x.GroupFooterType = DCFCalculated
```

Caption

Sets or returns the caption text to be used when displaying the field to the user.

Syntax

```
<object>.Caption [= <sCaption>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
sCaption	String - Caption text of the field

Example

```
DynamiCube1.Fields("RGN").Caption = "Region"
```

CurrentValue

Returns a DataItem Object relative to the current cursor position (SelStartRow, SelStartCol).

Syntax

```
<object>.CurrentValue [= <vCurentValue>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
vCurrentValue	Current value of the field, one of its data items

Example

```
For I = 0 to DynamiCube1.Fields.Count
Debug.Print DynamiCube1.Fields(I).CurrentValue
Next
```

DataItems (collection)

Returns a collection of DataItems in the specified field.

Syntax

```
<object>.DataItems
```

Argument	Description
object	Valid reference to a DynamiCube object

Example

```
For I = 0 to DynamiCube1.Fields(I).DataItems.Count
lstDItems.AddItem DynamiCube1.Fields(I).DataItems(I).Name
Next
```

DataType

Returns the fields data type.

Syntax

```
iType = <object>.DataType
```

Argument	Description
object	Valid reference to a DynamiCube field object
iType	Data type of the field

Example

```
For I = 0 to DynamiCube1.Fields.Count
Debug.Print DynamiCube1.Fields(I).DataType
Next
```

DisplayAs

Sets or retrieves the current DisplayAs setting of a data field. Data field values can be displayed as normal (same values retrieved from the data source) or as a percentage of column or row grand totals.

Syntax

```
<object>.DisplayAs [= <iDispAs>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
iDispAs	Integer setting of Display As

Settings

Setting	Description
0-DCDTNormal	Data is displayed as it was retrieved
1-DCDTPercentCol	Display as a percentage of column grand total
2-DCDTPercentRow	Display as a percentage of row grand total

Example

```
DynamiCube.DataFields(i).DisplayAs = DCDTPercentCol
DynamiCube.DataFields(i).NumberFormat = "% #.00"
DynamiCube.Refresh
```

GroupFooterCaption

Sets or retrieves the group footer caption used in the summary line.

Syntax

```
<object>.GroupFooterCaption [= <sCaption>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
sCaption	Group footer caption string

Example

```
DynamiCube1.Fields("Region").GroupFooterCaption = "Regions Total"
```

GroupFooterPicture

Sets or returns the picture of the group footer used in the summary line.

Syntax

```
<object>.GroupFooterPicture [ = <picture>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
picture	Valid reference to a picture

Example

```
DynamiCube1.Fields("Region").GroupFooterPicture = _
LoadPicture("map.ico")
```

GroupFooterPictureAlignment

Sets or returns the alignment of the group footer picture within the summary line caption cell.

Syntax

```
<object>.GroupFooterPictureAlignment [= <iAlignment>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
iAlignment	Valid alignment setting (as shown below)

Settings

Setting	Description
0-Left Top	Align to left-top of the heading cell
1-Center Top	Align to the center-top of the heading cell
2-Right Top	Align to the right-top of the heading cell
3-Left Center	Align to the left-center of the heading cell.
4-Center Center	Align to the center-center of the heading cell.
5-Right Center	Align to the right-center of the heading cell.
6-Left Bottom	Align to the left-bottom of the heading cell.
7-Center Bottom	Align to the center-bottom of the heading cell.
8-Right Bottom	Align to the right-bottom of the heading cell.

Example

```
DynamiCube1.Fields (0).GroupFooterPictureAlignment = 6
```

GroupFooterType

Sets or returns the group footer type of a field. A setting of 0-None hides the summary line. This type is calculated locally and is different from the data item aggregation setting.

Syntax

<object>.GroupFooterType [= <iType>]

Argument	Description
object	Valid reference to a DynamiCube field object
iAlignment	Valid footer type setting (as shown below)

Settings

Setting	Description
0-None	No group footer is required for this field.
1-Sum	Sum the data values for this field
2-Average	Average the data values for this field
3-Count	Count the data values for this field

Example

```
‘ Hide summary line
DynamiCube1.Fields(0).GroupFooterType = 0
```

Name

Sets or returns the name of a DynamiCube field object.

Syntax

```
<object>.Name [= <sFieldName>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
sFieldName	Name of the field as it is in the recordset

Example

```
‘ Fill List1 with field names in the cube
For I = 0 to DynamiCube1.Fields.Count
List1.AddItem DynamiCube1.Fields(I).Name
Next
```

NumberFormat

Sets or returns the format code for the DynamiCube field object (as a string). Valid only for Data Fields.

Syntax

```
<object>.NumberFormat = <sFormat>
```

Argument	Description
object	Valid reference to a DynamiCube field object
sFormat	Valid number formatting string

Example

```
DynamiCube1.Fields("Sales").NumberFormat = "$#####0.00"
```

Orientation

Sets or returns the location of the field in the DynamiCube object's pivot areas.

Syntax

```
<object>.Orientation [= <iOrientation>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
iOrientation	Valid orientation setting

Settings

Setting	Description
0-Hidden	Hidden field, does appear in the current layout
1-Column	Appears as a column in the current layout.
2-Row	Appears as a row in the current layout
3-Data	Appears as a data field in the current layout

Example

```
‘ Pivot the year field
DynamiCube1.Fields("Year").Orientation = 2
```

Position

Sets or returns the position of field within its orientation group (Rows, Columns, Data).

Syntax

```
<object>.Position [= <iPosition>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
iPosition	Integer - Position of the field

Example

```
DynamiCube1.Fields("Year").Position = 2
```

RankOn

Sets or returns the data field index which grand total value is used in ranking the data items for this field. Data field index starts with 0.

Syntax

```
<Object>.RankOn = <iDataField>
```

Argument	Description
object	Valid reference to a DynamiCube field object
iDataField	Data field index to be used in ranking.

Example

```
DynamiCube1.Fields("Region").RankOn = 1 ‘ Rank on Unit Sales Total
```

RankStyle

Sets or returns the ranking order, ascending, descending or off. This value is valid only if RankOn is set to valid data field index.

Syntax

```
<Object>.RankStyle = <iRankStyle>
```

Argument	Description
object	Valid reference to a DynamiCube field object
iRankStyle	A valid ranking style setting

Settings

Setting	Description
0-None	Default
1-Ascending	Sort the data items by rank field total value in ascending order.
2-Descending	Sort the data items by rank field total value in descending order.

Example

```
DynamiCube1.Fields("Region").RankStyle = 1‘ Sort Ascending
```

VarName

Sets a variable name for fields that are defined expressions in a SQL statement record source. If there is no alias for the field, DynamiCube uses the expression itself as a name for the field. If you need to use the field in local expressions (calculated fields), you will need to refer to the field by its VarName rather than its name. Setting or using this property for a field is not needed unless it is used in a local expression.

Syntax

```
<object>.VarName = <sVarName>
```

Argument	Description
object	Valid reference to a DynamiCube item object
sVarName	Variable name to be used instead of name

Example

```
DynamiCube1.DataFields(1).VarName = "Variance"
```

Visible

Defines or determines if a data item or a data field is visible in the current dynamic cube view.

Syntax

```
<object>.Visible [= <bVisible>]
```

Argument	Description
object	Valid reference to a DynamiCube item object
bVisible	Boolean. Item or data field is visible if True.

Example

```
DynamiCube1.Fields(0).DataItems(0).Visible = False
```

Width

Sets the width of a field when the field's orientation is set to Row. This would define the width of the row headings area as the total width of field in the RowFields collection. A setting of 0 allows the DynamiCube to do a Size-To-Fit sizing of the fields.

Syntax

```
<object>.value [= <lWidth>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
lWidth	Long. Width of the field, 0 for Size-To-Fit

Example

```
DynamiCube1.RowFields(0).Width = 300
```

Caption

Sets or returns the data item caption as it appears in the Dynamic Cube view. This can be used as a translation table of lookup fields in a view. For example region code "E" can have a caption of "East" displayed in the view.

Syntax

```
<object>.Caption [= <sCaption>]
```

Argument	Description
object	Valid reference to a DynamiCube item object
sCaption	New caption of the data item

Example

```
DynamiCube1.Fields("Region").DataItems("E").Caption = "East"
```

Name

Returns the name or value of the data item. A data field "region" would return the following data item names: "E", "W", "N", "S". This property is read only at runtime.

Syntax

```
<object>.Name
```

Argument	Description
object	Valid reference to a DynamiCube item object

Example

```
For I = 0 to DynamiCube1.Fields(0).DataItems.Count
List1.AddItem DynamiCube1.Fields(0).DataItems(I).Name
Next
```

Picture

Indicates what image appears in a data item cell. This property is definable at design time or runtime and defaults to displaying nothing.

Syntax

```
<object>.Picture = <picture>
```

Argument	Description
object	Valid reference to a DynamiCube item object
picture	A picture object to be displayed in the cell

Example

```
DynamiCube1.Fields(0).DataItems(0).Picture = _
LoadPicture("c:\test\test.ico")
```

PictureAlignment

Sets or returns the alignment of the bitmap in the data item cell.

Syntax

```
<object>.PictureAlignment [= <iAlignment>]
```

Argument	Description
object	Valid reference to a DynamiCube item object
iAlignment	Valid alignment setting (as shown below)

Settings

Setting	Description
0-Left Top	Align to left-top of the heading cell
1-Center Top	Align to the center-top of the heading cell
2-Right Top	Align to the right-top of the heading cell
3-Left Center	Align to the left-center of the heading cell.
4-Center Center	Align to the center-center of the heading cell.
5-Right Center	Align to the right-center of the heading cell.
6-Left Bottom	Align to the left-bottom of the heading cell.
7-Center Bottom	Align to the center-bottom of the heading cell.
8-Right Bottom	Align to the right-bottom of the heading cell.

Example

```
DynamiCube1.Fields(0).DataItems(0).PictureAlignment = 2
```

Visible

Defines or determines if a data item or a data field is visible in the current dynamic cube view.

Syntax

```
<object>.Visible [= <bVisible>]
```

Argument	Description
object	Valid reference to a DynamiCube item object
bVisible	Boolean. Item or data field is visible if True.

Example

```
DynamiCube1.Fields(0).DataItems(0).Visible = False
```

Width

Sets the width of a field when the field's orientation is set to Row. This would define the width of the row headings area as the total width of field in the RowFields collection. A setting of 0 allows the DynamiCube to do a Size-To-Fit sizing of the fields.

Syntax

```
<object>.value [= <lWidth>]
```

Argument	Description
object	Valid reference to a DynamiCube field object
lWidth	Long. Width of the field, 0 for Size-To-Fit

Example

```
DynamiCube1.RowFields(0).Width = 300
```

Events Reference

DataError

This event is fired when a data access error occurs and there is no Visual Basic code executing. Even if the application handles run-time errors, some errors might occur during an automatic data refresh.

Parameters

Parameter	Description
ErrorCode	One if the internal error codes listed below
SuppressErrorMessage	Default is 1, display an error message.

Error Codes

Code	Description
1	Cannot open a record set
2	Data Area Does not have any fields
3	DAO engine not found

Example

```
Private Sub DynamiCube1_DataError(ErrorCode AS Integer,
SuppressErrorMessage As Integer)
if ErrorCode = 1 Then SuppressErrorMessage = 1
End Sub
```

FetchAttributes

This event is fired for each data cell before painting to the screen. It is used to change any display attributes based on the value or position of the current cell. To set or retrieve the attributes of a cell, you will simply modify or read the properties of the 'Attrib' object as shown below.

Parameters

Parameter	Description
CellAttrib.DataFieldIndex	Index of data field that this cell represents in a multiple data fields view. A negative value specifies that the value is a summary value for that field. An example of two data fields (Sales, Profit) DataFieldIndex would be 1 for Sales and -1 for the summary line value of Sales. Valid ranges: 1 - ColumnCount and -ColumnCount -1
CellAttrib.DataValue	Value displayed in the cell
CellAttrib.ForeColor	Foreground color of the cell
CellAttrib.BackColor	Background color of the cell
CellAttrib.Row	Row that the cell is in
cellAttrib.Column	Column that the cell is in

Example

```
Private Sub DCube1_FetchAttributes(ByVal cellAttrib As
DynamiCubeLibCtl.Attrib)
If cellAttrib.Column = cellAttrib.Row Then
      cellAttrib.BackColor = vbBlue
      cellAttrib.ForeColor = vbGreen
End If
End Sub
```

FetchData

This event is fired only if the dcConnectType property is set to DCCT_UNBOUND, which instructs DynamiCube to fetch the summarized data rows at run-time instead of retrieving it from a predefined data source. The event is fired once, and the user can add the data using the AddRow method.

Parameters

None

Example

```
‘ This assumes a DynamiCube view with
‘ two dimensions Product, Region and
‘ one data field Sales.
Private Sub Form_Load()
Dim x As Object
Set x = DynamiCube1.Fields.Add("product", "Product", DCRow)
x.GroupFooterType = 1
Set x = DynamiCube1.Fields.Add("region", "Region", DCColumn)
x.GroupFooterType = 1
Set x = DynamiCube1.Fields.Add("Sales", "Sales", DCData)
x.AggregateFunc = 0
End Sub
Private Sub DynamiCube1_FetchData()
Dim valArray(3)
valArray(0) = "Product2"
valArray(1) = "Africa"
valArray(2) = 123
DynamiCube1.AddRow (valArray)
valArray(0) = "Product1"
valArray(1) = "Africa"
valArray(2) = 123
DynamiCube1.AddRow (valArray)
End Sub
```

Filter

This event is fired whenever the user manually filters fields in or out of a given view. This allows capturing the event and changing some DynamiCube's display properties at run-time as a result of the filtering action.

Parameters

None

Pivot

This event is fired whenever the user manually pivots fields between row and column orientation. This allows capturing the event and changing some DynamiCube's display properties at run-time as a result of the pivoting action.

Parameters

None

Example

```
Private Sub DynamiCube1_Pivot()
‘ Change the first column width
DynamiCube1.ColWidth(0) = 1200
End Sub
```

LoadCompleted

Fired after DynamiCube control is instantiated and initialized. This event can be useful when DynamiCube is used in an ActiveX container that loads its controls asynchronously or one that call it Form_Load() event before fully initializing its controls. User can perform the same pre-data fetching and pre-display property changes as in Form_Load().

Parameters

None

QueryCompleted

Fired after the internal summarization query is executed and all the internal data structures are initialized. It can be used to change display properties before the view is painted.

Parameters

None

Example

```
Private Sub DynamiCube1_QueryCompleted()
‘ Change the first column width
DynamiCube1.ColWidth(0) = 1200
End Sub
```

SelChange

Fired when the user changes the current selection or cursor position using keyboard and mouse movements, or if the values of current selection coordinate's properties were changed through code.

Parameters

None

Example

```
Private Sub DynamiCube1.SelChange()
lblStartRow = DynamiCube1.SelStartRow
lblEndCol = DynamiCube1.SelEndRow
lblStartCol = DyanmiCube1.SelStartCol
lblEndCol = DynamiCube1.SelEndCol
End Sub
```

Methods Reference

DynamiCube Object Methods

AddRow
CalcHit
ColFields
ColHeading
DataFields
DataPosition
DataValue
Fields
GetColDepth
GetNextCol
GetNextRow
GetRowDepth
HiddenFields
Load
Print
PrintPreview
Refresh
RefreshData
RowFields
RowHeading
Save
VisibleFields

Fields Collections Methods

Add
Count
Delete
DeleteAll
Item

Field Object Methods

ShowDetail

DataItems Collection Methods

Count
Delete
Item

AddRow

Used in FetchData event to load data into DynamiCube when used in unbound mode. Rows are added into the array using a single dimensional array of variants. The size and order of the array and its elements should match the number of dimensions and data items used in the DynamiCube view and the order they were added into the view.

Syntax

```
<object>.AddRow(aRow)
```

Argument	Description
object	Valid reference to a DynamiCube object
aRow	Variant Array of dimension values matching the order and number of dimensions and data fields.

Example

```
‘ 1- Setup your fields collection in Form_Load() like
Private Sub Form_Load()
Dim x As Object
Set x = DynaCube1.Fields.Add("product", "Product", DCRow)
x.GroupFooterType = 1
Set x = DynaCube1.Fields.Add("region", "Region", DCColumn)
x.GroupFooterType = 1
Set x = DynaCube1.Fields.Add("Sales", "Sales", DCData)
x.AggregateFunc = 0
2- Set dcConnectType to DCCT_UNBOUND
DynamiCube1.dcConnectType = DCCT_UNBOUND
End Sub
‘ 3- Add the following to the FetchData event handler which is called
once for all data to be loaded:
Private Sub FetchData()
Dim valArray(3)
valArray(0) = "Product2"
valArray(1) = "Africa"
valArray(2) = 123
DynaCube1.AddRow (valArray)
valArray(0) = "Product1"
valArray(1) = "Africa"
valArray(2) = 123
DynaCube1.AddRow (valArray)
End Sub
```

See Also

FetchData , DCConnectType

CalcHit

Creates a 'HitRec' object.

Syntax

<object>.CalcHit

Argument	Description
object	Valid reference to a DynamiCube object

Example

```
Private Sub dcdemo_MouseDown(Button As Integer, Shift As Integer, X As
Single, Y As Single)
Dim HitRec As Object
      On Error Resume Next
      Set HitRec = dcdemo.CalcHit
      If Button = vbRightButton Then
            ' First select the cell under the mouse
            If HitRec.Area = 1 Then
                  dcdemo.SelStartCol = HitRec.column
                  dcdemo.SelEndCol = HitRec.column
                  dcdemo.SelStartRow = HitRec.row
                  dcdemo.SelEndRow = HitRec.row
                  dcdemo.Refresh
            End If
      End If
      On Error GoTo 0
End Sub
```

ColFields

Returns a collection of DynamiCube fields displayed as columns in the current view.

Syntax

```
<object>.ColFields
```

Argument	Description
object	Valid reference to a DynamiCube object

Example

```
For I = 0 to DynamiCube1.ColFields.Count
lstCols.AddItem DynamiCube1.ColFields(I).Name
Next
```

ColHeading, RowHeading

Returns column or row heading string of the specified column or row at the specified depth.

Syntax

```
<sHeading> = <object>.ColHeading(<iPos>, <iDepth>)
```

Argument	Description
object	Valid reference to a DynamiCube object
sHeading	Returned heading string
iPos	Column or row position
iDepth	Depth of the requested heading.

DataFields

Returns a collection of DynamiCube fields displayed as data fields in the current view.

Syntax

```
<object>.DataFields
```

Argument	Description
object	Valid reference to a DynamiCube object

Example

```
For I = 0 to DynamiCube1.DataFields.Count
lstCols.AddItem DynamiCube1.DataFields(I).Name
Next
```

DataPosition

Returns the data field index at the requested row and column coordinates.

Syntax

```
<object>.DataPosition(<iRow>, <iCol>)
```

Argument	Description
Object	Valid reference to a DynamiCube object
Irow	Row coordinate of the requested cell data position
Icol	Column coordinate of the requested data position

Example

```
sCurrentCellName = _
DynamiCube1.DataFields(DynamiCube1.DataPosition(0, 0)).Name
```

DataValue

Returns data value in the cell at the requested row and column coordinates.

Syntax

```
<object>.DataValue(<iRow>, <iCol>)
```

Argument	Description
object	Valid reference to a DynamiCube object
iRow	Row coordinate of the requested cell data value
iCol	Column coordinate of the requested data value

Example

```
rows = DynamiCube1.SelEndRow
cols = DynamiCube1.SelEndCol
col = DynamiCube1.SelStartCol
i = 1
‘ For each column in the selection
Do While col <= cols
row = DynamiCube1.SelStartRow
j = 1
‘ For each row in the selection
Do While row <= rows
      grfDemo.ThisSet = j
      grfDemo.ThisPoint = i
      ‘ Set the graph data point to cell value
      grfDemo.GraphData = DynamiCube1.DataValue(row, col)
      ‘ Get the next row in the selection
      row = DynamiCube1.GetNextRow(row)
      j = j + 1
Loop
i = i + 1
‘ Get the next column in the selection
col = DynamiCube1.GetNextColumn(col)
Loop
```

Fields

Returns a collection of DynamiCube all fields in the current view.

Syntax

```
<object>.Fields
```

Argument	Description
object	Valid reference to a DynamiCube object

Example

```
For I = 0 to DynamiCube1.Fields.Count
lstCols.AddItem DynamiCube1.Fields(I).Name
Next
```

GetColDepth

Returns the level number of the requested column. It accounts for any rolled up fields and summary lines. In a view where there are three (3) column fields and none of the fields are rolled up, each data point cell will have a column depth of 3.

Syntax

```
<object>.GetColDepth(<iCol>)
```

Argument	Description
Object	Valid reference to a DynamiCube object
ICol	Requested column coordinate.

Example

```
i = dc.SelStartCol
Do While i <= dc.SelEndCol
iPoints = iPoints + 1
ReDim Preserve aLabels(iPoints)
‘ For each column level
For j = 1 To dc.GetColumnDepth(i)
      ‘ Concatenate Headings to create label
      aLabels(iPoints) = aLabels(iPoints) & _
      dc.ColumnHeading(i, j) & " "
Next
i = dc.GetNextColumn(i)
Loop
```

GetNextCol

Returns the next visible column in the current view following the requested column. It accounts for any rolled up fields and hidden detail levels.

Syntax

```
<object>.GetNextCol(<iCol>)
```

Argument	Description
object	Valid reference to a DynamiCube object
iCol	Requested column coordinate.

Example

```
rows = dc.SelEndRow
cols = dc.SelEndCol
col = dc.SelStartCol
i = 1
‘ For each column in the selection
Do While col <= cols
row = dc.SelStartRow
j = 1
‘ For each row in the selection
Do While row <= rows
      grfDemo.ThisSet = j
      grfDemo.ThisPoint = i
      ‘ Set the graph data point to cell value
      grfDemo.GraphData = dc.DataValue(row, col)
      ‘ Get the next row in the selection
      row = dc.GetNextRow(row)
      j = j + 1
Loop
i = i + 1
‘ Get the next column in the selection
col = dc.GetNextColumn(col)
Loop
```

GetNextRow

Returns the next row in the current selection following the requested row. It accounts for any rolled up fields and hidden detail levels.

Syntax

```
<object>.GetNextRow(<iRow>)
```

Argument	Description
object	Valid reference to a DynamiCube object
iRow	Requested row coordinate.

Example

```
rows = dc.SelEndRow
cols = dc.SelEndCol
col = dc.SelStartCol
i = 1
‘ For each column in the selection
Do While col <= cols
row = dc.SelStartRow
j = 1
‘ For each row in the selection
Do While row <= rows
      grfDemo.ThisSet = j
      grfDemo.ThisPoint = i
      ‘ Set the graph data point to cell value
      grfDemo.GraphData = dc.DataValue(row, col)
      ‘ Get the next row in the selection
      row = dc.GetNextRow(row)
      j = j + 1
Loop
i = i + 1
‘ Get the next column in the selection
col = dc.GetNextColumn(col)
Loop
```

GetRowDepth

Returns the level number of the requested row. It accounts for any rolled up fields and summary lines. In a view where there are three (3) row fields and none of the fields are rolled up, each data point cell will have a row depth of 3.

Syntax

```
<object>.GetRowDepth(<iRow>)
```

Example

```
‘ Start at the top of the current selection
i = dc.SelStartRow
‘ For each selected row
Do While i <= dc.SelEndRow
iSets = iSets + 1
ReDim Preserve aLegends(iSets)
‘ For each level of the current row
For j = 1 To dc.GetRowDepth(i)
      ‘ Concatenate the heading to create legend
      aLegends(iSets) = aLegends(iSets) & _
      dc.RowHeading(i, j) & " "
Next
‘ Get the next row number in the selection
i = dc.GetNextRow(i)
Loop
```

HiddenFields

Returns a collection of DynamiCube fields not visible in the current view.

Syntax

```
<object>.HiddenFields
```

Argument	Description
object	Valid reference to a DynamiCube object

Example

```
For I = 0 to DynamiCube1.HiddenFields.Count
lstCols.AddItem DynamiCube1.HiddenFields(I).Name
Next
```

Load

Loads DynamiCube data from a local microCube file.

Syntax

```
<object>.Load <sFileName>
```

Argument	Description
object	Valid reference to a DynamiCube object
sFileName	A qualified DOS path to local microCube file

Example

```
DynamiCube1.Load App.Path & "\MYCUBE.CUB"
DynamiCube1.Refresh
```

Print

Prints current Dynamic Cube view to the printer.

Syntax

```
<bResult> = <object>.Print(<bDispPrintDialog>, <iFromPage>, <iToPage>)
```

Argument	Description
object	Valid reference to a DynamiCube object
bResult	Returns True if printing was successful, False otherwise
bDispPrintDialog	Boolean - Displays a printer setup dialog if True. iFromPage and iToPage parameters are ignored if the dialog is displayed. Defaults to False.
iFromPage	Page number to start printing. Ignored if bDispPrintDialog is True. Defaults to first page.
iToPage	Page number to stop printing at. Ignored if bDispPrintDialog is True. Defaults to last page.

Example

```
‘ Print first two pages without displaying a print dialog
bRC = DynamiCube1.Print(False, 1, 2)
```

PrintPreview

Displays a non-modal print preview window of the current Dynamic Cube view.

Syntax

```
<object>.PrintPreview
```

Argument	Description
object	Valid reference to a DynamiCube object

Example

```
‘ open print preview window
DynamiCube1.PrintPreview
```

Refresh

Updates the DynamiCube display to reflect the latest setting changes.

Syntax

```
<object>.Refresh
```

Argument	Description
object	Valid reference to a DynamiCube object

Example

```
‘ Change background color and refresh the display
DynamiCube1.BackColor = vbYellow
DynamiCube1.Refresh
```

RefreshData

Refreshed the current data set. Used after adding a fields to the visible fields collection which would require the aggregation query to be executed again and the data set to be updated.

Syntax

```
<object>.RefreshData
```

Argument	Description
Object	Valid reference to a DynamiCube object

Example

```
DynamiCube1.Fields.Add "Region"
DynamiCube1.Fields("Region").Orientation = 1
DynamiCube1.RefreshData
```

RowFields

Returns a collection of DynamiCube fields displayed as rows in the current view.

Syntax

```
<object>.RowFields
```

Argument	Description
object	Valid reference to a DynamiCube object

Example

```
For I = 0 to DynamiCube1.RowFields.Count
lstCols.AddItem DynamiCube1.RowFields(I).Name
Next
```

Save

Saves the current DynamiCube layout and data to a local microCube file. The file can be loaded and viewed using with the load method or the DataPath property.

Syntax

```
<object>.Save <sFileName>, <iSaveLayout>
```

Argument	Description
object	Valid reference to a DynamiCube object
sFileName	A qualified DOS path to a local file where the data will be saved
iSaveLayout	Valid Option (as shown below)

Options:

```
0-Save only an uncompressed layout
1-Save an uncompressed layout with data
2-Save only a compressed layout
3-Save a compressed layout with data
```

Example

```
‘ Save the current data and layout

DynamiCube1.Save App.Path & "\MYCUBE.CUB", 1
```

VisibleFields

Returns a collection of DynamiCube fields displayed in the current view.

Syntax

```
<object>.VisibleFields
```

Argument	Description
object	Valid reference to a DynamiCube object

Example

```
For I = 0 to DynamiCube1.VisibleFields.Count
lstCols.AddItem DynamiCube1.VisibleFields(I).Name
Next
```

Add

Adds a new field to the collection.

Syntax

```
<fields>.Add <sName>, <sCaption>, <iOrientation>
```

Argument	Description
fields	Valid reference to a DynamiCube field collection.
sName	Name of the field as it appears in the data source recordset. This is used as a key in the collection.
sCaption	Caption of the field as it would be displayed in the DynamiCube view
iOrientation	Orientation property of the field

Example

```
'SETTING DynamiCube PROPERTIES
Cube1.DCConnectType = DCCT_DAO
Cube1.DCConnect = "Access"
Cube1.DCDataBaseName = "C:\DYNACUBE\samples\dcdemo.mdb"
Cube1.DCRecordSource = "budget"

'ADDING THE FIELDS
Set x = DCube1.Fields.Add("Company", "Company", 2)
Set x = DCube1.Fields.Add("Year", "Year", 1)
Set x = DCube1.Fields.Add("Budget", "Budget", 3)
x.AggregateFunc = 0
DynamiCube1.RefreshData
```

Count

Returns number of elements in the collection.

Syntax

```
<object>.Count
```

Argument	Description
object	Valid reference to a DynamiCube fields or items collection.

Example

```
For I = 0 to DynamiCube1.Fields.Count
lstFields.AddItem DynamiCube1.Fields(I).Name
Next
```

Delete

Deletes an element from the collection.

Syntax

```
<object>.Delete(<vElement>)
```

Argument	Description
object	Valid reference to a DynamiCube fields or items collection.
vElement	Field name reference

Example

```
DynamiCube1.Fields.Delete("Region")
DynamiCube1.RefreshData
```

DeleteAll

Deletes all elements in the fields or items collection.

Syntax

```
<object>.DeleteAll
```

Argument	Description
object	Valid reference to a DynamiCube fields or items collection.

Example

```
DynamiCube1.Fields.DeleteAll
```

Item

Returns a field object. This is useful for Delphi and PowerBuilder users where the reference to collection element does not work.

Syntax:

<object>.Item

Argument	Description
object	Valid reference to a DynamiCube fields collection.

Example

```
Debug.Print DynamiCube1.Fields.Item(0).Caption
```

ShowDetail

Defines or determines if the current field detail (sub levels of the outline) is visible or not. This method can be used to set the initial view of the DynamiCube to a summary view where all data items are collapsed. It is to be used after the data is loaded into DynamiCube in response to a QueryCompleted event.

Syntax

```
<object>.ShowDetail = <bDetail>
```

Argument	Description
object	Valid reference to a DynamiCube item object
bDetail	Boolean. Sub levels are displayed if true

Example

```
Private Sub QueryCompleted()
‘ Hide all details at the top level to start the cube in a summary view
DynamiCube1.RowFields(0).ShowDetail(False)
DynamiCube1.ColFields(0).ShowDetail(False)
End Sub
```

Count

Returns number of elements in the collection.

Syntax

```
<object>.Count
```

Argument	Description
object	Valid reference to a DynamiCube DataItems collection.

Example

```
For I = 0 to DynamiCube1.Fields(0).DataItems.Count
lstFields.AddItem DynamiCube1.Fields(I).DataItems.Caption
Next
```

Delete

Deletes an element from the collection.

Syntax

```
<object>.Delete(<vElement>)
```

Argument	Description
object	Valid reference to a DynamiCube fields or items collection.
vElement	Either integer element index or key in the collection

Example

```
DynamiCube1.Fields(0).DataItems("Africa").Delete
DynamiCube1.RefreshData
```

Item

Returns a DataItem object. This is useful for Delphi and PowerBuilder users where the reference to collection element does not work.

Syntax:

<object>.Item

Argument	Description
object	Valid reference to a DynamiCube fields or items collection.

Example

```
Debug.Print DynamiCube1.Fields(0).Item.Caption
```

Appendices

Optimizing DynamiCube

You can achieve better speed in the loading, displaying and pivoting operations of the DynamiCube by following these steps:

1. Create indexes on any fields that you wish to include as rows or columns in a DynamiCube view. Since aggregate SQL statements run against the data source, indexes will speed the retrieval of the record set.
2. Provide a filtered record set of the data that the user requests to view. For example, use this if your tables contain sales records for ten (10) years and you know the user will not request to view the complete data set. You can create forms that asking the user how many or which years will be used in viewing. Then, build your SQL statement based on their answers.
3. Minimize the number of dimensions in a single view, providing a drill down feature that displays another cube based on a selection from the current cube. For example, if you need to analyze ten (10) dimensions, you can improve usability and speed by splitting the dimensions into two levels with five (5) dimensions each. When the user double clicks on a cell in the first view, display another filtered view with less data and less dimensions. This reduces the complexity and increases the comprehension of the displayed result set.
4. Finally, if your record set is very large and client machines are not powerful enough for good performance. You can create pre-joined and summarized table through a batch process and let that table be the record source. For example, if your sales data is retrieved through a join of customer, product, category, invoice, and line item tables, you can run a SQL aggregate query on a join of all these tables. This will create a single table with only the fields that are needed as dimensions or data fields.

Technical Specification

System Requirements

Windows 95 or Windows NT compatible machine.
3 MB Hard Drive Space.
8 MB RAM (16 Recommended).

Included Files

DCUBE.OCX	DynamiCube OLE Custom Control
DCUBE.LIC	DynamiCube license file. Not for redistribution.
DCUBEPP.DLL	DynamiCube's Design-Time Property Sheets
DYNAODBC.DLL	DynamiCube ODBC support DLL.
DYNABDE.DLL	DynamiCube BDE support DLL
README.TXT	Latest updates to this manual
DCUBE.HLP	DynamiCube Help File
*.VBP, *.FR?, *.BAS	Sample files

Technical Support

Problems?

If you are having problems using the DynamiCube ActiveX, please make sure that the control was registered by the installation program. If not, use the RegSvr32.exe program to register the DCube.ocx file and the DCubepp.dll. The REGSVR32.EXE is included with your Visual Basic installation.

If the problem is not fixed, and a solution is not listed in the README.TXT file included with your installation, please contact our technical support staff. You can reach Data Dynamics Technical Support via:

Email:	support@datadynamics.com
Internet:	http://www.datadynamics.com
Fax:	614.899.2943
Telephone:	614.895.3142 (8 a.m. to 5 p.m. EST, M-F)

Please include a complete description of the problem and the version of the DynamiCube.

Technical Support options

Basic Support: $0

- Free telephone support for 30 days after registration
- Free updates to current version for first 90 days
- Unlimited email support
- Must pay $149 to get current version after initial 90 days
    - Receive additional releases for 90 days after payment

Subscription Service: $199 per year

- Includes Basic support
- Free updates
    - For 15 months if purchased with product
    - For 12 months if purchased subsequent to product

Gold Support: $349 per year

- Includes subscription
- Priority unlimited telephone and/or email support for 13 months
- Maximum 8 hour response time
- Eligible for Beta Testing

We welcome suggestions

We at Data Dynamics welcome your suggestions for improving DynamiCube. Much of the initial feedback has been included in this version of DynamiCube.

Fax your suggestions to us at 614.899.2943, email to support@datadynamics.com, or write to Data Dynamics, 2600 Tiller Lane, Columbus, OH 43231.

Disk Defects Warranty

Data Dynamics is committed to producing a quality product that undergoes an extensive series of tests and refinements at both the manufacturing and development levels. In the unfortunate case that you receive a damaged disk, Data Dynamics will replace your disk free of charge. Please contact us at the above address to get your replacement disks.