Contents

Finding the Information You Need

Part I BusinessObjects

Chapter 1 BusinessObjects Object Model 27

Part II Designer

Chapter 2 Designer Object Model 255

Part III Report Viewer Component

Chapter 3 Report Viewer Component Object Model 429

Part IV Appendix

Appendix A Obsolete Properties and Methods 455
Finding the Information You Need

In this chapter

- It's in the Documentation vi
  - A Documentation Service on the Web vi
  - Multimedia vii
  - Online Guides viii
  - Online Help x
  - What to Do for More Information x

- About this Guide xi
  - Audience xi
  - Conventions Used in this Guide xi
  - Programming Resources xii
  - Developer Suite Online xvi
Finding the Information You Need

It’s in the Documentation

Business Objects documentation continues to deliver timely and indepth coverage of product information. Not just facts about product features, but a world of knowledge in the way of tips, samples, and troubleshooting instructions.

For your convenience, Business Objects documentation comes in a variety of formats including Windows online help, HTML, Acrobat PDF, paper, and multimedia. What’s more, you can gain quick access to it at any time directly from the product you are working with.

Documentation has been carefully designed to meet your needs for speed and ease of navigation. All the information you need is there just a few mouse clicks away.

The next sections highlight the key features of our documentation.

A Documentation Service on the Web

From the Help menu of all our products, you can check out More Tips and Samples, the Business Objects documentation service on the Internet. From here, you can discover the latest updates, tips, samples, and troubleshooting.

You can also get there by pointing your browser to the following URL:

http://www.businessobjects.com/services/infocenter

From the Tips and Tricks page, registered customer support contacts can explore the electronic version of the Business Objects documentation set. It offers extensive information on all Business Objects products, updates, troubleshooting, tips, and much more.

In addition, registered DEVELOPER SUITE customers can download new documentation and code samples.
Multimedia

Business Objects multimedia comprises the BUSINESSOBJECTS Quick Tour, the INFOVIEW Quick Tour, and the BUSINESSMINER tutorial, all of which cover the essential features of these products.

The BusinessObjects Quick Tour
The BUSINESSOBJECTS Quick Tour is a multimedia presentation that takes you on a guided tour of the key features of BUSINESSOBJECTS. Its didactic approach makes it an ideal primer for first-time users of the product.

You may wish to use it as an accompaniment to the guide Getting Started with BusinessObjects.

The InfoView Quick Tour
The INFOVIEW Quick Tour is a multimedia presentation that highlights the key features of INFOVIEW. Intended primarily for new users, it offers an overview of all the features necessary for managing and distributing documents.

The INFOVIEW Quick Tour can be used as an accompaniment to the guide Getting Started with WebIntelligence.

The BusinessMiner Tutorial
This multimedia tutorial teaches novice users how to use the powerful desktop datamining software, BUSINESSMINER. Each lesson in this tutorial has a narrated, animated presentation which shows users how to answer a business question using BUSINESSMINER. Users can then try out the demonstrated tasks themselves by following the step-by-step exercises in the accompanying guide.
Online Guides

User’s Guides
All user’s guides are available as Acrobat Portable Document Format (PDF) files. Designed for online reading, PDF files enable you to view, navigate through, or print any of their contents. The full list of Business Objects guides is provided in the Deployment Guide.

From a Business Objects product, you can open a guide from the commands of the Help menu.

The Help menu of BusinessObjects provides commands for viewing documentation.

During installation, the BUSINESSOBJECTS installer program automatically copies these files to:

Business Objects\BusinessObjects 5.0\Online Guides\En

You can open a document from the Help menu provided that you have installed the Adobe Acrobat Reader, version 3.0 or higher on your machine. This Reader is available on the Business Objects CD-ROM. You can also download it for free from the Web site of Adobe Corporation.
The Error Message Guide

The Error Message Guide is a compilation of the error messages that can appear with ordinary use of Business Objects products.

This guide provides you with detailed troubleshooting information so that you can determine the reasons for an error and take the appropriate steps to resolve it. It allows you to search for error messages by code. Each error message appears with its probable cause and the recommended course of action.

You can open this online guide from any of the Business Objects products by selecting the Error Messages Explained command from the Help menu. From InfoView, click Error Messages in the navigation bar.

Click the List of Error Messages bookmark to view the error messages by code.

The Error Message Guide in PDF format.

Click an error message to go to its explanation in the guide.
Finding the Information You Need

Online Help
For Business Objects Windows desktop products, online help is available in the form of .hlp and .cnt files that comply with the standards of Microsoft Windows online help.
From INFOVIEW, online help is available for both INFOVIEW and WEBINTELLIGENCE.

What to Do for More Information
If you cannot find the information you are looking for, then we encourage you to let us know as soon as you can. We welcome any requests, tips, suggestions, or comments you may have regarding the contents of this or other Business Objects documentation. You can contact us by e-mail at:
documentation@businessobjects.com
To find out information about Business Objects products and services, visit our Web site at:
http://www.businessobjects.com
About this Guide

This guide details the BUSINESSOBJECTS object and event model and the DESIGNER object model. These features allow you to tailor your data presentation and retrieval for creating a powerful business intelligence solution using BUSINESSOBJECTS.

The events and classes described in this guide are listed alphabetically.

Audience

This guide is intended for people using the BUSINESSOBJECTS SDK.
To use the BUSINESSOBJECTS SDK you should have a working knowledge of Visual Basic for Applications (VBA). You should also be familiar with BUSINESSOBJECTS, DESIGNER and the type of data and the databases in your organization.

Conventions Used in this Guide

The conventions used in this guide are described in the table below.

<table>
<thead>
<tr>
<th>Convention</th>
<th>How Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALL CAPITALS</td>
<td>The names of all products such as BUSINESSOBJECTS, WEBINTELLIGENCE, SUPERVISOR, DESIGNER.</td>
</tr>
<tr>
<td>This font</td>
<td>Code, SQL syntax, computer programs. For example: @Select(Country\Country_Id)</td>
</tr>
</tbody>
</table>
### Programming Resources

**Developer Suite**

**BUSINESSOBJECTS DEVELOPER SUITE** includes the following programming resources:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVELOPER SUITE ONLINE</td>
<td>a web site containing documentation, programming samples, and tips</td>
</tr>
<tr>
<td>Introduction to Developer Suite</td>
<td>describes how to customize BUSINESSOBJECTS and WEBINTELLIGENCE with DEVELOPER SUITE</td>
</tr>
<tr>
<td>BusinessObjects SDK Reference Guide</td>
<td>details on the object models for BUSINESSOBJECTS, DESIGNER and the Report Viewer Component</td>
</tr>
<tr>
<td>WebIntelligence SDK Reference Guide</td>
<td>details on the object models for WEBINTELLIGENCE</td>
</tr>
<tr>
<td>Installing and Configuring Application Servers</td>
<td>describes how to install and configure the supported application servers</td>
</tr>
<tr>
<td>BusinessObjects SDK Object Model Diagram</td>
<td>object model diagrams for BUSINESSOBJECTS, DESIGNER and the Report Viewer Component</td>
</tr>
<tr>
<td>WebIntelligence SDK Object Model Diagram</td>
<td>object model diagrams for WEBINTELLIGENCE</td>
</tr>
<tr>
<td>BusinessObjects SDK Online Help</td>
<td>online details on the object models for BUSINESSOBJECTS, and DESIGNER</td>
</tr>
<tr>
<td>README</td>
<td>release information</td>
</tr>
</tbody>
</table>
### WebIntelligence JSP Tag Library

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebIntelligence JSP Tag Library Guide</td>
<td>how to use the WebIntelligence JSP Tag Library</td>
</tr>
<tr>
<td>Using the WebIntelligence JSP Tag Library Configuration Tool</td>
<td>how to use the configuration tool for the WebIntelligence JSP Tag Library</td>
</tr>
<tr>
<td>WebIntelligence JSP Tag Library Configuration Tool README</td>
<td>release information for the WebIntelligence JSP Tag Library Configuration Tool</td>
</tr>
</tbody>
</table>

### Samples

The following samples are available through Developer Suite Online:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP Samples</td>
<td></td>
</tr>
<tr>
<td>WebIntelligence JSP</td>
<td>an implementation of INFOVIEW and WEBINTELLIGENCE using JSP that you can use as a basis for your own web application</td>
</tr>
<tr>
<td>Creating a Session</td>
<td>creating a WEBINTELLIGENCE session and listing the documents in the Corporate repository</td>
</tr>
<tr>
<td>Creating a Document</td>
<td>creating a document with the Web Panel</td>
</tr>
<tr>
<td>Displaying a Document</td>
<td>displaying a document in standard or custom formats</td>
</tr>
<tr>
<td>Drilling in a Document</td>
<td>displaying a drillable document</td>
</tr>
<tr>
<td>Publishing a Document</td>
<td>publishing a document to the Corporate repository</td>
</tr>
<tr>
<td>Scheduling a Document</td>
<td>scheduling a document with WebIntelligence SDK and BROADCAST AGENT</td>
</tr>
</tbody>
</table>
Other Business Objects documentation

You might also find the documentation of other Business Objects products useful. All Business Objects documentation is available via the Online Customer Support website. For information on accessing this website see page xvi.

WebIntelligence System Administrator’s Guide

The WebIntelligence System Administrator’s Guide discusses how a standard deployment of WebIntelligence works. You should be familiar with this guide before you begin customizing WebIntelligence.

System administration

Business Objects also publishes the following system administration guides:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast Agent Administrator’s Guide</td>
<td>explains how to set up and run Broadcast Agent</td>
</tr>
<tr>
<td>Deployment Guide</td>
<td>discusses how to deploy BusinessObjects, WebIntelligence and Broadcast Agent</td>
</tr>
</tbody>
</table>
User guides
The following guides describe WEBINTELLIGENCE for end users.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebIntelligence User’s Guide</td>
<td>describes how to use WEBINTELLIGENCE to create, format, analyze, and publish reports</td>
</tr>
<tr>
<td>BusinessObjects User’s Guide</td>
<td>describes how to use BUSINESSOBJECTS to create, format, analyze, and publish reports</td>
</tr>
<tr>
<td>InfoView User’s Guide</td>
<td>describes how to view, manage and distribute documents on the corporate repository</td>
</tr>
</tbody>
</table>
Finding the Information You Need

Developer Suite Online

DEVELOPER SUITE ONLINE is an information resource for developers provided on the Online Customer Support web site. From this web site you can access:

- electronic versions of the DEVELOPER SUITE documentation
- regularly updated sample programs that demonstrate possible uses for DEVELOPER SUITE
- regularly updated tips and tricks on using DEVELOPER SUITE

You can contribute to this web site by sending in your own tips.

Accessing Developer Suite Online

You access DEVELOPER SUITE ONLINE via the Online Customer Support website. To access this web site, you must be a registered user of Worldwide Customer Support. For details about registering with Worldwide Customer Support go to www.businessobjects.com/services/

To access DEVELOPER SUITE ONLINE after you have registered with Worldwide Customer Support:

1. In a web browser, go to: www.techsupport.businessobjects.com
2. Click Enter.
3. Enter your login name and password. You get these when you register with Worldwide Customer Support.
5. Enter your DEVELOPER SUITE product license key. You only have to enter the product license key the first time you use DEVELOPER SUITE ONLINE.

Once your key has been verified, you can view the DEVELOPER SUITE ONLINE Web site.
To access DEVELOPER SUITE ONLINE after you have registered with Worldwide Customer Support:

1. In a web browser, go to www.techsupport.businessobjects.com

2. Enter your login name and password. You get these when you register with Worldwide Customer Support.

3. Click on the Developer Suite link of the Knowledge Base section.

4. Enter your DEVELOPER SUITE product licence key. You only have to enter the product licence key the first time you use DEVELOPER SUITE ONLINE.

Once your key has been verified, you can view the DEVELOPER SUITE ONLINE web site.
Part I
BusinessObjects

www.businessobjects.com/services/infocenter
## Class List for BusinessObjects

### Application Events 28
- DocumentActivate .................. 28
- DocumentAfterRefresh ........ 28
- DocumentBeforeClose ............ 29
- DocumentBeforeRefresh .......... 29
- DocumentBeforeSave .............. 30
- DocumentDeactivate .............. 31
- DocumentOpen ....................... 31
- NewDocument ........................ 31

### Application Class 32
- ActiveDocument Property ..... 32
- ActiveReport Property ...... 33
- Application Property .......... 33
- BreakOnVBAError Property ... 33
- Clipboard Property ............... 34
- CmdBars Property ................. 34
- Documents Property .............. 34
- ExchangeDomain Property ... 35
- ExchangeMode Property ...... 35
- ExecuteMacro Method .......... 36
- GetInstallDirectory Method ... 36
- GetVersion Method ............ 37
- Interactive Property .......... 38
- LoginAs Method ................. 38
- Name Property .................. 39
- Parent Property ................. 39
- Quit Method ..................... 39
- RegisterDPVBAProc Method 40
- UnregisterDPVBAProc Method  40
- SetInstallDirectory Method ... 41
- SetSecurityPrompt Method ... 41
- Universes Property .......... 42
- Variables Property .............. 42
- VBE Property .................. 43
- Version Property .................. 44
- Visible Property ................. 44
- Window Property ................. 44

### BlockStructure Class 45
- Application Property .......... 45
- ApplyStdStyle Method .......... 45
- Clear Method ..................... 45
- Delete Method .................... 45
- Name Property .................. 46
- Parent Property .................. 46
- Pivot Property ................. 46
- Type Property ................... 46

### CellStructure Class 47
- Application Property .......... 47
- ApplyStdStyle Method .......... 47
- Clear Method ..................... 47
- Delete Method .................... 47
- Parent Property .................. 48
- Type Property ................... 48
- ValueType Property ............ 49
- Variable Property .............. 49

### Class Class 50
- Application Property .......... 50
- Classes Property ................. 50
- Description Property .......... 50
- Name Property .................. 51
- Objects Property ............... 51
- Parent Property ................. 51
- PredefinedConditions Property 51

### Clipboard Class 54
- Application Property .......... 54
- Clear Method ..................... 55
- GetData Method ................. 55
- GetFormat Method .............. 55
- GetText Method ................. 56
- Parent Property .................. 56
- SetData Method .................. 56
- SetText Method .................. 57

### CmdBar Class 58
- Application Property .......... 58
- BuiltIn Property ................. 58
- Controls Property .............. 59
- Delete Method .................... 59
- Enabled Property ............... 59
- Name Property .................. 59
- Parent Property ................. 60
- Position Property ............... 60
- ShowPopup Method ............. 60
- Type Property .................. 61
- Visible Property ................. 61

### CmdBarButton Class 62
- BuiltInFace Property .......... 62
- CopyFace Method ................. 63
- Faceld Property ................. 63
- PasteFace Method ............... 63

### CmdBarControl Class 64
- Application Property .......... 64
- BuiltIn Property ................. 65
- Caption Property ............... 65
- Delete Method .................... 65
- DescriptionText Property ...... 66
- Execute Method .................. 66
- ID Property ..................... 66
BusinessObjects

Index Property..........................66
OnAction Property ..................67
Parent Property........................67
TooltipText Property ...............67
Type Property...........................68

CmdBarControls Class 69
Add Method .............................70
Application Property...............71
Count Property.........................71
Item Property............................71
Parent Property........................72

CmdBarPopup Class 73
CmdBar Property .....................73
Controls Property ....................73

CmdBars Class 74
ActiveMenuBar Property............75
Add Method .............................75
Application Property ..............76
Count Property.........................76
DisplayKeysInTooltips Property ..........76
DisplayTooltips Property ....76
Item Property............................77
LargeButtons Property............77
Parent Property........................77

Column Class 78
Application Property ..............78
Count Property.........................78
CustomSort Method .................78
Item Property............................79
Name Property...........................79
Parent Property........................79

Columns Class 80
Application Property ..............80
Count Property.........................80
Item Property............................80
Parent Property........................81

Condition Class 82
Application Property ..............82
Class Property........................82
Level Property.........................83
Object Property.......................83
Operand Property.....................83
OperandCount Property ..........84
OperandType Property ..............84
Operator Property....................84
Parent Property........................85

Conditions Class 86
Add Method..............................86
Application Property ..............87
Count Property.........................87
Format Property.......................87
Item Property............................89
Parent Property........................89
Remove Method ......................89

Daily Class 90
Application Property ..............90
DayOfWeek Property .................90
Parent Property.......................90
StartTime Property...............91
WeekPeriodicity Property ....91

DataProvider Class 92
Application Property ..............92
ChangeUniverse Method ......93
Columns Property .................93
ConvertTo Method ................93
Edit Method.............................94
ExportToRDBMS Method ....95
GetType Method ....................95
IsEditable Property ...............96
IsRefreshable Property ....97
LastExecutionTime Property ....97
Load Method.........................98
MaxDuration Property ..........98
MaxNbLines Property ..........99
Name Property.......................99
NbCubes Property.................99
NbRowsFetched Property ....100
Parent Property......................100
Queries Property..............100
Refresh Method.................100
SQL Property...................101
Universe Property..............102
UniverseName Property ......102
Unload Method.....................103

DataProviders Class 104
AddDPVBA Method.............104
AddQueryTechnique Method....105
Application Property............105
Count Property......................106
Item Property........................106
Parent Property......................106

DocAgentOption Class 107
Application Property............107
CategoryList Property ..........107
ContinueOnInvalidCategory Property ..........107
ContinueOnInvalidUser Property ..........107
CustomScript Property ..........108
Daily Property......................108
DistributionFolder Property 108
EndDate Property..................109
Every Property.......................109
FileWatcher Property............109
Hourly Property....................109
Monthly Property..................110
Overwrite Property...............110
Parent Property......................110
Priority Property................110
Refresh Property ................111
RefreshInTheNameOfRecipient Property ................111
ResetToDefault Method ....112
ScheduleMode Property ....114
Send Method.........................114
Server Property......................115
StartDate Property..............115
Title Property .......................... 115
UserDefined Property ............. 116
Users Property ....................... 116
Weekly Property ..................... 116

Document Class ........................ 117
Activate Method .................. 117
ActiveReport Property ........... 117
Application Property ............ 118
Author Property .................... 118
AutoRefreshWhenOpening Property........................................ 119
Close Method ...................... 120
Comments Property ............... 120
DataProviders Property ........ 121
DocAgentOption Property ...... 121
DocumentVariables Property ... 121
Evaluate Method ................... 122
ExecuteMacro Method ........... 124
ExportAsPDF Method ............. 124
ExportSheetsAsHtml Method ... 124
FullName Property ............... 125
GetSecurityPrompts Method.... 126
Installed Property ............... 126
IsAddIn Property ................. 126
Keywords Property ............... 127
LastPrintDate Property ....... 127
LastSaveDate Property ....... 127
Name Property .................... 127
Parent Property .................. 128
PartialResults Property ....... 128
Path Property ...................... 128
PrintDialog Method .............. 129
PrintOut Method ................ 129
Refresh Method .................. 130
Reports Property .................. 130
Save Method ...................... 130
SaveAs Method ................... 131
Saved Property ................. 132
Send Method ..................... 132
Subject Property ............... 134
Title Property .................... 134
Variables Property .............. 135
Windows Property .............. 135

Document Events .......................... 136
Activate .................................. 136
AfterRefresh ......................... 136
BeforeClose ......................... 137
BeforeRefresh ...................... 138
BeforeSave ........................... 139
Deactivate ........................... 139
Open ...................................... 140

Documents Class .......................... 141
Add Method ........................... 141
Application Property .......... 143
Count Property ..................... 143
Item Property ...................... 144
Open Method ......................... 144
Parent Property .................. 145
Receive Method ................... 145

DocumentVariable Class ............... 147
Application Property ............ 147
Delete Method ..................... 147
Formula Property ................ 148
IsDataProviderObject Property . 148
Name Property .................... 148
Parent Property .................. 148
Qualification Property ........ 149
Values Property ................... 149

DocumentVariables Class ............... 151
Add Method ........................... 152
Application Property ........... 152
Count Property .................... 152
Item Property ...................... 152
Parent Property .................. 153

DpVBAColumn Class .......................... 154
Add Method ........................... 154
Aggregation Property ........... 155
Delete Method ..................... 155
Index Property .................... 155
Item Property ...................... 156
MaxWidth Property .............. 156
Name Property ..................... 157
Parent Property .................. 157
Qualification Property ........ 157
Type Property ..................... 158

DpVBAColumns Class .......................... 159
Add Method ........................... 159
AddLine Method ................... 159
Count Property ..................... 160
Item Property ...................... 160
NbLines Property ............... 160
Parent Property .................. 160
SetNbColumns Method .......... 161

DpVBAColumns Class .......................... 162
DpVBAColumns Property ...... 162
Parent Property .................. 162

DpVBA Cubes Class ....................... 163
Count Property ..................... 163
Item Property ...................... 163
Parent Property .................. 163

DpVBA Interface Class ............... 164
CancelOnExit Property ......... 166
CheckDataIntegrity Method .... 166
DpVBA Cubes Property .......... 168
IsEdit Property ................... 168
Parent Property .................. 168
ShowSelectionDialogOnEdit Property .................. 169
UserString Property .......... 169
UserStringCount Property ... 169
Every Class 170
Application Property 170
Day Property 170
Every Property 171
MonthPeriodicity Property 171
Parent Property 172
StartHour Property 172

FileWatcher Class 173
Application Property 173
DeleteFileOnCompletion Property 173
FileWatcherOnly Property 173
FullFileName Property 174
Parent Property 174

Hourly Class 175
Application Property 175
FromHour Property 175
MinutesAfterHour Property 175
Parent Property 175
ToHour Property 176

ListofValues Class 177
Application Property 177
Edit Method 177
Parent Property 177
Purge Method 178
Refresh Method 178
Values Property 178

Monthly Class 179
Application Property 179
DayOfMonth Property 179
Parent Property 179
StartTime Property 180

Object Class 181
Application Property 181
Description Property 181
ListofValues Property 181
Name Property 181

Objects Property 182
Parent Property 182
Qualification Property 182
Type Property 182

Objects Class 184
Application Property 184
Count Property 184
Item Property 184
Parent Property 185

Pivot Class 186
Application Property 186
Apply Method 186
Body Property 186
BodyCount Property 187
Columns Property 187
ColumnsCount Property 187
Parent Property 187
Reset Method 187
Rows Property 188
RowsCount Property 188

PredefinedCondition Class 189
Application Property 189
Description Property 189
Name Property 189
Parent Property 189

PredefinedConditions Class 190
Application Property 190
Count Property 190
Item Property 190
Parent Property 191

Queries Class 192
Add Method 192
Application Property 193
Count Property 193
DeleteTrailingBlanks Property 193
DuplicateRows Property 193
Item Property 194
Parent Property 194
ScopeOfAnalysis Property 194
Sorts Property 195

Query Class 196
Application Property 196
Conditions Property 196
Name Property 196
Operator Property 197
Parent Property 197
Results Property 197

Report Class 198
Activate Method 198
AddComplexFilter Method 199
Application Property 199
ApplyTemplate Method 199
Delete Method 200
DrillMode Property 200
Duplicate Method 200
ExportAsHtml Method 200
ExportAsPDF Method 201
ExportAsRtf Method 202
ExportAsText Method 202
GeneralSectionStructure Property 202
Name Property 203
NumberOfPages Property 203
PageFooter Property 203
PageHeader Property 203
PrintOut Method 204
Reset Method 204

Reports Class 205
Add Method 205
Application Property 205
Count Property 205
CreateQuickReport Method 206
Item Property 206
Item Property .................. 245
Parent Property ............... 245
Chapter 1  BusinessObjects Object Model

In this chapter

- Details of the events, classes and class members in the BusinessObjects Object Model.
Application Events

DocumentActivate

Occurs when the specified document becomes the active document.

**Syntax**

Private Sub Application_DocumentActivate(doc As IDocument)

doc is the interface to the document.

**Comments**

The Activate event can occur only when a document is visible. The Activate event occurs only when you move the focus within BUSINESSOBJECTS. Switching to the VBA editor does not affect the activated status of a document. Closing or opening a dialog box in BUSINESSOBJECTS does not trigger the Activate event.

A document can become active by using the Activate method in code.

**Example**

The following example displays the full name of a document when the document is activated:

Private Sub App_DocumentActivate(ByVal Doc As busobj.IDocument)
    MsgBox Doc.FullName
End Sub

DocumentAfterRefresh

Occurs after the specified document has been refreshed.

**Syntax**

Private Sub Application_DocumentAfterRefresh(doc As IDocument)

doc is the interface to the document.

**Comments**

This event occurs after a document refresh where a refresh corresponds to the following actions:

- Selecting the Refresh Data option from the Data menu
- Clicking the Refresh button.
- Opening a document whose AutoRefreshWhenOpening property is set to TRUE
- Calling the method ActiveDocument.Refresh

**Note:** This event does not occur after ActiveDocument.DataProviders.Item(x).Refresh.
### DocumentBeforeClose

Occurs before the specified document is closed.

**Syntax**

Private Sub Application_DocumentBeforeClose(doc As IDocument)

doc is the interface to the document.

**Comments**

This event corresponds to the following cases:

- Selecting the Close option from the File menu
- Selecting the Login As... option from the Tools menu. This closes all documents.
- Calling the methods ActiveDocument.Close

### DocumentBeforeRefresh

Occurs before the specified document is refreshed.

**Syntax**

Private Sub Application_DocumentBeforeRefresh(doc As IDocument, Cancel As Boolean)

doc is the interface to the document.

Cancel has a value of FALSE if the event has been triggered successfully. It has a value of TRUE otherwise.

**Comments**

This event occurs before a document refresh where a refresh corresponds to the following actions:

- Selecting the Refresh Data option from the Data menu
- Opening a document whose AutoRefreshWhenOpening property is set to TRUE
- Calling the method ActiveDocument.Refresh

**Note:** This event does not occur before ActiveDocument.DataProviders.Item(x).Refresh.
Chapter 1 BusinessObjects Object Model

DocumentBeforeSave

Occurs before the specified document is saved.

Syntax

Private Sub Application_DocumentBeforeSave(doc As IDocument, Cancel As Boolean)

doc is the interface to the document.

Cancel has a value of FALSE if the event has been triggered successfully. It has a value of TRUE otherwise.

Comments

This event corresponds to the following cases:

- Selecting the Save option from the File menu
- Selecting the Save As... option from the File menu. The event occurs after you click the Save button on the Save As dialog box. If the file already exists, the event occurs after you click Yes in the Save As warning box.
- Calling the methods ActiveDocument.Save and ActiveDocument.SaveAs

Note: This event does not occur before publishing a document on Corporate Documents although a temporary copy of the document is saved. Likewise, this event does not occur before saving a file as HTML.
Application Events

**DocumentDeactivate**

Occurs when the specified document is no longer the active document.

**Syntax**

Private Sub Application_DocumentDeactivate(doc As IDocument)

doc is the interface to the document.

**Comments**

The Deactivate event can occur only when a document is visible. The Deactivate event occurs only when you move the focus within BUSINESSOBJECTS. Switching to the VBA editor does not affect the deactivated status of a document. Closing or opening a dialog box in BUSINESSOBJECTS does not trigger the deactivate event. A document can become deactivated when the Activate method is applied to another document.

*Note:* Do not include any window activity (message boxes, forms, etc.) in an implementation of this event. If you do, BUSINESSOBJECTS may fail. See also, “Deactivate” on page 139.

**DocumentOpen**

Occurs when the specified document has been opened.

**Syntax**

Private Sub Application_DocumentOpen(doc As IDocument)

doc is the interface to the document.

**NewDocument**

Occurs when a new document is created.

**Syntax**

Private Sub Application_NewDocument(doc As IDocument)

doc is the interface to the document.
Application Class

This class represents the BUSINESSOBJECTS application. An Application object contains BUSINESSOBJECTS settings and options. For example, you can show or hide BUSINESSOBJECTS (using the Visible property), or change the Exchange Domain.

You can also access top-level objects using properties such as ActiveDocument and ActiveReport.

You will need to use the Application object for developing almost any macro, since it contains all other BUSINESSOBJECTS objects.

Syntax

```vba
Dim var As Application
```

`var` is the name of the Application variable that you declare.

Comments

The properties ActiveDocument and ActiveReport can be used without the Application object qualifier. For example, instead of writing:

```vba
Application.ActiveDocument.Refresh
```

you can write:

```vba
ActiveDocument.Refresh
```

Example

This example displays the name of BUSINESSOBJECTS.

```vba
Dim msgText as String
msgText = Application.Name
MsgBox msgText
```

ActiveDocument Property

The active document.

Definition

`ActiveDocument` As Document (Read-only)

Syntax

```vba
var.ActiveDocument
```

`var` is the name of the Application variable that you declare.

Comments

If there is no active document, then Nothing is returned.

When writing macros, you should use ThisDocument to refer to the document containing the macro rather than Application.ActiveDocument. ThisDocument always refers to the document containing the code being executed, which may not necessarily be the active document.
ActiveReport Property

The active report.

**Definition**
ActiveReport As Report (Read-only)

**Syntax**
```
var.ActiveReport
```

**Comments**
If there is no active report then Nothing is returned.

Application Property

The Application object.

**Definition**
Application As Application (Read-only)

**Syntax**
```
var.Application
```

**Comments**
For the Application class, the object returned is the application itself.

BreakOnVBAError Property

Whether or not the user is notified of VBA errors.

**Definition**
BreakOnVBAError As Boolean (Read/Write)

**Syntax**
```
var.BreakOnVBAError
```

**Comments**
If BreakOnVBAError is set to TRUE, then, when a VBA error occurs, the program breaks. VBA will prompt the user to debug or stop the program.

If BreakOnVBAError is set to FALSE, the program will stop when a VBA error occurs. No notification is given that an error has occurred. (This is useful where a macro is being executed on a server or automatic schedule where debugging is not feasible.)

The default value is TRUE.
Chapter 1 BusinessObjects Object Model

Clipboard Property

The system clipboard object.

**Definition**

Clipboard As Clipboard (Read-only)

**Syntax**

`var.Clipboard`

`var` is the name of the Application variable that you declare.

CmdBars Property

The collection of command bars.

**Definition**

CmdBars As CmdBars (Read-only)

**Syntax**

`var.CmdBars`

`var` is the name of the Application variable that you declare.

**Example**

This example adds a command bar, named “Finance” to BUSINESSOBJECTS:

```vba
Dim FBar As CmdBar
Set FBar = Application.CmdBars.Add("Finance", boBarFloating)
```

Documents Property

The collection of open documents in BUSINESSOBJECTS.

**Definition**

Documents As Documents (Read-only)

**Syntax**

`var.Documents`

`var` is the name of the Application variable that you declare.

**Example**

This example displays the number of opened documents.

```vba
Dim docs As Documents
Set docs = Application.Documents
MsgBox docs.Count
```
Application Class

ExchangeDomain Property

The name of the default domain for the next Send or Retrieve procedure.

Definition

ExchangeDomain As String (Read/Write)

Syntax

var.ExchangeDomain

var is the name of the Application variable that you declare.

See Also

Application.ExchangeMode, Document.Send, Documents.Receive

ExchangeMode Property

The default exchange mode of the next Send or Retrieve procedure.

Definition

ExchangeMode As BoExchangeMode (Read/Write)

Syntax

var.ExchangeMode

var is the name of the Application variable that you declare.

Comments

BoExchangeMode is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Values for BoExchangeMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>boUsermode (=0)</td>
</tr>
<tr>
<td>boRepositoryMode (=1)</td>
</tr>
<tr>
<td>boRepositoryModeNoOverwrite (=2)</td>
</tr>
<tr>
<td>boDocAgentMode (=3)</td>
</tr>
</tbody>
</table>

BusinessObjects SDK Reference Guide  35
ExecuteMacro Method

Executes a VBA macro.

**Definition**

*Sub ExecuteMacro(MacroName As String)*

**Syntax**

`var.ExecuteMacro(macroName)`

`var` is the name of the Application variable that you declare.

`macroName` is a string that contains the name of the macro to execute. It takes the form:

`docName.rep!module.macroName`

**Comments**

This method is also available in the Document class.

For the `macroName` parameter, `docName.rep!` must be included for `Application.ExecuteMacro`. However, `module` may be omitted if it is unambiguous.

For the `Document.ExecuteMacro` method only `macroName` should be given.

GetInstallDirectory Method

The path of the directory where BusinessObjects is installed.

**Definition**

*Function GetInstallDirectory(DirectoryID As BoDirectoryID) As String*

**Syntax**

`var.GetInstallDirectory(dirID)`

`var` is the name of the Application variable that you declare.

`dirID` specifies the installation directory. It is of the type `BoDirectoryID`. `BoDirectoryID` is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Values for BoDirectoryID</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>boBusObjDirectory (=0)</td>
<td>BusinessObjects directory</td>
</tr>
<tr>
<td>boDocumentDirectory (=1)</td>
<td>UserDocs directory</td>
</tr>
<tr>
<td>boTemplateDirectory (=2)</td>
<td>Template directory</td>
</tr>
<tr>
<td>boUniverseDirectory (=3)</td>
<td>Universe directory</td>
</tr>
<tr>
<td>boScriptsDirectory (=4)</td>
<td>Scripts directory</td>
</tr>
<tr>
<td>boLocDataDirectory (=5)</td>
<td>Local Data directory</td>
</tr>
<tr>
<td>boSharedDataDirectory (=6)</td>
<td>Shared Data directory</td>
</tr>
</tbody>
</table>
This example displays the path of the installation directories.

```vba
Dim msg As String
msg = "Installation Directory" & Chr(9) & ": " & _
    Application.GetInstallDirectory(boBusObjDirectory) & Chr(10)
msg = msg & "UserDocuments" & Chr(9) & ": " & _
    Application.GetInstallDirectory(boDocumentDirectory) & Chr(10)
msg = msg & "Templates" & Chr(9) & ": " & _
    Application.GetInstallDirectory(boTemplateDirectory) & Chr(10)
msg = msg & "Universes" & Chr(9) & Chr(9) & ": " & _
    Application.GetInstallDirectory(boUniverseDirectory) & Chr(10)
msg = msg & "Scripts" & Chr(9) & Chr(9) & ": " & _
    Application.GetInstallDirectory(boScriptsDirectory) & Chr(10)
MsgBox msg, 64, "Path of installation directories"
```

### GetVersion Method

The version of BUSINESSOBJECTS.

**Definition**

Function `GetVersion((ByRef) Major As Integer, (ByRef) Minor As Integer, (ByRef) Maintenance As Integer) As Boolean`

**Syntax**

```vba
var.GetVersion(major, minor, maint)
```

*var* is the name of the Application variable that you declare.

*major* is the major version number.

*minor* is the secondary release level.

*maint* is the maintenance release level.

**Example**

This example displays the version of BUSINESSOBJECTS:

```vba
Dim maj As Integer
Dim min As Integer
Dim maint As Integer
Application.GetVersion maj, min, maint
MsgBox maj & "." & min & "." & maint
```
Interactive Property

Whether or not the application accepts actions from the user.

**Definition**

**Interactive** As Boolean (Read/Write)

**Syntax**

```vbnet
var.Interactive
```

`var` is the name of the Application variable that you declare.

**Default**

TRUE

**Comments**

User interaction is allowed when this property is set to TRUE
User interaction is *not* allowed when this property is set to FALSE
This method suppresses confirmative interaction and notification with the user.
Dialog boxes and other application prompts are not displayed when Interactive is set to FALSE. (This is useful where a macro is being executed on a server or automatic schedule where user input is not possible.)

**Note:** This property must be reset to TRUE for users to regain control of BUSINESSOBJECTS.

**Example**

This example sets the Interactive property to FALSE:

```vbnet
Application.Interactive = False
```

LoginAs Method

Changes the current BUSINESSOBJECTS user.

**Definition**

Function **LoginAs**([User As String], [Pass As String], [Offline As Boolean], [RepositoryName As String]) As Boolean

**Syntax**

```vbnet
var/LoginAs([userName], [password], [offline], [repositoryName])
```

`var` is the name of the Application variable that you declare.

`userName` is a string that contains the name of the user under which you are logging in. If this parameter is omitted, the User Identification dialog box is displayed.

`password` is the password of the `userName`. If this parameter is omitted, the User Identification dialog box is displayed.

`offline` is a string that contains TRUE if you wish to work “offline”, without access to a repository. If it contains FALSE you will be connected to a repository. The default value is FALSE.
repositoryName is a string containing the repository name. This may be omitted if there is only one repository.

Comments
If all parameters have not been provided, then the User Identification dialog box is displayed. The fields corresponding to supplied parameters are filled. Returns TRUE if the login has succeeded. An error is returned otherwise.

Name Property

The name of BUSINESSOBJECTS.

Definition
Name As String (Read/Write)

Syntax
var.Name

var is the name of the Application variable that you declare.

Example
This example displays the name of BUSINESSOBJECTS.
MsgBox Application.Name

Parent Property

The creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the Application variable that you declare.

Comments
For the Application class, the object returned is the application itself.

Quit Method

Quits BUSINESSOBJECTS and closes all open documents.

Definition
Sub Quit()

Syntax
var.Quit

var is the name of the Application variable that you declare.

Comments
If you insert this method, the rest of the program is ignored.

Example
This example quits BUSINESSOBJECTS using the global variable Application.
Application.Quit
Chapter 1 BusinessObjects Object Model

### RegisterDPVBAProc Method

Register a data provider.

**Definition**

```vba
Sub RegisterPVBAProc(procName As String, friendlyName As String, description As String)
```

**Syntax**

```vba
var.RegisterDPVBAProc(procName, friendlyName, description)
```

- `var` is the name of the Application variable that you declare.
- `procName` is a string that contains the internal name of the procedure.
- `friendlyName` is a string that contains the name that appears in the combo box.
- `description` is a string that contains the text that appears below the combo box.

**Comments**

This method allows you to register a VBA procedure for creating a data provider. It is triggered on the event Document_Open(). Once a procedure has been registered it appears in the combo box of a BUSINESSOBJECTS wizard with other data access types. The user can execute the procedure from this wizard.

**See Also**

Application.Unregister.

### UnregisterDPVBAProc Method

Unregister a data provider.

**Definition**

```vba
Sub UnregisterPVBAProc(procName As String)
```

**Syntax**

```vba
var.UnregisterDPVBAProc(procName)
```

- `var` is the name of the Application variable that you declare.
- `procName` is a string that contains the internal name of the procedure.

**Comments**

This method allows you to unregister a VBA procedure for creating a data provider. It is triggered on the event Document_Close(). Once a procedure has been unregistered it no longer appears in the combo box of a BUSINESSOBJECTS wizard with other data access types.

**See Also**

Application.Register.
SetInstallDirectory Method

Sets the path of the directory where BUSINESSOBJECTS is installed.

**Definition**

Sub *SetInstallDirectory*(DirectoryID As BoDirectoryID, Path As String, Temporary As Boolean)

**Syntax**

```
var.SetInstallDirectory(dirID, Path, Temporary)
```

*var* is the name of the Application variable that you declare.

*dirID* specifies the directory you want to set. It is of the type BoDirectoryID. BoDirectoryID is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boBusObjDirectory (=0)</td>
<td>BusinessObjects directory</td>
</tr>
<tr>
<td>boDocumentDirectory (=1)</td>
<td>UserDocs directory</td>
</tr>
<tr>
<td>boTemplateDirectory (=2)</td>
<td>Template directory</td>
</tr>
<tr>
<td>boUniverseDirectory (=3)</td>
<td>Universe directory</td>
</tr>
<tr>
<td>boScriptsDirectory (=4)</td>
<td>Scripts directory</td>
</tr>
<tr>
<td>boLocDataDirectory (=5)</td>
<td>Local Data directory</td>
</tr>
<tr>
<td>boSharedDataDirectory (=6)</td>
<td>Shared Data directory</td>
</tr>
</tbody>
</table>

*Path* is the new path you wish to assign to the directory variable.

*Temporary* is a Boolean which determines whether or not changes are lost when BUSINESSOBJECTS is closed. If it is set to TRUE, then changes are only applied while BUSINESSOBJECTS is running. If it is set to FALSE, changes are made permanently.

**See Also**

Application.GetInstallDirectory

SetSecurityPrompt Method

Sets the security prompt for OLAP access.

**Definition**

Sub *SetSecurityPrompt*(UserName As String, UserPasswordVarName As String, UserPassword As String, DBName As String, DBPasswordVarName As String, DBPassword As String)
Syntax

```plaintext
var.SetSecurityPrompt(UserName, UserPasswordVarName, UserPassword, DBName, DBPasswordVarName, DBPassword)
```

*var* is the name of the Application variable that you declare.

*UserName* is a user name for OLAP security.

*UserPasswordVarName* is a variable obtained from SecurityPrompts.GetPrompt.

*UserPassword* is a password for OLAP security.

*DBName* is a variable obtained from SecurityPrompts.GetPrompt.

*DBPasswordVarName* is a variable obtained from SecurityPrompts.GetPrompt.

*DBPassword* is a database password for an OLAP access pack.

Comments

You can find more information on OLAP access security in the documentation accompanying your access pack.

Universes Property

The list of all the universes of BUSINESSOBJECTS.

Definition

```plaintext
Universes As Universes (Read-only)
```

Syntax

```plaintext
var.Universes
```

*var* is the name of the Application variable that you declare.

Example

This example displays the names of all the universes in BusinessObjects.

```plaintext
Dim univs As Universes
Dim i As Integer
Set univs = Application.Universes
For i = 1 To univs.Count
    MsgBox univs.Item(i).LongName
Next
```

Variables Property

The list of all the variables in BUSINESSOBJECTS.

Definition

```plaintext
Variables As Variables (Read-only)
```

Syntax

```plaintext
var.Variables
```

*var* is the name of the Application variable that you declare.
Example 1

This example displays the names of all the variables in BusinessObjects and their values.

```vba
Dim vars As Variables
Dim i As Integer
Set vars = Application.Variables
For i = 1 To vars.Count
    MsgBox vars.Item(i).Name & " " & vars.Item(i).Value
Next
```

Example 2

This example determines which language version of BUSINESSOBJECTS is installed.

```vba
Select Case Application.Variables("BOLANGUAGE").Value
Case "EN"
    MsgBox "English Version"
Case "FR"
    MsgBox "Version Francaise"
'Other languages can go here...
End Select
```

VBE Property

The Microsoft Visual Basic Environment (VBE) object.

**Definition**

*VBE* As VBE (Read-only)

**Syntax**

`var.VBE`

*var* is the name of the Application variable that you declare.

**Comments**

VBE is the root object that contains all other objects and collections represented in VBA.

For more information on see the on-line help in BUSINESSOBJECTS VBE.

**Example**

The following example displays the version of VBE shipped with BusinessObjects:

```vba
MsgBox Application.VBE.Version
```
Chapter 1 BusinessObjects Object Model

Version Property

The version of BusinessObjects.

**Definition**

Version As String (Read-only)

**Syntax**

`var.Version`

`var` is the name of the Application variable that you declare.

Visible Property

Whether or not BUSINESSOBJECTS is visible to the user.

**Definition**

Visible As Boolean (Read/Write)

**Syntax**

`var.Visible`

`var` is the name of the Application variable that you declare.

**Comment**

If this is set to TRUE, then BUSINESSOBJECTS is visible. Otherwise it is hidden.

Window Property

The BUSINESSOBJECTS main window.

**Definition**

Window As Window (Read-only)

**Syntax**

`var.Window`

`var` is the name of the Application variable that you declare.

**Comments**

The Window object lets you modify the size, the position or the state of the BUSINESSOBJECTS main window.

If you close this window, you close BUSINESSOBJECTS.

**Example**

This example changes the width of the main window.

```vba
Dim wnd As Window
Set wnd = Application.Window
wnd.Width = 250
```
BlockStructure Class

Provides access to a block structure of a report. A block structure can be a table, a crosstab, or a chart.

This class inherits all the properties and methods in the class ReportStructureItem.

Syntax

Dim var As BlockStructure

var is the name of the BlockStructure variable that you declare.

See Also

“ReportStructureItem Class” on page 208, and “CellStructure Class” on page 47.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the BlockStructure variable that you declare.

ApplyStdStyle Method

Applies the default style to the block structure.

Definition

Sub ApplyStdStyle()

Syntax

var.ApplyStdStyle

var is the name of the BlockStructure variable that you declare.

Delete Method

Deletes the BlockStructure.

Definition

Sub Delete()

Syntax

var.Delete

var is the name of the BlockStructure variable that you declare.
Name Property

The name of a block structure.

Definition
Name As String (Read/Write)

Syntax
var.Name

var is the name of the BlockStructure variable that you declare.

Parent Property

The creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the BlockStructure variable that you declare.

Pivot Property

A Pivot object for applying pivot features to a block structure.

Definition
Pivot As Pivot (Read-only)

Syntax
var.Pivot

var is the name of the BlockStructure variable that you declare.

Type Property

The type of the block structure.

Definition
Type As BoReportItemType (Read-only)

Syntax
var.Type

var is the name of the BlockStructure variable that you declare.

Comments
BoReportItem is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Values for BoReportItem</th>
</tr>
</thead>
<tbody>
<tr>
<td>boCell (=0)</td>
</tr>
<tr>
<td>boTable (=1)</td>
</tr>
<tr>
<td>boCrosstab (=2)</td>
</tr>
<tr>
<td>boChart (=3)</td>
</tr>
</tbody>
</table>
CellStructure Class

Provides access to a cell structure in a report. You can format and delete cells, and access the data in a cell.

This class inherits all the properties and methods in the class ReportStructureItem.

Syntax

Dim var As CellStructure

var is the name of the CellStructure variable that you declare.

See Also

“ReportStructureItem Class” on page 208, and “BlockStructure Class” on page 45.

Application Property

The Application object.

 Definition

Application As Application (Read-only)

 Syntax

var/Application

var is the name of the CellStructure variable that you declare.

ApplyStdStyle Method

Applies the default style to the cell structure.

Definition

Sub ApplyStdStyle()

 Syntax

var.ApplyStdStyle

var is the name of the CellStructure variable that you declare.

Clear Method

Deletes the contents of a cell, without deleting the cell itself.

Definition

Sub Clear()

 Syntax

var.Clear

var is the name of the CellStructure variable that you declare.
Delete Method

Deletes a cell structure.

Definition

_sub Delete()

Syntax

var.Delete

_var is the name of the CellStructure variable that you declare.

Parent Property

The creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

_var is the name of the CellStructure variable that you declare.

Type Property

The type of the cell structure.

Definition

Type As BoReportItemType (Read-only)

Syntax

var.Type

_var is the name of the CellStructure variable that you declare.

Comments

BoReportItemType is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Values for BoReportItemType</th>
</tr>
</thead>
<tbody>
<tr>
<td>boCell (=0)</td>
</tr>
<tr>
<td>boCrosstab (=2)</td>
</tr>
<tr>
<td>boTable (=1)</td>
</tr>
<tr>
<td>boChart (=3)</td>
</tr>
</tbody>
</table>

For a CellStructure object, this is always boCell.
**ValueType Property**

The type of value contained in the cell structure.

**Definition**

```
ValueType As BoCellValueType (Read-only)
```

**Syntax**

```
var.ValueType
```

`var` is the name of the `CellStructure` variable that you declare.

**Comments**

BoCellValueType is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Values for BoCellValueType</th>
</tr>
</thead>
<tbody>
<tr>
<td>boDocumentVariable (=1)</td>
</tr>
<tr>
<td>boOLEObject (=2)</td>
</tr>
<tr>
<td>boEmpty (=3)</td>
</tr>
</tbody>
</table>

**Variable Property**

The variable contained in the cell structure.

**Definition**

```
Variable As DocumentVariable (Read-only)
```

**Syntax**

```
var.Variable
```

`var` is the name of the `CellStructure` variable that you declare.

**See Also**

“Variable Class” on page 231, and “DocumentVariable Class” on page 147.
Class Class

Provides access to a class. A class is a logical grouping of objects in a BusinessObjects universe. When you are building a query using BusinessObjects, classes are displayed as folders.

You can access the predefined conditions and objects that make up the class.

Syntax

Dim var As Class

var is the name of the Class variable that you declare.

See Also

“Object Class” on page 181, and “Universe Class” on page 225.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Class variable that you declare.

Classes Property

The subclasses of the class.

Definition

Classes As Classes (Read-only)

Syntax

var.Classes

var is the name of the Class variable that you declare.

Description Property

The description associated with the class.

Definition

Description As String (Read-only)

Syntax

var.Description

var is the name of the Class variable that you declare.
Name Property

The name of the class.

**Definition**
Name As String (Read-only)

**Syntax**
var.Name

*var* is the name of the Class variable that you declare.

Objects Property

The objects in the class.

**Definition**
Objects As Objects (Read-only)

**Syntax**
var.Objects

*var* is the name of the Class variable that you declare.

Parent Property

The creator of the object.

**Definition**
Parent As Object (Read-only)

**Syntax**
var.Parent

*var* is the name of the Class variable that you declare.

PredefinedConditions Property

The predefined conditions in a class.

**Definition**
PredefinedConditions As PredefinedConditions (Read-only)

**Syntax**
var.PredefinedConditions

*var* is the name of the Class variable that you declare.

**Comments**
You can apply one or more predefined conditions when you build a query. However, you can’t delete or change predefined conditions.
Classes Class

Provides access to the list of classes.

Syntax

Dim var As Classes

var is the name of the Classes variable that you declare.

See Also

“Class Class” on page 50, and “Object Class” on page 181.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Classes variable that you declare.

Count Property

The number of objects in the collection.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the Classes variable that you declare.

Item Property

A class in the collection.

Definition

Property Item(Index As Variant) As Class (Read-only)

Syntax

var.Item(ind)

var is the name of the Classes variable that you declare.

ind is a Variant that contains either the index of the list of classes or the class identifier.
Parent Property

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

```
var.Parent
```

`var` is the name of the Classes variable that you declare.
Clipboard Class

Provides access to the system clipboard.

Syntax

\[ \text{Dim } \text{var} \text{ As Clipboard} \]

\textit{var} is the name of the Clipboard variable that you declare.

Comments

The Clipboard object is used to manipulate text and graphics on the Clipboard. You can use this object to enable a user to copy, cut, and paste text or graphics in your application. Before copying any material to the Clipboard object, you should clear its contents by as performing a Clear method, such as Clipboard.Clear.

\textbf{Note:} Note that the Clipboard object is shared by all Windows applications, and thus, the contents are subject to change whenever you switch to another application.

The Clipboard object can contain several pieces of data as long as each piece is in a different format. For example, you can use the SetData method to put a bitmap on the Clipboard with the boClipboardFormatBitmap format, and then use the SetText method with the boClipboardFormatText format to put text on the Clipboard. You can then use the GetText method to retrieve the text or the GetData method to retrieve the graphic. Data on the Clipboard is lost when another set of data of the same format is placed on the Clipboard either through code or a menu command.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

\[ \text{var.Application} \]

\textit{var} is the name of the Clipboard variable that you declare.
Clear Method

Clears the contents of the system clipboard.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Sub Clear()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>var.Clear</td>
</tr>
<tr>
<td></td>
<td>var is the name of the Clipboard variable that you declare.</td>
</tr>
</tbody>
</table>

GetData Method

Returns a graphic from the system clipboard.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Function GetData([Format As Variant]) As StdPicture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>var.GetData([format])</td>
</tr>
<tr>
<td></td>
<td>var is the name of the Clipboard variable that you declare.</td>
</tr>
<tr>
<td></td>
<td>format specifies the format in which the graphic is returned. The following values are permissible:</td>
</tr>
<tr>
<td></td>
<td>0 GetData() automatically uses the appropriate format</td>
</tr>
<tr>
<td></td>
<td>2 A bitmap (.bmp) is returned</td>
</tr>
</tbody>
</table>

Comments

This method can only be called from within VBA for BUSINESSOBJECTS. If you are developing in a third-party environment then use the native method to get data from the clipboard. For example, from Visual Basic, use the Visual Basic method Clipboard.GetData.

GetFormat Method

Returns an integer indicating whether an item on the system clipboard matches a specified format.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Function GetFormat(Format As Long) As Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>var.GetFormat(format)</td>
</tr>
<tr>
<td></td>
<td>var is the name of the Clipboard variable that you declare.</td>
</tr>
<tr>
<td></td>
<td>format specifies the format with which you wish to compare the format of the system clipboard contents. The following values are permissible:</td>
</tr>
<tr>
<td></td>
<td>1 Text</td>
</tr>
<tr>
<td></td>
<td>2 Bitmap graphic (.bmp)</td>
</tr>
</tbody>
</table>
Comments
TRUE if an item on the system clipboard matches the specified format.
FALSE otherwise.

GetText Method
Returns a text string from the system clipboard.

Definition
Function GetText([Format As Variant]) As String

Syntax
var.GetText([format])

var is the name of the Clipboard variable that you declare.
format specifies the format in which the text string is returned.

Parent Property
The creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent
var is the name of the Clipboard variable that you declare.

SetData Method
Puts a picture on the system clipboard using the specified graphic format.

Definition
Sub SetData(Picture As StdPicture, [Format As Variant])

Syntax
var.SetData(image, [format])

var is the name of the Clipboard variable that you declare.
image is the graphic to be placed on the clipboard.
format specifies the object format.

Comments
This method can only be called from within VBA for BUSINESSOBJECTS. If you are
developing in a third-party environment then use the native method to place data
on the clipboard. For example, from Visual Basic, use the Visual Basic method
Clipboard.SetData.
**SetText Method**

Puts a text string on the system clipboard using the specified format.

**Definition**

```vbnet
Sub SetText(Str As String, [Format As Variant])
```

**Syntax**

- `var.SetText(str, [format])`
- `var` is the name of the Clipboard variable that you declare.
- `str` is the text to be placed on the clipboard.
- `format` specifies object format.
CmdBar Class

Provides access to a command bar.

Syntax

Dim var As CmdBar

var is the name of the CmdBar variable that you declare.

Comments

A command bar can represent a tool bar, menu bar or pop-up menu. Command bars contain controls (buttons and pop-ups) that provide users with access to the functions you provide using the command bar.

See Also

“CmdBars Class” on page 74, and “CmdBarControl Class” on page 64. Introduction to Developer Suite provides information on command bars and the controls they contain.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the CmdBar variable that you declare.

BuiltIn Property

Determines whether or not the command bar is part of the standard BUSINESSOBJECTS interface.

Definition

BuiltIn As Boolean (Read-only)

Syntax

var.BuiltIn

var is the name of the CmdBar variable that you declare.

Comments

TRUE when the command bar is part of the standard BUSINESSOBJECTS interface. FALSE when the command bar is not part of the standard BUSINESSOBJECTS interface. For example, a command bar that you have added with CmdBars.Add.
Controls Property

The collection of controls contained in the command bar.

**Definition**

`Controls` As `CmdBarControls` (Read-only)

**Syntax**

`var. Controls`

`var` is the name of the `CmdBar` variable that you declare.

Delete Method

Deletes the command bar from the `CmdBars` collection.

**Definition**

Sub `Delete()`

**Syntax**

`var. Delete`

`var` is the name of the `CmdBar` variable that you declare.

**Comment**

It is good programming practice to use this method to remove the command bar from the application’s collection of command bars (`CmdBars`). This reduces the risk of the collection growing too large.

Enabled Property

Determines whether or not the command bar is active.

**Definition**

`Enabled` As Boolean (Read/Write)

**Syntax**

`var. Boolean`

`var` is the name of the `CmdBar` variable that you declare.

**Comments**

This property only applies to command bars of type `boBarTypeNormal`. When a command bar is disabled it appears "grayed-out."

**See Also**

`CmdBar.Type`, `CmdBar.Visible`

Name Property

The name of the command bar.

**Definition**

`Name` As String (Read/Write)

**Syntax**

`var. Name`

`var` is the name of the `CmdBar` variable that you declare.
Parent Property

The creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the CmdBar variable that you declare.

Position Property

Where the command bar appears in the BUSINESSOBJECTS interface.

Definition

Position As BoBarPosition (Read/Write)

Syntax

var.Position

var is the name of the CmdBar variable that you declare.

BoBarPosition is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Values for BoBarPosition</th>
</tr>
</thead>
<tbody>
<tr>
<td>boBarLeft (=0)</td>
</tr>
<tr>
<td>boBarTop (=1)</td>
</tr>
<tr>
<td>boBarRight (=2)</td>
</tr>
<tr>
<td>boBarBottom (=3)</td>
</tr>
<tr>
<td>boBarFloating (=4)</td>
</tr>
<tr>
<td>boBarPopup (=5)</td>
</tr>
</tbody>
</table>

ShowPopup Method

Displays the command bar at optionally specified coordinates.

Definition

Sub ShowPopup([x As Variant], [y As Variant])

Syntax

var.ShowPopup([x], [y])

var is the name of the Document variable that you declare.

x, y specify the coordinate position of the popup. These parameters are optional. You can specify both x and y, or just one of the co-ordinates.

Default

If you only specify one co-ordinate, BUSINESSOBJECTS uses the value of the cursor’s current position for the co-ordinate you did not specify. If you do not specify any co-ordinates, the pop-up appears where the user clicks.

Comments

The method is only applicable to command bars of the type boBarTypePopup.
Type Property

The type of the command bar.

Definition  
**Type** As BoBarType (Read-only)

Syntax  
`var.Type`

`var` is the name of the CmdBar variable that you declare.

Comments  
BoBarType can have the following values:

<table>
<thead>
<tr>
<th>Values for BoBarType</th>
</tr>
</thead>
<tbody>
<tr>
<td>boBarTypeNormal (=0)</td>
</tr>
<tr>
<td>boBarTypeMenuBar (=1)</td>
</tr>
<tr>
<td>boBarTypePopup (=2)</td>
</tr>
</tbody>
</table>

Visible Property

Determines whether or not the command bar appears in the BUSINESSOBJECTS interface.

Definition  
**Visible** As Boolean (Read/Write)

Syntax  
`var.Visible`

`var` is the name of the CmdBars variable that you declare.

Comments  
TRUE when the command bar is visible.
FALSE when the command bar is not visible.
Chapter 1 BusinessObjects Object Model

CmdBarButton Class

Provides access to a command bar button.

Syntax

```vbnet
Dim var As CmdBarButton
```

*var* is the name of the CmdBarButton variable that you declare.

Comments

CmdBarButton inherits the properties and methods of CmdBarControl.

CmdBarButton is a specialization of the CmdBarControl class. It represents a button control on a command bar, or an item in a menu.

**Note:** Because controls can be nested, sometimes the auto-complete facility in Visual Basic cannot determine the type of an object in the “nest”. When this happens you need to enter the properties and methods manually. To reduce the frequency of this, make sure that all the variables you use are dimensioned. See Introduction to Developer Suite for more information on nested controls.

See Also

“CmdBarControl Class” on page 64, “CmdBarPopup Class” on page 73, “CmdBar Class” on page 58, and “Clipboard Class” on page 54. Introduction to Developer Suite describes the relationship between CmdBarButton and CmdBarControl in more detail.

BuiltInFace Property

Determines whether or not the button’s face is part of the standard BUSINESSOBJECTS interface.

**Definition**

`BuiltInFace As Boolean (Read-only)`

**Syntax**

```vbnet
var.BuiltInFace
```

*var* is the name of the CmdBarButton or CmdBarControl variable that you declare.

**Comments**

TRUE when the button uses the standard BUSINESSOBJECTS face.
FALSE when the button does not use the standard BUSINESSOBJECTS face. For example one that you have added with CmdBarButton.PasteFace.
CopyFace Method

Copies the face of the button to the system clipboard.

**Definition**
Sub CopyFace()

**Syntax**
`var.CopyFace`

`var` is the name of the CmdBarButton variable that you declare.

**Comments**
Once you have copied the face to the clipboard you can access the bitmap using the Clipboard class.

**See Also**
“Clipboard Class” on page 54.

FaceId Property

The identifier for the button’s face.

**Definition**
`FaceId` As Long (Read/Write)

**Syntax**
`var.FaceId`

`var` is the name of the CmdBarButton variable that you declare.

PasteFace Method

Pastes the bitmap in the system clipboard onto the face of the button.

**Definition**
Sub PasteFace()

**Syntax**
`var.PasteFace`

`var` is the name of the CmdBarButton variable that you declare.

**Comments**
Before you paste the bitmap onto the face of the button, you must first add it to the clipboard.

**See Also**
“Clipboard Class” on page 54.

*Introduction to Developer Suite* describes the procedure for adding a face to a button.
CmdBarControl Class

Provides access to a command bar control.

Syntax

```
Dim var As CmdBarControl
```

*var* is the name of the CmdBarControl variable that you declare.

Comments

CmdBarControl is a generic class that represents the buttons and pop-up menus contained in a command bar. Its methods are inherited by CmdBarButton and CmdBarPopup.

Having a class representing both buttons and pop-up menus allows all the controls for a command bar to be accessed from one collection—CmdBarControls—rather than one for buttons and one for pop-up menus.

To access the properties and methods of a control you can use either CmdBarControl or CmdBarButton/CmdBarPopup. If you use CmdBarControl, the operations you can perform depend on CmdBarControl.Type. For example, CmdBarControl.OnAction is only applicable to buttons.

Example

The following code fragment executes the action associated with the refresh button on the Standard tool bar in the BUSINESSOBJECTS interface.

```
Dim RefreshButton As CmdBarControl
Set RefreshButton = Application.CmdBars(5).Controls("&Refresh Data")
RefreshButton.Execute
```

This has the same effect as the user clicking the Refresh button.

See Also

“CmdBarControls Class” on page 69, “CmdBarButton Class” on page 62, and “CmdBarPopup Class” on page 73

Application Property

The Application object.

Definition

```
Application As Object (Read-only)
```

Syntax

```
var.Application
```

*var* is the name of the CmdBarControl, CmdBarButton, or CmdBarPopup variable that you declare.
CmdBarControl Class

BuiltIn Property

Determines whether or not the control is part of the standard BUSINESSOBJECTS interface.

**Definition**

**BuiltIn** As Boolean (Read-only)

**Syntax**

`var.BuiltIn`

`var` is the name of the CmdBarControl, CmdBarButton, or CmdBarPopup variable that you declare.

**Comments**

TRUE when the control is part of the standard BUSINESSOBJECTS interface.
FALSE when either the control is not part of the standard BUSINESSOBJECTS interface. Or when it is a standard BUSINESSOBJECTS control but has CmdBarControl.OnAction set.

Caption Property

The caption for the control.

**Definition**

**Caption** As String (Read/Write)

**Syntax**

`var.Caption`

`var` is the name of the CmdBarControl, CmdBarButton, or CmdBarPopup variable that you declare.

**Comments**

For controls that are menu items (pop-up menus) this property is used as the entry in the menu.

Delete Method

Deletes the control from the collection of CmdBarControls.

**Definition**

Sub **Delete()**

**Syntax**

`var.Delete()`

`var` is the name of the CmdBarControl, CmdBarButton, or CmdBarPopup variable that you declare.

**Comments**

It is good programming practice to use this method to remove the control from the command bar’s collection of controls (CmdBarControls). This reduces the risk of the collection growing too large.

**See Also**

CmdBarControls.Add
DescriptionText Property

The description of the control that appears in the status area when the cursor is over the control.

**Definition**

`DescriptionText As String (Read/Write)`

**Syntax**

```vba
var.DescriptionText
```

`var` is the name of the `CmdBarControl` variable that you declare.

Execute Method

Executes the action associated with the control.

**Definition**

`Sub Execute()`

**Syntax**

```vba
var.Execute
```

`var` is the name of the `CmdBarControl` variable that you declare.

**Comment**

This method is only applicable to buttons.

Calling this method is equivalent to the user clicking on the button.

**See Also**

`CmdBarControl.OnAction`

ID Property

The ID of the control.

**Definition**

`ID As Long (Read-only)`

**Syntax**

```vba
var.ID
```

`var` is the name of the `CmdBarControl` variable that you declare.

Index Property

The control’s position in its `CmdBarControls` collection.

**Definition**

`Index As Long (Read-only)`

**Syntax**

```vba
var.Index
```

`var` is the name of the `CmdBarControl` variable that you declare.

**Comments**

The first element in the collection has the index 1.

**See Also**

“CmdBarControls Class” on page 69.
OnAction Property

Returns or sets the name of the Visual Basic macro that will be run when the user clicks or changes the value of a command bar control.

**Definition**

`OnAction As String (Read/Write)`

**Syntax**

```
var.OnAction
```

*var* is the name of the CmdBarControl variable that you declare.

**Comments**

This method is only applicable to buttons.

The string defining the macro must have the following format:

```
"filename.extension!ModuleName.MacroName"
```

**Example**

The following code fragment configures a button so that when a user clicks on it, BUSINESSOBJECTS executes the Update sub-routine in the ToolsModule module of Tools.rea.

```vbnet
Dim Button As CmdBarControl 'could also be a CmdBarButton
...
```

**See Also**

CmdbarControl.Execute

Parent Property

The creator of the object.

**Definition**

`Parent As Object (Read-only)`

**Syntax**

```
var.Parent
```

*var* is the name of the CmdBarControl variable that you declare.

TooltipText Property

The text for the control’s tooltip.

**Definition**

`TooltipText As String (Read/Write)`

**Syntax**

```
var.TooltipText
```

*var* is the name of the CmdBarControl variable that you declare.
Type Property

The type of the control.

**Definition**

Type As BoControlType (Read-only)

**Syntax**

```
var.Type
```

*var* is the name of the CmdBarControl variable that you declare.

**Comments**

BoControlType can have the following values:

<table>
<thead>
<tr>
<th>Values for BoControlType</th>
</tr>
</thead>
<tbody>
<tr>
<td>boControlButton (=1)</td>
</tr>
<tr>
<td>boControlPopup (=2)</td>
</tr>
</tbody>
</table>
CmdBarControls Class

Provides access to the controls of a command bar.

Syntax

Dim var As CmdBarControls

var is the name of the CmdBarControls variable that you declare.

Comments

This class is a collection for the buttons and pop-up menus for a command bar. Using CmdControls.Add you can add controls to a command bar. You cannot add pop-up menus to tool bars.

You access a command bar’s set of controls (this collection) using CmdBar.Controls.

Note: Because controls can be nested, sometimes the auto-complete facility in Visual Basic cannot determine the type of an object in the “nest”. When this happens you need to enter the properties and methods manually. To reduce the frequency of this, make sure that all the variables you use are dimensioned. See Introduction to Developer Suite for details on nesting controls.

Example 1

The following sub-routine displays a message box showing the item number and name of all controls in the BUSINESSOBJECTS Charts command bar.

Sub ShowControls()
    Dim msg As String
    Dim i As Integer
    Dim ctls As CmdBarControls

    msg = "The available controls are:" & Chr(10)
    Set ctls = Application.CmdBars("Charts").Controls

    For i = 1 To ctls.Count
        msg = msg & i & " - " & ctls(i).Caption & Chr(10)
    Next i

    MsgBox msg
End Sub
Example 2

The following code fragment adds a button to the BUSINESSOBJECTS standard tool bar.

Sub AddControl()
    Dim ctls As CmdBarControls
    Dim button As CmdBarControl
    Set ctls = Application.CmdBars("Standard").Controls
    Set button = ctls.Add(boControlButton)
    ... 'set the properties of the button
    button.Delete 'clean up when finished
End Sub

See Also

“CmdBarControl Class” on page 64, and “CmdBar Class” on page 58.

Add Method

Adds a control to the command bar.

Definition

Function Add(Type As BoControlType, [ID As Long], [Before As Long]) As CmdBarControl

Syntax

var.Add(type, [id], [before])

var is the name of the CmdBarControls variable that you declare.

type is the type of control you wish to add. BoControlType can take the following values:

<table>
<thead>
<tr>
<th>Values for BoControlType</th>
</tr>
</thead>
<tbody>
<tr>
<td>boControlButton (=1)</td>
</tr>
<tr>
<td>boControlPopup (=2)</td>
</tr>
</tbody>
</table>

id is an integer that specifies a built-in control. If the value of this argument is 1, or if this argument is omitted, a blank custom control of the specified type will be added to the command bar.

before is a number that indicates the position of the new control on the command bar. The new control will be inserted before the control at this position. If this argument is omitted, the control is added at the end of the specified command bar.
CmdBarControls Class

Comments
It is good programming practice to remove controls from a collection when you have finished with them. This reduces the risk of the collection growing too large. To delete a control use CmdBarControl.Delete.

Application Property
The Application object.
Definition
Application As Application (Read-only)
Syntax
var.Application
var is the name of the CmdBarControls variable that you declare.

Count Property
The number of controls in the collection.
Definition
Count As Long (Read-only)
Syntax
var.Count
var is the name of the CmdBarControls variable that you declare.

Item Property
A control in the command bar’s collection of controls.
Definition
Item(Index As Variant) As CmdBarControl (Read-only)
Syntax
var.Item(ind)
var is the name of the CmdBarControls variable that you declare.
ind is a Variant that contains either the index of the list of command bars or the command bar identifier.
Comments
The item number starts at 1.
Example
The following example sets Popup to the Help menu of the long version of the BUSINESSOBJECTS menu bar.
Dim Popup As CmdBarControl
Set Popup = Application.Cmdbars(2).Controls.Item("&Help")
Parent Property

The creator of the object.

Definition: Parent As Object (Read-only)

Syntax: `var.Parent`

`var` is the name of the CmdBarControls variable that you declare.
CmdBarPopup Class

Provides access to a pop-up control.

Syntax

Dim var As CmdBarPopup

var is the name of the CmdBarPopup variable that you declare.

Comments

CmdBarPopup inherits the properties and methods of CmdBarControl.

CmdBarPopup is a specialization of the CmdBarControl class. It represents a popup control on a command bar.

Note: Because controls can be nested, sometimes the auto-complete facility in Visual Basic cannot determine the type of an object in the 'nest'. When this happens you need to enter the properties and methods manually. To reduce the frequency of this, make sure that all the variables you use are dimensioned. See Introduction to Developer Suite for more information on nested controls.

See Also

“CmdBarControl Class” on page 64, “CmdBarButton Class” on page 62, and “CmdBar Class” on page 58.

Introduction to Developer Suite describes the relationship between CmdBarButton and CmdBarControl in more detail.

CmdBar Property

The command bar containing the pop-up control.

Definition

CmdBar As CmdBar (Read-only)

Syntax

var.CmdBar

var is the name of the CmdBarPopup variable that you declare.

Controls Property

The collection of controls accessed via the pop-up control.

Definition

Controls As CmdBarControl (Read-only)

Syntax

var.Controls

var is the name of the CmdBarPopup variable that you declare.
CmdBars Class

Provides access to the list of command bars.

Syntax

Dim var As CmdBars

*var* is the name of the CmdBars variable that you declare.

Comments

The CmdBars class is the access point for all the command bars (menu bars, tool bars and pop-up menus) and controls (buttons and pop-up menus) in the BUSINESSOBJECTS interface.

Example

The following sub-routine displays a message box showing the item number and name of all the BUSINESSOBJECTS command bars.

```vba
Sub ShowCommandBars()
    Dim msg As String
    Dim i As Integer
    Dim cbars As CmdBars

    msg = "The available command bars are:" & Chr(10)
    Set cbars = Application.CmdBars

    For i = 1 To cbars.Count
        msg = msg & i & " - " & cbars(i).Name & Chr(10)
    Next i

    MsgBox msg
End Sub
```

*Note:* There are two menu bars. The first menu bar in the collection is the short one that appears when there is no open document. The second menu bar is the long one that appears when a document is open.

See Also

“CmdBar Class” on page 58.
**CmdBars Class**

**ActiveMenuBar Property**

The active menu bar.

**Definition**

ActiveMenuBar As CmdBar (Read-only)

**Syntax**

`var .ActiveMenuBar`  

`var` is the name of the CmdBars variable that you declare.

**Comments**

There are two menu bars. One for when there is no open document and one for when there is an open document.

This property is set to VT_EMPTY if there is no active menu bar.

**Add Method**

Adds a command bar.

**Definition**

Function Add(Name As String, Position As BoBarPosition) As CmdBar

**Syntax**

`var .Add(cmdbarName, position)`  

`var` is the name of the CmdBars variable that you declare.

`cmdbarName` identifies the name of the command bar.

`position` specifies where the command bar appears in the interface. BoBarPosition can have the following values:

<table>
<thead>
<tr>
<th>BoBarPosition</th>
<th>Values for BoBarPosition</th>
</tr>
</thead>
<tbody>
<tr>
<td>boBarLeft (=0)</td>
<td>boBarBottom (=3)</td>
</tr>
<tr>
<td>boBarTop (=1)</td>
<td>boBarFloating (=4)</td>
</tr>
<tr>
<td>boBarRight (=2)</td>
<td>boBarPopup (=5)</td>
</tr>
</tbody>
</table>

**Comments**

It is good programming practice to remove controls from a collection when you have finished with them. This reduces the risk of the collection growing too large. To delete a control use CmdBarControl.Delete.
Chapter 1 BusinessObjects Object Model

Application Property
The Application object.

Definition
Application As Application (Read-only)

Syntax
var.Application
var is the name of the CmdBars variable that you declare.

Count Property
The number of command bars in the collection.

Definition
Count As Long (Read-only)

Syntax
var.Count
var is the name of the CmdBars variable that you declare.

DisplayKeysInTooltips Property
Determines whether or not to display keys in tool tips.

Definition
DisplayKeysInTooltips As Boolean (Read/Write)

Syntax
var.DisplayKeysInTooltips
var is the name of the CmdBars variable that you declare.

Comments
TRUE when keys are to be displayed.
FALSE when keys are not to be displayed.

DisplayTooltips Property
Determines whether or not to display tool tips for the command bars in the collection.

Definition
DisplayTooltips As Boolean (Read/Write)

Syntax
var.DisplayTooltips
var is the name of the CmdBars variable that you declare.

Comments
TRUE when tool tips are to be displayed.
FALSE when tool tips are not to be displayed.
Item Property

A command bar in the collection of command bars.

**Definition**

Property *Item*(Index As Variant) As CmdBar (Read-only)

**Syntax**

```
var.Item(ind)
```

*var* is the name of the CmdBars variable that you declare.

*ind* is a Variant that contains either the index of the list of command bars or the command bar identifier.

**Comments**

The item number starts at 1.

**Example**

The following code fragment sets VB_Bar to the Visual Basic tool bar in the BUSINESSOBJECTS interface.

```
Dim VB_Bar As CmdBar
Set VB_Bar = Application.CmdBars.Item(13)
```

LargeButtons Property

Determines whether or not to display large buttons.

**Definition**

LargeButtons As Boolean (Read/Write)

**Syntax**

```
var.LargeButtons
```

*var* is the name of the CmdBars variable that you declare.

Parent Property

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

```
var.Parent
```

*var* is the name of the CmdBars variable that you declare.
Column Class

Provides access to a column of a data provider.

Syntax

Dim var As Column

var is the name of the Column variable that you declare.

Comments

The Column class corresponds to a column in the cube (data provider). In BusinessObjects you can view the names of the columns in a cube in the Results tab of the Data Manager dialog.

See Also

“DpVBAColumn Class” on page 154.

Application Property

The application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Column variable that you declare.

Count Property

The number of elements in the Column class.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the Column variable that you declare.

CustomSort Method

Sorts the column in the specified order.

Definition

Sub CustomSort(SortOrder As String)

Syntax

var.CustomSort(order)

var is the name of the Column variable that you declare.

order is a string that contains the order in which the column is to be sorted. Values in the string must be separated by “,” “|”, or tab (Chr(9)).
Comments
The sort applied by Column.CustomSort does not have a related object in the Sorts collection of a query. Column.CustomSort is applied to a column after the data has been retrieved.

Example
The following example sorts the Month column into chronological order (the default order is alphabetical).

```vba
Sub SortMonth()
  Dim dp As DataProvider
  Set dp = Application.ActiveDocument.DataProviders.Item(1)
  dp.Columns.Item("Month").CustomSort "January;February;March;" &
  "April;May;June;July;August;September;October;November;December"
End Sub
```

Item Property
The contents of a cell in the column.

Definition
Property **Item**(Index As Long) As Variant (Read-only)

Syntax
`var.Item(ind)`
`var` is the name of the Column variable that you declare.

`ind` is a Long that contains the index of the column or a string that contains the name of the column.

Name Property
The name of the column.

Definition
**Name** As String (Read-only)

Syntax
`var.Name`
`var` is the name of the Column variable that you declare.

Parent Property
The creator of the object.

Definition
**Parent** As Object (Read-only)

Syntax
`var.Parent`
`var` is the name of the Column variable that you declare.
Columns Class

Provides access to the columns of a data provider (cube).

Syntax

Dim var As Columns

var is the name of the Columns variable that you declare.

Application Property

The application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Columns variable that you declare.

Count Property

The number of columns in the collection.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the Columns variable that you declare.

Item Property

A column in the collection.

Definition

Property Item(Index As Variant) As Column (Read-only)

Syntax

var.Item(ind)

var is the name of the Columns variable that you declare.

ind is a Variant that contains either the index of the list of columns or the column identifier.
Parent Property

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Columns variable that you declare.
Condition Class

Provides access to the conditions applied to a query technique data provider.

Syntax

Dim var As Condition

var is the name of the Condition variable that you declare.

Comments

A condition is made up of:

- a class
- an object
- one or more operands.

An example of a condition is Year equal to ‘2000’. In this example, Year is the object, (its class, Time period, while not present in the statement is part of the Condition) and ‘2000’ is the operand.

Conditions can be combined using the Conditions collection (see “Conditions Class” on page 86). For example, Year equal to ‘2000’ And Sales Revenue less than $2 million.

Application Property

The application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Condition variable that you declare.

Class Property

The class (in the universe) that corresponds to the condition.

Definition

Class As String (Read-only)

Syntax

var.Class

var is the name of the Condition variable that you declare.
Level Property

The position of the condition in its Conditions collection.

**Definition**  
Level As Long (Read-only)

**Syntax**  

```
var.Level
```

*var* is the name of the Condition variable that you declare.

**Comments**  

The level of a condition determines its position a nested group of conditions. See Conditions.Format for an example of a nested condition.

**See Also**  
“Conditions Class” on page 86.

Object Property

The object (in the universe) on which the condition is applied.

**Definition**  
Object As String (Read-only)

**Syntax**  

```
var.Object
```

*var* is the name of the Condition variable that you declare.

Operand Property

The operands for the condition.

**Definition**  
Operand(Index As Long) As String (Read-only)

**Syntax**  

```
var.Operand(ind)
```

*var* is the name of the Condition variable that you declare.

*ind* is the index of the operand, “1” for the first operand, and so on.

**Comments**  

Condition.Operand is a list, starting at 1, that contains the operands in the condition. Use Condition.OperandCount to find out how many operands are in the list.

**Example**  

For the condition, Year Between ’1999’ and ’2001’, …Condition.Operand(1) is “1999” and …Condition.Operand(2) is “2001”.

OperandCount Property

The number of operands in the condition.

**Definition**

`OperandCount` As Long (Read-only)

**Syntax**

`var.OperandCount`

`var` is the name of the Condition variable that you declare.

**Example**

For the condition, `Year` is not null, `Condition.OperandCount` equals 0, and for the condition, `Year Between '1999' and '2001'`, `Condition.OperandCount` equals 2.

OperandType Property

The type of operand for the condition.

**Definition**

`OperandType(Index As Long) As String (Read-only)`

**Syntax**

`var.OperandType(ind)`

`var` is the name of the Condition variable that you declare.

`ind` is the index of the operand, “1” for the first operand, and so on.

**Comments**

There are two operand types: “Constant” and “Prompt”.

Operator Property

The operator for the condition.

**Definition**

`Operator` As String (Read-only)

**Syntax**

`var.Operator`

`var` is the name of the Condition variable that you declare.
### Comments
The operator can be one of the following:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Equal to&quot;</td>
<td>&quot;Different from&quot;</td>
</tr>
<tr>
<td>&quot;Greater than&quot;</td>
<td>&quot;Greater than or equal to&quot;</td>
</tr>
<tr>
<td>&quot;Less than&quot;</td>
<td>&quot;Less than or equal to&quot;</td>
</tr>
<tr>
<td>&quot;Between&quot;</td>
<td>&quot;Not between&quot;</td>
</tr>
<tr>
<td>&quot;In list&quot;</td>
<td>&quot;Not in list&quot;</td>
</tr>
<tr>
<td>&quot;Is null&quot;</td>
<td>&quot;Is not null&quot;</td>
</tr>
<tr>
<td>&quot;Matches pattern&quot;</td>
<td>&quot;Different from pattern&quot;</td>
</tr>
<tr>
<td>&quot;Both&quot;</td>
<td>&quot;Except&quot;</td>
</tr>
<tr>
<td>&quot;Null&quot;</td>
<td>&quot;None&quot;</td>
</tr>
</tbody>
</table>

### Parent Property
The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

```plaintext
var.Parent
```

`var` is the name of the Condition variable that you declare.
Conditions Class

Provides access to the conditions applied to a query.

**Syntax**

```vba
Dim var As Conditions
```

*var* is the name of the Conditions variable that you declare.

**Comments**

Only queries on universes (that is query technique data providers) have a Conditions collection.

You can have more than one condition for a query. To define how the conditions are combined (Condition 1 And/Or Condition 2... And/Or Condition n) use `Conditions.Format`.

If you change the parameters of a data provider (by adding or removing conditions or results etc.) you must refresh the data provider for the change to take effect.

**See Also**

“Condition Class” on page 82, “Query Class” on page 196, and “Results Class” on page 213.

**Add Method**

Adds a condition.

**Definition**

Function `Add(Class As String, ObjectOrCondition As String, [Operator As String], [Operand1 As String], [Operand1Type As String], [Operand2 As String], [Operand2Type As String]) As Condition`

**Syntax**

```vba
var.Add(class, objectorcondition, [operator], [operand1], [operand1type], [operand2], [operand2type])
```

*var* is the name of the Conditions variable that you declare.

*class* is a string that identifies the class.

*objectorcondition* is a string that identifies the condition.

*operator* identifies the operation. This parameter is optional.

*operand1, operand2* are each operands for the condition. These parameters are optional.

*operand1type, operand2type* are each operand types for the corresponding operator. These parameters are optional.
Example
The following sub-routine adds a simple condition (Year equal to 2000) to a
query.
Sub add_condition()
Dim DPs As New DataProviders 'the data providers for the document
Dim dp As DataProvider 'the data provider of interest
Set DPs = Application.ActiveDocument.DataProviders
Set dp = DPs.Item("Query 1 with EFASHION")
    dp.Queries.Item(1).Conditions.Add _ 'add a condition to
        "Time period", "Year", "Equal to", "2000"  'the first query
    dp.Refresh 'refresh the data provider
End Sub

Application Property
The application object.
Definition Application As Application (Read-only)
Syntax var.Application
var is the name of the Conditions variable that you declare.

Count Property
The number of conditions in the collection.
Definition Count As Long (Read-only)
Syntax var.Count
var is the name of the Conditions variable that you declare.

Format Property
The format for the condition object.
Definition Format As String (Read/Write)
Syntax var.Format
var is the name of the Conditions variable that you declare.
Comments

Conditions.Format defines how the conditions in the collection are combined. The string represents the format in the following general syntax:

(\#condition_1 \textit{operator} \#condition_2)...

where \#condition_1 is the index number of a condition in the Conditions collection, and \textit{operator} is a logical operator (AND, OR).

\textbf{Note:} The index of the conditions in the collection can change. For example, if you have three conditions (#1 AND (#2 OR #3) and set the format to (#3 AND #2), the next time you access Conditions.Format it will return (#1 AND #2). What was labeled #3 is now labeled #1.

Default Value

The default format is \#condition_1 And \#condition_2 And ... And \#condition_n.

Example 1

The following code fragment sets the format to Condition1 OR Condition 2.

```vba
Dim dp As DataProvider
Set dp = Application.ActiveDocument.DataProviders.Item(1)
dp.Queries.Item(1).Conditions.Format = "(#1 OR #2)"
```

Example 2

The format ( (#1 OR #2) AND (#3 AND #4) OR #5) represents the following conditions:
### Item Property

A condition in the collection.

**Definition**

Property `Item(Index As Long) As Condition (Read-only)`

**Syntax**

```vba
var.Item(ind)
```

- `var` is the name of the Conditions variable that you declare.
- `ind` is a Long that contains either the index of the list of conditions or the condition identifier.

### Parent Property

The creator of the object.

**Definition**

`Parent As Object (Read-only)`

**Syntax**

```vba
var.Parent
```

- `var` is the name of the Conditions variable that you declare.

### Remove Method

Removes a condition from the collection.

**Definition**

Sub `Remove(Index As Long)`

**Syntax**

```vba
var.Remove(ind)
```

- `var` is the name of the Conditions variable that you declare.
- `ind` is a Long that contains the index of the condition to be removed.
Daily Class

Provides access to a Daily object of BROADCAST AGENT.

**Syntax**

```vbnet
Dim var As Daily
```

*var* is the name of the Daily variable that you declare.

**Comments**

The Daily option processes the task on one or more specified days of the week at a specified time, at weekly intervals.

**Application Property**

The Application object.

**Definition**

```vbnet
Application As Application (Read-only)
```

**Syntax**

```vbnet
var.Application
```

*var* is the name of the Daily variable that you declare.

**DayOfWeek Property**

The day of the week for the Daily schedule of BROADCAST AGENT.

**Definition**

```vbnet
Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday As Boolean (Read/Write)
```

**Syntax**

```vbnet
var.DayOfWeek
```

*var* is the name of the Daily variable that you declare.

*DayOfWeek* specifies a day of the week: *Monday*, *Tuesday*, *Wednesday*, *Thursday*, *Friday*, *Saturday*, or *Sunday*.

**Comments**

If a day of the week has a value of TRUE then this day is included in the Daily schedule of BROADCAST AGENT. If a day of the week has a value of FALSE then it is not included in the Daily schedule.

**Parent Property**

The creator of the object.

**Definition**

```vbnet
Parent As Object (Read-only)
```

**Syntax**

```vbnet
var.Parent
```

*var* is the name of the Daily variable that you declare.
**StartTime Property**

The time at which the Daily schedule of BROADCAST AGENT begins.

**Definition**

`StartTime` As Date (Read/Write)

**Syntax**

`var StartTime`

`var` is the name of the Daily variable that you declare.

**Comments**

StartTime takes the form (hh:mm:ss) and can have values from 0:00:00 to 23:59:59. Times display according to the time format (either 12-hour or 24-hour) recognized by your computer.

**WeekPeriodicity Property**

The "Every n week(s)" setting for the Daily schedule of BROADCAST AGENT.

**Definition**

`WeekPeriodicity` As Long (Read/Write)

**Syntax**

`var WeekPeriodicity`

`var` is the name of the Daily variable that you declare.
DataProvider Class

The DataProvider class gives you access to the properties and functionality of a data provider.

**Syntax**

```vba
Dim var As DataProvider
```

*var* is the name of the DataProvider variable that you declare.

**Comments**

A data provider can be:
- Query on a universe
- Stored procedure
- Free-hand SQL
- Personal data files
- VBA procedure

See *Introduction to Developer Suite* for information on working with and accessing the elements of data providers.

**Example**

This example displays the name of the first data provider of the active document. The DataProvider class is used to declare the variable.

```vba
Dim dp as DataProvider
Dim msgtext as String
set dp = ActiveDocument.DataProviders.Item(1)
msgtext = dp.Name
MsgBox msgtext
```

Application Property

The application object.

**Definition**

`Application` As Application (Read-only)

**Syntax**

```vba
var.Application
```

*var* is the name of the Data Provider variable that you declare.
ChangeUniverse Method

Change the universe for the current document.

Definition
Sub ChangeUniverse(newUniverse As Universe)

Syntax
var.ChangeUniverse(newUniverse)

Comments
The new universe can have a different connection from the old one.
The new universe cannot contain incompatible objects, that is, objects belonging to different contexts. If the new universe contains incompatible objects the method generates an error.

See Also
For information on incompatible objects and contexts see Designers User’s Guide.

Columns Property

The columns of a data provider.

Definition
Columns As Columns (Read-only)

Syntax
var.Columns

See Also
“Columns Class” on page 80.

ConvertTo Method

Convert a microcube of the data provider into a designated format and stores the converted microcube in an external file.

Definition
Sub ConvertTo(Type As BoConvertToType, CubeNb As Long, [FileName As String])

Syntax
var.ConvertTo(type, cubeNum, [filename])

var is the name of the DataProvider variable that you declare.
**Chapter 1 BusinessObjects Object Model**

*type* indicates the file format of the destination file.

<table>
<thead>
<tr>
<th>Values for BoConvertToType</th>
</tr>
</thead>
<tbody>
<tr>
<td>boExpDbase (=1)</td>
</tr>
<tr>
<td>boExpAsciiCSV (=5)</td>
</tr>
<tr>
<td>boExpExcel (=2)</td>
</tr>
<tr>
<td>boExpExcel97 (=6)</td>
</tr>
<tr>
<td>boExpAsciiTab (=4)</td>
</tr>
</tbody>
</table>

*cubeNum* is an integer that indicates which microcube of the data provider is to be converted. If you enter 0, all microcubes of the data provider are converted. If you enter 1, the first microcube is converted; if you enter 2, the second is converted, and so on.

*filename* is a string that indicates the name of the external file in which the converted microcube is stored. This parameter is optional.

**Comments**

The destination file is saved in the UserDocs folder. The applied extension depends on the type of conversion.

If the filename is omitted, the Open dialog box appears.

You can display the microcubes of the data provider with the Data Providers command of the Data menu.

**Example**

This example converts the first data provider of the active document in an Excel format.

```vba
Dim dp As DataProvider
Set dp = ActiveDocument.DataProviders.Item(1)
call dp.ConvertTo(boExpExcel,1,"File.xl")
```

**Edit Method**

Displays the BUSINESSOBJECTS Edit Data Provider dialog for the data provider. This is the same as pressing the Edit Data Provider button in the BO tool bar.

**Definition**

Sub Edit()

**Syntax**

`var>Edit`

`var` is the name of theDataProvider variable that you declare.
Example

This example edits the first data provider of the active document.

```vba
Dim dp As DataProvider
Set dp = ActiveDocument.DataProviders.Item(1)
dp.Edit
```

Comments

Application.Interactive must be set to TRUE for this method to succeed.

See Also


ExportToRDBMS Method

Exports data for use in other applications as a relational database.

Definition

```vba
Sub ExportToRDBMS(ConnectionName As String, ConnectionType As BoConnectionType)
```

Syntax

```vba
var.ExportToRDBMS(cName, cType)
```

Where:

- `var` is the name of the DataProvider variable that you declare.
- `cName` is the name of the connection.
- `cType` is the connection type. BoConnectionType is an enumerated object which may take either of the following values:

<table>
<thead>
<tr>
<th>Values for BoConnectionType</th>
</tr>
</thead>
<tbody>
<tr>
<td>boPersonalConnection (=1)</td>
</tr>
<tr>
<td>boSharedConnection (=2)</td>
</tr>
</tbody>
</table>

GetType Method

The type of the data provider.

Definition

```vba
Function GetType() As String
```

Syntax

```vba
var.GetType
```

Where:

- `var` is the name of the DataProvider variable that you declare.
Comments

Returns one of the following strings:

<table>
<thead>
<tr>
<th>For this kind of data provider…</th>
<th>GetType returns this string…</th>
</tr>
</thead>
<tbody>
<tr>
<td>query technique</td>
<td>DPQTC</td>
</tr>
<tr>
<td>freehand SQL</td>
<td>DPSQLC</td>
</tr>
<tr>
<td>personal text file</td>
<td>DPASCC</td>
</tr>
<tr>
<td>VBA procedure</td>
<td>DPVBAC</td>
</tr>
<tr>
<td>stored procedure</td>
<td>DPSPC</td>
</tr>
</tbody>
</table>

Some properties and methods of the DataProvider class are only applicable to query technique data providers.

See Also

*Introduction to Developer Suite* for a discussion of the different types of data providers.

**IsEditable Property**

Whether or not the DataProvider can be edited.

**Definition**

**IsEditable** As Boolean (Read/Write)

**Syntax**

\[ var.IsEditable \]

`var` is the name of the Data Provider variable that you declare.

**Comments**

Corresponds to the Editable check box in the Definition pane of the Data Manager dialog.

**Note:** The value of this property depends on the user’s profile in SUPERVISOR.

See Also

DataProvider.Edit, DpVBAInterface.IsEdit
### IsRefreshable Property

Whether or not the data provider can be refreshed.

**Definition**  
IsRefreshable As Boolean (Read/Write)

**Syntax**  
`var.IsRefreshable`

`var` is the name of the Data Provider variable that you declare.

**Comments**  
Corresponds to the Refreshable check box in the Definition pane of the Data Manager dialog.

**Note:** The value of this property depends on the user’s profile in SUPERVISOR.

**See Also**  

### LastExecutionTime Property

Returns the date and time the data provider was last run.

**Definition**  
LastExecutionTime As String (Read-only)

**Syntax**  
`var.LastExecutionTime`

`var` is the name of the Data Provider variable that you declare.

**Example**  
This example displays the date and time the first data provider of the active document was last run.

```vbnet
Dim dp As DataProvider
Set dp = ActiveDocument.DataProviders.Item(1)
MsgBox dp.LastExecutionTime
```

**Comments**  
Corresponds to the value in the Definition pane of the Data Manager dialog.
Chapter 1 BusinessObjects Object Model

Load Method

Loads the data provider into memory for increased speed.

Definition

Sub Load()

Syntax

var.Load

var is the name of the DataProvider variable that you declare.

Comments

Every call to Load must have a corresponding call to DataProvider.Unload to remove the data provider from memory. See Introduction to Developer Suite for a discussion on loading and unloading data providers.

You only need to load and unload data providers when you are optimizing your code to make it “product ready”. For most cases—macros and simple add-ins—you do not need to load a data provider into memory to work with it.

You can load only one data provider at a time. You must remove a loaded data provider from memory before you load another data provider.

Example

The following example loads and unloads a data provider to and from memory.

Set boDP = myDPs.Item(1)
boDP.Load  'load the data provider into memory
...      'do some processing with the data provider
boDP.Unload 'unload the data provider from memory

See Also

DataProvider.Unload

MaxDuration Property

Sets or retrieves the maximum duration, in seconds, of the refresh of the data provider.

Definition

MaxDuration As Long (Read/Write)

Syntax

var.MaxDuration

var is the name of the Data Provider variable that you declare.

Comments

Corresponds to the Max Duration field in the Definition pane of the Data Manager dialog.

If the time for a refresh is greater than MaxDuration, the user is presented with three options:

- Stop and ignore the data retrieved so far
- Stop and keep the partial results
- Continue with the refresh.

98  BusinessObjects SDK Reference Guide
MaxNbLines Property

Sets or retrieves the maximum number of lines returned by the DP.

Definition
MaxNbLines As Long (Read/Write)

Syntax
var.MaxNbLines

var is the name of the Data Provider variable that you declare.

Comments
Corresponds to the Max # Rows field in the Definition pane of the Data Manager dialog.

Name Property

The name of the data provider.

Definition
Name As String (Read/Write)

Syntax
var.Name

var is the name of the Data Provider variable that you declare.

Example
This example displays the name of the first data provider in the active document.

Dim dp As DataProvider
Set dp = ActiveDocument.DataProviders.Item(1)
MsgBox dp.Name

NbCubes Property

The number of cubes of this data provider.

Definition
NbCubes As Long (Read-only)

Syntax
var.NbCubes

var is the name of the Data Provider variable that you declare.

See Also
DpVBAInterface.DpVBACube
Chapter 1 BusinessObjects Object Model

**NbRowsFetched Property**

The number of rows fetched by the data provider.

**Definition**

`NbRowsFetched` As Long (Read-only)

**Syntax**

```
var.NbRowsFetched
```

`var` is the name of the Data Provider variable that you declare.

**Comments**

Corresponds to the value in the #Rows column in the Definition pane of the Data Manager dialog box.

**Parent Property**

The creator of the object.

**Definition**

`Parent` As Object (Read-only)

**Syntax**

```
var.Parent
```

`var` is the name of the Data Provider variable that you declare.

**Queries Property**

The Queries object of the data provider.

**Definition**

`Queries` As Queries (Read-only)

**Syntax**

```
var.Queries
```

`var` is the name of the Data Provider variable that you declare.

**Comments**

If the data provider does not support a Queries object, this property is set to NULL.

**See Also**

“Queries Class” on page 192.

**Refresh Method**

Refreshes the data of the data provider.

**Definition**

Sub `Refresh()`

**Syntax**

```
var.Refresh
```

`var` is the name of the DataProvider variable that you declare.
DataProvider Class

Comments
DataProvider.Refresh only refreshes the data provider, it does not refresh other data providers in the DataProviders collection. Therefore there are no BeforeRefresh or AfterRefresh Document events generated when you call this method.

Example
This example refreshes the first data provider of the active document.
Dim dp As DataProvider
Set dp = ActiveDocument.DataProviders.Item(1)
dp.Refresh

See Also
See Introduction to Developer Suite for a discussion on refreshing data providers.

SQL Property

The SQL (Structured Query Language) of the data provider.

Definition
SQL As String (Read/Write)

Syntax
var.SQL

Comments
This method only works with data providers that query universes: query technique and freehand SQL data providers.

Note: This property is read only for freehand SQL data providers.

Example
This example displays the SQL associated with the first data provider of the active document.
Dim dp As DataProvider
Set dp = ActiveDocument.DataProviders.Item(1)
MsgBox dp.SQL

See Also
DataProvider.GetType
Chapter 1 BusinessObjects Object Model

Universe Property

The universe used as a data provider.

**Definition**

`Universe As Universe (Read-only)`

**Syntax**

`var.Universe

`var` is the name of the Data Provider variable that you declare.

**See Also**

“Universe Class” on page 225.

UniverseName Property

The name of the universe used as a data provider.

**Definition**

`UniverseName As String (Read-only)`

**Syntax**

`var.UniverseName

`var` is the name of the Data Provider variable that you declare.

**Comments**

This method only works with data providers that query universes: query technique and freehand SQL data providers.

**Example**

This example displays the name of the universe associated with the first data provider of the active document.

```vba
Dim dp As DataProvider
Set dp = ActiveDocument.DataProviders.Item(1)
MsgBox dp.UniverseName
```

**See Also**

DataProvider.GetType
Unload Method

Unloads the data provider from memory for it to be used by the rest of the application.

Definition

Sub Unload()

Syntax

var.Unload

var is the name of the DataProvider variable that you declare.

Comments

A call to Unload must accompany every call to DataProvider.Load. See Introduction to Developer Suite for a discussion on loading and unloading data providers.

You only need to load and unload data providers when you are optimizing your code to make it “product ready”. For most cases—macros and simple add-ins—you do not need to load a data provider into memory to work with it.

You can load only one data provider at a time. You must remove a loaded data provider from memory before you load another data provider.

Example

The following example loads and unloads a data provider to and from memory.

Set boDP = myDPs.Item(1)
boDP.Load    'load the data provider into memory
...      'do some processing with the data provider
boDP.Unload     'unload the data provider from memory

See Also

DataProvider.Load
DataProviders Class

Provides access to the list of data providers of a BusinessObjects document.

Syntax

```vba
Dim var As DataProviders
```

`var` is the name of the DataProviders variable that you declare.

Example

This example displays the name of the first data provider of the active document. The DataProviders class is used to declare the variable.

```vba
Dim dps As DataProviders
Dim msgtext as String
Set dps = ActiveDocument.DataProviders
msgtext = dps.Item(1).Name
MsgBox msgtext
```

AddDPVBA Method

Adds a VBA procedure data provider.

Definition

```vba
Function AddDPVBA(procName As String) As DataProvider
```

`AddDPVBA(procName)`

`procName` is a string containing the name of the procedure that defines the data provider. The variable procName has the following format `filename.extension!module.subroutine`

Comments

The subroutine you specify must have a parameter of the type DpVBAInterface in its list of arguments, however you do not need to create this object; BUSINESSOBJECTS creates this object for you as part of the process of adding the data provider.

Once the data provider has been created its name will appear in the Access Data from VBA dialog box of the New Report Wizard of BusinessObjects.

BUSINESSOBJECTS does not create a report for the cube. You can create one using Reports.CreateQuickReport.
DataProviders Class

Example
This example adds a data provider to the current document.

Dim DP As DataProvider
Dim DPs As DataProviders
Set DPs = ActiveDocument.DataProviders
Set DP = DPs.AddDPVBA("MyDPAccess.rea!Main.MyDPAccessRoutine")

See Also
“DpVBAInterface Class” on page 164, “Reports Class” on page 205, and the discussion on using data providers in Introduction to Developer Suite.

AddQueryTechnique Method

Creates a new query technique data provider object.

Definition
Function AddQueryTechnique(UniverseName As String, [UniverseDomainName As String]) As DataProvider

Syntax
var. AddQueryTechnique(universeName, [domainName])

universeName is a string that contains the long name of the universe to which you are adding the query technique.

domainName is the name of the domain. This parameter is optional.

Comments
This method does not create a report. To create a report with the new data provider use Reports.Add or Reports.CreateQuickReport.

Example
The following code fragment creates a query technique data provider.

Dim QT As DataProvider
Set QT = Application.ActiveDocument._
        DataProviders.AddQueryTechnique("eFashion", ")
QT.Queries.Item(1).Results.Add "Time period", "Year"

See Also
“Query Class” on page 196.

Reports.Add, Reports.CreateQuickReport

Application Property

The Application object.

Definition
Application As Application (Read-only)

Syntax
var. Application

var is the name of the Data Providers variable that you declare.
Chapter 1 BusinessObjects Object Model

Count Property
The number of data providers in the DataProviders collection.

**Definition**

Count As Long (Read-only)

**Syntax**

`var.Count`

`var` is the name of the Data Providers variable that you declare.

**Comments**

If the variable is created by `Document.DataProviders`, this property returns the number of data providers in the document.

Item Property
A data provider in the DataProviders collection.

**Definition**

Property Item(Index As Variant) As DataProvider

**Syntax**

`var.Item(ind)`

`var` is the name of the DataProviders variable that you declare.

`ind` is a Variant that contains the index of the list of data providers or a string that contains the name of the data provider.

**Comments**

The item number starts at 1.

**Example**
This example displays the Edit Data Provider dialog for the first data provider of the active document.

```vbnet
Dim dp As DataProvider
Set dp = ActiveDocument.DataProviders.Item(1)
dp.Edit
```

Parent Property
The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Data Providers variable that you declare.
DocAgentOption Class

Provides access to BROADCAST AGENT settings. BROADCAST AGENT enables BusinessObjects users to automatically process and publish their documents via the repository, an Intranet and the World Wide Web.

You can use this class to set various BROADCAST AGENT settings for a document, such as its schedule and refresh options.

**Syntax**

```
Dim var As DocAgentOption

var is the name of the DocAgentOption variable that you declare.
```

**Application Property**

The Application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```
var.Application

var is the name of the DocAgentOption variable that you declare.
```

**CategoryList Property**

The list of categories which can be assigned to a document.

**Definition**

```
CategoryList As String (Read/Write)
```

**Syntax**

```
var.CategoryList

var is the name of the DocAgentOption variable that you declare.
```

**ContinueOnInvalidCategory Property**

Whether or not the Send operation continues after encountering an invalid category.

**Definition**

```
ContinueOnInvalidCategory As Boolean (Read/Write)
```

**Syntax**

```
var.ContinueOnInvalidCategory

var is the name of the DocAgentOption variable that you declare.
```

**Comments**

If ContinueOnInvalidCategory is TRUE, the Send operation is still carried out. If it is set to FALSE (default) the operation is stopped if an invalid category is encountered.
ContinueOnInvalidUser Property

Whether or not the Send operation continues after encountering an invalid user name.

Definition: **ContinueOnInvalidUser** As Boolean (Read/Write)

Syntax: `var.ContinueOnInvalidUser`

Comments: If `ContinueOnInvalidUser` is TRUE, the invalid user is skipped and the Send operation is still carried out for subsequent users. If it is set to FALSE (default) the operation is stopped if an invalid user name is encountered.

CustomScript Property

The path to a custom script.

Definition: **CustomScript** As String (Read/Write)

Syntax: `var.CustomScript`

Daily Property

A Daily object.

Definition: **Daily** As Daily (Read-only)

Syntax: `var.Daily`

DistributionFolder Property

The Distribution Folder on the file system.

Definition: **DistributionFolder** As String (Read/Write)

Syntax: `var.DistributionFolder`
**EndDate Property**

The End Date for the BROADCAST AGENT schedule.

**Definition**

`EndDate` As Date (Read/Write)

**Syntax**

`var.EndDate`

`var` is the name of the DocAgentOption variable that you declare.

---

**Every Property**

An Every object.

**Definition**

`Every` As Every (Read-only)

**Syntax**

`var.Every`

`var` is the name of the DocAgentOption variable that you declare.

---

**FileWatcher Property**

A FileWatcher object.

**Definition**

`FileWatcher` As FileWatcher (Read-only)

**Syntax**

`var.FileWatcher`

`var` is the name of the DocAgentOption variable that you declare.

---

**Hourly Property**

An Hourly object.

**Definition**

`Hourly` As Hourly (Read-only)

**Syntax**

`var.Hourly`

`var` is the name of the DocAgentOption variable that you declare.

---

**See Also**

Hourly Class
Monthly Property

A Monthly object.

Definition

**Monthly** As Monthly (Read-only)

Syntax

```
var Monthly
```

*var* is the name of the DocAgentOption variable that you declare.

See Also

Monthly Class

Overwrite Property

Whether or not a file can be overwritten.

Definition

**Overwrite** As Boolean (Read/Write)

Syntax

```
var Overwrite
```

*var* is the name of the DocAgentOption variable that you declare.

Comments

If Overwrite is set to TRUE, the file can be overwritten. If it set to FALSE, the file can not be overwritten.

This is the Overwrite Mode option when distributing via the Repository.

Parent Property

The creator of the object.

Definition

**Parent** As Object (Read-only)

Syntax

```
var Parent
```

*var* is the name of the DocAgentOption variable that you declare.

Priority Property

The priority of a DocAgent option.

Definition

**Priority** As BoPriority (Read/Write)

Syntax

```
var Priority
```

*var* is the name of the DocAgentOption variable that you declare.
BoPriority is an enumerated type and can take the following values:

<table>
<thead>
<tr>
<th>Values for BoPriority</th>
</tr>
</thead>
<tbody>
<tr>
<td>boPriorityLow (=1)</td>
</tr>
<tr>
<td>boPriorityNormal (=2)</td>
</tr>
<tr>
<td>boPriorityHigh (=3)</td>
</tr>
</tbody>
</table>

**Refresh Property**

Refreshes the document.

**Definition**

Refresh As Boolean (Read/Write)

**Syntax**

`var.Refresh`

`var` is the name of the DocAgentOption variable that you declare.

**Comments**

If set to TRUE, the document is refreshed before any possible custom scripts are executed. The default value is FALSE.

Since custom scripts are only executed under Windows platforms, this property can be used to allow a BUSINESSOBJECTS document to be refreshed when sent to a BROADCAST AGENT server running under UNIX.

**RefreshInTheNameOfRecipient Property**

Whether or not a document is refreshed according to the profile of the users who will receive it.

**Definition**

RefreshInTheNameOfRecipient As Boolean (Read/Write)

**Syntax**

`var.RefreshInTheNameOfRecipient`

`var` is the name of the DocAgentOption variable that you declare.

**Comments**

The advantage of this option is that recipients do not see information that is private, or available to users with different profiles. For example, your user profile (set up by the supervisor) might enable you to retrieve 1000 rows of data from the database. Other users might only be able to retrieve 500 rows, for security reasons. This option ensures that parameters such as these are respected; users who open the refreshed document do not receive information they should not see.
This feature is also called report bursting.
This option:
• Is not available for corporate documents.
• Allows you to distribute the document via the repository only.

**ResetToDefault Method**

Resets the BROADCAST AGENT schedule to the default schedule.

**Definition**
Sub ResetToDefault()

**Syntax**

```plaintext
var.ResetToDefault
```

*var* is the name of the DocAgentOption variable that you declare.

**Comments**

The default schedule is:

```plaintext
DocAgentOption.
    CustomScript = ""
    StartDate = CurrentTime()
    EndDate = CurrentTime() + 1 year
    Title = ""
    Server = ""
    Priority = Normal
    Overwrite = TRUE
    ScheduleMode = Once
    ContinueOnInvaliduser = FALSE
    DistributionFolder = ""
    Users = ""
```
DocAgentOption Class

Daily.
  Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday = FALSE
  StartTime = CurrentTime()
  WeekPeriodicity = 1

Every.
  Day = 1
  Every = 1
  StartHour = CurrentTime()
  MonthPeriodicity = 1

Hourly.
  FromHour = CurrentTime()
  ToHour = CurrentTime() + 2 hours
  MinutesAfterHour = 30

Monthly.
  StartTime = CurrentTime()
  ToHour = CurrentTime() + 2 hours
  MinutesAfterHour = 30

Weekly.
  StartDay = 1
  StartTime = CurrentTime()
  WeekPeriodicity = 1

UserDefined.
  Every = 1
  Unit = Hours
Chapter 1 BusinessObjects Object Model

ScheduleMode Property

The schedule mode of a DocAgent option.

Definition

`ScheduleMode As BoScheduleMode (Read/Write)`

Syntax

```
var.ScheduleMode
```

*var* is the name of the DocAgentOption variable that you declare.

Comments

BoScheduleMode is an enumerated type that specifies the schedule mode of a DocAgent option. It can take the following values:

<table>
<thead>
<tr>
<th>Values for BoScheduleMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>boOnce = 1</td>
</tr>
<tr>
<td>boHourly = 2</td>
</tr>
<tr>
<td>boDaily = 3</td>
</tr>
<tr>
<td>boWeekly = 4</td>
</tr>
<tr>
<td>boMonthly = 5</td>
</tr>
<tr>
<td>boEvery = 6</td>
</tr>
<tr>
<td>boUserDefined = 7</td>
</tr>
<tr>
<td>boWeekly = 4</td>
</tr>
</tbody>
</table>

Send Method

Sends the request to BROADCAST AGENT.

Definition

Function `Send() As Long`

Syntax

```
var.Send
```

*var* is the name of the DocAgentOption variable that you declare.

Example

This example shows you how to attach a schedule to a document you plan to publish.

```
Sub PublishWithScheduler()
    Dim SendtoBCA As DocAgentOption
    Set SendtoBCA = Application.ActiveDocument.DocAgentOption
    'boRepositoryMode is a variable that refers to the document you plan to publish
    Application.ExchangeMode = boRepositoryMode
    Application.ExchangeDomain = "Document"
    With SendtoBCA
        ResetToDefault
        'Enter the name of your Broadcast Agent server
```
DocAgentOption Class

Server = "bca"
ScheduleMode = boOnce
Priority = boPriorityNormal
StartDate = Now()
ContinueOnInvalidCategory = True
RefreshInTheNameOfRecipient = True
ContinueOnInvalidUser = True
CustomScript = "ThisDocument.TaskToExecute"
Refresh = True
'Enter the name of the users or groups to whom you want to send the document to
Users(1) = "PublicationGroup"
Send
End With
End Sub

Server Property

The BROADCAST AGENT name.

**Definition**

Server As String (Read/Write)

**Syntax**

`var.Server`

*var* is the name of the DocAgentOption variable that you declare.

StartDate Property

The Start Date for the BROADCAST AGENT schedule.

**Definition**

StartDate As Date (Read/Write)

**Syntax**

`var.StartDate`

*var* is the name of the DocAgentOption variable that you declare.

Title Property

The title of the DocAgent option.

**Definition**

Title As String (Read/Write)

**Syntax**

`var.Title`
Chapter 1 BusinessObjects Object Model

*var* is the name of the DocAgentOption variable that you declare.

**UserDefined Property**

A UserDefined object.

<table>
<thead>
<tr>
<th>Definition</th>
<th>UserDefined As UserDefined (Read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><em>var</em>.UserDefined</td>
</tr>
<tr>
<td></td>
<td><em>var</em> is the name of the DocAgentOption variable that you declare.</td>
</tr>
<tr>
<td>See Also</td>
<td>UserDefined Class</td>
</tr>
</tbody>
</table>

**Users Property**

A user name from the BROADCAST AGENT’s users list according to the specified index.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Property Users(Index As Long) As String (Read/Write)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><em>var</em>.Users(<em>ind</em>)</td>
</tr>
<tr>
<td></td>
<td><em>var</em> is the name of the DocAgentOption variable that you declare.</td>
</tr>
<tr>
<td></td>
<td><em>ind</em> is a Long that contains the index of the list of users.</td>
</tr>
</tbody>
</table>

**Weekly Property**

A Weekly object.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Weekly As Weekly (Read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><em>var</em>.Weekly</td>
</tr>
<tr>
<td></td>
<td><em>var</em> is the name of the DocAgentOption variable that you declare.</td>
</tr>
<tr>
<td>See Also</td>
<td>Weekly class</td>
</tr>
</tbody>
</table>
Document Class

Provides access to a BusinessObjects document. A document is a file which contains reports. The Document class exposes many properties and methods for manipulating documents. You can automate almost every task you can perform in BusinessObjects, such as printing, saving, refreshing and so on.

You can create a new document, and subsequently variables, data providers, and reports.

A number of document events are also defined and are described earlier in this chapter.

Syntax

Dim var As Document

var is the name of the Document variable that you declare.

Example

This example displays the name of the first opened document. The Document class is used to declare the variable.

Dim doc as Document
set doc = Application.Documents.Item(1)
MsgBox doc.Name

Activate Method

Sets the current document as the default document. Activates the first window associated with the document.

Definition

Sub Activate()

Syntax

var.Activate

var is the name of the Document variable that you declare.

ActiveReport Property

The active report.

Definition

ActiveReport As Report (Read-only)

Syntax

var.ActiveReport

var is the name of the Document variable that you declare.

Comments

If there is no active report then Nothing is returned.
Application Property

The Application object.

**Definition**

Application As Application (Read-only)

**Syntax**

`var.Application`

`var` is the name of the Document variable that you declare.

Author Property

The author of the document.

**Definition**

Author As String (Read/Write)

**Syntax**

`var.Author`

`var` is the name of the Document variable that you declare.

**Comments**

If the name is omitted, this property returns a string that contains the name of the author of the document.

**Example**

This example changes and displays the name of author of the active document.

```vbnet
Application.Documents.Item(3).Activate
ActiveDocument.Author = "Alan"
MsgBox "The new author of the document is " & _
    ActiveDocument.Author
```
AutoRefreshWhenOpening Property

Whether or not a document is auto-refreshed when it is opened.

**Definition**

AutoRefreshWhenOpening As Boolean (Read/Write)

**Syntax**

`var.AutoRefreshWhenOpening`

`var` is the name of the Document variable that you declare.

**Comments**

If AutoRefreshWhenOpening is set to TRUE, then the document is autorefreshed when opened. If it is set to FALSE then the document is not autorefreshed when it is opened.

**Note:** When the Auto Refresh property of a document is TRUE, the sequence of application and document events when the document is opened is: BeforeRefresh, AfterRefresh, Open, then Activate. This means that during the BeforeRefresh and AfterRefresh the document object does not exist and you cannot refer to it within implementations of these events.

When the Auto Refresh property of a document is FALSE the sequence of application and document events when the document is opened is: Open, then Activate.
Close Method

Closes the document.

**Definition**

Sub Close([CloseOption As BoCloseOption])

**Syntax**

`var.Close([closeOption])`

`var` is the name of the Document variable that you declare.

*BoCloseOption* is an enumerated object that specifies the following close options. It can have the following values:

<table>
<thead>
<tr>
<th>Values for BoCloseOption</th>
</tr>
</thead>
<tbody>
<tr>
<td>boSaveIfModified (=1)</td>
</tr>
<tr>
<td>boDontSave (=2)</td>
</tr>
<tr>
<td>boPromptUser (=3)</td>
</tr>
</tbody>
</table>

**Comments**

You should not use the document after you implement this method. Close (Document Class) is not the same as Close (Window Class). Whereas Close (Document Class) closes the document and all associated windows, Close (Window Class) closes only the window. When a document is open in a single window, which is usually the case, Close (Document Class) and Close (Window Class) have the same effect.

**Example**

This example closes the active document without saving (the default value).

```vba
Dim doc as Document
set doc = ActiveDocument
doc.Close
```

Comments Property

The remarks associated with the current document.

**Definition**

Comments As String (Read/Write)

**Syntax**

`var.Comments`

`var` is the name of the Document variable that you declare.

**Comments**

If the comments are omitted, this property returns a string that contains the comments of the document.
DataProviders Property

The list of data providers in the document.

**Definition**  
DataProviders As DataProviders (Read-only)

**Syntax**  
var.DataProviders

*var* is the name of the Document variable that you declare.

**Example**  
This example displays the number of available data providers in the active document.

```vba
dim dps as DataProviders
set dps = ActiveDocument.DataProviders
MsgBox dps.Count
```

DocAgentOption Property

A DocAgentOption object.

**Definition**  
DocAgentOption As DocAgentOption (Read-only)

**Syntax**  
var.DocAgentOption

*var* is the name of the Document variable that you declare.

**See Also**  
DocAgentOption class

DocumentVariables Property

The list of variables in the document.

**Definition**  
DocumentVariables As DocumentVariables (Read-only)

**Syntax**  
var.DocumentVariables

*var* is the name of the Document variable that you declare.
Evaluate Method

Evaluates a formula.

**Definition**

Function **Evaluate**(Formula As String, [EvaluationMode As BoEvaluationMode])
As Variant

**Syntax**

`var.Evaluate(formula, [evalMode])`

`var` is the name of the Document variable that you declare.

`formula` is a string containing the formula to be evaluated. It takes the form “=<MeasureName/DimensionName>”.

`evalMode` is of the type BoEvaluation and can take the following values:

<table>
<thead>
<tr>
<th>Values for BoEvaluationMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bo41Behaviour (=0)</td>
</tr>
<tr>
<td>This is the default value.</td>
</tr>
<tr>
<td>For a dimension, a String containing the aggregate value is returned.</td>
</tr>
<tr>
<td>For a measure, a String containing the measure is returned if there is only one value. If there are more than one values, an error occurs.</td>
</tr>
</tbody>
</table>

| BoUniqueValues (=1)                  |
| For either a dimension or a measure, a Variant is returned containing the unique value(s). Where there are more than one values, a 1-based array is returned. |

| BoAllValues (=2)                     |
| For either a dimension or a measure, a Variant is returned containing all values. Where there are more than one values, a 1-based array is returned. |
Example

The following example displays a list box containing the value(s) for the formula "=<Number of guests>". You need to create a form with a TextBox, a ListBox, a Label, and a CommandButton. The values displayed depend on the number you enter (0, 1, or 2) in TextBox1. This number corresponds to BoEvaluationMode. This example is based on the report Annual.rep:

```vba
Sub EvalMode()
    UserForm1.TextBox1.Text = "1"
    UserForm1.Show
End Sub

Private Sub CommandButton1_Click()
On Error GoTo handler
    Dim DocEval As Variant
    Dim sVal As String
    Dim mode As Long
    Dim sText As String
    ListBox1.Clear
    mode = Val(TextBox1.Text)
    sText = "Evaluate(=<Number of guests>," & mode & ")"
    ListBox1.AddItem sText
    If mode = 0 Then
        sVal = ActiveDocument.Evaluate("=<Number of guests>", mode)
        ListBox1.AddItem sVal
    Else
        DocEval = ActiveDocument.Evaluate("=<Number of guests>", mode)
        For Each sepval In DocEval
            sText = sepval
            ListBox1.AddItem sText
        Next
    End If
    Label1.Caption = ListBox1.ListCount - 1
    Exit Sub
handler:
    MsgBox Err.Number & " - " & Err.Description
End Sub
```
**ExecuteMacro Method**

Executes a macro.

**Definition**

Sub `ExecuteMacro(MacroName As String)`

**Syntax**

`var.ExecuteMacro(macroName)`

`var` is the name of the Document variable that you declare.

`macroName` is a string that contains the macro.

**ExportAsPDF Method**

Save a document in Portable Document Format (PDF).

**Definition**

Sub `ExportAsPDF(FileName As String)`

**Syntax**

`var.ExportAsPDF(fileName)`

`var` is the name of the Document variable that you declare.

`fileName` is a string that contains the name of the PDF file.

**Comments**

This method adds the file extension .pdf if it is not already given in the file name you provide.

The default file location is `.../BusinessObjects/UserDocs`. You can save the file to another directory by specifying the full path name of the file in `fileName`.

**ExportSheetsAsHtml Method**

Saves the report in HTML format.

**Definition**

Function `ExportSheetsAsHtml(FileName As String, [SheetsName As String], [Graphs As Boolean], [Borders As Boolean], [Background As Boolean], [Foreground As Boolean], [FreeForm As Boolean], [Frames As Boolean], [AutoRefreshTime As Long], [BusObjDoc As Boolean], [HtmlLayout As BoHTMLLayout]) As Long`  

**Syntax**

`var.ExportSheetsAsHtml(fileName, [sheetsName], [graphs], [borders], [background], [foreground], [font], [freeform], [frames], [autoRefresh], [busObjDoc], [HtmlLayout])`  

`var` is the name of the Document variable that you declare.

`fileName` is a string that indicates the name of the html file.

`sheetsName` is a string specifying the sheet or report within the current document.

If this is an empty string, all reports within the document are exported. Specific reports within a document are separated by a bar, for example:  

“Report3|Report6”
The following Boolean characteristics set the formats of the html file. Each value is defaulted to TRUE:
- *graphs* to display charts and pictures
- *borders* to display borders
- *background* to display background color
- *foreground* to display text colors
- *font* to preserve current sheet fonts
- *freeform* to set free-form layout
- *frames* to use a frame

*autorefresh* sets automatic reload every x seconds (not minutes, as in the dialog box). The default value is 0.

When *busObjDoc* is set to TRUE, a copy of the report is generated which can be downloaded via the generated HTML file.

*HtmlLayout* specifies the type of page layout. *BoHTMLLayout* is an enumerated object that can have the following values:

<table>
<thead>
<tr>
<th>Values for <em>BoHTMLLayout</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>boHTMLOnePage (=0)</td>
</tr>
<tr>
<td>boHTMLSectionBySection (=1)</td>
</tr>
<tr>
<td>boHTMLBoth (=2)</td>
</tr>
</tbody>
</table>

### Fullname Property

The path and filename of the document.

**Definition**

*FullName* As String (Read-only)

**Syntax**

`var.FullName`

*var* is the name of the Document variable that you declare.

**Example**

This example displays the path and filename of the active document.

```vba
Application.Documents.Item(3).Activate
MsgBox "The total path of the document is " & _
ActiveDocument.FullName
```
GetSecurityPrompts Method

The security prompts for OLAP access.

**Definition**
Function GetSecurityPrompts() As SecurityPrompts (Read-only)

**Syntax**
`var.GetSecurityPromts()`

*var* is the name of the Document variable that you declare.

**Comments**
You can find more information on OLAP access security in the documentation accompanying your access pack.

Installed Property

Whether or not an add-in is installed.

**Definition**
`Installed` As BoInstallStatus (Read/Write)

**Syntax**
`var.Installed`

*var* is the name of the Document variable that you declare.

`BoInstallStatus` is an enumerated object that can have the following values:

<table>
<thead>
<tr>
<th>Values for BoInstallStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>boInstalled (=1)</td>
</tr>
<tr>
<td>boInstalledAndLocked(=2)</td>
</tr>
<tr>
<td>boNotInstalled (=3)</td>
</tr>
</tbody>
</table>

IsAddIn Property

Whether or not a document is an add-in.

**Definition**
`IsAddIn` As Boolean (Read/Write)

**Syntax**
`var.IsAddIn`

*var* is the name of the Document variable that you declare.
Keywords Property

The keywords specification of the document.

**Definition**

*Keywords* As String (Read/Write)

**Syntax**

```vbnet
var.Keywords
```

*var* is the name of the Document variable that you declare.

LastPrintDate Property

The date and time the document was last printed.

**Definition**

*LastPrintDate* As String (Read-only)

**Syntax**

```vbnet
var.LastPrintDate
```

*var* is the name of the Document variable that you declare.

**Example**

This example displays the date and time the active document was last printed.

```vbnet
MsgBox ActiveDocument.LastPrintDate
```

LastSaveDate Property

The date and time the document was last saved.

**Definition**

*LastSaveDate* As String (Read-only)

**Syntax**

```vbnet
var.LastSaveDate
```

*var* is the name of the Document variable that you declare.

**Example**

This example displays the date and time the active document was last modified.

```vbnet
MsgBox ActiveDocument.LastSaveDate
```

Name Property

The name of the document.

**Definition**

*Name* As String (Read-only)

**Syntax**

```vbnet
var.Name
```

*var* is the name of the Document variable that you declare.
Chapter 1 BusinessObjects Object Model

Example

This example displays the names of all open documents.

```vba
Dim I as integer
for I = 1 to Application/Documents/Count
MsgBox Application/Documents/Item(I)/Name
next
```

Parent Property

The creator of the object.

Definition

Parent As Object (Read-only)

Syntax

`var.Parent`

Comments

`var` is the name of the Document variable that you declare.

PartialResults Property

Whether or not partial results are returned.

Definition

PartialResults As Boolean (Read-only)

Syntax

`var.PartialResults`

Comments

`var` is the name of the Document variable that you declare.

Path Property

The name of the folder of the active document.

Definition

Path As String (Read-only)

Syntax

`var.Path`

Comments

This is the patch specification of the document not including the filename or extension.

Example

This example displays the path of the active document.

```vba
MsgBox ActiveDocument/Path
```
PrintDialog Method

Displays the Print dialog box to control how a document is printed, and then prints the document if you click OK.

Definition
Sub PrintDialog()

Syntax
var.PrintDialog
var is the name of the Document variable that you declare.

Example
This example displays the Print dialog box and then prints the active document if you click OK.
ActiveDocument.PrintDialog

Note: Calling this method from an implementation of the Document events Activate, BeforeRefresh or AfterRefresh could cause an error message to be displayed. See “Open” on page 140.

PrintOut Method

Prints the document using the printer name if specified or current print settings.

Definition
Sub PrintOut([PrinterName As String])

Syntax
var.PrintOut([printerName])
var is the name of the Document variable that you declare.

printerName is the name of the printer. This parameter is optional.

Example
This example prints the active document.
ActiveDocument.PrintOut

Note: Calling this method from an implementation of the Document events Activate, BeforeRefresh or AfterRefresh could cause an error message to be displayed. See “Open” on page 140.
Chapter 1 BusinessObjects Object Model

Refresh Method

Refreshes all the data of the document.

**Definition**

Sub Refresh()

**Syntax**

`var.Refresh`

`var` is the name of the Document variable that you declare.

**Example**

This example opens the document “Prices” and refreshes it.

```vba
dim doc as Document
set doc = Application.Documents.Open("Prices")
doc.Refresh
```

Reports Property

The list of all the reports of the document.

**Definition**

Reports As Reports (Read-only)

**Syntax**

`var.Reports`

`var` is the name of the Document variable that you declare.

**See Also**

Reports class

Save Method

Saves the document.

**Definition**

Sub Save()

**Syntax**

`var.Save`

`var` is the name of the Document variable that you declare.

**Comments**

You cannot modify the filename when you use this method. If you must change the filename, select the SaveAs method.
Example

This example opens the document "Prices", refreshes it, and then saves it.

```vba
Dim doc As Document
Set doc = Application.Documents.Open("Prices")
doc.Refresh
doc.Save
```

**Note:** Calling this method from an implementation of the Document events Activate, BeforeRefresh or AfterRefresh could cause an error message to be displayed. See “Open” on page 140.

---

**SaveAs Method**

Saves the active document with the specified name.

**Definition**

Sub `SaveAs([FileName As String], [ForAllUsers])`

**Syntax**

```vba
var.SaveAs([fileName], [all])
```

- `var` is the name of the Document variable that you declare.
- `fileName` is a string that contains the filename of the document without its extension, or the filename with its path and extension.
- `all` is a Boolean which determines whether or not the document is saved for all users.

**Comments**

If a document name is not specified, the Save As dialog box appears.

**Example**

This example opens the document "Prices", refreshes it then saves it with the name "Today".

```vba
Dim doc As Document
Set doc = Application.Documents.Open("Prices")
doc.Refresh
doc.SaveAs("Today")
```

**Note:** Calling this method from an implementation of the Document events Activate, BeforeRefresh or AfterRefresh could cause an error message to be displayed. See “Open” on page 140.
Saved Property

Whether or not the document has been saved since the last change.

**Definition**

Saved As Boolean (Read-only)

**Syntax**

```
var Saved
```

*var* is the name of the Document variable that you declare.

**Comments**

TRUE if the document has been saved since the last change.
FALSE otherwise.

Send Method

Sends the document to the repository. If a user name is not specified, the Send dialog box appears.

**Definition**

Sub Send([UserName As String], [StoreGeneratedHTML As Boolean], [HtmlLayout As BoHTMLLayout], [CategoryList As String], [ExchangeMode As BoExchangeMode], [ExchangeDomain As String])

**Syntax**

```
var Send([UserName], [StoreGenHTML], [HtmlLayout], [CatList], [xMode], [xDomain])
```

*var* is the name of the Document variable that you declare.

*UserName* is a string that contains the name of the user to whom you plan to address the document.

**Note:** *UserName* is only optional when Application.Interactive is TRUE.

StoreGenHTML is a Boolean which determines whether the document is stored in HTML.
HtmlLayout specifies the type of page layout. This is ignored if StoreGenHTML is FALSE. BoHTMLLayout is an enumerated object that can have the following values:

<table>
<thead>
<tr>
<th>Values for BoHTMLLayout</th>
</tr>
</thead>
<tbody>
<tr>
<td>boHTMLOnePage (=0)</td>
</tr>
<tr>
<td>boHTMLSectionBySection (=1)</td>
</tr>
<tr>
<td>boHTMLBoth (=2)</td>
</tr>
</tbody>
</table>

CatList is a string that identifies the document category list.

xMode is the exchange mode. BoExchangeMode is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Values for BoExchangeMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>boUsermode (=0)</td>
</tr>
<tr>
<td>boRepositoryMode (=1)</td>
</tr>
<tr>
<td>boRepositoryModeNoOverwrite (=2)</td>
</tr>
<tr>
<td>boDocAgentMode (=3)</td>
</tr>
</tbody>
</table>

xDomain is a string that identifies the exchange domain.

Comments

First parameter is mandatory when interactive mode is set to FALSE
Example

This example exports the document "Prices" to the current repository.

dim doc as Document
set doc = Application.Documents.Open("Prices")
Application.ExchangeMode = boRepositoryMode
Application.ExchangeDomain = "Document"
doc.Send("Company")

See Also

Application.Interactive on page 38.

Subject Property

The subject of the document.

Definition

Subject As String (Read/Write)

Syntax

var. Subject

var is the name of the Document variable that you declare.

Example

This example changes and displays the subject of the active document.

Application/Documents/Item(1).Activate
ActiveDocument.Subject = "Prices for demo"
MsgBox "The new subject of the document is " & _
    ActiveDocument.Subject

Title Property

The document's title.

Definition

Title As String (Read/Write)

Syntax

var. Title

var is the name of the Document variable that you declare.

Example

This example changes and displays the title of the active document.

Application/Documents/Item(1).Activate
ActiveDocument.Title = "Prices"
MsgBox "The new title of the document is " & ActiveDocument.Title
Variables Property

The list of variables of the document.

**Definition**

Variables As Variables (Read-only)

**Syntax**

`var.Variables`

`var` is the name of the Document variable that you declare.

Windows Property

The list of opened windows of the document.

**Definition**

Windows As Windows (Read-only)

**Syntax**

`var.Windows`

`var` is the name of the Document variable that you declare.

**Comments**

If you close the document, the windows must no longer be used.

**See Also**

Windows class

**Example**

This example iconizes all the windows of the active document.

```vba
dim wnds as Windows
dim I as integer
set wnds = ActiveDocument.Windows
for I = 1 to wnds.Count
    wnds.Item(I).State = Minimized
next
```
Document Events

Note: Document events also apply to add-ins.

See the Developer Suite Online web site for examples of implementing events.

Activate

Occurs when a document becomes the active document.

Syntax
Private Sub Document_Activate()

Comments
The Activate event can occur only when a document is visible. The Activate event occurs only when you move the focus within BUSINESSOBJECTS. Switching to the VBA editor does not affect the activated status of a document. Closing or opening a dialog box in BUSINESSOBJECTS does not trigger the Activate event.

A document can become active by using the Activate method in code.

Note: Calling the Document methods PrintOut(), PrintDialog(), Save(), or SaveAs() in an implementation of this event could cause an error message to be displayed. See “Open” on page 140.

AfterRefresh

Occurs after a document has been refreshed.

Syntax
Private Sub Document_AfterRefresh()

Comments
This event occurs after a document refresh where a refresh corresponds to the following actions:

• Selecting the Refresh Data option from the Data menu
• Clicking the Refresh button.
• Opening a document whose AutoRefreshWhenOpening property is set to TRUE
• Calling the method ActiveDocument.Refresh

Notes: This event does not occur after ActiveDocument.DataProviders.Item(x).Refresh.

Calling the Document methods PrintOut(), PrintDialog(), Save(), or SaveAs() in an implementation of this event could cause an error message to be displayed. See “Open” on page 140.

When the Auto Refresh property of a document is TRUE, the sequence of application and document events when the document is opened is: BeforeRefresh, AfterRefresh, Open, then Activate. This means that during the BeforeRefresh and AfterRefresh the document object does not exist and you cannot refer to it within implementations of these events.

When the Auto Refresh property of a document is FALSE the sequence of application and document events when the document is opened is: Open, then Activate.

BeforeClose

Occurs before a document is closed.

Syntax

Private Sub Document_BeforeClose(Cancel As Boolean)
Cancel has a value of FALSE if the event has been triggered successfully. It has a value of TRUE otherwise.

Comments

This event corresponds to the following cases:
• Selecting the Close option from the File menu
• Selecting the Login As option from the Tools menu. This closes all documents.
• Calling the methods ActiveDocument.Close
BeforeRefresh

Occurs before a document is refreshed.

Syntax

Private Sub Document_BeforeRefresh(Cancel As Boolean)

Cancel has a value of FALSE if the event has been triggered successfully. It has a value of TRUE otherwise.

Comments

This event occurs before a document refresh where a refresh corresponds to the following actions:

- Selecting the Refresh Data option from the Data menu
- Opening a document whose AutoRefreshWhenOpening property is set to TRUE
- Calling the method ActiveDocument.Refresh

Notes: This event does not occur after ActiveDocument.DataProviders.Item(x).Refresh.

Calling the Document methods PrintOut(), PrintDialog(), Save(), or SaveAs() in an implementation of this event could cause an error message to be displayed. See “Open” on page 140.

When the Auto Refresh property of a document is TRUE, the sequence of application and document events when the document is opened is: BeforeRefresh, AfterRefresh, Open, then Activate. This means that during the BeforeRefresh and AfterRefresh the document object does not exist and you cannot refer to it within implementations of these events.

When the Auto Refresh property of a document is FALSE the sequence of application and document events when the document is opened is: Open, then Activate.
BeforeSave

Occurs before a document is saved.

Syntax

Private Sub Document_BeforeSave(Cancel As Boolean)

Cancel has a value of FALSE if the event has been triggered successfully. It has a value of TRUE otherwise.

Comments

This event corresponds to the following cases:

- Selecting the Save option from the File menu
- Selecting the Save As... option from the File menu. The event occurs after you click the Save button on the Save As dialog box.
  - If the file already exists, the event occurs after you click Yes in the Save As warning box.
- Calling the methods ActiveDocument.Save and ActiveDocument.SaveAs

**Note:** This event does not occur before publishing a document on Corporate Documents although a temporary copy of the document is saved. Likewise, this event does not occur before saving a file as HTML.

Deactivate

Occurs when a document is no longer the active document.

Syntax

Private Sub Document_Deactivate()

Comments

The Deactivate event can occur only when a document is visible. The Deactivate event occurs only when you move the focus within BUSINESSOBJECTS. Switching to the VBA editor does not affect the deactivated status of a document. Closing or opening a dialog box in BUSINESSOBJECTS does not trigger the deactivate event.

A document can become deactivated when the Activate method is applied to another document.

**Note:** Do not include any window activity (message boxes, forms, etc.) in an implementation of this event. If you do, BUSINESSOBJECTS may fail. See also, “DocumentDeactivate” on page 31.
Open

Occurs when a document has been opened.

Syntax

Private Sub Document_Open()

Comments

If you perform a refresh as part of an implementation of this event, and include calls to the Document methods PrintOut(), PrintDialog(), Save(), or SaveAs() in implementations of the Activate, BeforeRefresh or AfterRefresh events, BUSINESSOBJECTS displays an error message.

For example, the following displays an error message:

Private Sub Document_AfterRefresh()
    ThisDocument.PrintOut
End Sub

Private Sub Document_Open()
    ThisDocument.Refresh()
End Sub

The refresh could also be triggered by the user selecting the Refresh Document When Opening option in the BUSINESSOBJECTS Options dialog.
Documents Class

Provides access to the list of BusinessObjects documents.

Syntax

Dim var As Documents

*var* is the name of the Documents variable that you declare.

Example

This example displays the name of the first opened document. The Documents class is used to declare the variable.

```vba
Dim docs as Documents
Dim msgtext as String
set docs = Application.Documents
msgtext = docs.Item(1).Name
MsgBox msgtext
```

Add Method

Creates a new document, adds it to the collection, and returns the document that was created.

Definition

Function *Add()* As Document

Syntax

*var*.Add

*var* is the name of the Documents variable that you declare.

Example

This example creates a new document using the Add method:

```vba
Sub CreateNewDoc()
' Reference the application Reporter.
    Dim myApp As Application
    Set myApp = Application
' Reference the documents of the application.
    Dim myDocs As Documents
    Set myDocs = Application.Documents
' Add a new document and set it as the active document.
    Dim myDoc As Document
    Set myDoc = myDocs.Add
' Create a new QueryTechnique on the Island Resorts Marketing universe.
' Reference the data providers of the document.
```
Dim myDPs As DataProviders
Set myDPs = myDoc.DataProviders
' Create a data provider.
Dim myDP As DataProvider
Set myDP = myDPs.AddQueryTechnique("Island Resorts Marketing")
' Load this data provider in memory to increase speed.
myDP.Load
' Reference the queries of the active document.
Dim myQueries As Queries
Set myQueries = myDP.Queries
' Reference the first query. This query has been created by the call to AddQueryTechnique.
Dim myQuery1 As Query
Set myQuery1 = myQueries.Item(1)
' Define the objects of the first query.
Dim myResults1 As Results
Set myResults1 = myQuery1.Results
Dim myResult1 As Result
Set myResult1 = myResults1.Add("Resort", "Country")
Set myResult1 = myResults1.Add("Resort", "Service")
Set myResult1 = myResults1.Add("Customer", "Country of origin")
Set myResult1 = myResults1.Add("Customer", "Customer")
Set myResult1 = myResults1.Add("Measures", "Revenue")
' Define the conditions of the first query.
Dim myConditions1 As Conditions
Set myCondition1 As Condition
Set myCondition1 = myConditions1.Add("Customer", _ "US customer")
Set myCondition1 = myConditions1.Add("Measures", _ "Revenue", "Greater than", "500")
Set myCondition1 = myConditions1.Add("Measures", "Revenue", _ "Between", "0", "Constant", "100000", "Constant")
' Define the sorts of the first query.
Dim mySorts1 As Sorts
Set mySorts1 = myQueries.Sorts
Dim mySort1 As Sort
Set mySort1 = mySorts1.Add("Resort", "Country", FALSE)
' Unload the data provider from memory and refresh it.
myDP.Unload
DoEvents
myDP.Refresh
' Create a quick report.
Dim myReports As Reports
Set myReports = myDoc.Reports
Dim myReport As Report
Set myReport = myReports.CreateQuickReport
' Choose a template to apply.
myReport.ApplyTemplate "Table Master Detail", FALSE
MsgBox "Done"
End Sub

Application Property

The Application object.

**Definition**

Application As Application (Read-only)

**Syntax**

`var.Application`

*var* is the name of the Documents variable that you declare.

Count Property

The number of documents in the Documents variable.

**Definition**

Count As Long (Read-only)

**Syntax**

`var.Count`

*var* is the name of the Documents variable that you declare.

**Comments**

If the variable is created by Application.Documents, this property returns the number of opened documents.
Example

This example displays the number of opened documents.

dim docs as Documents
set docs = Application.Documents
MsgBox docs.Count

Item Property

A Document type object based on its item number or its name.

Definition

Property Item(Index As Variant) As Document

Syntax

var.Item(ind)

var is the name of the Documents variable that you declare.

ind is a Variant that contains the index of the list of documents or a string that contains the name of the document.

Comments

The item number starts at 1.

Open Method

Opens the specified document. If a document name is not specified, the Open dialog box appears.

Definition

Function Open([FileName As String], [NoAutomaticRefresh As Boolean], [ReadOnly As Boolean], [Password As String], [WriteResPassword As String]) As Document

Syntax

var.Open([documentName], [noAutoRefresh], [readOnly], [protPasswd], [writeResPassword])

var is the name of the Documents variable that you declare.

documentName is a string that contains the filename of the document. If you specify just the filename without its extension the document must be in the UserDocs folder. If you specify the filename with its extension, then you must also provide the path to the file.

noAutoRefresh is a Boolean that specifies that the document should not be refreshed on opening if set to TRUE; the document is refreshed if this is set to FALSE.

readOnly is a Boolean which specifies whether or not the document is writable. The default is FALSE.
protPasswd is a string that contains the protection password to open a protected document. If this argument is omitted and the workbook requires a password, the user is prompted for the password, or an error will be returned.

writeResPasswd is a string that contains the password required to write to a write-reserved document. If this argument is omitted and the document requires a password, the user will be prompted for the password or an error will be returned. This Parameter is only useful if parameter ReadOnly is not set to TRUE.

Comments
You can open the same file formats that you can open from the File menu in BUSINESSOBJECTS. These are: *.rep;*.rea;*.bqy;*.wqy. If two or more files have a different format and the same name (for example, Sales.rep and Sales.wqy), then BUSINESSOBJECTS will discriminate according to the order in which the extensions are listed here.

Parent Property
The creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the Documents variable that you declare.

Receive Method
Retrieves a document from the repository.

Definition
Sub Receive([Name As String], [DestDir As String])

Syntax
var.Receive([documentName], [destDir])

var is the name of the Documents variable that you declare.

documentName is a string that contains the filename of the document. If you specify just the filename without its extension the document must be in the UserDocs folder. If you specify the filename with its extension, then you must also provide the path to the file.

destDir is a string that contains the optional name of the destination directory.

Comments
If a document name is not specified, the Retrieve dialog box appears which lists the available documents.
You can modify the import mode with Application.ExchangeMode.

**Note:** You must make sure that Application.ExchangeDomain is set before calling Documents.Receive.

You can receive the following file types from the repository: *.rep;*.ret;*.rea;*.bqy;*.wqy. If two or more files have a different format and the same name (for example, Sales.rep and Sales.wqy), then BusinessObjects will discriminate according to the order in which the extensions are listed here.

**Example**

This example imports the document "Prices" from the current repository then refreshes it.

```vbscript
dim doc as Document
Application.Documents.Receive("Prices")
set doc = Application.Documents.Open("Prices")
doc.Refresh
```
DocumentVariable Class

Provides access to a document variable.

Syntax

```vbnet
Dim var As DocumentVariable
```

`var` is the name of the DocumentVariable variable that you declare.

Comments

A document variable can be a local variable, formula or constant.

Users can access the document variables for a document using the Variables… item of the Data menu.

**Note:** Be careful not to confuse the DocumentVariable class with the Variable class.

See Also

“DocumentVariables Class” on page 151, “Variables Class” on page 235.

*Introduction to Developer Suite* discusses how to use the DocumentVariable class.

The *BusinessObjects User's Guide* discusses the syntax of formulas and how users define local variables.

Application Property

The Application object.

**Definition**

`Application As Application (Read-only)`

**Syntax**

```vbnet
var.Application
```

`var` is the name of the DocumentVariable variable that you declare.

Delete Method

Deletes a document variable.

**Definition**

`Sub Delete()`

**Syntax**

```vbnet
var.Delete()
```

`var` is the name of the DocumentVariable variable that you declare.
**Chapter 1 BusinessObjects Object Model**

**Formula Property**

The formula of the document variable.

**Definition**

Formula As String (Read/Write)

**Syntax**

`var`.Formula

`var` is the name of the DocumentVariable variable that you declare.

**Example**

When used as a column in a report, the following formula displays the sales revenue if it is greater than 10 000 otherwise it displays 10 000.

`= If (<Sales revenue> > 10000) Then <Sales revenue> Else 10000`

**See Also**

The *BusinessObjects User’s Guide* discusses the syntax of formulas.

**IsDataProviderObject Property**

Whether or not the document variable is part of a data provider.

**Definition**

IsDataProviderObject As Boolean (Read-only)

**Syntax**

`var`.IsDataProviderObject

`var` is the name of the DocumentVariable variable that you declare.

**Name Property**

The name of the document variable.

**Definition**

Name As String (Read/Write)

**Syntax**

`var`.Name

`var` is the name of the DocumentVariable variable that you declare.

**Comments**

Formulas and constants have an empty Name property.

**Parent Property**

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var`.Parent

`var` is the name of the DocumentVariable variable that you declare.
Qualification Property

The document variable’s qualification.

**Definition**

**Qualification** As BoObjectQualification (Read/Write)

**Syntax**

```vbnet
var.Qualification
```

*var* is the name of the DocumentVariable variable that you declare.

BoObjectQualification is an enumerated type and can take the following values:

<table>
<thead>
<tr>
<th>Values for BoObjectQualification</th>
</tr>
</thead>
<tbody>
<tr>
<td>boDimension (=1)</td>
</tr>
<tr>
<td>boDetail (=2)</td>
</tr>
<tr>
<td>boMeasure (=3)</td>
</tr>
</tbody>
</table>

**Comments**

Formulas and constants have a Qualification of boDetail.

Values Property

The values of the document variable.

**Definition**

**Values**(EvaluationMode As BOEvaluationMode) As Variant (Read-only)

**Syntax**

```vbnet
var.Values(evalMode)
```

*var* is the name of the DocumentVariable variable that you declare.

*evalMode* is the evaluation mode. BoEvaluationMode is an enumerated type and can take the following values:

<table>
<thead>
<tr>
<th>Values for BoEvaluationMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bo41Behaviour (=0)</td>
</tr>
<tr>
<td>For a dimension…</td>
</tr>
<tr>
<td>A String containing the aggregate value is returned.</td>
</tr>
<tr>
<td>For a measure…</td>
</tr>
<tr>
<td>A String containing the measure is returned if there is only one value. If there are more than one values, an error occurs.</td>
</tr>
</tbody>
</table>
Chapter 1 BusinessObjects Object Model

<table>
<thead>
<tr>
<th>BoUniqueValues (=1)</th>
<th>A Variant is returned containing the unique value(s). Where there is more than one value, a 1-based array is returned.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoAllValues (=2)</td>
<td>A Variant is returned containing all values. Where there is more than one value, a 1-based array is returned.</td>
</tr>
</tbody>
</table>

Default

The default value of the evalMode parameter is Bo41Behaviour.

Comments

Use the UBound() Visual Basic method to get the number of elements in the Values array.

To access the elements of the array returned by DocumentVariables.Values use the following syntax:

```
DocumentVariable.Values(BoEvaluationMode)(index)
```

Example

The following code fragment creates a string using the values of the document variable called ‘Year’.

```vba
Dim strYearVals As String
Dim bodvrYear As DocumentVariable
Set bodvrYear = ThisDocument.DocumentVariables("Year")
For i = 1 To UBound(bodvrYear.Values(BoAllValues))
    strYearVals = strYearVals & " " & bodvrYear.Values(BoAllValues)(i)
Next i
```

See Also

Document.Evaluate
**DocumentVariables Class**

This collection contains the definitions of local variables, formulas and constants.

**Syntax**

Dim \( \text{var} \) As DocumentVariables

\( \text{var} \) is the name of the DocumentVariables variable that you declare.

**Comments**

This collection contains the same information that users can access from the Variables… item of the Data menu.

**BUSINESSOBJECTS** uses this collection to store the details of document elements such as report titles, column headings, dates, and page numbers.

**See Also**

“DocumentVariable Class” on page 147, “Variables Class” on page 235. *Introduction to Developer Suite* discusses how to use the DocumentVariables class. The *BusinessObjects User’s Guide* discusses the syntax of formulas and how variables are defined.
Chapter 1 BusinessObjects Object Model

Add Method

Add Method adds a document variable.

**Definition**

Function `Add(Formula As String, [Name As String]) As DocumentVariable`

**Syntax**

`var.Add(formula, docVarName)`

*var* is the name of the DocumentVariables variable that you declare.

*formula* is a combination of functions, operators, variables and/or local variables, displayed in a cell, that calculates a new value based on existing values.

*docVarName* is a string that identifies the name of the document variable.

**Comments**

Adding a document variable using the Add method is the same as adding a document variable using the Variables dialog box.

Application Property

The Application object.

**Definition**

**Application** As Application (Read-only)

**Syntax**

`var.Application`

*var* is the name of the DocumentVariables variable that you declare.

Count Property

The number of document variables.

**Definition**

**Count** As Long (Read-only)

**Syntax**

`var.Count`

*var* is the name of the DocumentVariables variable that you declare.

Item Property

A document variable in the collection.

**Definition**

**Item(Index As Variant) As DocumentVariable (Read-only)**

**Syntax**

`var.Item(ind)`

*var* is the name of the DocumentVariables variable that you declare.

*ind* is a variable that contains either the index of the list of document variables or the document variable identifier.
Parent Property

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the `DocumentVariables` variable that you declare.
DpVBAColumn Class

Provides access to a column in a microcube.

Syntax

Set \( \text{var} = \text{oColumns.Item}(n) \)

\( \text{var} \) is the name of the DpVBAColumn variable that you declare.

\( \text{oColumns} \) is a variable of type DpVBAColumns.

\( n \) is the position of the column in the DpVBAColumns collection.

Comments

Using this class, you can access the individual values in a microcube.

Example

The following example multiplies the values of a column by a constant.

```vba
Dim Pi As Single
Pi = 3.1415
For i = 1 To 10
    Col.Item(i) = Col.Item(i) * Pi
Next i
```

Add Method

Adds one or more values to the column and returns the index (line number) of the last insertion.

Definition

Function Add(Val As Variant) As Long

Syntax

\( \text{var}.\text{Add}(\text{val}) \)

\( \text{var} \) is the name of the DpVBAColumn variable that you declare.

\( \text{val} \) is a simple value or an array of values which is added.

Comments

You can add several values at a time by declaring \( \text{val} \) as an array.

This method increases the value of DpVBAColumns.NbLines as necessary.

See Also

DpVBAColumns.Add
Aggregation Property

The type of aggregation performed on the column.

**Definition**

*Aggregation* As BoObjectAggregation (Read/Write)

**Syntax**

`var.Aggregation`

`var` is the name of the DpVBAColumn variable that you declare.

**Comments**

Aggregation can have one of the following values:

<table>
<thead>
<tr>
<th>Values for BoObjectAggregation</th>
</tr>
</thead>
<tbody>
<tr>
<td>boAggregateBySumObject (=1)</td>
</tr>
<tr>
<td>boAggregateByMaxObject (=2)</td>
</tr>
<tr>
<td>boAggregateByMinObject (=3)</td>
</tr>
</tbody>
</table>

Delete Method

Deletes a column.

**Definition**

Function *Delete()* As Boolean

**Syntax**

`var.Delete()`

`var` is the name of the DpVBAColumn variable that you declare.

**Comments**

Returns TRUE if the column was successfully deleted. Returns FALSE if the column was not deleted or if the column does not exist.

Index Property

The index (position) of the column in the Columns collection.

**Definition**

Property *Index* As Long (Read-only)

**Syntax**

`var.Index`

`var` is the name of the DpVBAColumn variable that you declare.

**Comments**

This property enables you to access a column directly without having to search for it in the collection.
Example

The following example shows how to use DpVBAColumn.Index.

```vba
Dim Cols As DpVBAColumns
Dim NameColumn, AnotherColumn As DpVBAColumn 'columns in a DpVBA cube
Dim NameColumnIndex As Long 'index to the Name column
...
Set Col = Cols.Item(1) 'set the properties of the name column
Col.Name = "Name"
NameColumnIndex = Col.Index 'remember the column’s index
...
Set AnotherColumn = Cols.Item(NameColumnIndex) 'get the name column
```

Item Property

A value in the column.

**Definition**

Property **Item**(Index As Long) As Variant (Read/Write)

**Syntax**

```vba
var.Item(ind) = val
```

*var* is the name of the DpVBAColumn variable that you declare.

*ind* is a Long that specifies which line in the column you want to access.

*val* is a Variant that contains the value that will be inserted at line *ind* in the column. *val* cannot be an array.

**Example**

The following example sets the contents of the cell in line two of the column *currentColumn* to "Yes".

```vba
currentColumn.Item(2,) = "Yes"
```

MaxWidth Property

The maximum width of a column.

**Definition**

**MaxWidth** As Long (Read/Write)

**Syntax**

```vba
var.MaxValue
```

*var* is the name of the DpVBAColumn variable that you declare.

**Default Value**

256.

**Comments**

The maximum number of characters in a column is 256.
DpVBAColumn Class

Name Property

The name of the column.

Definition
Name As String (Read/Write)

Syntax
var.Name

var is the name of the DpVBAColumn variable that you declare.

Parent Property

The creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the DpVBAColumn variable that you declare.

Qualification Property

The qualification (type of information) of data contained in the column.

Definition
Qualification As BoObjectQualification (Read/Write)

Syntax
var.Qualification

var is the name of the DpVBAColumn variable that you declare.

Comments
Qualification can have one of the following values:

<table>
<thead>
<tr>
<th>Values for BoObjectQualification</th>
</tr>
</thead>
<tbody>
<tr>
<td>boDimension (=1)</td>
</tr>
<tr>
<td>boDetail (=2)</td>
</tr>
<tr>
<td>boMeasure (=3)</td>
</tr>
</tbody>
</table>

See Also
**Chapter 1 BusinessObjects Object Model**

**Type Property**

The type of data in the column.

**Definition**

**Type** As `BoObject` (Read/Write)

**Syntax**

`var`.Type

`var` is the name of the DpVBAColumn variable that you declare.

**Comments**

Type can have one of the following values:

<table>
<thead>
<tr>
<th>Values for <code>BoObject</code></th>
<th><code>boNullObject</code> (=1)</th>
<th><code>boCharacterObject</code> (=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>boNumericObject</code> (=2)</td>
<td><code>boDateObject</code> (=4))</td>
</tr>
</tbody>
</table>

**Note:** `BoObject` also defines the values `boBlobCharacterObject` and `boBlobObject`, however these are not applicable to the VBA data provider.
DpVBAColumns Class

Provides access to the collection of columns within a microcube.

**Syntax**

Set `var = oCube.DpVBAColumns`  
`var` is the name of the DpVBAColumns variable that you declare.  
o`Cube` is a variable of type DpVBACube.

**Add Method**

Adds a new column (DpVBAColumn) to the microcube and returns its index in the collection.

**Definition**

Function **Add**(sColName As String) As Long

**Syntax**

`var.Add(scolName)`  
`var` is the name of the DpVBAColumns variable that you declare.  
s`colName` is the name of the column you wish to add.

**See Also**

DpVBAColumn.Add

**AddLine Method**

Adds a new line to the microcube and returns the index.

**Definition**

Function **AddLine**(newValue As Variant) As Long

**Syntax**

`var.AddLine(newval)`  
`var` is the name of the DpVBAColumns variable that you declare.  
`newval` is an array of values which is added to the columns in the microcube.

**Comments**

`newval` must be declared as a VBA array. This array can be:

- A one-dimensional array with the same size as the number of columns in the microcube.
- A two-dimensional array. The first dimension represents the rows and the second dimension represents the columns. The second dimension must be equal to the number of columns in the microcube.

**Comments**

DpVBAColumns.NbLines is automatically increased.
Chapter 1 BusinessObjects Object Model

Count Property

The number of columns in the collection.

**Definition**

Count As Long (Read-only)

**Syntax**

```vba
var.Count
```

`var` is the name of the DpVBAColumns variable that you declare.

Item Property

A column in the DpVBAColumns collection.

**Definition**

Property Item(Index As Long) As DpVBAColumn (Read-only)

**Syntax**

```vba
var.Item(ind)
```

`var` is the name of the DpVBAColumns variable that you declare.

`ind` is a Long that contains the index of the list of columns.

NbLines Property

The number of lines in the collection of columns.

**Definition**

NbLines As Long (Read/Write)

**Syntax**

```vba
varNbLines
```

`var` is the name of the DpVBAColumns variable that you declare.

Parent Property

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

```vba
var.Parent
```

`var` is the name of the DpVBAColumns variable that you declare.
SetNbColumns Method

Sets the number of columns in the collection.

**Definition**
Sub SetNbColumns(nNbColumns As Long)

**Syntax**
var.SetNbColumns(nCols)

*var* is the name of the DpVBAColumns variable that you declare.

*nCols* declares the number of objects of type DpVBAColumn to be created in the microcube.

**Comments**
Use this method when you are defining the parameters of a microcube.
DpVBACube Class

Provides access to a microcube.

**Syntax**

Set `var = dpInterface.DpVBACubes.Item(1)`

*var* is the name of the DpVBACube variable that you declare.

**Note:** The index for `Item()` must always be set to one, since there can only be one microcube.

DpVBAColumns Property

The collection of columns in the microcube.

**Definition**

*DpVBAColumns* As DpVBAColumns (Read-only)

**Syntax**

`var.DpVBAColumns`

*var* is the name of the DpVBACube variable that you declare.

Parent Property

The creator of the object.

**Definition**

*Parent* As Object (Read-only)

**Syntax**

`var.Parent`

*var* is the name of the DpVBACube variable that you declare.
DpVBACubes Class

Provides access to the collection of microcubes for a VBA procedure data provider.

Syntax
Set var = dpInterface.DpVBACubes

var is the name of the DpVBACubes variable that you declare.

Comments
A VBA procedure data provider can have only one microcube.

Count Property

The number of microcubes in the collection.

Definition
Count As Long (Read-only)

Syntax
var.Count

var is the name of the DpVBACubes variable that you declare.

Comments
The number of microcubes is always one.

Item Property

A microcube in the DpVBACubes collection.

Definition
Property Item(Index As Long) As DpVBACube (Read-only)

Syntax
var.Item(ind)

var is the name of the DpVBACubes variable that you declare.

Comments
There can only be one microcube so the value Index must always be 1.

Parent Property

The creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the DpVBACubes variable that you declare.
DpVBAInterface Class

The DpVBAInterface class gives you the ability to define a VBA procedure data provider.

You create an instance of this class by defining a procedure that has an argument of type DpVBAInterface. When BUSINESSOBJECTS calls your procedure it creates a DpVBAInterface object that you can use in your procedure.

**Note:** You cannot create an instance of DpVBAInterface using New or CreateObject() you must wait for BUSINESSOBJECTS to make one for you. This means that you cannot have a call to the procedure in your code.

**Syntax**

Sub DataProviderName(dpInterface As DpVBAInterface)

End Sub

**DataProviderName** is the name of the VBA procedure data provider.

**dpInterface** is the object created by BUSINESSOBJECTS that you use to access the microcube of the data provider.

**Comments**

Once you have defined a VBA procedure data provider, you can execute it using the New Report and New Data wizards, editing the data provider with the Edit Data Provider dialog, or by refreshing the data provider with DataProvider.Refresh.

You can use DpVBAInterface.IsEdit to determine how your procedure was executed; was the data provider executed as a result of an edit, or as a refresh.

**Example**

The following example defines a VBA procedure data provider that fills a microcube using the Microsoft Outlook 9.0 object library. To access the Outlook object library you need to include it in your project’s references:

1. In the VBA environment, from the Tools menu, select References...

2. Select Microsoft Outlook 9.0 object library from the list and click OK.

Once you have done this you can use the Outlook object library.

```vba
Public Sub Outlook_DPVBA(dpInterface As DpVBAInterface)
    Dim olkapp As New Outlook.Application 'reference to outlook
    Dim nspNameSpace As NameSpace

    Dim objInboxFolder As Object 'the outlook folder we're interested in
    Dim objMail As Object 'the section(s) of the folder
```
Dim oCube As DpVBACube 'somewhere to put the data
Dim oColumns As DpVBAColumns 'the columns in the cube
Dim oCol As DpVBAColumn 'specific column to hold the data
Dim i As Integer
Dim ColumnIndex As Long

Set nsNameSpace = olkapp.GetNamespace("MAPI")
Set objInboxFolder = nsNameSpace.GetDefaultFolder(olFolderInbox)

'Define the attributes of the data provider and cube
dpInterface.UserString(1) = "User String for Outlook data provider"
Set oCube = dpInterface.DpVBACubes.Item(1)
Set oColumns = oCube.DpVBAColumns
oColumns.SetNbColumns (2) 'there are two columns in the cube

'fill in the first column in the cube
Set oCol = oColumns.Item(1) 'look at the first column
oCol.Name = "From" 'the name that appears in the report
oCol.Type = boCharacterObject 'what sort of information is it?
oCol.Qualification = boDimension 'qualify the information further
For i = 1 To 10
    Set objMail = objInboxFolder.Items.Item(i) 'get the data
    oCol.Item(i) = objMail.SenderName 'put it in the cube
Next i

'repeat for the other columns in the cube
Set oCol = oColumns.Item(2)
oCol.Name = "To"
oCol.Type = boCharacterObject
oCol.Qualification = boDimension
For i = 1 To 10
    Set objMail = objInboxFolder.Items.Item(i)
    oCol.Item(i) = objMail.To
Next i
'check to make sure everything's okay
dpInterface.CheckDataIntegrity (boCheckAll)
End Sub

See Also
“DataProvider Class” on page 92, “DpVBACubes Class” on page 163, and “DpVBAColumns Class” on page 159.

Document.AddDPVBA

Introduction to Developer Suite discusses how to create and work with VBA procedure data providers.

CancelOnExit Property

Whether or not the operation that caused the VBA procedure data provider to execute was successful.

Definition: CancelOnExit As Boolean (Read/Write)

Syntax: var.CancelOnExit

var is the name of the DpVBAInterface variable that you declare.

Default value: FALSE

Comments:
If CancelOnExit is TRUE then the operation was not successful.
If CancelOnExit is FALSE then the was successful.
If this property is set to TRUE when the VBA procedure data provider finishes executing, BUSINESSOBJECTS cancels the operation (refresh etc.) that caused the procedure to execute.
If it is the first time the procedure is run, the data provider is not created.

CheckDataIntegrity Method

Checks that BUSINESSOBJECTS has successfully created the DpVBAInterface object and microcube and that there are no conflicts.

Definition: CheckDataIntegrity(Level As BoCheckLevel)

Syntax: var.CheckDataIntegrity(level)

var is the name of the DpVBAInterface variable that you declare.
level defines the level of checking which is to take place. It can have one of the following values:

<table>
<thead>
<tr>
<th>Values for BoCheckLevel</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boNoCheck (= 0)</td>
<td>The data in the microcube is not checked. This can be used to save time but should only be applied where the integrity of the data is certain.</td>
</tr>
<tr>
<td>boCheckNumberOfCube (= 1)</td>
<td>Check that at least one microcube is created.</td>
</tr>
<tr>
<td>boCheckColumnsName (= 2)</td>
<td>Check that the names of the columns are different and not empty. If the column has no name it sets it to: “Col # n”. If there is a name conflict between columns it sets the name to: “ColumnName # n”</td>
</tr>
<tr>
<td>boCheckAutoTyping (= 4)</td>
<td>If the type of a column has not been set explicitly, then check that it does not contain conflicting types. If all the data is the same type it sets the column’s type to the type of the data, otherwise it sets the type to String.</td>
</tr>
<tr>
<td>boCheckChangeType (= 8)</td>
<td>If a column has been typed explicitly all its values are converted into this type.</td>
</tr>
<tr>
<td>boCheckAll (= -1)</td>
<td>All checks listed above are carried out.</td>
</tr>
</tbody>
</table>
Checking levels can be combined. For example, CheckDataIntegrity (boCheckColumnsName & boCheckAutoTyping).

If you do not call CheckDataIntegrity as part of your VBA procedure data provider, BUSINESSOBJECTS calls it with level set to boCheckAll. In this case, if there is a problem, no exception is generated, but an error message is displayed.

If the DpVBAInterface object is not correctly populated, an exception is thrown that you can trap with the VBA statement “On Error”.

DpVBACubes Property

The collection of microcubes for the data provider. There can only be one microcube.

Definition: DpVBACubes As DpVBACubes (Read-only)

Syntax: var.DpVBACubes

Comments: Returns an object of type DpVBACubes that contains the collection of microcubes for a VBA procedure data provider.

IsEdit Property

Determines if the VBA data provider has been called via the Refresh command or from the DpVBA dialog box.

Definition: IsEdit As Boolean (Read-only)

Syntax: var.IsEdit

Comments: If the return value is TRUE, then the data provider is being edited. If the return value is FALSE, then the data provider is being refreshed.

Parent Property

The creator of the object.

Definition: Parent As Object (Read-only)

Syntax: var.Parent

Comments: var is the name of the DpVBAInterface variable that you declare.
ShowSelectionDialogOnEdit Property

Controls display of the Access Data From VBA dialog box.

**Definition**

`ShowSelectionDialogOnEdit` As Boolean (Read/Write)

**Syntax**

`var.ShowSelectionDialogOnExit`  
`var` is the name of the DpVBAInterface variable that you declare.

**Default Value**

TRUE

**Comments**

When this property is set to TRUE, BUSINESSOBJECTS displays the Access Data From VBA dialog box when the user chooses to edit the data provider through either the user interface (for example the New Report Wizard) or the `DataProvider.Edit()` method.

When this property is set to FALSE, BUSINESSOBJECTS does not display the Access Data From VBA dialog box and executes the sub-routine associated with the current data provider.

UserString Property

A string attached to a microcube.

**Definition**

`UserString(Index As Long) As String (Read/Write)`

**Syntax**

`var.UserString(ind)`  
`var` is the name of the DpVBAInterface variable that you declare.  
`ind` is a Long that contains the index of the array.

**Comments**

Any number of user strings can be defined. User strings are saved with the document and can be reused whenever the data provider is called. A user string can be used to store user input which can then be referenced within a procedure.

UserStringCount Property

The total number of UserStrings associated with the microcube.

**Definition**

`UserStringCount` As Long (Read/Write)

**Syntax**

`var.UserStringCount`  
`var` is the name of the DpVBAInterface variable that you declare.

**Comments**

If you set count to 0 then all UserStrings associated with the microcube are deleted.
Every Class

This class enables you to program the date, hour and frequency that the document(s) will be sent to BROADCAST AGENT.

The Every class corresponds to the Monthly Interval in the BROADCAST AGENT schedule.

The advantage in sending documents for delayed processing (at night, for example) is that you do not overload the server. By processing the tasks on a remote server, you do not block your own PC.

Syntax

Dim var As Every

var is the name of the Every variable that you declare.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Every variable that you declare.

Day Property

The “Day” setting for the Monthly Interval schedule of BROADCAST AGENT.

Definition

Day As BoDayDA (Read/Write)

Syntax

var.Day
Every Class

var is the name of the Every variable that you declare. It can have one of the following values:

<table>
<thead>
<tr>
<th>Values for BoDayDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>boUserDefinedMinutes (= 0)</td>
</tr>
<tr>
<td>boUserDefinedDays (= 2)</td>
</tr>
<tr>
<td>boUserDefinedWeekendDays (= 3)</td>
</tr>
<tr>
<td>boUserDefinedBusinessDays (= 4)</td>
</tr>
<tr>
<td>boUserDefinedMondays (= 5)</td>
</tr>
<tr>
<td>boUserDefinedTuesdays (= 6)</td>
</tr>
<tr>
<td>boUserDefinedWednesdays (= 7)</td>
</tr>
<tr>
<td>boUserDefinedThursdays (= 8)</td>
</tr>
<tr>
<td>boUserDefinedFridays (= 9)</td>
</tr>
<tr>
<td>boUserDefinedSaturdays (= 10)</td>
</tr>
<tr>
<td>boUserDefinedSundays (= 11)</td>
</tr>
</tbody>
</table>

Comments
Specifies the day of the week that a document is sent to BROADCAST AGENT. The property Every.Every specifies how many times this occurs in a month.

Every Property

The "Every n" setting for the Monthly Interval schedule of BROADCAST AGENT.

Definition
Every As Long (Read/Write)

Syntax

var.Every

var is the name of the Every variable that you declare.

Comments
Specifies how often a document will be sent to BROADCAST AGENT in a month. This value determines how many times in a month the value for Every.Day will occur. For example, if Every.Every has a value of “first” (1) and Every.Day has a value of “Tuesday” (5), then the document will be sent to BROADCAST AGENT on the first Tuesday of each month.

MonthPeriodicity Property

The "Every n Month(s)” setting for the Monthly Interval schedule of BROADCAST AGENT.

Definition
MonthPeriodicity As Long (Read/Write)

Syntax

var.MonthPeriodicity

var is the name of the Every variable that you declare.
Chapter 1 BusinessObjects Object Model

Comments
Specifies the month periodicity with which a document is sent to BROADCAST AGENT. For example, if Every.Every has a value of “first” (1), Every.Day has a value of “Tuesday” (5), and Every.MonthPeriodicity has a value of 2, then the task will be processed by BROADCAST AGENT on the first Tuesday of every second month.

Parent Property
The creator of the object.
Definition Parent As Object (Read-only)
Syntax var.Parent

StartHour Property
The “at hour” setting for the Monthly Interval schedule of BROADCAST AGENT.
Definition StartHour As Date (Read/Write)
Syntax var.StartHour

Comments This property specifies the time, in the Monthly Interval schedule, at which a document is sent to BROADCAST AGENT.
FileWatcher Class

The File Watcher gives you control over when BROADCAST AGENT processes the document. You specify a file that must be present before processing can begin. You can also set up the File Watcher so that the specified file must be present every time BROADCAST AGENT is set to process the document.

Syntax

Dim var As FileWatcher

var is the name of the FileWatcher variable that you declare.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the FileWatcher variable that you declare.

DeleteFileOnCompletion Property

Whether or not the file is deleted after completion.

Definition

DeleteFileOnCompletion As Boolean (Read/Write)

Syntax

var.FileFullName

var is the name of the FileWatcher variable that you declare.

FileWatcherOnly Property

Whether or not the FileWatcher option is set.

Definition

FileWatcherOnly As Boolean (Read/Write)

Syntax

var.FileWatcherOnly

var is the name of the FileWatcher variable that you declare.
FullFileName Property

The full name of the file.

Definition

FullFileName As String (Read/Write)

Syntax

var.FullFileName

var is the name of the FileWatcher variable that you declare.

Parent Property

The creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the FileWatcher variable that you declare.
Hourly Class

Provides access to the Hourly object of BROADCAST AGENT. The Hourly option causes the task to be executed once an hour during the interval specified.

Syntax

Dim var As Hourly

var is the name of the Hourly variable that you declare.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Hourly variable that you declare.

FromHour Property

The "From" setting for the Hourly schedule of BROADCAST AGENT.

Definition

FromHour As Date (Read/Write)

Syntax

var.FromHour

var is the name of the Hourly variable that you declare.

MinutesAfterHour Property

The "Minutes after the hour" setting for the Hourly schedule of BROADCAST AGENT.

Definition

MinutesAfterHour As Long (Read/Write)

Syntax

var.MinutesAfterHour

var is the name of the Hourly variable that you declare.

Parent Property

The creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the Hourly variable that you declare.
ToHour Property

The “To” setting for the Hourly schedule of BROADCAST AGENT.

**Definition**

ToHour As Date (Read/Write)

**Syntax**

```
var.ToHour
```

*var* is the name of the Hourly variable that you declare.
ListofValues Class

Provides access to the list of values. A list of values presents the values returned by an object. BusinessObjects lets you use lists of values in order to select the value(s) you need when defining conditions on objects in a query, or when running a query that contains a prompt.

Syntax

```vba
Dim var As ListofValues
```

`var` is the name of the ListofValues variable that you declare.

Application Property

The Application object.

Definition

```vba
Application As Application (Read-only)
```

Syntax

```vba
var.Application
```

`var` is the name of the ListofValues variable that you declare.

Edit Method

Modifies the list of values.

Definition

```vba
Sub Edit()
```

Syntax

```vba
var.Edit
```

`var` is the name of the ListofValues variable that you declare.

Parent Property

The creator of the object.

Definition

```vba
Parent As Object (Read-only)
```

Syntax

```vba
var.Parent
```

`var` is the name of the ListofValues variable that you declare.
Purge Method

Purges the list of values.

**Definition**
Sub `Purge()`

**Syntax**
```vba
var.Purge
```

`var` is the name of the ListofValues variable that you declare.

Refresh Method

Refreshes the list of values.

**Definition**
Sub `Refresh()`

**Syntax**
```vba
var.Refresh
```

`var` is the name of the ListofValues variable that you declare.

Values Property

The values in the list of values.

**Definition**
Values As Variant (Read-only)

**Syntax**
```vba
var.Values
```

`var` is the name of the ListofValues variable that you declare.

**Comments**
Use the `UBound` VB function to get the number of items in the table.
Monthly Class

Provides access to the Monthly object of BROADCAST AGENT. The Monthly option processes the task on selected days of the month at a specified hour.

Syntax

Dim var As Monthly

var is the name of the Monthly variable that you declare.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Monthly variable that you declare.

DayOfMonth Property

The "Day(s) of the month" setting for the Monthly schedule of BROADCAST AGENT.

Definition

Day1 | Day2 | Day3 | Day4 | Day5 | Day6 | Day7 | Day8 | Day9 | Day10 |
Day11 | Day12 | Day13 | Day14 | Day15 | Day16 | Day17 | Day18 | Day19 |
Day29 | Day30 | Day31 | LastDayOfMonth As Boolean (Read/Write)

Syntax

var.DayOfMonth

var is the name of the Monthly variable that you declare.


Parent Property

The creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the Monthly variable that you declare.
StartTime Property

The "Hour (hh:mm)" setting for the Monthly schedule of BROADCAST AGENT.

**Definition**

StartTime As Date (Read/Write)

**Syntax**

`var StartTime`

`var` is the name of the Monthly variable that you declare.
Object Class

Provides access to a BusinessObjects object. Objects are elements in a BusinessObjects universe that correspond to a selection of data in the database.

Syntax

Dim var As Object
var is the name of the Object variable that you declare.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application
var is the name of the Object variable that you declare.

Description Property

The description of the object.

Definition

Description As String (Read/Write)

Syntax

var.Description
var is the name of the Object variable that you declare.

ListofValues Property

The list of values in the object.

Definition

ListofValues As ListofValues (Read-only)

Syntax

var.ListofValues
var is the name of the Object variable that you declare.

Name Property

The name of the object.

Definition

Name As String (Read-only)

Syntax

var.Name
var is the name of the Object variable that you declare.
Objects Property

The collection of objects.

**Definition**  
**Objects** As Objects (Read-only)

**Syntax**

```
var.Objects
```

*var* is the name of the Object variable that you declare.

Parent Property

The creator of the object.

**Definition**  
**Parent** As Object (Read-only)

**Syntax**

```
var.Parent
```

*var* is the name of the Object variable that you declare.

Qualification Property

The object qualification.

**Definition**  
**Qualification** As BoObjectQualification (Read-only)

**Syntax**

```
var.Qualification
```

*var* is the name of the Object variable that you declare.

**Comments**

Qualification can have one of the following values:

<table>
<thead>
<tr>
<th>Values for BoObjectQualification</th>
</tr>
</thead>
<tbody>
<tr>
<td>boDimension (=1)</td>
</tr>
<tr>
<td>boDetail (=2)</td>
</tr>
<tr>
<td>boMeasure (=3)</td>
</tr>
</tbody>
</table>

Type Property

The object type.

**Definition**  
**Type** As BoObjectType (Read-only)

**Syntax**

```
var.Type
```

*var* is the name of the Object variable that you declare.
BoObject Type is an enumerated type and can have one of the following values:

<table>
<thead>
<tr>
<th>Values for BoObjectType</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>boNullObject (= 1)</td>
<td>boDateObject (= 4)</td>
</tr>
<tr>
<td>boNumericObject (= 2)</td>
<td>boBlobObject (= 5)</td>
</tr>
<tr>
<td>boCharacterObject (= 3)</td>
<td>boBlobCharacterObject (= 6)</td>
</tr>
</tbody>
</table>
Objects Class

Provides access to the list of objects.

Syntax

Dim \textit{var} As Objects

\textit{var} is the name of the Objects variable that you declare.

Application Property

The Application object.

Definition

\textbf{Application} As Object (Read-only)

Syntax

\textit{var}.Application

\textit{var} is the name of the Objects variable that you declare.

Count Property

The number of objects in the collection.

Definition

\textbf{Count} As Long (Read-only)

Syntax

\textit{var}.Count

\textit{var} is the name of the Objects variable that you declare.

Item Property

An object in the collection.

Definition

Property \textbf{Item}(\textit{Index} As Variant) As Object (Read-only)

Syntax

\textit{var}.Item(\textit{ind})

\textit{var} is the name of the Objects variable that you declare.

\textit{ind} is a Variant that contains either the index of the list of objects or the object identifier.
Parent Property

The creator of the object.

**Definition**
Parent As Object (Read-only)

**Syntax**
var.Parent

*var* is the name of the Objects variable that you declare.
Pivot Class

Provides access to a pivot object. Pivoting is a technique that enables you to switch the position of data in a report, in order to view the data from different standpoints.

Syntax

Dim var As Pivot

var is the name of the Pivot variable that you declare.

Application Property

The Application object.

Definition

Application As Object (Read-only)

Syntax

var.Application

var is the name of the Pivot variable that you declare.

Apply Method

Applies the associated Pivot type object.

Definition

Sub Apply()

Syntax

var.Apply

var is the name of the Pivot variable that you declare.

Body Property

A Body object.

Definition

Property Body(Index As Long) As DocumentVariable (Read/Write)

Syntax

var.Body(ind)

var is the name of the Pivot variable that you declare.

ind is a Long that contains the body index identifier.
BodyCount Property

The number of body objects in the block.

**Definition**

BodyCount As Long (Read-only)

**Syntax**

`var.BodyCount`

`var` is the name of the Pivot variable that you declare.

Columns Property

A Columns object.

**Definition**

Property Columns(Index As Long) As DocumentVariable (Read/Write)

**Syntax**

`var.Columns(ind)`

`var` is the name of the Pivot variable that you declare.

`ind` is a Long that contains either the index of the list of columns or the column identifier.

ColumnsCount Property

The number of columns in the block.

**Definition**

ColumnsCount As Long (Read-only)

**Syntax**

`var.ColumnsCount`

`var` is the name of the Pivot variable that you declare.

Parent Property

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Pivot variable that you declare.
Reset Method

Resets the associated Pivot type object.

**Definition**
Sub Reset()

**Syntax**
var. Reset

*var* is the name of the Pivot variable that you declare.

Rows Property

A Pivot type object.

**Definition**
Property Rows(Index As Long) As DocumentVariable (Read/Write)

**Syntax**
var.Rows(ind)

*var* is the name of the Pivot variable that you declare.

*ind* is a Long that contains either the index of the list of rows or the row identifier.

RowsCount Property

The number of rows in the block.

**Definition**
RowsCount As Long (Read-only)

**Syntax**
var.RowsCount

*var* is the name of the Pivot variable that you declare.
PredefinedCondition Class

Provides access to a predefined condition. This is a condition defined by the universe designer using DESIGNER that you apply to queries in the Query Panel.

Syntax

```vbnet
Dim var As PredefinedCondition
```

`var` is the name of the PredefinedCondition variable that you declare.

Application Property

The Application object.

Definition

```vbnet
Application As Application (Read-only)
```

Syntax

```vbnet
var.Application
```

`var` is the name of the PredefinedCondition variable that you declare.

Description Property

The description of the predefined condition.

Definition

```vbnet
Description As String (Read-only)
```

Syntax

```vbnet
var.Description
```

`var` is the name of the PredefinedCondition variable that you declare.

Name Property

The name of the predefined condition.

Definition

```vbnet
Name As String (Read-only)
```

Syntax

```vbnet
var.Name
```

`var` is the name of the PredefinedCondition variable that you declare.

Parent Property

The creator of the object.

Definition

```vbnet
Parent As Object (Read-only)
```

Syntax

```vbnet
var.Parent
```

`var` is the name of the PredefinedCondition variable that you declare.
PredefinedConditions Class

Provides access to the list of predefined conditions.

**Syntax**

```vbnet
Dim var As PredefinedConditions

var is the name of the PredefinedConditions variable that you declare.
```

**Application Property**

The Application object.

**Definition**

```vbnet
Application As Application (Read-only)
```

**Syntax**

```vbnet
var.Application

var is the name of the PredefinedConditions variable that you declare.
```

**Count Property**

The number of predefined conditions.

**Definition**

```vbnet
Count As Long (Read-only)
```

**Syntax**

```vbnet
var.Count

var is the name of the PredefinedConditions variable that you declare.
```

**Item Property**

A PredefinedCondition type object.

**Definition**

Property **Item**(Index As Variant) As PredefinedCondition (Read-only)

**Syntax**

```vbnet
var.Item(ind)

var is the name of the PredefinedConditions variable that you declare.

ind is a Variant that contains either the index of the list of predefined conditions or the predefined conditions identifier.
```
**Parent Property**

The creator of the object.

**Definition**  
`Parent` As Object (Read-only)

**Syntax**

`var`.Parent

`var` is the name of the PredefinedConditions variable that you declare.
**Queries Class**

A data provider that queries a universe can be made up of more than one query: this is called a “combined query”. The Queries class gives you access to the queries used to access data from a data provider.

**Syntax**

```vbnet
Dim var As Queries
```

`var` is the name of the Queries variable that you declare.

**Comments**

The only data providers that have a Queries object are those that query universes: query technique and freehand SQL data providers. You use the Queries class when you create a query technique data provider. Part of creating this kind of data provider is building the query on a universe.

**Combined Queries**

If a data provider has more than one Query object it is called a “combined query”. The queries you build with the Queries and Query classes must conform to the rules for combined queries which are set out in the *BusinessObjects User’s Guide*.

**Example**

The following example creates a query technique data provider and adds a result to the Results collection. When run/refreshed this data provider will return the Year object in the Time Period class of the eFashion universe.

```vbnet
Dim QT As DataProvider
Set QT = Application.ActiveDocument._
    DataProviders.AddQueryTechnique("eFashion", "")
QT.Queries.Item(1).Results.Add "Time period", "Year"
QT.Refresh 'fill the cube
```

**See Also**

“Query Class” on page 196, “Results Class” on page 213, “Conditions Class” on page 86, and “Sorts Class” on page 223. *Introduction to Developer Suite* covers how to access and create data providers and queries using the BUSINESSOBJECTS SDK.

**Add Method**

Adds a new query and returns the query object.

**Definition**

Function `Add()` As Query

**Syntax**

```vbnet
var.Add
```

`var` is the name of the Queries variable that you declare.
**Application Property**

The Application object.

**Definition**

Application As Application (Read-only)

**Syntax**

`var.Application`

*var* is the name of the Queries variable that you declare.

**Count Property**

The number of queries in the Queries collection.

**Definition**

Count As Long (Read-only)

**Syntax**

`var.Count`

*var* is the name of the Queries variable that you declare.

**DeleteTrailingBlanks Property**

A flag that determines whether or not to delete trailing blanks.

**Definition**

DeleteTrailingBlanks As Boolean (Read/Write)

**Syntax**

`var.DeleteTrailingBlanks`

*var* is the name of the Queries variable that you declare.

**Comments**

TRUE when deleting trailing blanks is allowed.

FALSE when deleting trailing blanks is *not* allowed.

Corresponds to the Delete Trailing Blanks check box in the Query Options dialog in the Query Panel.

**DuplicateRows Property**

A flag that determines whether or not duplicate rows are allowed.

**Definition**

DuplicateRows As Boolean (Read/Write)

**Syntax**

`var.DuplicateRows`

*var* is the name of the Queries variable that you declare.

**Comments**

TRUE when duplicate rows are allowed.

FALSE when duplicate rows are *not* allowed.

Corresponds to the radio buttons in the Query Options dialog in the Query Panel.
Item Property

A Query object in the Queries collection.

Definition

Property **Item**(Index As Variant) As Query (Read-only)

Syntax

```vb
var.Item(ind)
```

*var* is the name of the Queries variable that you declare.

*ind* is a Variant that contains the index of the list of queries or a string that contains the name of the query.

Comments

The item number starts at 1.

Parent Property

The creator of the object.

Definition

**Parent** As Object (Read-only)

Syntax

```vb
var.Parent
```

*var* is the name of the Queries variable that you declare.

ScopeOfAnalysis Property

The scope of analysis for the query.

Definition

**ScopeOfAnalysis** As BoScopeOfAnalysis (Read/Write)

Syntax

```vb
var.ScopeOfAnalysis
```

*var* is the name of the Queries variable that you declare, it can take the following values:

<table>
<thead>
<tr>
<th>Values for BoScopeOfAnalysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>boZeroLevelsDown = 0</td>
</tr>
<tr>
<td>boTwoLevelsDown = 2</td>
</tr>
<tr>
<td>boOneLevelsDown = 1</td>
</tr>
<tr>
<td>boThreeLevelsDown = 3</td>
</tr>
</tbody>
</table>
Sorts Property

The Sorts object containing the sorts applied to the results of the queries.

**Definition**

Sorts As Sorts

**Syntax**

`var.Sort`  

`var` is the name of the Queries variable that you declare.

**See Also**

“Sorts Class” on page 223.
Query Class

Provides access to a BUSINESSOBJECTS query. A query is a type of data provider, built in the Query Panel and run on a universe.

Syntax

Dim var As Query

var is the name of the Query variable that you declare.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Query variable that you declare.

Conditions Property

The conditions to be applied to the query.

Definition

Conditions As Conditions (Read/Write)

Syntax

var.Conditions

var is the name of the Query variable that you declare.

See Also

“Conditions Class” on page 86.

Name Property

The name of the query.

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the Query variable that you declare.
Operator Property

The operator for the query.

**Definition**

**Operator** As BoQueryOperator (Read/Write)

**Syntax**

`var.Operator`

`var` is the name of the Query variable that you declare.

<table>
<thead>
<tr>
<th>Values for BoQueryOperator</th>
</tr>
</thead>
<tbody>
<tr>
<td>boOperatorNone (= 0)</td>
</tr>
<tr>
<td>boOperatorUnion (= 1)</td>
</tr>
</tbody>
</table>

**Comment**

The first query in the Queries list must be set to boOperatorNone. This property controls how the results of the query are combined with the results of other queries in the data provider.

Parent Property

The creator of the object.

**Definition**

**Parent** As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Query variable that you declare.

Results Property

The results objects in the universe that the query fetches.

**Definition**

**Results** As Results (Read/Write)

**Syntax**

`var.Results`

`var` is the name of the Query variable that you declare.

See Also

“Results Class” on page 213.
Report Class

Provides access to a BusinessObjects report. A report is the part of a document where data is displayed. A document can contain many reports; each report has a tab at the bottom of the application window.

A report can display only a subset of data from the document. You can use any dimension—displayed or not—as input or output for a calculation.

Syntax

Dim var As Report

var is the name of the Report variable that you declare.

Comments

If you change the structure of a report, with ReportStructureItem.Delete say, you must use Report.ForceCompute to return the document to a stable state.

Example

This example displays the name of the first report of the active document. The Report class is used to declare the variable.

Dim rep As Report
Dim msgtext As String
Set rep = ActiveDocument.Reports.Item(1)
msgtext = rep.Name
MsgBox msgtext

Activate Method

Sets the current report as the default report. Activates the report in the document’s frontmost window.

Definition

Sub Activate()

Syntax

var.Activate

var is the name of the Report variable that you declare.
AddComplexFilter Method

Add a complex filter to the report. The filter is global to the report.

**Definition**

Sub AddComplexFilter(Variable As Variant, Formula As String)

**Syntax**

var.AddComplexFilter(\ nvariableName, formula)

where:

- `var` is the name of the Report variable that you declare.
- `variableName` contains the name of the report variable that you want to filter.
- `formula` is a string that contains the filter conditions.

**Example**

This example adds a complex filter to the document "Annual" to show all years where `<Number of guests>` is greater than 50.

```vbnet
Dim rpt as Report
Application.Documents.Open("Annual")
set rpt = ActiveReport
rpt.AddComplexFilter "Year", "=<Number of guests> > 50"
```

Application Property

The Application object.

**Definition**

`Application` As Application (Read-only)

**Syntax**

`var.Application`

where:

- `var` is the name of the Report variable that you declare.

ApplyTemplate Method

Applies a template to the active report.

**Definition**

Sub ApplyTemplate([FileName As Variant], [StyleAndStructure As Boolean])

**Syntax**

`var.ApplyTemplate(fileName, styleAndStructure)`

where:

- `var` is the name of the Report variable that you declare.
- `fileName` indicates the template
- `styleAndStructure` is a Boolean that is TRUE for Apply Structure and Style in the Template Options and FALSE for Apply Only the Style.
Delete Method

Deletes a report from a document.

Definition
Sub Delete()

Syntax
var.Delete

Comments
The variable Report is no longer used after you apply this method.

Example
This example deletes the first report of the active document.

dim rep as Report
set rep = ActiveDocument.Reports.Item(1)
rep.Delete

DrillMode Property

Whether or not drill mode is enabled.

Definition
DrillMode As Boolean (Read-only)

Syntax
var.DrillMode

Duplicate Method

Duplicates a report from a document.

Definition
Function Duplicate() As Report

Syntax
var.Duplicate()

ExportAsHtml Method

Saves the report in HTML format.

Definition
Function ExportAsHtml(FileName As String, [Graphs As Boolean], [Borders As Boolean], [Background As Boolean], [Foreground As Boolean], [Font As Boolean], [FreeForm As Boolean], [Frames As Boolean], [AutoRefreshTime As Long], [HtmlLayout As BoHTMLLayout]) As Boolean

Syntax
var.ExportAsHtml(fileName, [graphs], [borders], [background], [foreground], [font], [freeform], [frames], [autoRefresh], [HtmlLayout])
var is the name of the Report variable that you declare.

fileName is a string that indicates the name of the html file.
The following Boolean characteristics set the formats of the html file:
graphs displays charts and pictures
borders displays borders
background displays background color
foreground displays text colors
font sets fonts
freeform sets a free-form layout
frames uses a frame
autorefresh sets automatic reload every x seconds (rather than minutes as in the dialog box)

HtmlLayout specifies the type of page layout. BoHTMLLayout is an enumerated object that, can have the following values:

<table>
<thead>
<tr>
<th>Values for BoHTMLLayout</th>
</tr>
</thead>
<tbody>
<tr>
<td>boHTMLOnePage (=0)</td>
</tr>
<tr>
<td>All in one page. This is the default</td>
</tr>
<tr>
<td>boHTMLSectionBySection (=1)</td>
</tr>
<tr>
<td>Show each section of the report on a separate page. A table of contents column appears on the left of the page.</td>
</tr>
<tr>
<td>boHTMLBoth (=2)</td>
</tr>
<tr>
<td>Both. A table of contents column appears on the left of the page</td>
</tr>
</tbody>
</table>

ExportAsPDF Method

Save a report in Portable Document Format (PDF).

**Definition**

Sub ExportAsPDF(FileName As String)

**Syntax**

var.ExportAsPDF(fileName)

**Comments**

This method adds the file extension .pdf if it is not already given in the file name you provide.
The default file location is ./BusinessObjects/UserDocs. You can save the file to another directory by specifying the full path name of the file in fileName.
Chapter 1 BusinessObjects Object Model

ExportAsRtf Method

Saves the report in RTF format.

Definition
Function ExportAsRtf(FileName As String) As Boolean

Syntax
var.ExportAsRtf(fileName)

var is the name of the Report variable that you declare.
fileName is a string that indicates the name of the rtf file to be exported.

ExportAsText Method

Saves the report in text file format.

Definition
Function ExportAsText(FileName As String) As Boolean

Syntax
var.ExportAsText(fileName)

var is the name of the Report variable that you declare.
fileName is a string that indicates the name of the text file to be exported.

ForceCompute Method

Forces the report to be recomputed.

Definition
Sub ForceCompute()

Syntax
var.ForceCompute

var is the name of the Report variable that you declare.
Comments
Not all changes to a report initiate a computation. It is recommended that you use
this method after changing a report to ensure that the document is in a stable
state.

GeneralSectionStructure Property

The list of sections in the report.

Definition
GeneralSectionStructure As SectionStructure (Read-only)

Syntax
var.GeneralSectionStructure

var is the name of the Report variable that you declare.
Name Property

- The name of the report.

**Definition**

- **Name** As String (Read/Write)

**Syntax**

- `var.Name`

- `var` is the name of the Report variable that you declare.

**Example**

- This example changes and displays the name of the first report of the active document.

```
  dim rep as Report
  set rep = ActiveDocument.Reports.Item(1)
  rep.Name = "firstone"
  MsgBox rep.Name
```

NumberOfPages Property

- The number of pages in the report.

**Definition**

- **NumberOfPages** As Long (Read-only)

**Syntax**

- `var.NumberOfPages`

- `var` is the name of the Report variable that you declare.

PageFooter Property

- The report page footer.

**Definition**

- **PageFooter** As ReportStructureItems (Read-only)

**Syntax**

- `var.PageFooter`

- `var` is the name of the Report variable that you declare.

PageHeader Property

- The report page header.

**Definition**

- **PageHeader** As ReportStructureItems (Read-only)

**Syntax**

- `var.PageHeader`

- `var` is the name of the Report variable that you declare.
Chapter 1 BusinessObjects Object Model

Parent Property

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Report variable that you declare.

PrintOut Method

Prints the report using the printer name if specified or the current print settings.

**Definition**

Sub **PrintOut**(PrinterName As Variant)

**Syntax**

`var.PrintOut(printerName)`

`var` is the name of the Report variable that you declare.

`printerName` is the name of the printer. This parameter is optional.

**Example**

This example prints the active report.

`ActiveReport.PrintOut`

Reset Method

Resets the report.

**Definition**

Sub **Reset**()

**Syntax**

`var.Reset()`

`var` is the name of the Report variable that you declare.
Reports Class

Provides access to the list of reports of a BusinessObjects document.

**Syntax**

Dim *var* As Reports

*var* is the name of the Reports variable that you declare.

**Add Method**

Adds a report.

**Definition**

Function *Add()* As Report

**Syntax**

*var*.Add()

*var* is the name of the Reports variable that you declare.

**Application Property**

The Application object.

**Definition**

*Application* As Application (Read-only)

**Syntax**

*var*.Application

*var* is the name of the Reports variable that you declare.

**Count Property**

The number of reports in the Reports variable.

**Definition**

*Count* As Long (Read-only)

**Syntax**

*var*.Count

*var* is the name of the Reports variable that you declare.

**Comments**

If the variable is created by Document.Reports, this property returns the number of reports in the document.

**Example**

This example displays the number of available reports in the active document.

dim reps as Reports
set reps = ActiveDocument.Reports
MsgBox reps.Count
CreateQuickReport Method

Creates a new report in the document using the available objects of all data providers associated with the document, or only the objects from the specified data provider.

**Definition**

Function `CreateQuickReport([DPName As Variant]) As Report`

**Syntax**

```vba
var.CreateQuickReport([dataprovierenName])
```

*var* is the name of the Reports variable that you declare.

*dataprovierenName* is a Variant string enclosed in double quotes that specifies the name of the data source file.

**Example**

This example creates a new report and renames it Budget.

```vba
dim rep as Report
set rep = ActiveDocument.Reports.CreateQuickReport
rep.Name = "Budget"
```

Item Property

A report in the collection.

**Definition**

Property `Item(Index As Variant) As Report (Read-only)`

**Syntax**

```vba
var.Item(ind)
```

*var* is the name of the Reports variable that you declare.

*ind* is a Variant that contains the index of the list of reports or a string that contains the name of the report.

**Comments**

The item number starts at 1.

**Example**

This example gets the first report of the active document and deletes it.

```vba
dim rep as Report
set rep = ActiveDocument.Reports.Item(1)
rep.Delete
```
Parent Property

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Reports variable that you declare.
ReportStructureItem Class

Provides access to a report structure item. This is one of the items which make up a report. The types of item this may be are listed under the Type property.

Syntax

```
Dim var As ReportStructureItem
```

`var` is the name of the ReportStructureItem variable that you declare.

Comments

If you change the structure of a report, with `ReportStructureItem.Delete` say, you must use `Report.ForceCompute` to return the document to a stable state.

Application Property

The Application object.

Definition

```
Application As Application (Read-only)
```

Syntax

```
var.Application
```

`var` is the name of the ReportStructureItem variable that you declare.

ApplyStdStyle Method

Applies the default style to the report structure item.

Definition

```
Sub ApplyStdStyle()
```

Syntax

```
var.ApplyStdStyle
```

`var` is the name of the ReportStructureItem variable that you declare.

Delete Method

Deletes the report structure item.

Definition

```
Sub Delete()
```

Syntax

```
var.Delete
```

`var` is the name of the ReportStructureItem variable that you declare.

Comments

After using this method, you must use `Report.ForceCompute` to return the document to a stable state.
**Parent Property**

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

var.Parent

*var* is the name of the ReportStructureItem variable that you declare.

**Type Property**

The type of the report structure item.

**Definition**

Type As BoReportItemType (Read-only)

**Syntax**

var.Type

*var* is the name of the ReportStructureItem variable that you declare.

**Comments**

BoReportItemType is an enumerated type that can take the following values:

<table>
<thead>
<tr>
<th>Values for BoReportItemType</th>
</tr>
</thead>
<tbody>
<tr>
<td>boCell (=0)</td>
</tr>
<tr>
<td>boCrosstab (=2)</td>
</tr>
<tr>
<td>boTable (=1)</td>
</tr>
<tr>
<td>boChart (=3)</td>
</tr>
</tbody>
</table>
**ReportStructureItems Class**

Provides access to the list of report items.

**Syntax**

```vba
Dim var As ReportStructureItems
```

*var* is the name of the `ReportStructureItems` variable that you declare.

**Application Property**

The Application object.

**Definition**

`Application` As Application (Read-only)

**Syntax**

```vba
var.Application
```

*var* is the name of the `ReportStructureItems` variable that you declare.

**Count Property**

The number of report items.

**Definition**

`Count` As Long (Read-only)

**Syntax**

```vba
var.Count
```

*var* is the name of the `ReportStructureItems` variable that you declare.

**Item Property**

A report structure item in the collection.

**Definition**

`Item(Index As Long)` As ReportStructureItem (Read-only)

**Syntax**

```vba
var.Item(ind)
```

*var* is the name of the `ReportStructureItems` variable that you declare.

*ind* is a Long that contains the index of the list of report items.

**Parent Property**

The creator of the object.

**Definition**

`Parent` As Object (Read-only)

**Syntax**

```vba
var.Parent
```

*var* is the name of the `ReportStructureItems` variable that you declare.
Result Class

Provides access to a result part of a query.

Syntax

Dim var As Result

*var* is the name of the Result variable that you declare.

Comments

This property is equivalent to an element in the Results pane of the Query Panel. Only data providers that query universes can have objects of the type Result.

A result is defined by:

- a class
- an object in that class

For example, using the eFashion universe a possible result could be the Lines object of the Product class.

For an example that uses the Result class see “Results Class” on page 213.

Application Property

The Application object.

**Definition**

*Application* As Application (Read-only)

**Syntax**

var.Application

*var* is the name of the Result variable that you declare.

Class Property

The universe class that contains the Result.Object universe object.

**Definition**

*Class* As String (Read-only)

**Syntax**

var.Class

*var* is the name of the Result variable that you declare.

**See Also**

Result.Object
Chapter 1 BusinessObjects Object Model

Object Property

The object (in the universe) that defines the result.

**Definition**

`Object` As String (Read-only)

**Syntax**

`var.Object`

`var` is the name of the Result variable that you declare.

**Comment**

This property is equivalent to an element in the Results pane of the Query Panel.

**See Also**

Result.Class

Parent Property

The creator of the object.

**Definition**

`Parent` As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Result variable that you declare.
Results Class

Provides access to the results for a query.

Syntax

Dim var As Results

var is the name of the Results variable that you declare.

Comments

Adding a result to the collection is equivalent to dragging an object from
the Classes and Objects pane to the Results pane in the Query Panel.

Only queries on universes have a Results collection.

Example

The following example displays a list of the results defined for a query.

Sub ListResults()
Dim resultlist As String
Dim i As Integer
Dim qry As Query
resultlist = "The results for the first query are:" & Chr(10) & Chr(10)
Set qry = Application.ActiveDocument.DataProviders.Item(1)._  
Queries.Item(1) 'qry is the first query in the Queries collection
For i = 1 To qry.Results.Count  'For the results in the collection
  resultlist = resultlist & qry.Results.Item(i).Class_ 'get the class
  & " - " & qry.Results.Item(i).Object & Chr(10)  'and the object
Next i
MsgBox resultlist 'display the results
End Sub

See Also

“Queries Class” on page 192, and “Result Class” on page 211.

Add Method

Adds a result of the query definition.

Definition

Function Add(Class As String, Object As String) As Result
Chapter 1 BusinessObjects Object Model

Syntax

\[ \text{var}.\text{Add}(\text{className}, \text{objectName}) \]

\textit{var} is the name of the Results variable that you declare.
\textit{className} is a string that identifies the class containing the object that you are adding.
\textit{objectName} is a string that contains the name of the object that you are adding.

Application Property

The Application object.

Definition

\texttt{Application} As Application (Read-only)

Syntax

\[ \text{var}.\text{Application} \]

\textit{var} is the name of the Results variable that you declare.

Count Property

The number of results in the collection.

Definition

\texttt{Count} As Long (Read-only)

Syntax

\[ \text{var}.\text{Count} \]

\textit{var} is the name of the Results variable that you declare.

Item Property

A result in the collection.

Definition

Property \texttt{Item}(\textit{Index} As Long) As Result (Read-only)

Syntax

\[ \text{var}.\text{Item}(\textit{ind}) \]

\textit{var} is the name of the Results variable that you declare.
\textit{ind} is a Long that contains the index of the list of results.

Comments

The item number starts at 1.
**Parent Property**

The creator of the object.

**Definition**

`Parent` As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Results variable that you declare.

**Remove Method**

Removes the Result type object based on its class name and object name.

**Definition**

Sub `Remove(Class As String, Object As String)`

**Syntax**

`var.Remove(className, objectName)`

`var` is the name of the Results variable that you declare.

`className` is a string that identifies the class.

`objectName` is a string that identifies the object.

**RemoveByIndex Method**

Removes the Result type object based on its item number or its identifier.

**Definition**

Sub `RemoveByIndex(Index As Long)`

**Syntax**

`var.RemoveByIndex(ind)`

`var` is the name of the Results variable that you declare.

`ind` is a Long that contains the index of the result to be removed.
SectionStructure Class

Provides access to a section structure of a report. Only applies if a report is divided into sections.

Syntax

```
Dim var As Section
```

`var` is the name of the SectionStructure variable that you declare.

Application Property

The Application object.

Definition

```
Application As Application (Read-only)
```

Syntax

```
var.Application
```

`var` is the name of the SectionStructure variable that you declare.

Body Property

The body.

Definition

```
Body As ReportStructureItems (Read-only)
```

Syntax

```
var.Body
```

`var` is the name of the SectionStructure variable that you declare.

Delete Method

Deletes the associated SectionStructure type object.

Definition

```
Sub Delete()
```

Syntax

```
var.Delete
```

`var` is the name of the SectionStructure variable that you declare.

Footer Property

The footer.

Definition

```
Footer As ReportStructureItems (Read-only)
```

Syntax

```
var.Footer
```

`var` is the name of the SectionStructure variable that you declare.
Header Property

The header.

**Definition**

Header As ReportStructureItems (Read-only)

**Syntax**

var.Header

*var* is the name of the SectionStructure variable that you declare.

IsGeneral Property

Whether the property is set (TRUE) or not (FALSE).

**Definition**

IsGeneral As Boolean (Read-only)

**Syntax**

var.IsGeneral

*var* is the name of the SectionStructure variable that you declare.

IsTerminal Property

Whether the property is set (TRUE) or not (FALSE).

**Definition**

IsTerminal As Boolean (Read-only)

**Syntax**

var.IsTerminal

*var* is the name of the SectionStructure variable that you declare.

Master Property

The master.

**Definition**

Master As DocumentVariable (Read/Write)

**Syntax**

var.Master

*var* is the name of the SectionStructure variable that you declare.

Parent Property

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

var.Parent

*var* is the name of the SectionStructure variable that you declare.
SubSectionStructure Property

A subsection.

**Definition**

SubSectionStructure As SectionStructure (Read-only)

**Syntax**

var.SubSectionStructure

*var* is the name of the SubSection variable that you declare.
SecurityPrompts Class

Provides access to security prompts. Security prompts must be filled when accessing external data sources.

Syntax

Dim var As SecurityPrompts

var is the name of the SecurityPrompts variable that you declare.

Application Property

The Application object.

Definition

Application As Object (Read-only)

Syntax

var.Application

var is the name of the SecurityPrompts variable that you declare.

Count Property

The number of security prompts.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the SecurityPrompts variable that you declare.

GetPrompt Method

A security prompt.

Definition

Function GetPrompt(Index As Long, Title As String, UserName As String, UserPasswordVarName As String, DBName As String, DBPasswordVarName As String) As Boolean

Syntax

var.GetPrompt(Index, Title, UserName, UserPasswordVarName, DBName, DBPasswordVarName)

var is the name of the SecurityPrompts variable that you declare.

Index specifies the security prompt to return.

Title is a string containing the security prompt title.

UserName is a user name for OLAP security.

UserPasswordVarName is an OLAP security variable.
Chapter 1 BusinessObjects Object Model

DBName is the database name.
DBPasswordVarName is an OLAP security variable.

Comments You can find more information on OLAP access security in the documentation accompanying your access pack.

Parent Property

The creator of the object.

Definition Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the SecurityPrompts variable that you declare.
Sort Class

Provides access to a sort on a query technique data provider.

Syntax

Dim var As Sort

var is the name of the Sort variable that you declare.

Comments

Only queries on universes can have a sort object.

The sort defined by a Sort object is applied to the cube before it is displayed.

To apply a sort to a data provider that is not a query technique, use Column.CustomSortOrder().

To add a new sort to a query, use Sorts.Add()

Some numerical objects are represented by strings. When you sort objects like this, the numbers will be ordered as if they were strings. That is “1”, “10”, “2”, “20”, “3”, “4” not “1”, “2”, “3”, “4”, “10”, “20”.

See Also

“Column Class” on page 78, and “Sorts Class” on page 223.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Sort variable that you declare.

Ascending Property

The direction of the sort: ascending or descending.

Definition

Ascending As Boolean (Read-only)

Syntax

var.Ascending

var is the name of the Sort variable that you declare.

Comments

TRUE for ascending sort (a - Z, 1.0 - 10.0)
FALSE for descending sort (Z - a, 10.0 - 1.0).
Class Property

The universe class that contains the Sort.Object universe object.

Definition Class As String (Read-only)
Syntax var.Class

var is the name of the Sort variable that you declare.

Object Property

The universe object to which the sort is applied.

Definition Object As String (Read-only)
Syntax var.Object

var is the name of the Sort variable that you declare.

Parent Property

The creator of the object.

Definition Parent As Object (Read-only)
Syntax var.Parent

var is the name of the Sort variable that you declare.
Sorts Class

Provides access to the collection of sorts applied to a query technique data provider.

**Syntax**

```vba
Dim var As Sorts
```

*var* is the name of the Sorts variable that you declare.

**Comments**

Only query technique data providers have a Sorts collection. To sort the columns of other types of data provider, use `Column.CustomSortOrder`.

**Example**

The following example adds a sort to a query technique data provider. The new sort sorts the Name object of the Customer class in descending order.

```vba
Sub AddSort()
    Dim dp As DataProvider
    Set dp = Application.ActiveDocument.DataProviders.Item(1)
    dp.Queries.Sorts.Add Customer, Name, FALSE
End Sub
```

**Add Method**

Sorts the corresponding object.

**Definition**

```vba
Function Add(Class As String, Object As String, Ascending As Boolean) As Sort
```

*Add* is the name of the Sorts variable that you declare.

*Class* is a string that identifies the class.

*Object* is a string that identifies the object.

*Ascending* identifies the direction of the sort. TRUE is ascending, and FALSE is descending.

**See Also**

See the example above.

**Application Property**

The Application object.

**Definition**

```vba
Application As Application (Read-only)
```

*Application* is the name of the Sorts variable that you declare.
Chapter 1 BusinessObjects Object Model

Count Property

The number of sorts in the collection.

Definition
Count As Long (Read-only)

Syntax
var.Count

var is the name of the Sorts variable that you declare.

Item Property

A sort in the collection.

Definition
Property Item(Index As Long) As Sort (Read-only)

Syntax
var.Item(ind)

var is the name of the Sorts variable that you declare.

ind is a Long that contains the index of the list of sorts.

Comments
The item number starts at 1.

Parent Property

The creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the Sorts variable that you declare.

Remove Method

Removes a sort.

Definition
Sub Remove(Item As Long)

Syntax
var.Remove(ind)

var is the name of the Sorts variable that you declare.

ind is a Long that contains the index of the sort to be removed.
Universe Class

Provides access to a BusinessObjects universe. A universe is the semantic layer between you and a database, consisting of classes and objects. The objects in the universe map to data in the database, and enable you to build queries on the universe when creating or working on reports.

Syntax

```vbnet
Dim var As Universe

var is the name of the Universe variable that you declare.
```

Application Property

The Application object.

Definition

```vbnet
Application As Application (Read-only)
```

Syntax

```vbnet
var.Application

var is the name of the Universe variable that you declare.
```

Classes Property

The collection of classes in the BUSINESSOBJECTS universe.

Definition

```vbnet
Classes As Classes (Read-only)
```

Syntax

```vbnet
var.Classes

var is the name of the Universe variable that you declare.
```

DomainName Property

The universe domain name.

Definition

```vbnet
DomainName As String (Read-only)
```

Syntax

```vbnet
var.DomainName

var is the name of the Universe variable that you declare.
```
Chapter 1 BusinessObjects Object Model

LongName Property

The long name of the universe.

Definition  
LongName As String (Read-only)

Syntax  
var.LongName

var is the name of the Universe variable that you declare.

Parent Property

The creator of the object.

Definition  
Parent As Object (Read-only)

Syntax  
var.Parent

var is the name of the Universe variable that you declare.

ShortName Property

The short name of the universe.

Definition  
ShortName As String (Read-only)

Syntax  
var.ShortName

var is the name of the Universe variable that you declare.
Universes Class

Provides access to the list of universes.

Syntax

Dim var As Universes

var is the name of the Universes variable that you declare.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Universes variable that you declare.

Count Property

The number of universes in the Universes variable.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the Universes variable that you declare.

Item Property

A universe in the collection.

Definition

Property Item(Index As Variant, [RepositoryName As String]) As Universe (Read-only)

Syntax

var.Item(ind, [repositoryName])

var is the name of the Universes variable that you declare.

ind is a Variant that contains either the index of the list of universes or the universe identifier.

repositoryName optionally specifies the name of the repository.
Parent Property

The creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`varParent`

`var` is the name of the Universes variable that you declare.
UserDefined Class

Provides access to a UserDefined object of BROADCAST AGENT. This processes tasks at an interval specified by you.

Syntax
Dim var As UserDefined

var is the name of the UserDefined variable that you declare.

Application Property

The Application object.

Definition
Application As Application (Read-only)

Syntax
var.Application

var is the name of the UserDefined variable that you declare.

Every Property

The “Every x” setting for the UserDefined schedule of BROADCAST AGENT.

Definition
Every As Long (Read/Write)

Syntax
var.Every

var is the name of the UserDefined variable that you declare.

Parent Property

The creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the UserDefined variable that you declare.

Unit Property

The unit setting for the UserDefined schedule of BROADCAST AGENT.

Definition
Unit As BoDayDA (Read/Write)

Syntax
var.Unit

var is the name of the UserDefined variable that you declare.
BoDayDA can have one of the following values:

<table>
<thead>
<tr>
<th>Values for BoDayDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>boUserDefinedMinutes (= 0)</td>
</tr>
<tr>
<td>boUserDefinedDays (= 2)</td>
</tr>
<tr>
<td>boUserDefinedDays (= 2)</td>
</tr>
<tr>
<td>boUserDefinedWeekendDays (= 3)</td>
</tr>
<tr>
<td>boUserDefinedBusinessDays (= 4)</td>
</tr>
<tr>
<td>boUserDefinedMondays (= 5)</td>
</tr>
</tbody>
</table>
Variable Class

Provides access to either an application variable or a document variable.

Syntax

Dim var As Variable

var is the name of the Variable variable that you declare.

Example

The following code fragment displays the names and values of all the user prompts associated with the active document.

Dim bovarTemp As Variable
Dim strMsg As String

For i = 1 To ActiveDocument.Variables.Count
  Set bovarTemp = ActiveDocument.Variables(i)
  If bovarTemp.IsUserPrompt Then
    strMsg = bovarTemp.Name & " can take the following values:" & Chr(10) & bovarTemp.PossibleValues
    strMsg = strMsg & Chr(10) & "The user chose:" & Chr(10) & bovarTemp.Value
  End If
  MsgBox strMsg
Next i

Comments

Variables accessed via the Variables collection of the Application object contain application information. Variables accessed via the Variables collection of a Document object contain document information.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Variable variable that you declare.

Delete Method

Deletes a variable from the Variables collection.

Definition

Sub Delete()
Chapter 1 BusinessObjects Object Model

Syntax

`var.Delete`

`var` is the name of the Variable variable that you declare.

Comments

You can only delete variables created using the BUSINESSOBJECTS object model.

See Also

“Variables Class” on page 235.

Variables.Add

Example

The following code fragment adds and deletes a variable named Sample.

```vbnet
Dim bovarSample as Variable

Set bovarSample = Application.Variables.Add("Sample")
bovarSample.Value = "Sample Value"
bovarSample.Delete
```

InterpretAs Property

The type which a Variable is interpreted as.

Definition

`InterpretAs` As BoVariableInterpretAs (Read/Write)

Syntax

`var.InterpretAs`

`var` is the name of the Variable variable that you declare.

BoVariableInterpretAs is an enumerated object that specifies the type of a variable. It can have one of the following values:

<table>
<thead>
<tr>
<th>Values for BoVariableInterpretAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>boStringVariable (= 0)</td>
</tr>
<tr>
<td>boNumericVariable (= 1)</td>
</tr>
<tr>
<td>boDateVariable (= 2)</td>
</tr>
</tbody>
</table>

Comments

This property is undefined for user-defined variables.
IsUserPrompt Property

Whether or not the variable contains values chosen from a user prompt.

**Definition**  
*IsUserPrompt* As Boolean (Read-only)

**Syntax**  
```
var.IsUserPrompt
```

*var* is the name of the Variable variable that you declare.

**Comments**  
TRUE when the values in *Variable.Value* were retrieved from a user prompt, FALSE otherwise.

**See Also**  
*Variable.Value*.

MultiValued Property

Whether or not *Variable.Value* is a user prompt that contains more than one value.

**Definition**  
*MultiValued* As Boolean (Read-only)

**Syntax**  
```
var.MultiValued
```

*var* is the name of the Variable variable that you declare.

**Comments**  
TRUE when *Variable.IsUserPrompt* is TRUE and *Variable.Value* contains more than one value, FALSE otherwise.

**See Also**  
*Variable.IsUserPrompt, Variable.Value*.

Name Property

The name of a variable.

**Definition**  
*Name* As String (Read/Write)

**Syntax**  
```
var.Name
```

*var* is the name of the Variable variable that you declare.

Parent Property

The creator of the object.

**Definition**  
*Parent* As Object (Read-only)

**Syntax**  
```
var.Parent
```

*var* is the name of the Variable variable that you declare.
**PossibleValues Property**

The values that a user prompt can have.

**Definition**

PossibleValues As String (Read-only)

**Syntax**

```
var.PossibleValues
```

*var* is the name of the Variable variable that you declare.

**Comments**

This property only applies to variables that contain values received from a user prompt. When Variable.IsUserPrompt is FALSE, this property is undefined.

The values in the returned string are separated by a semi-colon.

This property returns the same list that users see when they click Values… in the user prompt dialog box.

**See Also**


**Value Property**

The value of a variable.

**Definition**

Value As String (Read/Write)

**Syntax**

```
var.Value
```

*var* is the name of the Variable variable that you declare.

**Comments**

When this property contains more than one value, the values in the returned string are separated by a semi-colon.

**See Also**

Variables Class

Provides access to either the collection of application variables or document variables.

Syntax

Dim var As Variables

var is the name of the Variables variable that you declare.

Comments

This collection appears beneath the Application class and beneath the Document class. The collection of variables beneath the Application class contains system and environment variables, for example the path defined in autoexec.bat. BUSINESSOBJECTS uses the collection of variables beneath the Document class to store the values of user prompts entered by a user to form a query.

You can add your own variables with your own meanings to both collections of Variables.

The variables you add can be accessed by other macros and add-ins.

Document variables are saved and stored when the document is saved; you can access them as long as the document is open. Application variables, however, are lost when the busobj.exe process is killed.

Example

This example displays the name of all the BUSINESSOBJECTS system variables and their values.

Dim bovarsApp as Variables

Set bovarsApp = Application.Variables
For i = 1 to bovarsApp.Count
    MsgBox bovarsApp(i).Name & " " & bovarsApp.(i).Value
Next i

See Also

“Variable Class” on page 231.

Add Method

Adds a Variable type object with the specified name.

Definition

Function Add(Name As String) As Variable

Syntax

var.Add(name)

var is the name of the Variables variable that you declare.

name is a string that contains the name of the variable to be added.
Example

This code fragment adds and deletes an application variable named Sample.

```vba
Dim boVarSample as Variable
Set boVarSample = Application.Variables.Add("Sample")
boVarSample.value = "Sample Value"
MsgBox boVarSample.Name & " " & boVarSample.Value
boVarSample.Delete
```

Application Property

The Application object.

Definition

**Application** As Application (Read-only)

Syntax

```vba
var.Application
```

_var_ is the name of the Variables variable that you declare.

Count Property

The number of variables in the collection.

Definition

**Count** As Long (Read-only)

Syntax

```vba
var.Count
```

_var_ is the name of the Variables variable that you declare.

Item Property

A variable in the collection.

Definition

**Item**(Index As Variant) As Variable (Read-only)

Syntax

```vba
var.Item(ind)
```

_var_ is the name of the Variables variable that you declare.

_ind_ is a Variant that contains the index of the list of variables or a string that contains the name of the variable.
### Parent Property

The creator of the object.

**Definition**  
`Parent` As Object (Read-only)

**Syntax**  
`var.Parent`  
`var` is the name of the Variables variable that you declare.
Chapter 1 BusinessObjects Object Model

Weekly Class

Provides access to a Weekly object of BROADCAST AGENT. The Weekly option processes the task on one day of the week at the specified time.

Syntax

```vbnet
Dim var As Weekly
```

`var` is the name of the Weekly variable that you declare.

Application Property

The Application object.

Definition

```vbnet
Application As Application (Read-only)
```

Syntax

```vbnet
var.Application
```

`var` is the name of the Weekly variable that you declare.

Parent Property

The creator of the object.

Definition

```vbnet
Parent As Object (Read-only)
```

Syntax

```vbnet
var.Parent
```

`var` is the name of the Weekly variable that you declare.

StartDay Property

The "Which day" setting for the Weekly schedule of BROADCAST AGENT.

Definition

```vbnet
StartDay As Long (Read/Write)
```

Syntax

```vbnet
var.StartDay
```

`var` is the name of the Weekly variable that you declare.

StartTime Property

The "Hour (hh:mm)" setting for the Weekly schedule of BROADCAST AGENT.

Definition

```vbnet
StartTime As Date (Read/Write)
```

Syntax

```vbnet
var.StartTime
```

`var` is the name of the Weekly variable that you declare.
**WeekPeriodicity Property**

The "Every n week(s)" setting for the Weekly schedule of BROADCAST AGENT.

**Definition**

WeekPeriodicity As Long (Read/Write)

**Syntax**

var. WeekPeriodicity

var is the name of the Weekly variable that you declare.
Window Class

Provides access to a BUSINESSOBJECTS window. The Window object is a member of the Windows collection. The Windows collection for the Application object contains all the windows in the application.

Syntax

Dim var As Window

var is the name of the Window variable that you declare.

Example

This example changes the width of the main window.

dim wnd as Window
set wnd = Application.Window
wnd.Width = 75

Activate Method

Sets the current window as the active window.

Definition

Sub Activate()

Syntax

var.Activate

var is the name of the Window variable that you declare.

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Window variable that you declare.

Caption Property

The title of the active window.

Definition

Caption As String (Read-only)

Syntax

var.Caption

var is the name of the Window variable that you declare.

Comments

If the caption is empty, this property returns a string that contains the title of the window.
Example

This example changes and displays the title of the main window.

```vba
Dim wnd as Window
set wnd = Application.Window
MsgBox "The title of the new main window is " & wnd.Caption
```

Close Method

Closes the window.

**Definition**

Sub Close()

**Syntax**

`var.Close`

`var` is the name of the Window variable that you declare.

**Comments**

You should not use the window after you implement this method.

Window.Close is not the same as Document.Close. Whereas Document.Close closes the document and all associated windows, Window.Close closes only the window. When a document is open in a single window, which is usually the case, Document.Close and Window.Close have the same effect.

**Example**

This example closes the main window and exits BUSINESSOBJECTS.

```vba
Dim wnd as Window
set wnd = Application.Window
wnd.Close
```

Height Property

The height of the window.

**Definition**

`Height As Long (Read/Write)`

**Syntax**

`var.Height`

`var` is the name of the Window variable that you declare.

**Example**

This example changes and displays the height of the main window.

```vba
Dim wnd as Window
set wnd = Application.Window
wnd.Height = 100
MsgBox "the new window height is " & wnd.Height
```
Chapter 1 BusinessObjects Object Model

**Left Property**

The horizontal position in pixels.

**Definition**

**Left** As Long (Read/Write)

**Syntax**

```
var.Left
```

*var* is the name of the Window variable that you declare.

**Example**

This example changes and displays the horizontal position of the main window.

```vba
dim wnd as Window
set wnd = Application.Window
wnd.Left = 75
MsgBox "the new window position is " & wnd.Left
```

**Parent Property**

The creator of the object.

**Definition**

**Parent** As Object (Read-only)

**Syntax**

```
var.Parent
```

*var* is the name of the Window variable that you declare.

**State Property**

The sizing of the window.

**Definition**

**State** As BoWindowState (Read/Write)

**Syntax**

```
var.State
```

*var* is the name of the Window variable that you declare.

BoWindowState is an enumerator object that specifies the state of the window. It can have one of the following values:

<table>
<thead>
<tr>
<th>Values for BoWindowState</th>
</tr>
</thead>
<tbody>
<tr>
<td>boNormal (= 1)</td>
</tr>
<tr>
<td>boMinimized (= 2)</td>
</tr>
<tr>
<td>boMaximized (= 3)</td>
</tr>
</tbody>
</table>

**Comments**

A window is either maximized, minimized or normal.
Top Property

The vertical position in pixels.

**Definition**

Top As Long (Read/Write)

**Syntax**

`var.Top`

`var` is the name of the Window variable that you declare.

**Example**

This example changes and displays the vertical position of the main window.

```vba
dim wnd as Window
set wnd = Application.Window
wnd.Top = 75
MsgBox "the new window position is " & wnd.Top
```

Width Property

The width of the active document window in pixels.

**Definition**

Width As Long (Read/Write)

**Syntax**

`var.Width`

`var` is the name of the Window variable that you declare.
Windows Class

Provides access to the list of windows of a BUSINESSOBJECTS document.

Syntax

Dim var As Windows

var is the name of the Windows variable that you declare.

Example

This example iconizes all the windows of the active document.

dim wnds as Windows

dim I as integer

set wnds = ActiveDocument.Windows

for I = 1 to wnds.Count
    wnds.Item(I).State = boMinimized
next

Application Property

The Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Windows variable that you declare.

Count Property

The number of windows in the collection.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the Windows variable that you declare.

Comments

If the variable is created by Document.Windows, this property returns the number of opened windows of the document.

Example

This example displays the number of opened windows in the active document.

dim wins as Windows

set wins = ActiveDocument.Windows

MsgBox wins.Count
Item Property

A window in the collection.

**Definition**

Property **Item**(Index As Variant) As Window

**Syntax**

```
var.Item(ind)
```

*var* is the name of the Windows variable that you declare.

*ind* contains the index of the list of windows.

**Comments**

The item number starts at 1.

**Example**

This example iconizes all the windows of the active document.

```
dim I as integer
for I = 1 to ActiveDocument.Windows.Count
    ActiveDocument.Windows.Item(I).State = boMinimized
next
```

Parent Property

The creator of the object.

**Definition**

**Parent** As Object (Read-only)

**Syntax**

```
var.Parent
```

*var* is the name of the Windows variable that you declare.
# Class List for Designer

## Application Class 256
- ActiveUniverse Property .... 256
- Application Property ......... 256
- CmdBars Property .............. 257
- Connections Property .......... 257
- GetInstallDirectory Method .. 257
- Interactive Property .......... 258
- LoginAs Method ................. 258
- Name Property ................... 259
- NetworkLayers Property ...... 259
- Parent Property ................. 260
- Quit Method ...................... 260
- UniverseDomains Property . 260
- Universes Property ............ 260
- Variables Property ............. 261
- Version Property ............... 261
- Visible Property ............... 261
- Window Property ............... 261

## CandidateClass Class 263
- Application Property .......... 263
- CandidateClasses Property .. 263
- CandidateObjects Property .. 263
- CandidatePredefinedConditions Property ................... 264
- Insert Method .................... 264
- Name Property ................... 264
- Parent Property ................. 264

## CandidateClasses Class 265
- Application Property .......... 265
- Count Property ................. 265
- Item Property ................... 265
- Parent Property ................ 265
- Refresh Method ................ 266

## CandidateJoin Class 267
- Application Property .......... 267
- Cardinality Property .......... 267
- Expression Property ........... 267
- FirstColumn Property .......... 268
- FirstTable Property ............ 268
- Insert Method ................... 268
- OuterJoin Property ............. 268
- Parent Property ................. 269
- SecondColumn Property ....... 269
- SecondTable Property .......... 269

## CandidateJoins Class 270
- Application Property .......... 270
- Count Property ................. 270
- Item Property ................... 270
- Parent Property ................ 270
- Refresh Method ................ 271

## CandidateObject Class 272
- Application Property .......... 272
- Insert Method ................... 272
- Name Property ................... 272
- Parent Property ................ 272

## CandidateObjects Class 273
- Application Property .......... 273
- Count Property ................. 273
- Item Property ................... 273
- Parent Property ................ 273

## CandidatePredefined-Condition Class 274
- Application Property .......... 274

## CheckedItem Class 277
- Application Property .......... 277
- AssociatedObject Property ... 277
- CheckError Property .......... 277
- CheckErrorDescription Property .................. 279
- Filter Property ................. 279
- Parent Property ................. 279
- SubItems Property ............. 279
- Type Property ................... 279

## CheckedItems Class 281
- Application Property .......... 281
- Count Property ................. 281
- Filter Property ................ 281
- Item Property ................... 282
- Parent Property ................. 282
- ParseLevel Property .......... 283

## Class Class 284
- Application Property .......... 284
- Classes Property ............... 284
- Delete Method .................. 284
- Description Property .......... 284
- Name Property ................... 285
- Objects Property .............. 285
Designer

Parent Property ........................ 285
PredefinedConditions Property .... 286
RootClass Property ........................ 286
Show Property ................................ 286

Classes Class .......................... 287
Add Method .............................. 287
Application Property .................... 287
Count Property ........................... 287
FindClass Property ...................... 288
Item Property ................................ 288
Parent Property .......................... 288

CmdBar Class .................................. 289
Application Property .................... 289
Name Property ................................ 289
Parent Property .......................... 289
Visible Property ........................... 289

CmdBars Class .................................. 290
Application Property .................... 290
Count Property ................................ 290
Item Property ................................ 290
Parent Property .......................... 290

Column Class .................................. 292
Application Property .................... 292
Delete Method ................................ 292
Key Property ................................ 292
Name Property ................................ 293
Parent Property .......................... 293
Type Property ................................ 293

Columns Class .................................. 294
Add Method .................................. 294
Application Property .................... 294
Count Property ........................... 294
Item Property ................................ 295
Parent Property .......................... 295

Connection Class .................................. 296
ActiveState Property ..................... 296
ActiveTime Property ..................... 296
Application Property .................... 297
ArrayFetchSize Property ............. 297
DatabaseEngine Property ............. 297
DatabaseSource Property ............ 297
Delete Method .............................. 298
IsAsyncMode Property .................. 298
Name Property .............................. 298
NetworkLayer Property ............... 299
Parent Property .......................... 299
Password Property ...................... 299
PerformCostEstimateProperty ........ 299
ServerProperty ............................ 300
TestMethod .................................. 300
Type Property ................................ 300
UseBOUserPassProperty .............. 301
UserNameProperty ....................... 301

Constraints Class .......................... 302
Add Method .............................. 302
AddDialogMethod ......................... 303
Application Property .................... 303
Count Property ................................ 303
Item Property ................................ 303
Parent Property .......................... 304

Context Class .................................. 305
Application Property .................... 305
Delete Method ................................ 305
DescriptionProperty .................... 305
Joins Property ................................ 305
NameProperty .............................. 305
ParentProperty ............................ 306

CustomHierarchies Class ................. 313
Add Method .................................. 313
Application Property .................... 313
Count Property ........................... 313
Item Property ................................ 314
Parent Property .......................... 314

CustomHierarchy Class ..................... 315
Application Property .................... 315
Delete Method ................................ 315
Dimensions Property .................... 315
Name Property .............................. 315
Parent Property .......................... 316

DatabaseEngine Class ...................... 317
Application Property .................... 317
Name Property .............................. 317
Parent Property .......................... 317
<table>
<thead>
<tr>
<th>Class</th>
<th>Properties</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatabaseEngines Class</td>
<td>Application Property, Count Property, Item Property, Parent Property</td>
<td>318</td>
</tr>
<tr>
<td>DBColumn Class</td>
<td>Application Property, Key Property, Name Property, Parent Property, Type Property</td>
<td>319</td>
</tr>
<tr>
<td>DBColumns Class</td>
<td>Application Property, Count Property, Item Property, Parent Property, Refresh Method</td>
<td>321</td>
</tr>
<tr>
<td>DBTable Class</td>
<td>Application Property, DBColumns Property, Insert Method, Name Property, Parent Property</td>
<td>323</td>
</tr>
<tr>
<td>DBTables Class</td>
<td>Application Property, Count Property, Item Property, Parent Property, Refresh Method</td>
<td>325</td>
</tr>
<tr>
<td>DefaultHierarchies Class</td>
<td>Application Property, Count Property, Item Property, Parent Property</td>
<td>327</td>
</tr>
<tr>
<td>DefaultHierarchy Class</td>
<td>Application Property, Dimensions Property, Name Property, Parent Property</td>
<td>329</td>
</tr>
<tr>
<td>Join Class</td>
<td>Application Property, Cardinality Property, Delete Method, Expression Property, FirstColumns Property, FirstTable Property, OuterJoin Property, Parent Property, Parse Method, Parent Property, SecondColumns Property, SecondTable Property, SetCardinality Method, Shortcut Property</td>
<td>331</td>
</tr>
<tr>
<td>Joins Class</td>
<td>Add Method, Application Property, Count Property, Detect Method, Item Property, Parent Property</td>
<td>335</td>
</tr>
<tr>
<td>JoinStrategies Class</td>
<td>Application Property, Count Property, Item Property, Parent Property</td>
<td>337</td>
</tr>
<tr>
<td>JoinStrategy Class</td>
<td>Application Property, Help Property, Name Property, Parent Property</td>
<td>339</td>
</tr>
<tr>
<td>LinkedUniverse Class</td>
<td>Application Property, ChangeSource Method, Delete Method, Description Property, FullName Property, LongName Property, Merge Method, Name Property, Parent Property</td>
<td>341</td>
</tr>
<tr>
<td>LinkedUniverses Class</td>
<td>Add Method, Application Property, Count Property, Item Property, Parent Property</td>
<td>344</td>
</tr>
<tr>
<td>ListofValues Class</td>
<td>Application Property, Edit Method, Name Property, Parent Property, Purge Method, Refresh Method, Values Property</td>
<td>346</td>
</tr>
<tr>
<td>NetworkLayer Class</td>
<td>Application Property, DatabaseEngines Property, Name Property, Parent Property</td>
<td>348</td>
</tr>
<tr>
<td>NetworkLayers Class</td>
<td>Application Property, Count Property, Item Property, Parent Property</td>
<td>349</td>
</tr>
<tr>
<td>Object Class</td>
<td>ActiveMonth Property, ActiveQuarter Property</td>
<td>351</td>
</tr>
</tbody>
</table>
ActiveYear Property ..............351
AggregateFunction Property ..... 352
AllowUserToEditLov Property .. 352
Application Property..............352
AssociatedDimension Property. 353
AutomaticLovRefreshBeforeUse
e Property..........................353
CanBeUsedCondition Property. 353
CanBeUsedResult Property..353
CanBeUsedSort Property .......354
DatabaseFormat Property.....354
Delete Method .....................354
Description Property .............354
ExportLovWithUniverse Property.................................354
HasListoValues Property ...355
ListOfValues Property ........355
MonthName Property ............355
Name Property .................355
Objects Property ..............356
Parent Property ..............356
Parse Method ....................356
Qualification Property .........357
QuarterName Property .........357
RootClass Property ............357
SecurityAccessLevel Property.... 358
Select Property ..................358
Show Property ..................358
Tables Property ..............359
Type Property .................359
Where Property ..............359
YearName Property ............360
ObjectStrategies Class 363
Application Property ........363
Count Property .............363
Item Property ...............363
Parent Property ..........364
ObjectStrategy Class 365
Application Property ........365
Help Property ...............365
Name Property ..............365
Parent Property ..........366
Owner Class 367
Application Property ........367
Name Property ..............367
Parent Property ..........367
Owners Class 368
Application Property ........368
Count Property .............368
Item Property ...............368
Parent Property ..........368
Refresh Method .............369
PredefinedCondition Class 370
Application Property ........370
Delete Method ...............370
Description Property .........370
Name Property ..............370
Parent Property ..........371
Parse Method ...............371
RootClass Property .........371
Show Property ..............372
Tables Property .............372
Where Property ..............372
PredefinedConditions Class 373
Add Method ....................373
Application Property ........373
Count Property .............373
Item Property ...............374
Parent Property ..........374
Qualifier Class 375
Application Property ........375
Name Property ..............375
Parent Property ..........375
Qualifiers Class 376
Application Property ........376
Count Property .............376
Item Property ...............376
Parent Property ..........376
Refresh Method .............377
SQLOption Class 378
Application Property ........378
ComplexOperators Property.378
MultipleSQLForContext Property.......................378
MultipleSQLForMeasure Property.......................379
Operators Property ..........379
Parent Property ..........379
PreventCartesianProducts Property.......................379
SelectMultipleContexts Property.......................380
Subqueries Property ........380
StoredUniverse Class 381
Application Property ........381
LockBy Property ..........381
Name Property ..............381
Parent Property ..........381
StoredUniverses Class 382
Application Property ........382
Count Property .............382
Item Property ...............382
Parent Property ..........382
Table Class 383
Table Class
Application Property .......... 383
Columns Property .......... 383
CreateAlias Method .......... 383
CreateClass Method .......... 383
Delete Method .......... 384
IncompatibleObjects Property .. 384
IncompatiblePredefConditions Property .......... 384
IsAlias Property .......... 384
Name Property .......... 384
OriginalTable Property .......... 385
Parent Property .......... 385
Weight Property .......... 385
Tables Class 386
Tables Class
Add Method .......... 386
Application Property .......... 386
Count Property .......... 386
DetectIncompatibility Method .. 386
Item Property .......... 387
Parent Property .......... 387
TableStrategies Class 388
TableStrategies Class
Application Property .......... 388
Count Property .......... 388
Item Property .......... 388
Parent Property .......... 389
TableStrategy Class 390
TableStrategy Class
Application Property .......... 390
Help Property .......... 390
Name Property .......... 390
Parent Property .......... 391
Universe Class 392
Universe Class
Activate Method .......... 392
Application Property .......... 392
ArrangeTables Method .......... 392
Author Property .......... 392
CandidateClasses Property .............. 393
CandidateJoins Property .............. 393
CheckIntegrity Method .......... 393
Classes Property .......... 394
Close Method .......... 394
Comments Property .......... 394
Connection Property .......... 394
Contexts Property .......... 395
CreationProperty .......... 395
CreationDate Property .......... 395
CurrentJoinStrategy Property .......... 395
CurrentObjectStrategy Property .......... 396
CurrentOwner Property .......... 396
CurrentQualifier Property .......... 396
CurrentTableStrategy Property .......... 396
CustomHierarchies Property .......... 397
DBTables Property .......... 397
DefaultHierarchies Property .......... 397
Description Property .......... 397
FullName Property .......... 398
Joins Property .......... 398
JoinStrategies Property .......... 398
LinkedUniverses Property .......... 398
LongName Property .......... 399
ModificationDate Property .......... 399
Modifier Property .......... 399
ObjectStrategies Property .......... 400
Owners Property .......... 400
OwnerSupported Property .......... 400
Parent Property .......... 400
Path Property .......... 401
PrintDialog Method .......... 401
PrintOut Method .......... 401
Qualifiers Property .......... 401
QualifierSupported Property .......... 402
RefreshStructure Method .......... 402
RevisionNumber Property .......... 402
Save Method .......... 402
SaveAs Method .......... 403
Saved Property .......... 403
SQLOption Property .......... 403
Tables Property .......... 403
TableStrategies Property .......... 404
UseCustomHierarchies Property .......... 404
Windows Property .......... 404
UniverseDomain Class 405
UniverseDomain Class
Application Property .......... 405
Name Property .......... 405
Parent Property .......... 405
StoredUniverses Property .......... 405
Users Property .......... 406
UniverseDomains Class 407
UniverseDomains Class
Application Property .......... 407
Count Property .......... 407
Item Property .......... 407
Parent Property .......... 407
Universes Class 408
Universes Class
Add Method .......... 408
Application Property .......... 408
Count Property .......... 408
Export Method .......... 409
ExportEx Method .......... 410
Import Method .......... 410
Item Property .......... 411
Open Method .......... 411
Parent Property .......... 411
User Class 412
User Class
Application Property .......... 412
Name Property .......... 412
Parent Property .......... 412
Users Class 413
Users Class
Application Property .......... 413
Count Property .......... 413
Item Property .......... 413
Parent Property .......... 414
Variable Class 415
Application Property..........415
Delete Method ..................415
InterpretAs Property ........416
MultiValued Property .......416
Name Property .................416
Parent Property ...............417
Value Property .................417

Variables Class 418
Add Method ....................418
Application Property ..........418
Count Property .................419
Item Property ..................419
Parent Property ...............419

Window Class 420
Activate Method ..............420
Application Property ........420
Caption Property ..............420
Close Method ..................421
Height Property ...............421
Left Property ..................421
Parent Property ...............422
State Property .................422
Top Property ..................423
Width Property ...............423

Windows Class 424
Application Property .......424
Count Property ...............424
Item Property .................424
Parent Property ...............424
In this chapter

- Details of the classes and class members in the Designer Object Model.
Application Class

The Application class represents the complete DESIGNER application. It defines settings and options within DESIGNER as well as methods that return objects such as Universe and Join.

Syntax

Dim var As Application

var is the name of the Application variable that you declare.

Example

This example launches DESIGNER and makes it visible:

Dim dsgnr As Designer.Application
Set dsgnr = CreateObject("Designer.Application")
dsgnr.Visible = True

ActiveUniverse Property

Returns the active universe, or VT_EMPTY if none.

Definition

ActiveUniverse As Universe (Read-only)

Syntax

var.ActiveUniverse

var is the name of the Application variable that you declare.

Example

This example displays the name of the active universe.

If dsgnr.Universes.Count > 0 Then
    MsgBox (dsgnr.ActiveUniverse.Name)
End If

See Also

Universe Class

Application Property

Returns the application object.

Definition

Application As Application (Read Only)

Syntax

var.Application

var is the name of the Application variable that you declare.

Comments

For the Application class, the object returned is the application itself.
Example

The following lines of code both produce the same result; they both display the name of the DESIGNER application.

MsgBox dsgnr.Name
MsgBox dsgnr.Application.Name

CmdBars Property

Returns a CmdBars object that contains the collection of command bars in DESIGNER.

Definition  
CmdBars As CmdBars (Read-only)

Syntax  
var.CmdBars

var is the name of the Application variable that you declare.

See Also  
CmdBars Class

Connections Property

Returns a Connections object which contains the available connection identifiers.

Definition  
Connections As Connections (Read-only)

Syntax  
var.Connections

var is the name of the Application variable that you declare.

See Also  
Connections Class

GetInstallDirectory Method

Returns a string with the path of the directory identified by the DirectoryID argument.

Definition  
Function GetInstallDirectory(DirectoryID As DsDirectoryID) As String

Syntax  
var.GetInstallDirectory(directoryID)

var is the name of the Application variable that you declare.
directoryID identifies the directory. DsDirectoryID is an enumerated type can take
the following values:

<table>
<thead>
<tr>
<th>Values for DsDirectoryID</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsDesignerDirectory (=0)</td>
</tr>
<tr>
<td>dsDocumentDirectory (=1)</td>
</tr>
<tr>
<td>dsTemplateDirectory (=2)</td>
</tr>
<tr>
<td>dsUniverseDirectory (=3)</td>
</tr>
<tr>
<td>dsScriptsDirectory (=4)</td>
</tr>
</tbody>
</table>

Interactive Property

Sets or returns True if the application accepts actions from the user, and False
otherwise.

**Definition**

Interactive As Boolean (Read/Write)

**Syntax**

```
var.Interactive
```

*var* is the name of the Application variable that you declare.

**Comments**

The default value is True.

This method suppresses confirmative interaction and notification with the user. Dialog boxes and other application prompts are not displayed when Interactive is set to False. (This is useful where a macro is being executed on a server or automatic schedule where user input is not possible.)

**Note:** This property must be reset to True for users to regain control of DESIGNER.

**Example**

This example sets the Interactive property to False:

```
dsgnr.Interactive = False
```

LoginAs Method

Changes the current user of DESIGNER.

**Definition**

```
Sub LoginAs([User As String], [Pass As String], [Offline As Boolean], [RepositoryName As String])
```
Syntax

var.LoginAs([userName], [password], [offline], [repositoryName])

var is the name of the Application variable that you declare.

userName is a string that contains the name of the user under which you are logging in.

password is the password of the user.

offline is a Boolean value which specifies False if there is a connection to a repository or True otherwise.

repositoryName is a string containing the name of the repository to which you are connecting.

Comments

If you omit the username or password, the Login dialog box is displayed.

Example

This example starts an offline DESIGNER session.

dsgnr.LoginAs "Manager", "boss", True

Name Property

Returns a string with the name of the DESIGNER application.

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the Application variable that you declare.

Example

This example displays the name of the DESIGNER application.

MsgBox dsgnr.Name

NetworkLayers Property

Provides access to the NetworkLayers object. The network layer is the layer between DESIGNER and the repository, for example an ODBC Driver, or Oracle client.

Definition

NetworkLayers As NetworkLayers

Syntax

var.NetworkLayers

var is the name of the Application variable that you declare.

See Also

NetworkLayers Class
Parent Property

Returns the creator of the object.

**Definition**

`Parent` As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Application variable that you declare.

**Comments**

For the Application class, the object returned is the application itself.

Quit Method

Quits the DESIGNER module.

**Definition**

Sub `Quit()`

**Syntax**

`var.Quit`

`var` is the name of the Application variable that you declare.

**Comments**

If you insert this method, the rest of the script is ignored.

**Example**

This example quits DESIGNER using the global variable Application.

`Application.Quit`

UniverseDomains Property

Returns a UniverseDomains object that contains the list of all the universes of DESIGNER.

**Definition**

`UniverseDomains` As UniverseDomains (Read-only)

**Syntax**

`var.UniverseDomains`

`var` is the name of the Application variable that you declare.

**See Also**

UniverseDomains Class

Universes Property

Returns a Universes object that contains the list of all the universes of DESIGNER.

**Definition**

`Universes` As Universes (Read-only)

**Syntax**

`var.Universes`

`var` is the name of the Application variable that you declare.

**See Also**

Universes Class
Variables Property

Returns a Variables object that lets you use the DESIGNER variables.

**Definition**
Variables As Variables (Read-only)

**Syntax**
`var.Variables`

`var` is the name of the Application variable that you declare.

**See Also**
Variables Class

Version Property

Returns a string with the version DESIGNER.

**Definition**
Version As String (Read-only)

**Syntax**
`var.Version`

`var` is the name of the Application variable that you declare.

Visible Property

Determines whether DESIGNER is visible to the user.

**Definition**
Visible As Boolean (Read/Write)

**Syntax**
`var.Visible`

`var` is the name of the Application variable that you declare.

**Example**
This example makes DESIGNER visible.

`dsgnr.Visible = True`

Window Property

Returns a Window object that lets you modify the size, the position or the state of the DESIGNER window.

**Definition**
Window As Window (Read-only)

**Syntax**
`var.Window`

`var` is the name of the Application variable that you declare.

**Comments**
If you close this window, you close DESIGNER.
Example

This example changes the width of the main window.
Dim wnd As Designer.Window
dsgrn.Visible = True
Set wnd = dsgrn.Window
wnd.Width = 75
CandidateClass Class

Provides access to a candidate class. A candidate class is defined using external strategies which are explained in the BusinessObjects Designer’s Guide.

**Syntax**

```
Dim var As CandidateClass
```

`var` is the name of the CandidateClass variable that you declare.

**Example**

This example adds a number of candidate classes to a universe.

```
For I = 1 To ADEUniverse.CandidateClasses.Count
    Set ADECandidateClass = ADEUniverse.CandidateClasses(I)
    ADECandidateClass.Insert
Next I
```

**Application Property**

Returns the application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```
var.Application
```

`var` is the name of the CandidateClass variable that you declare.

**CandidateClasses Property**

Returns a CandidateClasses object that contains a list of all the candidate classes in DESIGNER.

**Definition**

```
CandidateClasses As CandidateClasses (Read-only)
```

**Syntax**

```
var.CandidateClasses
```

`var` is the name of the CandidateClass variable that you declare.

**CandidateObjects Property**

Returns a CandidateObjects type object that contains a list of all the candidate objects in DESIGNER.

**Definition**

```
CandidateObjects As CandidateObjects (Read-only)
```

**Syntax**

```
var.CandidateObjects
```

`var` is the name of the CandidateClass variable that you declare.
CandidatePredefinedConditions Property

Returns a CandidatePredefinedConditions type object that contains a list of all the candidate predefined conditions in DESIGNER.

**Definition**

CandidatePredefinedConditions As CandidatePredefinedConditions (Read-only)

**Syntax**

```
var.CandidatePredefinedConditions
```

*var* is the name of the CandidateClass variable that you declare.

Insert Method

Inserts a candidate class.

**Definition**

Function Insert() As Class

**Syntax**

```
var.Insert()
```

*var* is the name of the CandidateClass variable that you declare.

Name Property

Returns a string with the name of the candidate class.

**Definition**

Name As String (Read-only)

**Syntax**

```
var.Name
```

*var* is the name of the CandidateClass variable that you declare.

Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

```
var.Parent
```

*var* is the name of the CandidateClass variable that you declare.
CandidateClasses Class

Provides access to the list of candidate classes.

**Syntax**

```vba
Dim var As CandidateClasses
```

*var* is the name of the CandidateClasses variable that you declare.

**Application Property**

Returns the application object.

**Definition**

```vba
Application As Application (Read-only)
```

**Syntax**

```vba
var.Application
```

*var* is the name of the CandidateClasses variable that you declare.

**Count Property**

Returns the number of candidate classes.

**Definition**

```vba
Count As Long (Read-only)
```

**Syntax**

```vba
var.Count
```

*var* is the name of the CandidateClasses variable that you declare.

**Item Property**

Returns a CandidateClass type object based on its item number or its identifier.

**Definition**

```vba
Property Item(Index As Variant) As CandidateClass (Read-only)
```

**Syntax**

```vba
var.Item(ind)
```

*var* is the name of the CandidateClasses variable that you declare.

*ind* is a Variant that contains the index of the item.

**Parent Property**

Returns the creator of the object.

**Definition**

```vba
Parent As Object (Read-only)
```

**Syntax**

```vba
var.Parent
```

*var* is the name of the CandidateClasses variable that you declare.
Refresh Method

Refreshes the data of the candidate classes.

**Definition**
Sub Refresh()

**Syntax**
var.Refresh

*var* is the name of the CandidateClasses variable that you declare.
CandidateJoin Class

Provides access to a candidate join.

Syntax

Dim var As CandidateJoin

var is the name of the CandidateJoin variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the CandidateJoin variable that you declare.

Cardinality Property

Sets or returns the cardinality of the candidate join.

Definition

Cardinality As DsCardinality (Read-only)

Syntax

var.Cardinality

var is the name of the CandidateJoin variable that you declare.

DsCardinality is an enumerated type that identifies the cardinality. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsCardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsUnknownCardinality (=0)</td>
</tr>
<tr>
<td>dsManytoOneCardinality (=3)</td>
</tr>
<tr>
<td>dsOnetoOneCardinality (=1)</td>
</tr>
<tr>
<td>dsManytoManyCardinality (=4)</td>
</tr>
<tr>
<td>dsOnetoManyCardinality (=2)</td>
</tr>
</tbody>
</table>

Expression Property

Sets or returns a string containing the expression of the candidate join.

Definition

Expression As String (Read-only)

Syntax

var.Expression

var is the name of the CandidateJoin variable that you declare.
FirstColumn Property

Returns a string with the name of the first column of the candidate join.

Definition

FirstColumn As String (Read-only)

Syntax

var.FirstColumn

var is the name of the CandidateJoin variable that you declare.

FirstTable Property

Returns a string with the name of the first table of the candidate join.

Definition

FirstTable As String (Read-only)

Syntax

var.FirstTable

var is the name of the CandidateJoin variable that you declare.

Insert Method

Inserts a candidate join.

Definition

Function Insert() As Join

Syntax

var.Insert()

var is the name of the CandidateJoin variable that you declare.

OuterJoin Property

Sets or returns the outer join status of the candidate join.

Definition

OuterJoin As DsOuterJoin (Read-only)

Syntax

var.OuterJoin

var is the name of the CandidateJoin variable that you declare.
DsOuterJoin is an enumerated type that identifies the outer join status. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsOuterJoin</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsNoOuter (=1)</td>
</tr>
<tr>
<td>dsOuterLeft (=2)</td>
</tr>
<tr>
<td>dsOuterRight (=3)</td>
</tr>
</tbody>
</table>

**Parent Property**

Returns the creator of the object.

**Definition**

`Parent` As Object (Read-only)

**Syntax**

```plaintext
var.Parent
```

*var* is the name of the CandidateJoin variable that you declare.

**SecondColumn Property**

Returns a string with the name of the second column of the candidate join.

**Definition**

`SecondColumn` As String (Read-only)

**Syntax**

```plaintext
var.SecondColumn
```

*var* is the name of the CandidateJoin variable that you declare.

**SecondTable Property**

Returns a string with the name of the second table of the candidate join.

**Definition**

`SecondTable` As String (Read-only)

**Syntax**

```plaintext
var.SecondTable
```

*var* is the name of the CandidateJoin variable that you declare.
CandidateJoins Class

Provides access to the list of candidate joins.

Syntax

Dim var As CandidateJoins

var is the name of the CandidateJoins variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the CandidateJoins variable that you declare.

Count Property

Returns the number of candidate joins.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the CandidateJoins variable that you declare.

Item Property

Returns a CandidateJoin type object based on its item number or its identifier.

Definition

Property Item(Index As Variant) As CandidateJoin (Read-only)

Syntax

var.Item(ind)

var is the name of the CandidateJoins variable that you declare.

ind is a Variant that contains the index of the item.

Parent Property

Returns the creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the CandidateJoins variable that you declare.
Refresh Method

Refreshes the data of the candidate joins.

**Definition**

Sub *Refresh*( )

**Syntax**

`var.Refresh`

`var` is the name of the CandidateJoins variable that you declare.
CandidateObject Class

Provides access to a candidate object.

Syntax

Dim var As CandidateObject

var is the name of the CandidateObject variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the CandidateObject variable that you declare.

Insert Method

Definition

Function Insert(Class As Class) As Object

Syntax

var.Insert(class)

var is the name of the CandidateObject variable that you declare.

class is the name of the class.

Name Property

Returns a string with the name of the candidate object.

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the CandidateObject variable that you declare.

Parent Property

Returns the creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the CandidateObject variable that you declare.
CandidateObjects Class

Provides access to the list of candidate objects.

**Syntax**

Dim *var* As CandidateObjects

*var* is the name of the CandidateObjects variable that you declare.

**Application Property**

Returns the application object.

**Definition**

*Application* As Application (Read-only)

**Syntax**

*var*.Application

*var* is the name of the CandidateObjects variable that you declare.

**Count Property**

Returns the number of candidate objects.

**Definition**

*Count* As Long (Read-only)

**Syntax**

*var*.Count

*var* is the name of the CandidateObjects variable that you declare.

**Item Property**

Returns a CandidateObject type object based on its item number or its identifier.

**Definition**

Property *Item*(Index As Variant) As CandidateObject (Read-only)

**Syntax**

*var*.Item(*ind*)

*var* is the name of the CandidateObjects variable that you declare.

*ind* is a Variant that contains the index of the item.

**Parent Property**

Returns the creator of the object.

**Definition**

*Parent* As Object (Read-only)

**Syntax**

*var*.Parent

*var* is the name of the CandidateObjects variable that you declare.
CandidatePredefinedCondition Class

Provides access to a candidate predefined condition.

Syntax

Dim var As CandidatePredefinedCondition

var is the name of the CandidatePredefinedCondition variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the CandidatePredefinedCondition variable that you declare.

Insert Method

Inserts a candidate predefined condition.

Definition

Function Insert(Class As Class) As PredefinedCondition

Syntax

var.Insert(class)

var is the name of the CandidatePredefinedCondition variable that you declare.

class is the name of the class.

Name Property

Returns a string with the name of the candidate predefined condition.

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the CandidatePredefinedCondition variable that you declare.

Parent Property

Returns the creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the CandidatePredefinedCondition variable that you declare.
CandidatePredefinedConditions Class

Provides access to the list of candidate predefined conditions.

Syntax

Dim var As CandidatePredefinedConditions

var is the name of the CandidatePredefinedConditions variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the CandidatePredefinedConditions variable that you declare.

Count Property

Returns the number of candidate predefined conditions.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the CandidatePredefinedConditions variable that you declare.

Item Property

Returns a CandidatePredefinedConditions type object based on its item number or its identifier.

Definition

Property Item(Index As Variant) As CandidatePredefinedCondition (Read-only)

Syntax

var.Item(ind)

var is the name of the CandidatePredefinedConditions variable that you declare.

ind is a Variant that contains the index of the item.
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

```
var.PARENT
```

`var` is the name of the CandidatePredefinedConditions variable that you declare.
CheckedItem Class

Provides access to a checked item.
A checked item is a structure, join, cardinality, object, context, or condition whose integrity has been validated. You can check the entire universe or only certain of its components.

**Syntax**

```
Dim var As CheckedItem
```

*var* is the name of the CheckedItem variable that you declare.

**Application Property**

Returns the application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```
var.Application
```

*var* is the name of the CheckedItem variable that you declare.

**AssociatedObject Property**

Returns the associated object.

**Definition**

```
AssociatedObject As Object
```

**Syntax**

```
var.AssociatedObject
```

*var* is the name of the CheckedItem variable that you declare.

**CheckError Property**

Returns the error generated by the check.

**Definition**

```
CheckError As DsCheckError
```

**Syntax**

```
var.CheckError
```

*var* is the name of the CheckedItem variable that you declare.
DsCheckError is an enumerated type. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsCheckError</th>
</tr>
</thead>
<tbody>
<tr>
<td>DsCheckErrorConnection (=1)</td>
</tr>
<tr>
<td>DsCheckErrorTableIsNotInDataStructure (=2)</td>
</tr>
<tr>
<td>DsCheckErrorTableHasChanged (=3)</td>
</tr>
<tr>
<td>DsCheckErrorColumnHasChanged (=4)</td>
</tr>
<tr>
<td>DsCheckErrorColumnNotFound (=5)</td>
</tr>
<tr>
<td>DsCheckErrorColumnHasBeenAdded (=6)</td>
</tr>
<tr>
<td>DsCheckErrorTableNotLinked (=7)</td>
</tr>
<tr>
<td>DsCheckErrorObjectParseFailed (=8)</td>
</tr>
<tr>
<td>DsCheckErrorJoinParseFailed (=9)</td>
</tr>
<tr>
<td>DsCheckErrorPredefinedConditionParseFailed (=10)</td>
</tr>
<tr>
<td>DsCheckErrorJoinMissingCardinality (=11)</td>
</tr>
<tr>
<td>DsCheckErrorJoinUnknownCardinality (=12)</td>
</tr>
<tr>
<td>DsCheckErrorJoinBadCardinality (=13)</td>
</tr>
<tr>
<td>DsCheckErrorLoopSolveByContext (=14)</td>
</tr>
<tr>
<td>DsCheckErrorLoopNotSolveByContext (=15)</td>
</tr>
<tr>
<td>DsCheckErrorLoopSolveByContextWithLoop (=16)</td>
</tr>
<tr>
<td>DsCheckErrorLoopNotSolveButContext (=17)</td>
</tr>
<tr>
<td>DsCheckErrorContextInvolvedInLoop (=18)</td>
</tr>
<tr>
<td>DsCheckErrorContextHasLoop (=19)</td>
</tr>
<tr>
<td>DsCheckErrorLoopInContext (=20)</td>
</tr>
<tr>
<td>DsCheckErrorContextIsolatedJoin (=21)</td>
</tr>
</tbody>
</table>
CheckedItem Class

**CheckErrorDescription Property**
Returns a string containing the description of the error returned by the check.

**Definition**
CheckErrorDescription As String

**Syntax**
var.CheckErrorDescription

*var* is the name of the CheckedItem variable that you declare.

**Filter Property**
Filter for the checked item.

**Definition**
Filter As Long

**Syntax**
var.Filter

*var* is the name of the CheckedItem variable that you declare.

**Parent Property**
Returns the creator of the object.

**Definition**
Parent As Object

**Syntax**
var.Parent

*var* is the name of the CheckedItem variable that you declare.

**SubItems Property**
Returns all subitems.

**Definition**
SubItems As CheckedItems

**Syntax**
var.SubItems

*var* is the name of the CheckedItem variable that you declare.

**Type Property**
Returns the type of the checked item.

**Definition**
Type As DsCheckItemType

**Syntax**
var.Type

*var* is the name of the CheckedItem variable that you declare.
DsCheckItemType is an enumerated type. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsCheckItemType</th>
</tr>
</thead>
<tbody>
<tr>
<td>DsCheckedNull (= 0)</td>
</tr>
<tr>
<td>DsCheckedTable (=1)</td>
</tr>
<tr>
<td>DsCheckedColumn (=2)</td>
</tr>
<tr>
<td>DsCheckedObject (=3)</td>
</tr>
<tr>
<td>DsCheckedJoin (=4)</td>
</tr>
<tr>
<td>DsCheckedPredefinedCondition (= 5)</td>
</tr>
<tr>
<td>DsCheckedLoop (= 6)</td>
</tr>
<tr>
<td>DsCheckedContext (= 7)</td>
</tr>
</tbody>
</table>
CheckedItems Class

Provides access to a list of checked items.
A checked item is a structure, join, cardinality, object, context, or condition whose integrity has been validated. You can check the entire universe or only certain of its components.

Syntax

```
Dim var As CheckedItems
```

*var* is the name of the CheckedItems variable that you declare.

**Application Property**

Returns the application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```
var.Application
```

*var* is the name of the CheckedItems variable that you declare.

**Count Property**

Returns the number of items in the collection.

**Definition**

```
Count As Long (Read-only)
```

**Syntax**

```
var.Count
```

*var* is the name of the CheckedItems variable that you declare.

**Filter Property**

Determines the type of check.

**Definition**

```
Filter As DsCheckFilter
```

**Syntax**

```
var.Filter
```

*var* is the name of the CheckedItems variable that you declare.
DsCheckFilter is an enumerated type. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsCheckFilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsCheckStructure (= 1)</td>
</tr>
<tr>
<td>dsCheckObject (= 2)</td>
</tr>
<tr>
<td>dsCheckJoin (= 4)</td>
</tr>
<tr>
<td>dsCheckCondition (= 8)</td>
</tr>
</tbody>
</table>

**Item Property**

Returns a CheckedItem object.

**Definition**

Item(Index As Long) As CheckedItem (Read-only)

**Syntax**

`var.Item(index)`

*var* is the name of the CheckedItems variable that you declare.

*index* is a Long that contains the index of the list of checked items.

**Parent Property**

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

*var* is the name of the CheckedItems variable that you declare.
CheckedItems Class

ParseLevel Property

Determines the level at which DESIGNER validates the checking of an item.

**Definition**

ParseLevel As DsCheckParseLevel

**Syntax**

`var.ParseLevel`

`var` is the name of the CheckedItems variable that you declare.

**Comments**

DsCheckParseLevel is an enumerated type. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsCheckParseLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>DsCheckQuickParsing (= 0)</td>
</tr>
</tbody>
</table>
Class Class

Provides access to a class of a DESIGNER universe. A class is a logical grouping of objects within a universe.

Syntax

```vba
Dim var As Class
```

*var* is the name of the Class variable that you declare.

Application Property

Returns the application object.

Definition

```vba
Application As Application (Read-only)
```

Syntax

```vba
var.Application
```

*var* is the name of the Class variable that you declare.

Classes Property

Returns a Classes type object that contains the collection of classes in the DESIGNER universe.

Definition

```vba
Classes As Classes (Read-only)
```

Syntax

```vba
var.Classes
```

*var* is the name of the Class variable that you declare.

Delete Method

Deletes the class.

Definition

```vba
Sub Delete()
```

Syntax

```vba
var.Delete
```

*var* is the name of the Class variable that you declare.

Description Property

Sets or returns a string with the description of the class.

Definition

```vba
Description As String (Read/Write)
```

Syntax

```vba
var.Description
```

*var* is the name of the Class variable that you declare.
Name Property

Returns a string with the name of the class.

**Definition**

Name As String (Read/Write)

**Syntax**

`var.Name`

`var` is the name of the Class variable that you declare.

Objects Property

Returns an Objects type object that contains the collection of objects in the class.

**Definition**

Objects As Objects (Read-only)

**Syntax**

`var.Objects`

`var` is the name of the Class variable that you declare.

Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Class variable that you declare.
PredefinedConditions Property

Provides access to the predefined conditions.

**Definition**

PredefinedConditions As PredefinedConditions (Read-only)

**Syntax**

```
var.PredefinedConditions
```

*var* is the name of the Class variable that you declare.

RootClass Property

Returns the root class.

**Definition**

RootClass As Class (Read-only)

**Syntax**

```
var.RootClass
```

*var* is the name of the Class variable that you declare.

Show Property

Whether or not the class is displayed for end-users making queries on the universe.

**Definition**

Show As Boolean (Read-only)

**Syntax**

```
var.Show
```

*var* is the name of the Class variable that you declare.

**Comments**

This property corresponds to the show/hide setting for a class in the Designer interface.

- If Class.Show is TRUE end-users can use the class elements to make queries on the universe. However, if there are elements (objects or conditions) in the class that are set to be hidden, they will not be available to the user for queries.
- If Class.Show is FALSE end-users cannot use the elements in the class to make queries on the universe.
Classes Class

Provides access to the list of classes.

Syntax

Dim var As Classes

var is the name of the Classes variable that you declare.

Add Method

Adds a class.

Definition

Function Add(Name As String) As Class

Syntax

var.Add(name)

var is the name of the Classes variable that you declare.

name is the name of the class.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Classes variable that you declare.

Count Property

Returns the number of classes.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the Classes variable that you declare.
Chapter 2 Designer Object Model

FindClass Property

Finds a class in the active universe.

**Definition**
FindClass(Name As String) As Class (Read-only)

**Syntax**
var.FindClass(name)

*var* is the name of the Classes variable that you declare.

*name* is the class name.

Item Property

Returns a Class type object based on its item number or its identifier.

**Definition**
Property Item(Index As Variant) As Class (Read-only)

**Syntax**
var.Item(ind)

*var* is the name of the Classes variable that you declare.

*ind* is a Variant that contains either the index of the list of classes or the class identifier.

Parent Property

Returns the creator of the object.

**Definition**
Parent As Object (Read-only)

**Syntax**
var.Parent

*var* is the name of the Classes variable that you declare.
CmdBar Class

Provides access to a command bar.

Syntax

```vba
Dim var As CmdBar
```

*var* is the name of the CmdBar variable that you declare.

Application Property

Returns the application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```vba
var.Application
```

*var* is the name of the CmdBar variable that you declare.

Name Property

Sets or returns a string with the name of the command bar.

**Definition**

```
Name As String (Read/Write)
```

**Syntax**

```vba
var.Name
```

*var* is the name of the CmdBar variable that you declare.

Parent Property

Returns the creator of the object.

**Definition**

```
Parent As Object (Read-only)
```

**Syntax**

```vba
var.Parent
```

*var* is the name of the CmdBar variable that you declare.

Visible Property

Sets or returns a Boolean indicating whether the Visible property of the command bar is set (True) or not (False).

**Definition**

```
Visible As Boolean (Read/Write)
```

**Syntax**

```vba
var.Visible
```

*var* is the name of the CmdBar variable that you declare.
CmdBars Class

Provides access to the list of command bars.

Syntax

Dim var As CmdBars

var is the name of the CmdBars variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the CmdBars variable that you declare.

Count Property

Returns the number of command bars.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the CmdBars variable that you declare.

Item Property

Returns a CmdBars object based on its item number or its identifier.

Definition

Property Item(Index As Variant) As CmdBar (Read-only)

Syntax

var.Item(ind)

var is the name of the CmdBars variable that you declare.

ind is a Variant that contains either the index of the list of command bars or the command bar identifier.

Parent Property

Returns the creator of the object.

Definition

Parent As Object (Read-only)
Syntax

`var.Parent`

`var` is the name of the CmdBars variable that you declare.

Example

This example displays the number of command bars in DESIGNER.

```vbnet
MsgBox dsgnr.CmdBars.Count
```
Chapter 2 Designer Object Model

Column Class

Provides access to a DESIGNER column.

Syntax

Dim var As Column

var is the name of the Column variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Column variable that you declare.

Delete Method

Deletes a column.

Definition

Sub Delete()

Syntax

var.Delete

var is the name of the Column variable that you declare.

Key Property

Sets or returns the column key.

Definition

Key As DsColumnKey (Read/Write)

Syntax

var.Key

var is the name of the Column variable that you declare.

DsColumnKey is an enumerated type that specifies the type of key. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsColumnKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsUnknownKey (= 0)</td>
</tr>
<tr>
<td>dsAllKey (= 1)</td>
</tr>
<tr>
<td>dsNoKey (=2)</td>
</tr>
<tr>
<td>dsPrimaryKey (=3)</td>
</tr>
<tr>
<td>dsSecondaryKey (=4)</td>
</tr>
<tr>
<td>dsExternKey (=5)</td>
</tr>
</tbody>
</table>
Column Class

**Name Property**

Returns a string with the name of the column.

**Definition**

Name As String (Read-only)

**Syntax**

`var.Name`

`var` is the name of the Column variable that you declare.

**Parent Property**

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Column variable that you declare.

**Type Property**

Sets or returns the column type.

**Definition**

Type As DsColumnType (Read/Write)

**Syntax**

`var.Type`

`var` is the name of the Column variable that you declare.

DsColumnType is an enumerated type that specifies the type of column. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsColumnType</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsNullColumn (= 0)</td>
</tr>
<tr>
<td>dsDateColumn (=3)</td>
</tr>
<tr>
<td>dsNumericColumn (=1)</td>
</tr>
<tr>
<td>dsTextColumn (=4)</td>
</tr>
<tr>
<td>dsCharacterColumn (=2)</td>
</tr>
<tr>
<td>dsUnknownColumn (=5)</td>
</tr>
</tbody>
</table>
Columns Class

Provides access to the list of DESIGNER columns.

Syntax

Dim var As Columns

var is the name of the Columns variable that you declare.

Add Method

Adds a column.

Definition

Function Add(Name As String) As Column

Syntax

var.Add(name)

var is the name of the Columns variable that you declare.
name is the name of the column.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Columns variable that you declare.

Count Property

Returns the number of columns in the Columns variable.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the Columns variable that you declare.
Columns Class

Item Property

Returns a Column type object based on its identifier.

**Definition**
Property **Item**(Index As Variant) As Column (Read-only)

**Syntax**
`var.Item(ind)`

*var* is the name of the Columns variable that you declare.

*ind* is a Variant that contains the index of the item.

Parent Property

Returns the creator of the object.

**Definition**
Property **Parent** As Object (Read-only)

**Syntax**
`var.Parent`

*var* is the name of the Columns variable that you declare.
Connection Class

Provides access to a DESIGNER connection.

Syntax

Dim var As Connection

var is the name of the Connection variable that you declare.

Comments

Connections are a set of parameters that provide access to a database. The available connections are listed in the Connections dialog box from the Tools menu.

ActiveState Property

Sets or returns the state of the connection.

Definition

ActiveState As DsConnectionState (Read/Write)

Syntax

var.ActiveState

var is the name of the Connection variable that you declare.

DsConnectionState is an enumerated type that identifies the state of the connection. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsConnectionState</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsDisconnectAfterEachTransaction (= 1)</td>
</tr>
<tr>
<td>dsKeepActiveForLimitedTime (= 2)</td>
</tr>
<tr>
<td>dsKeepActiveForWholeSession (=3)</td>
</tr>
</tbody>
</table>

Comments

This is the setting of the Connection Properties section of the Advanced tab of an individual network layer dialog box.

ActiveTime Property

Indicates or changes the number of minutes the connection is kept active.

Definition

ActiveTime As Long (Read/Write)

Syntax

var.ActiveTime

var is the name of the Connection variable that you declare.
Comments

This is the minutes value in the Connection Properties section of the Advanced tab of an individual network layer dialog box. It is applicable only if dsKeepActiveForLimitedTime is selected for the ActiveState property.

**Application Property**

Returns the application object.

**Definition**

*Application* As Application (Read-only)

**Syntax**

`var.Application`

*var* is the name of the Connection variable that you declare.

**ArrayFetchRequest Property**

Indicates or changes the array fetch size.

**Definition**

*ArrayFetchRequest* As Long (Read/Write)

**Syntax**

`var.ArrayFetchRequest`

*var* is the name of the Connection variable that you declare.

**Comments**

This is the Array fetch size value of the Advanced tab of an individual network layer dialog box.

**DatabaseEngine Property**

Sets or returns a string with the name of the database engine.

**Definition**

*DatabaseEngine* As String (Read/Write)

**Syntax**

`var.DatabaseEngine`

*var* is the name of the Connection variable that you declare.

**Comments**

This is the Database engine list box of the Login tab of an individual network layer dialog box.

**DatabaseSource Property**

Sets or returns a string with the name of the database source.

**Definition**

*DatabaseSource* As String (Read/Write)

**Syntax**

`var.DatabaseSource`

*var* is the name of the Connection variable that you declare.
Comments
This is the Data source name text box of the Login tab of an individual network layer dialog box.

Delete Method
Deletes the connection.

Definition
Sub Delete()

Syntax
var.Delete

Comments
This is equivalent to the Remove button of the Connections dialog box, selectable from the Tools menu.

IsAsyncMode Property
Sets or returns True if the connection is asynchronous, and False otherwise.

Definition
IsAsyncMode As Boolean (Read/Write)

Syntax
var.IsAsyncMode

Comments
This is the Connection Mode option button selection of the Advanced tab of an individual network layer dialog box.

Name Property
Sets or returns a string with the name of the connection class.

Definition
Name As String (Read-only)

Syntax
var.Name

Comments
This is the Name text box of the Login tab of an individual network layer dialog box.
NetworkLayer Property
Returns a string containing the network layer for the connection.

**Definition**  
`NetworkLayer` As String (Read-only)

**Syntax**  
`var.NetworkLayer`

**Comments**  
The network layers appear in the Network Layer list box of the Add a connection dialog box.

Parent Property
Returns the creator of the object.

**Definition**  
`Parent` As Object (Read-only)

**Syntax**  
`var.Parent`

**Comments**  
This is the Password text box of the Login tab of an individual network layer dialog box.

Password Property
Sets or returns a string with the password.

**Definition**  
`Password` As String (Read/Write)

**Syntax**  
`var.Password`

**Comments**  
This is the Password text box of the Login tab of an individual network layer dialog box.

PerformCostEstimate Property
Sets or returns True to perform cost estimate, and False otherwise.

**Definition**  
`PerformCostEstimate` As Boolean (Read/Write)

**Syntax**  
`var.PerformCostEstimate`

**Comments**  
This is the option button of the Cost Estimate section of the Advanced tab of an individual network layer dialog box.
Server Property

Sets or returns a string with the server name.

Definition
Server As String (Read/Write)

Syntax
var.Server

var is the name of the Connection variable that you declare.

Test Method

An error occurs if the test fails.

Definition
Sub Test()

Syntax
var.Test()

var is the name of the Connection variable that you declare.

Comments
This is the Test button for the connection of an individual network layer.

Type Property

Sets or returns the type of connection.

Definition
Type As DsConnectionType (Read-only)

Syntax
var.Type

var is the name of the Connection variable that you declare.

DsConnectionType is an enumerated type that identifies the connection type. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsConnectionType</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsPersonal (= 1)</td>
</tr>
<tr>
<td>dsSecured (=2)</td>
</tr>
<tr>
<td>dsShared (=3)</td>
</tr>
</tbody>
</table>

Comments
This is the Type list box of the Login tab of an individual network layer dialog box.
UsedBOUserPass Property

Determines whether the connection uses a user password.

**Definition**

UsedBOUserPass As Boolean (Read/Write)

**Syntax**

`var.UsedBOUserPass`

`var` is the name of the Connection variable that you declare.

UserName Property

Sets or returns a string with the user name of the connection class.

**Definition**

UserName As String (Read/Write)

**Syntax**

`var.UserName`

`var` is the name of the Connection variable that you declare.

**Comments**

This is the User name text box of the Login tab of an individual network layer dialog box.
Connections Class

Provides access to the list of DESIGNER connections.

**Syntax**

```vba
Dim var As Connections
```

*var* is the name of the Connections variable that you declare.

**Comments**

Connections are a set of parameters that provide access to a database. The available connections are listed in the Connections dialog box from the Tools menu.

If you create more than one instance of this collection, you might encounter locking problems. When this class is created it attempts to open and lock some files related to shared and personal connections. It releases the locks when it is destroyed.

**Add Method**

Adds a connection.

**Definition**

```vba
Function Add(Name As String, Type As DsConnectionType, NetworkLayer As String, DatabaseEngine As String, [DatabaseSource As String], [UserName As String], [Password As String], [Server As String]) As Connection
```

**Syntax**

```vba
var.Add(name, type, layer, dbengine, [dbsource], [user], [password], [server])
```

*var* is the name of the Connections variable that you declare.

*name* is a string that identifies the connection. It is the Name property of the Connection class.

*type* is a connection identifier defined by the DsConnectionType and may have a value of dsGlobalConnection, dsPersonalConnection, or dsSharedConnection.

*layer* is a string that identifies the network layer. It is the NetworkLayer property of the Connection class.

*dbengine* is a string that identifies the database engine. It is the DatabaseEngine property of the Connection class.

*dbsource* is a string that identifies the database source. This parameter is optional. It is the DatabaseSource property of the Connection class.

*user* is a string that identifies the name of the user. This parameter is optional. It is the UserName property of the Connection class.

*password* is a string that identifies the password. This parameter is optional. It is the Password property of the Connection class.
server is a string that identifies the server. This parameter is optional. It is the Server property of the Connection class.

**AddDialog Method**

Adds a dialog box which lets you select a network layer with which you can create a connection.

**Definition**
Function `AddDialog()` As Connection

**Syntax**
```vbnet
var.AddDialog
```
`var` is the name of the Connections variable that you declare.

**Application Property**

Returns the application object.

**Definition**
`Application` As Application (Read-only)

**Syntax**
```vbnet
var.Application
```
`var` is the name of the Connections variable that you declare.

**Count Property**

Returns a count of the number of connections.

**Definition**
`Count` As Long (Read-only)

**Syntax**
```vbnet
var.Count
```
`var` is the name of the Connections variable that you declare.

**Item Property**

Returns a Connection type object based on its identifier.

**Definition**
Property `Item(Index As Variant)` As Connection (Read-only)

**Syntax**
```vbnet
var.Item(ind)
```
`var` is the name of the Connections variable that you declare.
`ind` is a Variant that contains the index of the item.

**Comments**
The available connections are listed in the Connections dialog box from the Tools menu.

*BusinessObjects SDK Reference Guide 303*
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Connections variable that you declare.
Context Class

Provides access to a DESIGNER context.

Syntax

Dim var As Context

var is the name of the Context variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Context variable that you declare.

Delete Method

Deletes the context.

Definition

Sub Delete()

Syntax

var.Delete

var is the name of the Context variable that you declare.

Description Property

Sets or returns a string with the description of the context.

Definition

Description As String (Read/Write)

Syntax

var.Description

var is the name of the Context variable that you declare.

Joins Property

Returns the joins making up the context.

Definition

Joins As Joins (Read-only)

Syntax

var.Joins

var is the name of the Context variable that you declare.
Name Property

Sets or returns a string with the name of the context.

**Definition**
Name As String (Read/Write)

**Syntax**
```
var.Name
```

*var* is the name of the Context variable that you declare.

Parent Property

Returns the creator of the object.

**Definition**
Parent As Object (Read-only)

**Syntax**
```
var.Parent
```

*var* is the name of the Context variable that you declare.
Contexts Class

Provides access to the list of DESIGNER contexts.

Syntax

Dim var As Contexts

var is the name of the Contexts variable that you declare.

Add Method

Adds a context.

Definition

Function Add(Name As String) As Context

Syntax

var.Add(name)

var is the name of the Contexts variable that you declare.

name is a string that specifies the join.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Contexts variable that you declare.

Count Property

Returns the number of contexts.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the Contexts variable that you declare.

Detect Method

Detects Contexts.

Definition

Sub Detect()

Syntax

var.Detect(name)

var is the name of the Contexts variable that you declare.
Chapter 2 Designer Object Model

**Item Property**

Returns a context type object based on its identifier.

**Definition**

Property `Item(Index As Variant) As Context (Read-only)`

**Syntax**

```vbnet
var.Item(ind)
```

`var` is the name of the Contexts variable that you declare.

`ind` is a Variant that contains the index of the item.

**Parent Property**

Returns the creator of the object.

**Definition**

`Parent As Object (Read-only)`

**Syntax**

```vbnet
var.Parent
```

`var` is the name of the Contexts variable that you declare.
ControlOption Class

Provides access to a DESIGNER control option.

Syntax

Dim var As ControlOption

var is the name of the ControlOption variable that you declare.

Comments

The ControlOption class relates to the Controls tab of the Universe Parameters dialog box.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the ControlOption variable that you declare.

CostEstimateExceededValue Property

Sets or retrieves the value by which cost estimates are considered exceeded. It is applicable only if the WarnIfCostEstimateExceeded property is True.

Definition

CostEstimateExceededValue As Long (Read/Write)

Syntax

var.CostEstimateExceededValue

var is the name of the ControlOption variable that you declare.

Comments

This is the number of minutes shown in the "minutes" box for the "Warn if cost estimate exceeds:" option button in the Query Limits section of the Controls tab, Universe Parameters dialog box.

LimitExecutionTime Property

Sets or indicates whether there is a limit to the execution time (True) or not (False). The maximum number of minutes is specified by the LimitExecutionTimeValue property.

Definition

LimitExecutionTime As Boolean (Read/Write)

Syntax

var.LimitExecutionTime

var is the name of the ControlOption variable that you declare.
Comments
This is the "Limit execution time to:" option button in the Query Limits section of the Controls tab, Universe Parameters dialog box.

LimitExecutionTimeValue Property
Sets or retrieves the value of the execution time limit. It is applicable only if the LimitExecutionTime property is True.
Definition
LimitExecutionTimeValue As Long (Read/Write)
Syntax
var.LimitExecutionTimeValue
var is the name of the ControlOption variable that you declare.
Comments
This is the number of minutes shown in the "minutes" box for the "Limit execution time to:" option button in the Query Limits section of the Controls tab, Universe Parameters dialog box.

LimitSizeofLongTextObject Property
Sets or indicates whether the size of long text objects is limited (True) or not (False). The maximum number of characters is specified by the LimitSizeofLongTextObjectsValue property.
Definition
LimitSizeofLongTextObject As Boolean (Read/Write)
Syntax
var.LimitSizeofLongTextObject
var is the name of the ControlOption variable that you declare.
Comments
This is the "Limit size of long text objects to:" option button in the Query Limits section of the Controls tab, Universe Parameters dialog box.

LimitSizeofLongTextObjectValue Property
Sets or retrieves the size limit of long text objects. It is applicable only if the LimitSizeofLongTextObject property is True.
Definition
LimitSizeofLongTextObjectValue As Long (Read/Write)
Syntax
var.LimitSizeofLongTextObjectValue
var is the name of the ControlOption variable that you declare.
Comments
This is the number of characters shown in the "characters" box for the "Limit size of long text objects to:" option button in the Query Limits section of the Controls tab, Universe Parameters dialog box.
LimitSizeofResultSet Property

Sets or indicates whether the size of the result set is limited (True) or not (False). The maximum number of characters is specified by the LimitSizeofResultSetValue property.

Definition
LimitSizeofResultSet As Boolean (Read/Write)

Syntax
var.LimitSizeofResultSet

Comments
This is the “Limit size of result set to:” option button in the Query Limits section of the Controls tab, Universe Parameters dialog box.

LimitSizeofResultSetValue Property

Sets or retrieves the value of the size limit of the result set. It is applicable only if the LimitSizeofResultSet property is True.

Definition
LimitSizeofResultSetValue As Long (Read/Write)

Syntax
var.LimitSizeofResultSetValue

Comments
This is the number of rows shown in the “rows” box for the “Limit size of result set to:” option button in the Query Limits section of the Controls tab, Universe Parameters dialog box.

Parent Property

Returns the creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

Comments
var is the name of the ControlOption variable that you declare.
WarnIfCostEstimateExceeded Property

Sets or indicates whether to warn if the cost estimate is exceeded (True) or not (False). The maximum number of minutes is specified by the CostEstimateExceededValue property.

**Definition**

WarnIfCostEstimateExceeded As Boolean (Read/Write)

**Syntax**

`var.WarnIfCostEstimateExceeded`

`var` is the name of the ControlOption variable that you declare.

**Comments**

This is the “Warn if cost estimate exceeds:” option button in the Query Limits section of the Controls tab, Universe Parameters dialog box.
CustomHierarchies Class

Provides access to the list of DESIGNER custom hierarchies.

Syntax

Dim var As CustomHierarchies

var is the name of the CustomHierarchies variable that you declare.

Add Method

Adds a custom hierarchy.

Definition

Function Add(Name As String) As CustomHierarchy

Syntax

var.Add(name)

var is the name of the CustomHierarchies variable that you declare.

name is a string that specifies the hierarchy.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the CustomHierarchies variable that you declare.

Count Property

Returns the number of custom hierarchies.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the CustomHierarchies variable that you declare.
Chapter 2 Designer Object Model

**Item Property**

Returns a custom hierarchy type object based on its identifier.

**Definition**

Property `Item(Index As Variant) As CustomHierarchy (Read-only)`

**Syntax**

```
var.Item(ind)
```

*var* is the name of the `CustomHierarchies` variable that you declare.

*ind* is a Variant that contains the index of the item.

**Parent Property**

Returns the creator of the object.

**Definition**

`Parent As Object (Read-only)`

**Syntax**

```
var.Parent
```

*var* is the name of the `CustomHierarchies` variable that you declare.
CustomHierarchy Class

Provides access to a DESIGNER custom hierarchy.

Syntax

Dim var As CustomHierarchy

var is the name of the CustomHierarchy variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the CustomHierarchy variable that you declare.

Delete Method

Deletes the custom hierarchy.

Definition

Sub Delete()

Syntax

var.Delete

var is the name of the CustomHierarchy variable that you declare.

Dimensions Property

Returns a dimensions object.

Definition

Property Dimensions As Objects (Read/Write)

Syntax

var.Dimensions

var is the name of the CustomHierarchy variable that you declare.

Name Property

Returns a string containing the name of the custom hierarchy.

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the CustomHierarchy variable that you declare.
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

*var* is the name of the CustomHierarchy variable that you declare.
**DatabaseEngine Class**

**Syntax**

Dim var As DatabaseEngine

*var* is the name of the DatabaseEngine variable that you declare.

**Application Property**

Returns the application object.

**Definition**

*Application* As Application (Read-only)

**Syntax**

*var*.Application

*var* is the name of the DatabaseEngine variable that you declare.

**Name Property**

Returns a string containing the name of the database engine.

**Definition**

*Name* As String (Read-only)

**Syntax**

*var*.Name

*var* is the name of the DatabaseEngine variable that you declare.

**Parent Property**

Returns the creator of the object.

**Definition**

*Parent* As Object (Read-only)

**Syntax**

*var*.Parent

*var* is the name of the DatabaseEngine variable that you declare.
DatabaseEngines Class

Syntax
Dim var As DatabaseEngines

var is the name of the DatabaseEngines variable that you declare.

Application Property

Returns the application object.

Definition
Application As Application (Read-only)

Syntax
var.Application

var is the name of the DatabaseEngines variable that you declare.

Count Property

Returns the number of database engines in the database engines collection.

Definition
Count As Long (Read-only)

Syntax
var.Count

var is the name of the DatabaseEngines variable that you declare.

Item Property

Returns a DatabaseEngine type object based on its identifier.

Definition
Item(Index As Variant) As DatabaseEngine (Read-only)

Syntax
var.Item(ind)

var is the name of the DatabaseEngines variable that you declare.

ind is a Variant that specifies the index of the item.

Parent Property

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the DatabaseEngines variable that you declare.
DBColumn Class

Provides access to a DESIGNER DB column.

Syntax

Dim var As DBColumn

var is the name of the DBColumn variable that you declare.

Comments

You select database tables and columns when creating initial classes and objects. Database columns are displayed, for example, in the "Database tables and columns" list box (step 2 of the Quick Design Wizard).

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the DBColumn variable that you declare.

Key Property

Sets or returns the column key.

Definition

Key As DsColumnKey (Read-only)

Syntax

var.Key

var is the name of the DBColumn variable that you declare.

Name Property

Returns a string with the name of the column.

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the DBColumn variable that you declare.

Comments

The database column names are displayed in the lists of database tables and columns, for example, in the "Database tables and columns" list box (step 2 of the Quick Design Wizard).
Parent Property

Returns the creator of the object.

**Definition**

`Parent` As Object (Read-only)

**Syntax**

```
var.Parent
```

`var` is the name of the DBColumn variable that you declare.

Type Property

Sets or returns the column type.

**Definition**

`Type` As Long (Read-only)

**Syntax**

```
var.Type
```

`var` is the name of the DBColumn variable that you declare.
DBColumns Class

Provides access to the list of DESIGNER DB columns.

Syntax

Dim var As DBColumns

`var` is the name of the DBColumns variable that you declare.

Comments

You select database tables and columns when creating initial classes and objects. Database columns are displayed, for example, in the "Database tables and columns" list box (step 2 of the Quick Design Wizard).

Application Property

Returns the application object.

Definition

`Application` As Application (Read-only)

Syntax

`var`.Application

`var` is the name of the DBColumns variable that you declare.

Count Property

Returns the number of database columns in the database table.

Definition

`Count` As Long (Read-only)

Syntax

`var`.Count

`var` is the name of the DBColumns variable that you declare.

Item Property

Returns a DBColumn type object based on its identifier.

Definition

Property `Item`(Index As Variant) As DBColumn (Read-only)

Syntax

`var`.Item(ind)

`var` is the name of the DBColumns variable that you declare.
`ind` is a Variant that contains the index of the item.

Parent Property

Returns the creator of the object.

Definition

`Parent` As Object (Read-only)
syntax

var.Parent

*var* is the name of the DBColumns variable that you declare.

### Refresh Method

Refreshes the list of database columns.

**Definition**

Sub **Refresh()**

**Syntax**

`var.Refresh`

*var* is the name of the DBColumns variable that you declare.
DBTable Class

Provides access to a DESIGNER DB table.

Syntax

Dim var As DBTable

*var* is the name of the DBTable variable that you declare.

Comments

You select database tables and columns when creating initial classes and objects. Database tables are displayed, for example, in the "Database tables and columns" list box (step 2 of the Quick Design Wizard).

Application Property

Returns the application object.

Definition

*Application* As Application (Read-only)

Syntax

*var*.Application

*var* is the name of the DBTable variable that you declare.

DBColumns Property

Returns a DBColumns type object that contains the collection of database columns in the database table.

Definition

*DBColumns* As DBColumns (Read-only)

Syntax

*var*.DBColumns

*var* is the name of the DBTable variable that you declare.

Insert Method

Inserts to a database table.

Definition

Function *Insert()* As Table

Syntax

*var*.Insert()

*var* is the name of the DBTable variable that you declare.
**Name Property**

Returns a string with the name of the database table.

**Definition**

*Name* As String (Read-only)

**Syntax**

`var.Name`

`var` is the name of the `DBTable` variable that you declare.

**Comments**

The database table names are displayed in the lists of database tables and columns, for example, in the "Database tables and columns" list box (step 2 of the Quick Design Wizard).

**Parent Property**

Returns the creator of the object.

**Definition**

*Parent* As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the `DBTable` variable that you declare.
**DBTables Class**

Provides access to the list of DESIGNER DB tables.

**Syntax**

```
Dim var As DBTables
```

*var* is the name of the DBTables variable that you declare.

**Comments**

You select database tables and columns when creating initial classes and objects. Database tables are displayed, for example, in the "Database tables and columns" list box (step 2 of the Quick Design Wizard).

**Application Property**

Returns the application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```
var.Application
```

*var* is the name of the DBTables variable that you declare.

**Count Property**

Returns a count of the number of database tables.

**Definition**

```
Count As Long (Read-only)
```

**Syntax**

```
var.Count
```

*var* is the name of the DBTables variable that you declare.

**Item Property**

Returns a database tables type object based on its identifier.

**Definition**

```
Property Item(Index As Variant) As DBTable (Read-only)
```

**Syntax**

```
var.Item(ind)
```

*var* is the name of the DBTables variable that you declare.

*ind* is a Variant that contains the index of the item.
Chapter 2 Designer Object Model

Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

```vbnet
var.Parent
```

*var* is the name of the DBTables variable that you declare.

Refresh Method

Refreshes the list of database tables.

**Definition**

Sub Refresh()

**Syntax**

```vbnet
var.Refresh
```

*var* is the name of the DBTables variable that you declare.
DefaultHierarchies Class

Provides access to the list of DESIGNER default hierarchies.

**Syntax**

```vba
Dim var As DefaultHierarchies
```

`var` is the name of the DefaultHierarchies variable that you declare.

**Comments**

Default hierarchies are listed in the Hierarchies Editor dialog box, by selecting Hierarchies from the Tools menu.

**Application Property**

Returns the application object.

**Definition**

```vba
Application As Application (Read-only)
```

**Syntax**

```vba
var.Application
```

`var` is the name of the DefaultHierarchies variable that you declare.

**Count Property**

Returns a count of the number of default hierarchies.

**Definition**

```vba
Count As Long (Read-only)
```

**Syntax**

```vba
var.Count
```

`var` is the name of the DefaultHierarchies variable that you declare.

**Item Property**

Returns a default hierarchy type object based on its identifier.

**Definition**

```vba
Property Item(Index As Variant) As DefaultHierarchy (Read-only)
```

**Syntax**

```vba
var.Item(ind)
```

`var` is the name of the DefaultHierarchies variable that you declare. `ind` is a Variant that contains the index of the item.
Parent Property

Returns the creator of the object.

**Definition**

*Parent* As Object (Read-only)

**Syntax**

`var`.Parent

`var` is the name of the DefaultHierarchies variable that you declare.
DefaultHierarchy Class

Provides access to a DESIGNER default hierarchy.

Syntax

Dim var As DefaultHierarchy

var is the name of the DefaultHierarchy variable that you declare.

Comments

Default hierarchies are listed in the Hierarchies Editor dialog box, by selecting Hierarchies from the Tools menu.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the DefaultHierarchy variable that you declare.

Dimensions Property

Returns a dimensions object based on its identifier.

Definition

Property Dimensions As Objects (Read/Write)

Syntax

var.Dimensions

var is the name of the DefaultHierarchy variable that you declare.

Comments

Dimension objects of default hierarchies are listed in the Hierarchies Editor dialog box, by selecting Hierarchies from the Tools menu and clicking the plus sign (+) of the default hierarchy.

Name Property

Returns a string containing the name of the default hierarchy.

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the DefaultHierarchy variable that you declare.
**Parent Property**

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the DefaultHierarchy variable that you declare.
Join Class

Provides access to a Join object.

Syntax

Dim var As Join

*var* is the name of the Join variable that you declare.

Comments

The Join class relates to the join capabilities of DESIGNER, principally the Edit Join dialog box, displayed from selecting Join on the Insert menu.

**Application Property**

Returns the application object.

**Definition**

*Application* As Application (Read-only)

**Syntax**

*var*.Application

*var* is the name of the Join variable that you declare.

**Cardinality Property**

Sets or returns the cardinality of the join.

**Definition**

*Cardinality* As DsCardinality(Read/Write)

**Syntax**

*var*.Cardinality

*var* is the name of the Join variable that you declare.

DsCardinality is an enumerated type that identifies the cardinality. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsCardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsUnknwonCardinality (=0)</td>
</tr>
<tr>
<td>dsManytoOneCardinality (=3)</td>
</tr>
<tr>
<td>dsOneToOneCardinality (=1)</td>
</tr>
<tr>
<td>dsManytoManyCardinality (=4)</td>
</tr>
<tr>
<td>dsOneToManyCardinality (=2)</td>
</tr>
</tbody>
</table>

Comments

The cardinality is shown in the Cardinality section of the Edit Join dialog box.
Chapter 2 Designer Object Model

Delete Method

Deletes the join.

Definition
Sub Delete()

Syntax
var.Delete

Comments
This is equivalent to selecting a join in the structure window and deleting it (for example, with the Delete key).

Expression Property

Sets or returns a string containing the join formula.

Definition
Expression As String (Read/Write)

Syntax
var.Expression

Comments
The formula is displayed in the Expression text box of the Edit Join dialog box, or in the entry area of the formula bar, when a join is selected.

FirstColumns Property

Returns a column object as the first column.

Definition
FirstColumns As Columns (Read-only)

Syntax
var.FirstColumns

Comments
The first column is the selected column from the list of columns below the Table1 list box in the Edit Join dialog box.

FirstTable Property

Returns a table object as the first table.

Definition
FirstTable As Table (Read-only)

Syntax
var.FirstTable

Comments
The first table is shown in the Table1 list box in the Edit Join dialog box.
OuterJoin Property

Sets or returns the outer join status.

**Definition**

*OuterJoin* as *DsOuterJoin* (Read/Write)

**Syntax**

`var OUTERJOIN

*var* is the name of the Join variable that you declare.

*DsOuterJoin* is an enumerated type that identifies the outer join status. It can take the following values:

<table>
<thead>
<tr>
<th>Values for <em>DsOuterJoin</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>dsNoOuter (=1)</td>
</tr>
<tr>
<td>dsOuterLeft (=2)</td>
</tr>
<tr>
<td>dsOuterRight (=3)</td>
</tr>
</tbody>
</table>

**Comments**

The outer join status is shown in the two Outer join check boxes in the Edit Join dialog box.

Parent Property

Returns the creator of the object.

**Definition**

*Parent* as *Object* (Read-only)

**Syntax**

`var Parent

*var* is the name of the Join variable that you declare.

Parse Method

Parses the join.

**Definition**

Sub *Parse*

**Syntax**

`var Parse

*var* is the name of the Join variable that you declare.

**Comments**

This method is equivalent to the Parse button of the Edit Join dialog box.
SecondColumns Property
Returns a column object as the second column.

**Definition**
SecondColumns As Columns (Read-only)

**Syntax**
```
var.SecondColumns
```

`var` is the name of the Join variable that you declare.

**Comments**
The second column is the selected column from the list of columns below the Table2 list box in the Edit Join dialog box.

SecondTable Property
Returns a table object as the second table.

**Definition**
SecondTable As Table (Read-only)

**Syntax**
```
var.SecondTable
```

`var` is the name of the Join variable that you declare.

**Comments**
The second table is shown in the Table2 list box in the Edit Join dialog box.

SetCardinality Method
Sets the cardinality.

**Definition**
Sub SetCardinality()

**Syntax**
```
var.SetCardinality
```

`var` is the name of the Join variable that you declare.

**Comments**
This method is equivalent to checking the Cardinality check box of the Edit Join dialog box.

Shortcut Property
Sets or indicates whether the join is a shortcut join (True) or not (False).

**Definition**
Shortcut As Boolean (Read/Write)

**Syntax**
```
var.Shortcut
```

`var` is the name of the Join variable that you declare.

**Comments**
This method is equivalent to checking the Shortcut join check box of the Edit Join dialog box.
Joins Class

Provides access to the list of Joins.

**Syntax**

```vba
Dim var As Joins
```

`var` is the name of the Joins variable that you declare.

**Comments**

The Joins class relates to the join capabilities of DESIGNER, principally the Edit Join dialog box, displayed from selecting Join on the Insert menu.

### Add Method

Adds a join.

**Definition**

Function `Add(Expression As String) As Join`

**Syntax**

```vba
var.Add(expression)
```

`var` is the name of the Joins variable that you declare.

`expression` is a string that specifies the join formula. The general format is:

```
tablename1.columnname1 operator tablename2.columnname2
```

**Comments**

This method adds a join by defining a formula as displayed in the Expression text box of the Edit Join dialog box.

### Application Property

Returns the application object.

**Definition**

`Application As Application (Read-only)`

**Syntax**

```vba
var.Application
```

`var` is the name of the Joins variable that you declare.

### Count Property

Returns the number of joins.

**Definition**

`Count As Long (Read-only)`

**Syntax**

```vba
var.Count
```

`var` is the name of the Joins variable that you declare.

**Comments**

The count is the number of joins currently in the universe. Joins are shown graphically in the structure window.
Detect Method

Detects joins.

**Definition**

Sub Detect()

**Syntax**

var.Detect

*var* is the name of the Joins variable that you declare.

**Comments**

This method is equivalent to choosing the Detect Joins command from the Tools menu.

Item Property

Returns a join type object based on its identifier.

**Definition**

Property **Item**(Index As Variant) As Join (Read-only)

**Syntax**

*var*.Item(*ind*)

*var* is the name of the Joins variable that you declare.

*ind* is a Variant that contains the index of the item.

**Comments**

The formula defining the join is displayed in the Expression text box of the Edit Join dialog box, or in the entry area of the formula bar, when the join is selected. Joins are shown graphically in the structure window.

Parent Property

Returns the creator of the object.

**Definition**

**Parent** As Object (Read-only)

**Syntax**

*var*.Parent

*var* is the name of the Joins variable that you declare.
JoinStrategies Class

Provides access to the list of JoinStrategy objects.

**Syntax**

```vba
Dim var As JoinStrategies
```

*var* is the name of the JoinStrategies variable that you declare.

**Comments**

The JoinStrategies class relates to the Strategies tab of the Universe Parameters dialog box.

**Application Property**

Returns the application object.

**Definition**

```vba
Application As Application (Read-only)
```

**Syntax**

```vba
var.Application
```

*var* is the name of the JoinStrategies variable that you declare.

**Count Property**

Returns the number of join strategies.

**Definition**

```vba
Count As Long (Read-only)
```

**Syntax**

```vba
var.Count
```

*var* is the name of the JoinStrategies variable that you declare.

**Comments**

The count is the number of entries in the Joins list box on the Strategies tab of the Universe Parameters dialog box.

**Item Property**

Returns a JoinStrategy object based on its identifier.

**Definition**

```vba
Property Item(Index As Variant) As JoinStrategy (Read-only)
```

**Syntax**

```vba
var.Item(ind)
```

*var* is the name of the JoinStrategies variable that you declare.

*ind* is a Variant that contains the index of the item.

**Comments**

Each item is shown in the Joins list box on the Strategies tab of the Universe Parameters dialog box.
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the JoinStrategies variable that you declare.
JoinStrategy Class

Provides access to a JoinStrategy object.

**Syntax**

```vba
Dim var As JoinStrategy
```

`var` is the name of the JoinStrategy variable that you declare.

**Comments**

The JoinStrategy class relates to the Strategies tab of the Universe Parameters dialog box.

**Application Property**

Returns the application object.

**Definition**

```vba
Application As Application (Read-only)
```

**Syntax**

```vba
var.Application
```

`var` is the name of the JoinStrategy variable that you declare.

**Help Property**

**Definition**

```vba
Help As String (Read-only)
```

**Syntax**

```vba
var.Help
```

`var` is the name of the JoinStrategy variable that you declare.

**Comments**

The Help text is displayed below the Joins list box for the selected join strategy, on the Strategies tab of the Universe Parameters dialog box.

**Name Property**

Returns a string with the name of the join strategy.

**Definition**

```vba
Name As String (Read-only)
```

**Syntax**

```vba
var.Name
```

`var` is the name of the JoinStrategy variable that you declare.

**Comments**

The name is displayed in the Joins list box on the Strategies tab of the Universe Parameters dialog box.
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the JoinStrategy variable that you declare.
LinkedUniverse Class

Provides access to a DESIGNER linked universe.

Syntax

Dim var As LinkedUniverse

var is the name of the LinkedUniverse variable that you declare.

Comments

This class relates to the Links tab of the Universe Parameters dialog box.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the LinkedUniverse variable that you declare.

ChangeSource Method

Changes the path of a core universe.

Definition

Sub ChangeSource(Path As String)

Syntax

var.ChangeSource(path)

var is the name of the LinkedUniverse variable that you declare.

path is the new location of the core universe. You must include the extension .unv in the universe name.

Comments

This is equivalent to the Change Source button of the Links tab, Universe Parameters dialog box.

Delete Method

Deletes the linked universe.

Definition

Sub Delete()

Syntax

var.Delete

var is the name of the LinkedUniverse variable that you declare.

Comments

This is equivalent to the Remove Link button of the Links tab, Universe Parameters dialog box.
Chapter 2 Designer Object Model

Description Property
Sets or returns a string with the description of the linked universe.

Definition
Description As String (Read/Write)

Syntax
var.Description

Comments
The description is shown in the Description box of the Links tab, Universe Parameters dialog box.

FullName Property
Sets or returns a string with the full name of the linked universe.

Definition
FullName As String (Read-only)

Syntax
var.FullName

Comments
This is the full pathname of the .unv Universe file name.

LongName Property
Returns a string with the long name of the linked universe.

Definition
LongName As String (Read-only)

Syntax
var.LongName

Comments
The long name of the universe is defined in the Name text box of the Definition tab of the Universe Properties dialog box.

Merge Method
Merges the linked universe.

Definition
Sub Merge()

Syntax
var.Merge

Comments
var is the name of the LinkedUniverse variable that you declare.
**Name Property**

Sets or returns a string with the name of the linked universe.

**Definition**  
Name As String (Read/Write)

**Syntax**  
`var.Name`  
`var` is the name of the LinkedUniverse variable that you declare.

**Comments**  
This is the .unv Universe file name.

**Parent Property**

Returns the creator of the object.

**Definition**  
Parent As Object (Read-only)

**Syntax**  
`var.Parent`  
`var` is the name of the LinkedUniverse variable that you declare.
LinkedUniverses Class

Provides access to the list of linked universes.

Syntax

Dim var As Universes

var is the name of the Universes variable that you declare.

Comments

This class relates to the Links tab of the Universe Parameters dialog box.

Add Method

Adds a linked universe.

Definition

Function Add(Name As String) As LinkedUniverse

Syntax

var.Add(name)

var is the name of the LinkedUniverses variable that you declare.

name is a string that specifies the universe.

Comments

This method performs the Add Link button operation, with the name selected in the File Name box of the Universe to Link dialog box.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the LinkedUniverses variable that you declare.

Count Property

Returns the number of linked universes.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the LinkedUniverses variable that you declare.

Comments

It is the number of entries in the table of the Links tab of the Universe Parameters dialog box.
Item Property

Returns a linked universe object based on its identifier.

**Definition**
Property `Item(Index As Variant) As LinkedUniverse` (Read-only)

**Syntax**
```
var.Item(ind)
```

*var* is the name of the `LinkedUniverses` variable that you declare.

*ind* is a Variant that contains the index of the item.

**Comments**
Each item is shown in the table of the Links tab of the Universe Parameters dialog box.

Parent Property

Returns the creator of the object.

**Definition**
Property `Parent As Object` (Read-only)

**Syntax**
```
var.Parent
```

*var* is the name of the `LinkedUniverses` variable that you declare.
ListofValues Class

Provides access to the list of values.

**Syntax**

Dim *var* As ListofValues

*var* is the name of the ListofValues variable that you declare.

**Application Property**

Returns the application object.

**Definition**

*Application* As Application (Read-only)

**Syntax**

*var*.Application

*var* is the name of the ListofValues variable that you declare.

**Edit Method**

Modifies the list of values.

**Definition**

Sub *Edit*

**Syntax**

*var*.Edit

*var* is the name of the ListofValues variable that you declare.

**Name Property**

Sets or returns a string with the name property of the list of values.

**Definition**

*Name* As String (Read/Write)

**Syntax**

*var*.Name

*var* is the name of the ListofValues variable that you declare.

**Parent Property**

Returns the creator of the object.

**Definition**

*Parent* As Object (Read-only)

**Syntax**

*var*.Parent

*var* is the name of the ListofValues variable that you declare.
ListofValues Class

Purge Method

Purges the list of values.

Definition
Sub Purge()

Syntax
var.Purge

var is the name of the ListofValues variable that you declare.

Refresh Method

Refreshes the list of values.

Definition
Sub Refresh()

Syntax
var.Refresh

var is the name of the ListofValues variable that you declare.

Values Property

Returns the values in a ListofValues.

Definition
Values As Array of String (Read-only)

Syntax
var.Values

var is the name of the ListofValues variable that you declare.

Comments
This returns an array containing the ListofValues. Use the UBound VB function to get the number of items in the array. The lower bound of the array is always 1.

Example
This example displays all the values in the Country object’s ListofValues.

Dim vars As Variant
For i = LBound(vars) To UBound(vars)
    MsgBox vars(i)
Next i
NetworkLayer Class

Provides access to the network layer object. This is the layer between DESIGNER and the repository, for example an ODBC Driver, or Oracle client.

Syntax

Dim var As NetworkLayer

var is the name of the NetworkLayer variable that you declare.

Application Property

Returns the Application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the NetworkLayer variable that you declare.

DatabaseEngines Property

Returns the database engines available in the network layer.

Definition

DatabaseEngines As DatabaseEngines (Read-only)

Syntax

var.DatabaseEngines

var is the name of the NetworkLayer variable that you declare.

Name Property

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the NetworkLayer variable that you declare.

Parent Property

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the NetworkLayer variable that you declare.
NetworkLayers Class

Provides access to a list of NetworkLayer objects.

**Syntax**

```
Dim var As NetworkLayers
```

*var* is the name of the NetworkLayers variable that you declare.

**Comments**

To create a new NetworkLayers object, you must first dimension a variable, using the Dim statement.

To refer to a method or property of the newly created object, use the syntax: `var.property` or `var.method`.

**Application Property**

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```
var.Application
```

*var* is the name of the NetworkLayers variable that you declare.

**Count Property**

Returns the number of network layers.

**Definition**

```
Count As Long (Read-only)
```

**Syntax**

```
var.Count
```

*var* is the name of the NetworkLayers variable that you declare.

**Item Property**

Returns a NetworkLayer object based on its item number or its identifier.

**Definition**

```
Item(Index As Variant) As NetworkLayer (Read-only)
```

**Syntax**

```
var.Item(ind)
```

*var* is the name of the NetworkLayers variable that you declare.

*ind* is a Variant that contains either the index of the list of network layers or the network layer identifier.
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

```plaintext
var.Parent
```

*var* is the name of the NetworkLayers variable that you declare.
Object Class

Provides access to an object of a DESIGNER class.

Syntax

Dim var As Object

var is the name of the Object variable that you declare.

Comments

An object is a member of a class that is mapped from data in the universe. Objects are listed by name in the Universe window.

ActiveMonth Property

Determines whether a time hierarchy for the object is created. This enables end users to drill through a hierarchy consisting of months.

Definition

ActiveMonth As Boolean

Syntax

var.ActiveMonth

var is the name of the Object variable that you declare.

Comments

Only applies to a dimension object of type date.

ActiveQuarter Property

Determines whether a time hierarchy for the object is created. This enables end users to drill through a hierarchy consisting of quarters.

Definition

ActiveQuarter As Boolean

Syntax

var.ActiveQuarter

var is the name of the Object variable that you declare.

Comments

Only applies to a dimension object of type date.

ActiveYear Property

Determines whether a time hierarchy for the object is created. This enables end users to drill through a hierarchy consisting of years.

Definition

ActiveYear As Boolean

Syntax

var.ActiveYear

var is the name of the Object variable that you declare.

Comments

Only applies to a dimension object of type date.
AggregateFunction Property

Sets or returns the aggregation function of the associated object. It applies only if the Qualification property is dsMeasureObject.

**Definition**

AggregateFunction As DsObjectAggregate (Read/Write)

**Syntax**

```
var.AggregateFunction
```

`var` is the name of the Object variable that you declare.

DsObjectAggregate is an enumerator object that specifies the type of aggregate function. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsObjectAggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsAggregateBySumObject (=1)</td>
</tr>
<tr>
<td>dsAggregateByAvgObject (=4)</td>
</tr>
<tr>
<td>dsAggregateByMaxObject (=2)</td>
</tr>
<tr>
<td>dsAggregateByCountObject (=5)</td>
</tr>
<tr>
<td>dsAggregateByMinObject (=3)</td>
</tr>
<tr>
<td>dsAggregateByNullObject (=6)</td>
</tr>
</tbody>
</table>

AllowUserToEditLov Property

Sets or returns a Boolean indicating whether to allow the user to edit the list of values (True) or not (False).

**Definition**

AllowUserToEditLov As Boolean (Read/Write)

**Syntax**

```
var.AllowUserToEditLov
```

`var` is the name of the Object variable that you declare.

**Comments**

A list of values is edited from the Query Panel.

Application Property

Returns the application object.

**Definition**

Application As Application (Read-only)

**Syntax**

```
var.Application
```

`var` is the name of the Object variable that you declare.
**AssociatedDimension Property**

Returns the dimension associated with the object.

**Definition**

`AssociatedDimension` As Object (Read/Write)

**Syntax**

`var.AssociatedDimension`

`var` is the name of the Object variable that you declare.

---

**AutomaticLovRefreshBeforeUse Property**

Sets or returns a Boolean indicating whether to perform automatic refresh before use (True) or not (False).

**Definition**

`AutomaticLovRefreshBeforeUse` As Boolean (Read/Write)

**Syntax**

`var.AutomaticLovRefreshBeforeUse`

`var` is the name of the Object variable that you declare.

---

**CanBeUsedCondition Property**

Sets or returns a Boolean indicating whether the object can be used in conditions (True) or not (False).

**Definition**

`CanBeUsedCondition` As Boolean (Read/Write)

**Syntax**

`var.CanBeUsedCondition`

`var` is the name of the Object variable that you declare.

---

**CanBeUsedResult Property**

Sets or returns a Boolean indicating whether the object can be used in results (True) or not (False).

**Definition**

`CanBeUsedResult` As Boolean (Read/Write)

**Syntax**

`var.CanBeUsedResult`

`var` is the name of the Object variable that you declare.
Chapter 2 Designer Object Model

CanBeUsedSort Property

Sets or returns a Boolean indicating whether the object can be used in sorts (True) or not (False).

**Definition**

CppType As Boolean (Read/Write)

**Syntax**

`var.CtoBeUsedSort`  
`var` is the name of the Object variable that you declare.

DatabaseFormat Property

Sets or returns a string with the database format of the object.

**Definition**

DatabaseFormat As String (Read/Write)

**Syntax**

`var.DatabaseFormat`  
`var` is the name of the Object variable that you declare.

Delete Method

Deletes the object.

**Definition**

Sub Delete()  

**Syntax**

`var.Delete`  
`var` is the name of the Object variable that you declare.

Description Property

Sets or returns a string with the description of the object.

**Definition**

Description As String (Read/Write)

**Syntax**

`var.Description`  
`var` is the name of the Object variable that you declare.

**Comments**

The description is shown in the Description list box on the Definition tab of the object’s Edit Properties dialog box.

ExportLovWithUniverse Property

Sets or returns a Boolean indicating whether to export the list of values with the universe (True) or not (False).
### ExportLovWithUniverse Property

Sets or returns a Boolean indicating whether there is a list of values for the object (True) or not (False).

**Definition**

**ExportLovWithUniverse** As Boolean (Read/Write)

**Syntax**

```plaintext
var.ExportLovWithUniverse
```

*var* is the name of the Object variable that you declare.

### HasListofValues Property

Sets or returns a Boolean indicating whether there is a list of values for the object (True) or not (False).

**Definition**

**HasListofValues** As Boolean (Read/Write)

**Syntax**

```plaintext
var.HasListofValues
```

*var* is the name of the Object variable that you declare.

### ListofValues Property

Provides access to the list of values.

**Definition**

**ListofValues** As ListofValues (Read-only)

**Syntax**

```plaintext
var.ListofValues
```

*var* is the name of the Object variable that you declare.

### MonthName Property

Returns a string containing the name of the month associated with the object.

**Definition**

**MonthName** As String (Read-only)

**Syntax**

```plaintext
var.MonthName
```

*var* is the name of the Object variable that you declare.

**Comments**

By default, the name is Month of ObjectName.

Only applies to a dimension object of type date.

### Name Property

Sets or returns a string with the name of the object.

**Definition**

**Name** As String (Read/Write)

**Syntax**

```plaintext
var.Name
```

*var* is the name of the Object variable that you declare.
Chapter 2 Designer Object Model

Comments
The name appears in the Universe window and elsewhere to identify the object. The name can be edited in the Definition tab of the object’s Edit Properties dialog box.

Objects Property
Returns an Objects type object that contains the collection of objects in the class.

Definition
Objects As Objects (Read-only)

Syntax
var.Objects

var is the name of the Object variable that you declare.

Parent Property
Returns the creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the Object variable that you declare.

Parse Method
Parses the object.

Definition
Sub Parse()

Syntax
var.Parse

var is the name of the Object variable that you declare.
Qualification Property

Sets or returns the qualification of the associated object.

**Definition**

Qualification As DsObjectQualification (Read/Write)

**Syntax**

`var.Qualification`  
`var` is the name of the Object variable that you declare.

DsObjectQualification is an enumerated type that specifies the type of qualification. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsObjectQualification</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsDimensionObject (=1)</td>
</tr>
<tr>
<td>dsDetailObject (=2)</td>
</tr>
<tr>
<td>dsMeasureObject (=3)</td>
</tr>
</tbody>
</table>

**Comments**

It relates to the Qualification section on the Properties tab of the object's Edit Properties dialog box

QuarterName Property

Returns a string containing the name of the quarter associated with the object.

**Definition**

QuarterName As String (Read-only)

**Syntax**

`var.QuarterName`  
`var` is the name of the Object variable that you declare.

**Comments**

By default, the name is Quarter of ObjectName.  
Only applies to a dimension object of type date.

RootClass Property

Returns the root class.

**Definition**

RootClass As Class

**Syntax**

`var.RootClass`  
`var` is the name of the Object variable that you declare.
SecurityAccessLevel Property

Sets or returns the security access level of the associated object.

**Definition**  
`SecurityAccessLevel` As `DsObjectSecurityAccess` (Read/Write)

**Syntax**  
`var.SecurityAccessLevel`

`var` is the name of the Object variable that you declare.

`DsObjectSecurityAccess` is an enumerated type that specifies the security access level. It can take the following values:

<table>
<thead>
<tr>
<th>Values for <code>DsObjectSecurityAccess</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>dsPublicAccess (0)</td>
</tr>
<tr>
<td>dsConfidentialAccess (3)</td>
</tr>
<tr>
<td>dsControlledAccess (1)</td>
</tr>
<tr>
<td>dsPrivateAccess (4)</td>
</tr>
<tr>
<td>dsRestrictedAccess (2)</td>
</tr>
</tbody>
</table>

Select Property

Sets or returns a string that contains the Select statement of the object's SQL definition.

**Definition**  
`Select` As `String` (Read/Write)

**Syntax**  
`var.Select`

`var` is the name of the Object variable that you declare.

**Comments**
The Select statement is shown in the Select text box on the Definition tab of the object's Edit Properties dialog box.

Show Property

Sets or returns a Boolean indicating whether the Show property of the object is set (True) or not (False). A hidden object is not visible in `BusinessObjects`.

**Definition**  
`Show` As `Boolean` (Read/Write)

**Syntax**  
`var.Show`

`var` is the name of the Object variable that you declare.

**Comments**
Setting Show to False is equivalent to clicking Hide Item on the Edit menu with the object selected in the Universe window. A hidden object is displayed in italics in the Universe window.
Tables Property

Returns the list of tables that make up the object.

**Definition**

*Tables* As *Tables*

**Syntax**

```plaintext
var.Tables
```

*var* is the name of the Object variable that you declare.

Type Property

Sets or returns the object type.

**Definition**

*Type* As *DsObjectType* (Read/Write)

**Syntax**

```plaintext
var.Type
```

*var* is the name of the Object variable that you declare.

*DsObjectType* is an enumerated type that specifies the type of object. It can take the following values:

<table>
<thead>
<tr>
<th>Values for <em>DsObjectType</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>dsNullObject (=0)</td>
</tr>
<tr>
<td>dsNumericObject (=1)</td>
</tr>
<tr>
<td>dsCharacterObject (=2)</td>
</tr>
<tr>
<td>dsDateObject (=3)</td>
</tr>
<tr>
<td>dsBlobObject (=4)</td>
</tr>
<tr>
<td>dsUnknownObject (=5)</td>
</tr>
</tbody>
</table>

**Comments**

The type is shown in the Type list box on the Definition tab of the object's Edit Properties dialog box.

Where Property

Sets or returns a string with the Where property of the object.

**Definition**

*Where* As *String* (Read/Write)

**Syntax**

```plaintext
var.Where
```

*var* is the name of the Object variable that you declare.

**Comments**

The Where clause is shown in the Where text box on the Definition tab of the object's Edit Properties dialog box.
YearName Property

Returns a string containing the name of the year associated with the object.

**Definition**

\[ \text{YearName As String} \]

**Syntax**

\[ \text{var.YearName} \]

\[ \text{var is the name of the Object variable that you declare.} \]

**Comments**

By default, the name is Year of ObjectName.

Only applies to a dimension object of type date.
Objects Class

Provides access to the list of objects.

Syntax

\[
\text{Dim } \text{var As Objects}
\]

\text{var} is the name of the Objects variable that you declare.

Comments

An object is a member of a class that is mapped from data in the universe. Objects are listed by name in the Universe window.

Add Method

Adds an object instance.

Definition

\[
\text{Function Add(\text{Name As String, [\text{ClassName As String}] As Object}}
\]

Syntax

\[
\text{var}.\text{Add(name, [class])}
\]

\text{var} is the name of the Objects variable that you declare.

\text{name} is the name of the object.

\text{class} is the name of the class containing the object. This parameter should be used in cases where more than one objects exist with the same name.

Application Property

Returns the application object.

Definition

\[
\text{Application As Application (Read-only)}
\]

Syntax

\[
\text{var}.\text{Application}
\]

\text{var} is the name of the Objects variable that you declare.

Count Property

Returns a count of the number of objects in the class.

Definition

\[
\text{Count As Long (Read-only)}
\]

Syntax

\[
\text{var}.\text{Count}
\]

\text{var} is the name of the Objects variable that you declare.
Chapter 2 Designer Object Model

Item Property

Returns an object based on its identifier.

**Definition**

Property `Item(Index As Variant) As Object (Read-only)`

**Syntax**

```
var.Item(ind)
```

*var* is the name of the `Objects` variable that you declare.

*ind* is a Variant that contains the index of the item.

Parent Property

Returns the creator of the object.

**Definition**

`Parent As Object (Read-only)`

**Syntax**

```
var.Parent
```

*var* is the name of the `Objects` variable that you declare.
ObjectStrategies Class

Provides access to the list of object strategies.

**Syntax**

```
Dim var As ObjectStrategies
```

*var* is the name of the ObjectStrategies variable that you declare.

**Comments**
The ObjectStrategies class relates to the Strategies tab of the Universe Parameters dialog box.

---

**Application Property**

Returns the application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```
var.Application
```

*var* is the name of the ObjectStrategies variable that you declare.

**Count Property**

Returns the number of object strategies.

**Definition**

```
Count As Long (Read-only)
```

**Syntax**

```
var.Count
```

*var* is the name of the ObjectStrategies variable that you declare.

**Comments**
The count is the number of entries in the Objects list box on the Strategies tab of the Universe Parameters dialog box.

---

**Item Property**

Returns an ObjectStrategy object based on its identifier.

**Definition**

```
Property Item(Index As Variant) As ObjectStrategy (Read-only)
```

**Syntax**

```
var.Item(ind)
```

*var* is the name of the ObjectStrategies variable that you declare.

*ind* is a Variant that contains the index of the item.

**Comments**
Each item is shown in the Objects list box on the Strategies tab of the Universe Parameters dialog box.
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the ObjectStrategies variable that you declare.
ObjectStrategy Class

Provides access to an object strategy of a DESIGNER class.

Syntax

Dim var As ObjectStrategy

var is the name of the ObjectStrategy variable that you declare.

Comments

The ObjectStrategy class relates to the Strategies tab of the Universe Parameters dialog box.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the ObjectStrategy variable that you declare.

Help Property

Definition

Help As String (Read-only)

Syntax

var.Help

var is the name of the ObjectStrategy variable that you declare.

Comments

The Help text is displayed below the Objects list box for the selected object strategy, on the Strategies tab of the Universe Parameters dialog box.

Name Property

Returns a string with the name of the object strategy.

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the ObjectStrategy variable that you declare.

Comments

The name is displayed in the Objects list box on the Strategies tab of the Universe Parameters dialog box.
Parent Property

Returns the creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the ObjectStrategy variable that you declare.
Owner Class

Provides access to a DESIGNER owner class.

**Syntax**

```
Dim var As Owner
```

*var* is the name of the Owner variable that you declare.

---

**Application Property**

Returns the application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```
var.Application
```

*var* is the name of the Owner variable that you declare.

---

**Name Property**

Returns a string with the name of the owner.

**Definition**

```
Name As String (Read-only)
```

**Syntax**

```
var.Name
```

*var* is the name of the Owner variable that you declare.

---

**Parent Property**

Returns the creator of the object.

**Definition**

```
Parent As Object (Read-only)
```

**Syntax**

```
var.Parent
```

*var* is the name of the Owner variable that you declare.
Owners Class

Provides access to the list of DESIGNER owners.

Syntax

Dim var As Owners

*var* is the name of the Owners variable that you declare.

**Application Property**

Returns the application object.

**Definition**

`Application` As Application (Read-only)

**Syntax**

`var`.Application

*var* is the name of the Owners variable that you declare.

**Count Property**

Returns the number of owners.

**Definition**

`Count` As Long (Read-only)

**Syntax**

`var`.Count

*var* is the name of the Owners variable that you declare.

**Item Property**

Returns an owner type object based on its identifier.

**Definition**

Property `Item`(Index As Variant) As Owner (Read-only)

**Syntax**

`var`.Item(*ind*)

*var* is the name of the Owners variable that you declare.  
*ind* is a Variant that contains the index of the item.

**Parent Property**

Returns the creator of the object.

**Definition**

`Parent` As Object (Read-only)

**Syntax**

`var`.Parent

*var* is the name of the Owners variable that you declare.
## Refresh Method

Refreshes the list of owners.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Sub Refresh()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>var.Refresh</td>
</tr>
</tbody>
</table>

*var* is the name of the Owners variable that you declare.
PredefinedCondition Class

Provides access to a predefined condition.

**Syntax**

```vbnet
Dim var As PredefinedCondition
```

*var* is the name of the PredefinedCondition variable that you declare.

**Application Property**

Returns the application object.

**Definition**

```vbnet
Application As Application (Read-only)
```

**Syntax**

```vbnet
var.Application
```

*var* is the name of the PredefinedCondition variable that you declare.

**Delete Method**

Deletes the predefined condition.

**Definition**

```vbnet
Sub Delete()
```

**Syntax**

```vbnet
var.Delete
```

*var* is the name of the PredefinedCondition variable that you declare.

**Description Property**

Sets or returns a string with the description of the predefined condition.

**Definition**

```vbnet
Description As String (Read/Write)
```

**Syntax**

```vbnet
var.Description
```

*var* is the name of the PredefinedCondition variable that you declare.

**Name Property**

Sets or returns a string with the name of the predefined condition.

**Definition**

```vbnet
Name As String (Read/Write)
```

**Syntax**

```vbnet
var.Name
```

*var* is the name of the PredefinedCondition variable that you declare.
PredefinedCondition Class

Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the PredefinedCondition variable that you declare.

Parse Method

Parses the predefined condition.

**Definition**

Sub Parse()

**Syntax**

`var.Parse`

`var` is the name of the PredefinedCondition variable that you declare.

RootClass Property

Returns the root class.

**Definition**

RootClass As Class

**Syntax**

`var.RootClass`

`var` is the name of the PredefinedCondition variable that you declare.
Chapter 2 Designer Object Model

Show Property

Determines whether or not the class is shown.

Definition

Show As Boolean

Syntax

var.Show

var is the name of the PredefinedCondition variable that you declare.

Tables Property

Returns the list of tables that make up the class.

Definition

Tables As Tables

Syntax

var.Tables

var is the name of the PredefinedCondition variable that you declare.

Where Property

Sets or returns a string with the Where property of the predefined condition.

Definition

Where As String (Read/Write)

Syntax

var.Where

var is the name of the PredefinedCondition variable that you declare.
PredefinedConditions Class

Provides access to the list of predefined conditions.

**Syntax**

```
Dim var As PredefinedConditions
```

*var* is the name of the PredefinedConditions variable that you declare.

### Add Method

Adds a predefined condition.

**Definition**

```
Function Add(Name As String, [ClassName As String]) As PredefinedCondition
```

**Syntax**

```
var.Add(name, class)
```

*var* is the name of the PredefinedConditions variable that you declare.

*name* is the name of the predefined condition.

*class* is the name of the class containing the predefined condition. This parameter should be used in cases where more than one predefined conditions exist with the same name.

### Application Property

Returns the application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```
var.Application
```

*var* is the name of the PredefinedConditions variable that you declare.

### Count Property

Returns the number of predefined conditions.

**Definition**

```
Count As Long (Read-only)
```

**Syntax**

```
var.Count
```

*var* is the name of the PredefinedConditions variable that you declare.
Chapter 2 Designer Object Model

**Item Property**

Returns a PredefinedCondition object based on its item number or its identifier.

**Definition**

Property `Item(Index As Variant) As PredefinedCondition (Read-only)`

**Syntax**

```vbs
var.Item(ind)
```

*var* is the name of the `PredefinedConditions` variable that you declare.

*ind* is a Variant that contains either the index of the list of predefined conditions or the predefined conditions identifier.

**Parent Property**

Returns the creator of the object.

**Definition**

Parent `As Object (Read-only)`

**Syntax**

```vbs
var.Parent
```

*var* is the name of the `PredefinedConditions` variable that you declare.
Qualifier Class

Provides access to a DESIGNER qualifier.

**Syntax**

```
Dim var As Qualifier
```

*var* is the name of the Qualifier variable that you declare.

**Application Property**

Returns the application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```
var.Application
```

*var* is the name of the Qualifier variable that you declare.

**Name Property**

Returns a string with the name of the qualifier.

**Definition**

```
Name As String (Read-only)
```

**Syntax**

```
var.Name
```

*var* is the name of the Qualifier variable that you declare.

**Parent Property**

Returns the creator of the object.

**Definition**

```
Parent As Object (Read-only)
```

**Syntax**

```
var.Parent
```

*var* is the name of the Qualifier variable that you declare.
Qualifiers Class

Provides access to the list of DESIGNER qualifiers.

Syntax

Dim var As Qualifiers

var is the name of the Qualifiers variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Qualifiers variable that you declare.

Count Property

Returns the number of qualifiers.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the Qualifiers variable that you declare.

Item Property

Returns a qualifier type object based on its identifier.

Definition

Property Item(Index As Variant) As Qualifier (Read-only)

Syntax

var.Item(ind)

var is the name of the Qualifiers variable that you declare.

ind is a Variant that contains the index of the item.

Parent Property

Returns the creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the Qualifiers variable that you declare.
Refresh Method

Refreshes the list of qualifiers.

**Definition**

Sub **Refresh**()

**Syntax**

`var.Refresh`

`var` is the name of the Qualifiers variable that you declare.
Chapter 2 Designer Object Model

**SQLOption Class**

Provides access to a DESIGNER SQL option.

**Syntax**

```vbnet
Dim var As SQLOption
```

`var` is the name of the SQLOption variable that you declare.

**Comments**

This class defines the settings of the SQL tab of the Universe Parameters dialog box.

**Application Property**

Returns the application object.

**Definition**

```vbnet
Application As Application (Read-only)
```

**Syntax**

```vbnet
var.Application
```

`var` is the name of the SQLOption variable that you declare.

**ComplexOperators Property**

Sets or indicates whether complex operators in the Query Panel are allowed (True) or not (False).

**Definition**

```vbnet
ComplexOperators As Boolean (Read/Write)
```

**Syntax**

```vbnet
var.ComplexOperators
```

`var` is the name of the SQLOption variable that you declare.

**Comments**

This is the "Allow complex operators in Query Panel” option button in the Query section of the SQL tab, Universe Parameters dialog box.

**MultipleSQLForContext Property**

Sets or indicates whether multiple SQL statements for contexts are allowed (True) or not (False).

**Definition**

```vbnet
MultipleSQLForContext As Boolean (Read/Write)
```

**Syntax**

```vbnet
var.MultipleSQLForContext
```

`var` is the name of the SQLOption variable that you declare.

**Comments**

This is the "Multiple SQL statements for each context” option button in the Multiple Paths section of the SQL tab, Universe Parameters dialog box.
MultipleSQLForMeasure Property

Sets or indicates whether multiple SQL statements for measures are allowed (True) or not (False).

Definition: **MultipleSQLForMeasure** As Boolean (Read/Write)

Syntax:
```
var.MultipleSQLForMeasure
```

*var* is the name of the SQLOption variable that you declare.

Comments: This is the "Multiple SQL statements for each measure" option button in the Multiple Paths section of the SQL tab, Universe Parameters dialog box.

Operators Property

Sets or indicates whether union, intersect and minus operators are allowed (True) or not (False).

Definition: **Operators** As Boolean (Read/Write)

Syntax:
```
var.Operators
```

*var* is the name of the SQLOption variable that you declare.

Comments: This is the "Allow use of union, intersect and minus operators" option button in the Query section of the SQL tab, Universe Parameters dialog box.

Parent Property

Returns the creator of the object.

Definition: **Parent** As Object (Read-only)

Syntax:
```
var.Parent
```

*var* is the name of the SQLOption variable that you declare.

PreventCartesianProducts Property

Sets or indicates whether Cartesian products are to be prevented (True) or a warning issued (False).

Definition: **PreventCartesianProducts** As Boolean (Read/Write)

Syntax:
```
var.PreventCartesianProducts
```

*var* is the name of the SQLOption variable that you declare.
Chapter 2 Designer Object Model

Comments
This is the button selection in the Cartesian Products section of the SQL tab, Universe Parameters dialog box. It may be either Prevent or Warn.

SelectMultipleContexts Property

Sets or indicates whether multiple contexts are allowed (True) or not (False).

Definition
SelectMultipleContexts As Boolean (Read/Write)

Syntax
var.SelectMultipleContexts

Comments
This is the "Allow selection of multiple contexts" option button in the Multiple Paths section of the SQL tab, Universe Parameters dialog box.

Subqueries Property

Sets or indicates whether subqueries are allowed (True) or not (False).

Definition
Subqueries As Boolean (Read/Write)

Syntax
var.Subqueries

Comments
This is the "Allow use of subqueries" option button in the Query section of the SQL tab, Universe Parameters dialog box.
StoredUniverse Class

Provides access to a DESIGNER stored universe.

Syntax

Dim var As StoredUniverse

var is the name of the StoredUniverse variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the StoredUniverse variable that you declare.

LockBy Property

Returns a string with the containing the name of the person who locked the universe.

Definition

LockBy As String (Read-only)

Syntax

var.LockBy

var is the name of the StoredUniverse variable that you declare.

Name Property

Returns a string with the name of the stored universe.

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the StoredUniverse variable that you declare.

Parent Property

Returns the creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the StoredUniverse variable that you declare.
Chapter 2 Designer Object Model

**StoredUniverses Class**

Provides access to the list of stored universes.

**Syntax**

Dim var As StoredUniverses

*var* is the name of the StoredUniverses variable that you declare.

**Application Property**

Returns the application object.

**Definition**

*Application* As Application (Read-only)

**Syntax**

var.Application

*var* is the name of the StoredUniverses variable that you declare.

**Count Property**

Returns the number of stored universes.

**Definition**

*Count* As Long (Read-only)

**Syntax**

var.Count

*var* is the name of the StoredUniverses variable that you declare.

**Item Property**

Returns a StoredUniverse type object based on its identifier.

**Definition**

Property *Item*(Index As Variant) As StoredUniverse (Read-only)

**Syntax**

var.Item(ind)

*var* is the name of the StoredUniverses variable that you declare.
*ind* is a Variant that contains the index of the item.

**Parent Property**

Returns the creator of the object.

**Definition**

*Parent* As Object (Read-only)

**Syntax**

var.Parent

*var* is the name of the StoredUniverses variable that you declare.
Table Class

Provides access to a DESIGNER table.

Syntax

Dim var As Table

var is the name of the Table variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Table variable that you declare.

Columns Property

Returns a Columns type object that contains the collection of columns in the table.

Definition

Columns As Columns (Read-only)

Syntax

var.Columns

var is the name of the Table variable that you declare.

CreateAlias Method

Creates an alias table object.

Definition

Function CreateAlias(Name As String) As Table

Syntax

var.CreateAlias()

var is the name of the Table variable that you declare.

CreateClass Method

Creates a table class object.

Definition

Function CreateClass(Name As String) As Class

Syntax

var.CreateClass()

var is the name of the Table variable that you declare.
Delete Method

Definition
Sub Delete()

Syntax
var.Delete

var is the name of the Table variable that you declare.

IncompatibleObjects Property

Definition
IncompatibleObjects As Objects

Syntax
var.IncompatibleObjects

var is the name of the Variable variable that you declare.

IncompatiblePredefConditions Property

Definition
IncompatiblePredefConditions As PredefinedConditions.

Syntax
var.IncompatiblePredefConditions

var is the name of the Variable variable that you declare.

IsAlias Property

Definition
IsAlias As Boolean (Read-only)

Syntax
var.IsAlias

var is the name of the Variable variable that you declare.

Name Property

Definition
Name As String (Read/Write)

Syntax
var.Name

var is the name of the Table variable that you declare.
**OriginalTable Property**

Returns a table object as the original table.

**Definition**

OriginalTable As Table (Read-only)

**Syntax**

`var.OriginalTable`

`var` is the name of the Table variable that you declare.

**Parent Property**

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Table variable that you declare.

**Weight Property**

Sets or returns the weight value of the table.

**Definition**

Weight As Long (Read/Write)

**Syntax**

`var.Weight`

`var` is the name of the Table variable that you declare.
Tables Class

Provides access to the list of DESIGNER tables.

Syntax
Dim var As Tables

var is the name of the Tables variable that you declare.

Add Method

Adds a table.

Definition
Function Add(Name As String) As Table

Syntax
var.Add(name)

var is the name of the Tables variable that you declare.
name is the name of the table.

Application Property

Returns the application object.

Definition
Application As Application (Read-only)

Syntax
var.Application

var is the name of the Tables variable that you declare.

Count Property

Returns the number of tables.

Definition
Count As Long (Read-only)

Syntax
var.Count

var is the name of the Tables variable that you declare.

DetectIncompatibility Method

Automatically populates the collections Table.IncompatibleObjects and Table.IncompatiblePredefinedConditions.

Definition
Sub DetectIncompatibility()
Syntax \( var\.DetectIncompatibility \)
\( var \) is the name of the Table variable that you declare.

**Item Property**

Returns a Table type object based on its identifier.

**Definition**
Property \( \text{Item}(\text{Index As Variant}) \text{ As Table (Read-only)} \)

**Syntax**
\( var\.Item(ind) \)
\( var \) is the name of the Tables variable that you declare.
\( ind \) is a Variant that contains the index of the item.

**Parent Property**

Returns the creator of the object.

**Definition**
Parent As Object (Read-only)

**Syntax**
\( var\.Parent \)
\( var \) is the name of the Tables variable that you declare.
Chapter 2 Designer Object Model

TableStrategies Class

Provides access to the list of table strategies.

**Syntax**

```vbscript
Dim var As TableStrategies
```

*var* is the name of the TableStrategies variable that you declare.

**Comments**

The TableStrategies class relates to the Strategies tab of the Universe Parameters dialog box.

**Application Property**

Returns the application object.

**Definition**

```vbscript
Application As Application (Read-only)
```

**Syntax**

```vbscript
var.Application
```

*var* is the name of the TableStrategies variable that you declare.

**Count Property**

Returns the number of table strategies.

**Definition**

```vbscript
Count As Long (Read-only)
```

**Syntax**

```vbscript
var.Count
```

*var* is the name of the TableStrategies variable that you declare.

**Comments**

The count is the number of entries in the Tables list box on the Strategies tab of the Universe Parameters dialog box.

**Item Property**

Returns a TableStrategy object based on its identifier.

**Definition**

```vbscript
Property Item(Index As Variant) As TableStrategy (Read-only)
```

**Syntax**

```vbscript
var.Item(ind)
```

*var* is the name of the TableStrategies variable that you declare.

*ind* is a Variant that contains the index of the item.

**Comments**

Each item is shown in the Tables list box on the Strategies tab of the Universe Parameters dialog box.
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the TableStrategies variable that you declare.
TableStrategy Class

Provides access to a DESIGNER table strategy.

Syntax

Dim var As TableStrategy

var is the name of the TableStrategy variable that you declare.

Comments

The TableStrategy class relates to the Strategies tab of the Universe Parameters dialog box.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the TableStrategy variable that you declare.

Help Property

Accesses the Help property of the table strategy.

Definition

Help As String (Read-only)

Syntax

var.Help

var is the name of the TableStrategy variable that you declare.

Comments

The Help text is displayed below the Tables list box for the selected table browser strategy, on the Strategies tab of the Universe Parameters dialog box.

Name Property

Returns a string with the name of the table strategy.

Definition

Name As String (Read-only)

Syntax

var.Name

var is the name of the TableStrategy variable that you declare.

Comments

The name is displayed in the Tables list box on the Strategies tab of the Universe Parameters dialog box.
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the `TableStrategy` variable that you declare.
Chapter 2 Designer Object Model

Universe Class

Provides access to a DESIGNER universe.

Syntax

```vbnet
Dim var As Universe

var is the name of the Universe variable that you declare.
```

Activate Method

Activates the universe.

Definition

```vbnet
Sub Activate()
```

Syntax

```vbnet
var.Activate

var is the name of the Universe variable that you declare.
```

Application Property

Returns the application object.

Definition

```vbnet
Application As Application (Read-only)
```

Syntax

```vbnet
var.Application

var is the name of the Universe variable that you declare.
```

ArrangeTables Method

Arranges tables of the universe.

Definition

```vbnet
Sub ArrangeTables()
```

Syntax

```vbnet
var.ArrangeTables

var is the name of the Universe variable that you declare.
```

Author Property

Returns a string with the name of the universe author.

Definition

```vbnet
Author As String (Read-only)
```

Syntax

```vbnet
var.Author

var is the name of the Universe variable that you declare.
```
CandidateClasses Property

Returns a CandidateClasses type object that contains the list of candidate classes of the universe.

Definition: CandidateClasses As CandidateClasses (Read-only)

Syntax: var.CandidateClasses

var is the name of the Universe variable that you declare.

CandidateJoins Property

Returns a CandidateJoins type object that contains the list of candidate joins of the universe.

Definition: CandidateJoins As CandidateJoins (Read-only)

Syntax: var.CandidateJoins

var is the name of the Universe variable that you declare.

CheckIntegrity Method

Verifies the integrity of the universe.

Definition: Function CheckIntegrity(Filter As DsCheckFilter, ParseLevel As DsCheckParseLevel) As CheckedItems

Syntax: var.CheckIntegrity(filter, parseLevel)

var is the name of the Universe variable that you declare.

filter can take the following values:

<table>
<thead>
<tr>
<th>Values for DsCheckFilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsCheckStructure (= 1)</td>
</tr>
<tr>
<td>dsCheckObject (= 2)</td>
</tr>
<tr>
<td>dsCheckJoin (= 4)</td>
</tr>
<tr>
<td>dsCheckCondition (= 8)</td>
</tr>
</tbody>
</table>
parseLevel is defines the level of parsing. DsCheckParseLevel is an enumerated type. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsCheckParseLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>DsCheckQuickParsing (= 0)</td>
</tr>
<tr>
<td>DsCheckThoroughParsing (= 1)</td>
</tr>
</tbody>
</table>

### Classes Property

Returns a Classes type object that contains the list of classes of the universe.

**Definition**

Classes As Classes (Read-only)

**Syntax**

`var.Classes`

`var` is the name of the Universe variable that you declare.

### Close Method

Closes the universe.

**Definition**

Sub **Close**()

**Syntax**

`var.Close`

`var` is the name of the Universe variable that you declare.

**Comments**

This is equivalent to the Close operation of the File menu.

**See Also**

Universes.Import, Universes.Export, Universes.ExportEx

### Comments Property

Sets or returns a string with the universe comments.

**Definition**

Comments As String (Read/Write)

**Syntax**

`var.Comments`

`var` is the name of the Universe variable that you declare.

**Comments**

This is the Comments text box of the Summary tab of the Universe Parameters dialog box.

### Connection Property

Sets or returns a string with the connection name of the universe.
**Universe Class**

**Definition**

**Connection** As String (Read/Write)

**Syntax**

`var.Connection`

`var` is the name of the Universe variable that you declare.

**Comments**

This is the Connection list box of the Definition tab of the Universe Properties dialog box.

**Contexts Property**

Returns a Contexts type object that contains the list of contexts of the universe.

**Definition**

**Contexts** As Contexts (Read-only)

**Syntax**

`var.Contexts`

`var` is the name of the Universe variable that you declare.

**ControlOption Property**

Returns a ControlOption type object.

**Definition**

**ControlOption** As ControlOption (Read-only)

**Syntax**

`var.ControlOption`

`var` is the name of the Universe variable that you declare.

**CreationDate Property**

Returns a string with the creation date of the universe.

**Definition**

**CreationDate** As String (Read-only)

**Syntax**

`var.CreationDate`

`var` is the name of the Universe variable that you declare.

**Comments**

This is the Created date value of the Summary tab of the Universe Parameters dialog box.

**CurrentJoinStrategy Property**

Sets or returns a string with the current join strategy.

**Definition**

**CurrentJoinStrategy** As String (Read/Write)

**Syntax**

`var.CurrentJoinStrategy`
**CurrentObjectStrategy Property**

Sets or returns a string with the current object strategy.

**Definition**  
`CurrentObjectStrategy` As String (Read/Write)

**Syntax**  
`var.CurrentObjectStrategy`  
`var` is the name of the Universe variable that you declare.

**Comments**  
This is the current selection in the Objects list box of the Strategies tab of the Universe Parameters dialog box.

**CurrentOwner Property**

Sets or returns a string with the current owner.

**Definition**  
`CurrentOwner` As String (Read/Write)

**Syntax**  
`var.CurrentOwner`  
`var` is the name of the Universe variable that you declare.

**CurrentQualifier Property**

Sets or returns a string with the current qualifier.

**Definition**  
`CurrentQualifier` As String (Read/Write)

**Syntax**  
`var.CurrentQualifier`  
`var` is the name of the Universe variable that you declare.

**CurrentTableStrategy Property**

Sets or returns a string with the current table strategy.

**Definition**  
`CurrentTableStrategy` As String (Read/Write)

**Syntax**  
`var.CurrentTableStrategy`  
`var` is the name of the Universe variable that you declare.

**Comments**  
This is the current selection in the Tables list box of the Strategies tab of the Universe Parameters dialog box.
CustomHierarchies Property

Returns a CustomHierarchies type object that contains the list of custom hierarchies of the universe.

Definition

CustomHierarchies As CustomHierarchies (Read-only)

Syntax

var.CustomHierarchies

var is the name of the Universe variable that you declare.

DBTables Property

Returns a DBTables type object that contains the list of DB tables of the universe.

Definition

DBTables As DBTables (Read-only)

Syntax

var.DBTables

var is the name of the Universe variable that you declare.

DefaultHierarchies Property

Returns a DefaultHierarchies type object that contains the list of default hierarchies of the universe.

Definition

DefaultHierarchies As DefaultHierarchies (Read-only)

Syntax

var.DefaultHierarchies

var is the name of the Universe variable that you declare.

Description Property

Sets or returns a string with the description of the universe.

Definition

Description As String (Read/Write)

Syntax

var.Description

var is the name of the Universe variable that you declare.

Comments

This is the Description text box of the Definition tab of the Universe Properties dialog box.
Chapter 2 Designer Object Model

FullName Property

Sets or returns a string with the full name of the universe.

Definition: `FullName` As String (Read-only)

Syntax: `var.FullName`

`var` is the name of the Universe variable that you declare.

Comments: This is the full pathname of the .unv Universe file name.

Joins Property

Returns a Joins type object that contains the list of joins of the universe.

Definition: `Joins` As Joins (Read-only)

Syntax: `var.Joins`

`var` is the name of the Universe variable that you declare.

JoinStrategies Property

Returns a JoinStrategies type object that contains the list of join strategies of the universe.

Definition: `JoinStrategies` As JoinStrategies (Read-only)

Syntax: `var.JoinStrategies`

`var` is the name of the Universe variable that you declare.

Comments: This is the Joins list box of the Strategies tab of the Universe Parameters dialog box. The current selection is the CurrentJoinStrategy property.

LinkedUniverses Property

Returns a LinkedUniverses type object that contains the list of linked universes.

Definition: `LinkedUniverses` As LinkedUniverses (Read-only)

Syntax: `var.LinkedUniverses`

`var` is the name of the Universe variable that you declare.
LongName Property

Sets or returns a string with the long name of the universe.

**Definition**

`LongName` As String (Read/Write)

**Syntax**

```
var.LongName
```

*var* is the name of the Universe variable that you declare.

**Comments**

The long name of the universe is defined in the Name text box of the Definition tab of the Universe Properties dialog box.

ModificationDate Property

Returns a string with the modification date of the universe.

**Definition**

`ModificationDate` As String (Read-only)

**Syntax**

```
var.ModificationDate
```

*var* is the name of the Universe variable that you declare.

**Comments**

This is the Modified date value of the Summary tab of the Universe Parameters dialog box.

Modifier Property

Returns a string with the modifier name of the universe.

**Definition**

`Modifier` As String (Read-only)

**Syntax**

```
var.Modifier
```

*var* is the name of the Universe variable that you declare.

Name Property

Sets or returns a string with the name of the universe.

**Definition**

`Name` As String (Read-only)

**Syntax**

```
var.Name
```

*var* is the name of the Universe variable that you declare.

**Comments**

This is the .unv Universe file name.
ObjectStrategies Property

Returns an ObjectStrategies type object that contains the list of object strategies of the universe.

Definition

ObjectStrategies As ObjectStrategies (Read-only)

Syntax

var. ObjectStrategies

var is the name of the Universe variable that you declare.

Comments

This is the Objects list box of the Strategies tab of the Universe Parameters dialog box. The current selection is the CurrentObjectStrategy property.

Owners Property

Returns an Owners type object that contains the list of owners of the universe.

Definition

Owners As Owners (Read-only)

Syntax

var. Qualifiers

var is the name of the Universe variable that you declare.

OwnerSupported Property

Indicates whether the universe is owner supported (True) or not (False).

Definition

OwnerSupported As Boolean (Read-only)

Syntax

var. OwnerSupported

var is the name of the Universe variable that you declare.

Parent Property

Returns the creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var. Parent

var is the name of the Universe variable that you declare.
Path Property

Returns a string with the path name of the universe.

**Definition**
Path As String (Read-only)

**Syntax**
var.Path

*var* is the name of the Universe variable that you declare.

PrintDialog Method

Displays the Print dialog box to control how the data is printed, and then prints if you click OK.

**Definition**
Sub PrintDialog()

**Syntax**
var.PrintDialog

*var* is the name of the Universe variable that you declare.

PrintOut Method

Prints the data in the universe using the printer name if specified or current print settings.

**Definition**
Sub PrintOut([PrinterName As Variant])

**Syntax**
var.PrintOut([printer])

*var* is the name of the Universe variable that you declare.

*printer* is the name of the printer. This parameter is optional.

**Comments**
This is equivalent to the Print operation of the File menu, plus optionally selecting a printer from the Printer section of the Print dialog box.

Qualifiers Property

Returns a Qualifiers type object that contains the list of qualifiers of the universe.

**Definition**
Qualifiers As Qualifiers (Read-only)

**Syntax**
var.Qualifiers

*var* is the name of the Universe variable that you declare.
QualifierSupported Property

Indicates whether the universe is qualifier supported (True) or not (False).

**Definition**

QualifierSupported As Boolean (Read-only)

**Syntax**

```
var.QualifierSupported
```

*var* is the name of the Universe variable that you declare.

RefreshStructure Method

Updates the contents of the Structure window based on modifications made to the database.

**Definition**

Sub RefreshStructure()

**Syntax**

```
var.RefreshStructure
```

*var* is the name of the Universe variable that you declare.

RevisionNumber Property

Returns the revision number of the universe.

**Definition**

RevisionNumber As Long (Read-only)

**Syntax**

```
var.RevisionNumber
```

*var* is the name of the Universe variable that you declare.

**Comments**

This is the Revision value of the Summary tab of the Universe Parameters dialog box.

Save Method

Saves the data in the universe.

**Definition**

Sub Save()

**Syntax**

```
var.Save
```

*var* is the name of the Universe variable that you declare.

**Comments**

This is equivalent to the Save operation of the File menu.
SaveAs Method

Saves the data in the universe under a different file name.

**Definition**

Sub **SaveAs**(Name As Variant)

**Syntax**

\[ var.SaveAs([name]) \]

\( var \) is the name of the Universe variable that you declare.

\( name \) is the name of the file where the data is to be saved. If it is not specified, a dialog box appears.

**Comments**

This is equivalent to the Save As operation of the File menu. The type of file is always .unv and is not selectable in the SaveAs sub.

Saved Property

Indicates whether the universe has been saved since the last change.

**Definition**

**Saved** As Boolean (Read-only)

**Syntax**

\[ var.Saved \]

\( var \) is the name of the Universe variable that you declare.

SQLOption Property

Returns an SQLOption type object.

**Definition**

**SQLOption** As SQLOption (Read-only)

**Syntax**

\[ var.SQLOption \]

\( var \) is the name of the Universe variable that you declare.

Tables Property

Returns a Tables type object that contains the list of tables of the universe.

**Definition**

**Tables** As Tables (Read-only)

**Syntax**

\[ var.Tables \]

\( var \) is the name of the Universe variable that you declare.
TableStrategies Property

Returns a TableStrategies type object that contains the list of external table strategies that have been defined for the universe.

**Definition**

```
TableStrategies As TableStrategies (Read-only)
```

**Syntax**

```
var.TableStrategies
```

*var* is the name of the Universe variable that you declare.

**Comments**

This is the Tables list box of the Strategies tab of the Universe Parameters dialog box. The current selection is the CurrentTableStrategy property.

UseCustomHierarchies Property

Sets or indicates whether to use custom hierarchies (True) or not (False). If not, default hierarchies are used.

**Definition**

```
UseCustomHierarchies As Boolean (Read-only)
```

**Syntax**

```
var.UseCustomHierarchies
```

*var* is the name of the Universe variable that you declare.

Windows Property

Returns a Windows type object that contains the list of open windows of the universe.

**Definition**

```
Windows As Windows(Read-only)
```

**Syntax**

```
var.Windows
```

*var* is the name of the Universe variable that you declare.
UniverseDomain Class

Provides access to a DESIGNER universe domain.

Syntax

Dim var As UniverseDomain

var is the name of the UniverseDomain variable that you declare.

Application Property

Returns the application object.

Definition
Application As Application (Read-only)

Syntax
var.Application

var is the name of the UniverseDomain variable that you declare.

Name Property

Returns a string with the name of the universe domain.

Definition
Name As String (Read-only)

Syntax
var.Name

var is the name of the UniverseDomain variable that you declare.

Parent Property

Returns the creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the UniverseDomain variable that you declare.

StoredUniverses Property

Returns a StoredUniverses type object that contains the list all the stored universes in DESIGNER.

Definition
StoredUniverses As StoredUniverses (Read-only)

Syntax
var.StoredUniverses

var is the name of the UniverseDomain variable that you declare.
Users Property

Returns the list of users.

**Definition**

Users As Users (Read-only)

**Syntax**

```
var.Users
```

*var* is the name of the UniverseDomain variable that you declare.
UniverseDomains Class

Provides access to the list of universe domains.

Syntax

Dim var As UniverseDomains
var is the name of the UniverseDomains variable that you declare.

Application Property

Returns the application object.

Definition
  Application As Application (Read-only)

Syntax
  var.Application
  var is the name of the UniverseDomains variable that you declare.

Count Property

Returns the number of universe domains.

Definition
  Count As Long (Read-only)

Syntax
  var.Count
  var is the name of the UniverseDomains variable that you declare.

Item Property

Returns a UniverseDomain type object based on its identifier.

Definition
  Property Item(Index As Variant) As UniverseDomain (Read-only)

Syntax
  var.Item(ind)
  var is the name of the UniverseDomains variable that you declare.
  ind is a Variant that contains the index of the item.

Parent Property

Returns the creator of the object.

Definition
  Parent As Object (Read-only)

Syntax
  var.Parent
  var is the name of the UniverseDomains variable that you declare.
Universes Class

Provides access to the list of universes.

**Syntax**

```vba
Dim var As Universes
```

*var* is the name of the Universes variable that you declare.

**Comments**

To create a new Universes object, you must first dimension a variable, using the `Dim` statement.

To refer to a method or property of the newly created object, use the syntax:

```vba
var.property or var.method.
```

**Add Method**

Adds a universe.

**Definition**

```vba
Function Add() As Universe
```

**Syntax**

```vba
var.Add
```

*var* is the name of the Universes variable that you declare.

**Application Property**

Returns the application object.

**Definition**

```vba
Application As Application (Read-only)
```

**Syntax**

```vba
var.Application
```

*var* is the name of the Universes variable that you declare.

**Count Property**

Returns the number of universes in the Universes variable.

**Definition**

```vba
Count As Long (Read-only)
```

**Syntax**

```vba
var.Count
```

*var* is the name of the Universes variable that you declare.
Export Method

Exports a universe.

**Definition**

Sub **Export**(DomainName As String, GroupName As String, UniverseName As String, [Lock As Variant])

**Syntax**

`var.Export(domain, group, universe, [lock])`

- `var` is the name of the Universes variable that you declare.
- `domain` is a string that contains the name of the universe domain.
- `group` is a string that corresponds to the User class.
- `universe` is a string that contains the name of the universe. You can use either the full or short name (Universe.FullName or StoredUniverse.Name) of the universe, with or without the file extension. You can also use the short name and provide the path explicitly.
- `lock` determines if the universe is locked.

**Comments**

Make sure you close the universe before exporting it.

**See Also**

Universes.Import, Universe.Close, Universes.ExportEx
Chapter 2 Designer Object Model

ExportEx Method

Exports a universe to multiple groups.

**Definition**

Sub ExportEx(DomainName As String, GroupName As Array of String, UniverseName As String, [Lock As Variant])

**Syntax**

`var.ExportEx(domain, grouplist, universe, [lock])`

- `domain` is a string that contains the name of the universe domain.
- `grouplist` is an array of string that corresponds to the list of groups to which the universe is exported.
- `universe` is a string that contains the name of the universe. You can use either the full or short name (Universe.FullName or StoredUniverse.Name) of the universe, with or without the file extension. You can also use the short name and provide the path explicitly.
- `lock` determines if the universe is locked.

Lock is a Variant whose value is interpreted as either TRUE or FALSE. It is considered to be FALSE when set to 0 and TRUE otherwise.

**Comments**

Make sure you close the universe before exporting it.

**See Also**

Universes.Import, Universe.Close, Universes.Export

Import Method

Imports a universe.

**Definition**

Sub Import(UniverseDomainName As String, UniverseName As String, [Lock])

**Syntax**

`var.Import(domain, universe, [lock])`

- `domain` is a string that contains the name of the universe domain.
- `universe` is a string that contains the short name of the universe. You must use the short name (StoredUniverse.Name) of the universe, with or without the file extension.
- `lock` determines if the universe is locked.

Lock is a Variant whose value is interpreted as either TRUE or FALSE. It is considered to be FALSE when set to 0 and TRUE otherwise.

**See Also**

Universe.Close, Universes.Export, Universes.ExportEx
Item Property

Returns a Universe type object based on its identifier.

**Definition**

Property `Item(Index As Variant) As Universe (Read-only)`

**Syntax**

```plaintext
var.Item(ind)
```

*var* is the name of the Universes variable that you declare.

*ind* is a Variant that contains the index of the item.

Open Method

Opens a universe.

**Definition**

Function `Open([UniverseName As String]) As Universe`

**Syntax**

```plaintext
var.Open([universe])
```

*var* is the name of the Universes variable that you declare.

*universe* is the name of the universe.

Parent Property

Returns the creator of the object.

**Definition**

`Parent As Object (Read-only)`

**Syntax**

```plaintext
var.Parent
```

*var* is the name of the Universes variable that you declare.
User Class

Provides access to a user object.

Syntax

```vba
Dim var As User
```

*var* is the name of the User variable that you declare.

Application Property

Returns the application object.

Definition

```vba
Application As Application (Read-only)
```

Syntax

```vba
var.Application
```

*var* is the name of the User variable that you declare.

Name Property

Returns a string containing the user name.

Definition

```vba
Name As String (Read-only)
```

Syntax

```vba
var.Name
```

*var* is the name of the User variable that you declare.

Parent Property

Returns the creator of the object.

Definition

```vba
Parent As Object (Read-only)
```

Syntax

```vba
var.Parent
```

*var* is the name of the User variable that you declare.
Users Class

Provides access to a list of User objects.

Syntax

Dim var As Users

var is the name of the Users variable that you declare.

Application Property

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Users variable that you declare.

Count Property

Returns the number of users.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the Users variable that you declare.

Item Property

Returns a User object based on its item number or its identifier.

Definition

Item(Index As Variant) As User (Read-only)

Syntax

var.Item(ind)

var is the name of the Users variable that you declare.

ind is a Variant that contains either the index of the list of users or the user identifier.
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var`.Parent

`var` is the name of the Users variable that you declare.
**Variable Class**

Provides access to a DESIGNER variable.

**Syntax**

```vbnet
Dim var As Variable
```

*var* is the name of the Variable variable that you declare.

**Example**

This displays the name of all the DESIGNER system variables and their value.

```vbnet
Dim vars As Designer.Variables
Dim I As Integer
Set vars = dsgnr.Application.Variables
For I = 1 To vars.Count
    MsgBox vars.Item(I).Name & " " & vars.Item(I).Value
Next
```

**Application Property**

Returns the application object.

**Definition**

**Application** As Application (Read-only)

**Syntax**

```vbnet
var.Application
```

*var* is the name of the Variable variable that you declare.

**Delete Method**

Deletes the associated Variable type object. You can delete a variable only if it was created by a script.

**Definition**

Sub Delete()

**Syntax**

```vbnet
var.Delete
```

*var* is the name of the Variable variable that you declare.

**Example**

This example deletes a variable named Sample after adding it via the script.

```vbnet
Dim var as Variable
set var = Application.Variables.Add("Sample")
var.value = "Sample Value"
MsgBox var.Name & " " & var.Value
var.Delete
```
InterpretAs Property

Indicates or changes the type of a Variable.

**Definition**

InterpretAs As DsVariableType (Read/Write)

**Syntax**

`var.InterpretAs`

`var` is the name of the Variable variable that you declare.

DsVariableType is an enumerated type that specifies the type of a variable. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsVariableType</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsStringVariable (=0)</td>
</tr>
<tr>
<td>dsNumericVariable (=1)</td>
</tr>
<tr>
<td>dsDateVariable (=2)</td>
</tr>
</tbody>
</table>

MultiValued Property

Indicates whether the variable is multi-valued (True) or not (False).

**Definition**

MultiValued As Boolean (Read-only)

**Syntax**

`var.MultiValued`

`var` is the name of the Variable variable that you declare.

Name Property

Returns a string with the name of a variable.

**Definition**

Name As String (Read-only)

**Syntax**

`var.Name`

`var` is the name of the Variable variable that you declare.
**Variable Class**

**Example**

This example displays the name of all the variables in DESIGNER.

```vba
Dim var as Variable
Dim I as integer
for I = 1 to Application.Variables.Count
    set var = Application.Variables.Item(I)
    MsgBox var.Name & " " & var.Value
next
```

**Parent Property**

Returns the creator of the object.

**Definition**

`Parent` As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Variable variable that you declare.

**Value Property**

Returns or changes the value of a variable.

**Definition**

`Value` As String (Read/Write)

**Syntax**

`var.Value`

`var` is the name of the Variable variable that you declare.

**Example**

This example displays the name of all the variables in DESIGNER and their values.

```vba
Dim var as String
Dim I as integer
for I = 1 to Application.Variables.Count
    set var = Application.Variables.Item(I)
    MsgBox var.Name & " " & var.Value
next
```
Variables Class

Provides access to the list of DESIGNER variables.

Syntax

```vba
Dim var As Variables
```

`var` is the name of the Variables variable that you declare.

Example

This example displays the name of all the DESIGNER system variables and their value.

```vba
Dim vars as Variables
Dim I as integer
set vars = Application.Variables
for I = 1 to vars.Count
    MsgBox vars.Item(I).Name & " " & vars.Item(I).Value
next
```

Add Method

Adds a variable.

Definition

Function `Add(Name As String) As Variable`

Syntax

```vba
var.Add(name)
```

`var` is the name of the Variables variable that you declare.

`name` is the variable name.

Application Property

Returns the application object.

Definition

`Application As Application (Read-only)`

Syntax

```vba
var.Application
```

`var` is the name of the Variables variable that you declare.
Count Property

Returns the number of variables in the Variables variable.

**Definition**

Count As Long (Read-only)

**Syntax**

`var.Count`

*var* is the name of the Variables variable that you declare.

**Example**

This example displays the number of defined variables in DESIGNER.

```vba
Dim vars as Variables
set vars = Application.Variables
MsgBox vars.Count
```

Item Property

Returns a Variable type object based on its item number or its name.

**Definition**

Property Item(Index As Variant) As Variable

**Syntax**

`var.Item(ind)`

*var* is the name of the Variables variable that you declare.

*ind* is a Variant that contains the index of the list of variables or a string that contains the name of the variable.

**Comments**

The item number starts at 1.

**Example**

This example displays all the variables of DESIGNER.

```vba
Dim var as Variable
Dim I as integer
for I = 1 to Application.Variables.Count
    set var = Application.Variables.Item(I)
    MsgBox var.Name & " " & var.Value
next
```

Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

*var* is the name of the Variables variable that you declare.
Window Class

Provides access to a DESIGNER window.

**Syntax**

```vbnet
Dim var As Window
```

*var* is the name of the Window variable that you declare.

**Example**

This example changes the width of the main window.

```vbnet
Dim wnd As Designer.Window
dsgnr.Visible = True
Set wnd = dsgnr.Window
wnd.Width = 75
```

Activate Method

Sets the current window as the active window.

**Definition**

Sub Activate()

**Syntax**

```vbnet
var.Activate
```

*var* is the name of the Window variable that you declare.

Application Property

Returns the application object.

**Definition**

```
Application As Application (Read-only)
```

**Syntax**

```vbnet
var.Application
```

*var* is the name of the Window variable that you declare.

Caption Property

Returns or changes the title of the active window.

**Definition**

```
Caption As String (Read/Write)
```

**Syntax**

```vbnet
var.Caption
```

*var* is the name of the Window variable that you declare.

**Comments**

If the caption is omitted, this property returns a string that contains the title of the window.
Close Method

Closes the window.

**Definition**

Sub Close()

**Syntax**

`var.Close`

`var` is the name of the Window variable that you declare.

**Comments**

You should not use the window after you implement this method.

**Example**

This example closes the main window and exits DESIGNER.

```vbnet
Dim wnd as Window
set wnd = Application.Window
wnd.Close
```

Height Property

Returns or changes the height of the window.

**Definition**

`Height As Long (Read/Write)`

**Syntax**

`var.Height`

`var` is the name of the Window variable that you declare.

**Example**

This example changes and displays the height of the main window.

```vbnet
Dim wnd as Window
set wnd = Application.Window
wnd.Height = 100
MsgBox "the new window height is " & wnd.Height
```

Left Property

Returns or changes the horizontal position in pixels.

**Definition**

`Left As Long (Read/Write)`

**Syntax**

`var.Left`

`var` is the name of the Window variable that you declare.
Example

This example changes and displays the horizontal position of the main window.

```vbnet
Dim wnd as Window
set wnd = Application.Window
wnd.Left = 75
MsgBox "the new window position is " & wnd.Left
```

Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Window variable that you declare.

State Property

Returns or changes the sizing of the window.

**Definition**

State As DsWindowState (Read/Write)

**Syntax**

`var.State`

`var` is the name of the Window variable that you declare.

**Comments**

DsWindowState is an enumerated type. It can take the following values:

<table>
<thead>
<tr>
<th>Values for DsWindowState</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsNormal (=1)</td>
</tr>
<tr>
<td>dsMinimized (=2)</td>
</tr>
<tr>
<td>dsMaximized (=3)</td>
</tr>
</tbody>
</table>
Top Property

Returns or changes the vertical position in pixels.

**Definition**

*Top* As Long (Read/Write)

**Syntax**

```
var.Top
```

*var* is the name of the Window variable that you declare.

**Example**

This example changes and displays the vertical position of the main window.

```
Dim wnd as Window
set wnd = Application.Window
wnd.Top = 75
MsgBox "the new window position is " & wnd_Top
```

Width Property

Returns or changes the width of the active window in pixels.

**Definition**

*Width* As Long (Read/Write)

**Syntax**

```
var.Width
```

*var* is the name of the Window variable that you declare.

**Example**

This example changes the width of the main window.

```
Dim wnd as Window
set wnd = Application.Window
wnd.ScreenWidth = 800
wnd.Width = 75
```
Windows Class

Provides access to a collection of Designer windows.

Syntax

Dim var As Windows

*var* is the name of the Windows variable that you declare.

Application Property

Returns the application object.

Definition

*Application* As Application (Read-only)

Syntax

*var*.Application

*var* is the name of the Windows variable that you declare.

Count Property

Returns the number of windows in the collection.

Definition

*Count* As Long (Read-only)

Syntax

*var*.Count

*var* is the name of the Windows variable that you declare.

Item Property

Returns a Window type object based on its index.

Definition

Property *Item*(Index As Variant) As Window

Syntax

*var*.Item(index)

*var* is the name of the Windows variable that you declare.

*index* specifies the index of the list of windows.

Parent Property

Returns the creator of the object.

Definition

*Parent* As Object

Syntax

*var*.Parent

*var* is the name of the Windows variable that you declare.
Class List for Report Viewer Component

RptCategories Class 430
 Count Property .......... 430
 Item Method .......... 430
 Parent Property ...... 430

RptCategory Class 431
 Name Property .......... 431
 Parent Property ...... 431

RptDocument Class 432
 Close Method .......... 432
 Name Property .......... 432
 Open Method .......... 432
 Parent Property ...... 432
 PrintOut Method ........ 433
 Refresh Method .......... 433
 Reports Method .......... 433
 Type Property .......... 434

RptDocuments Class 435
 Count Property .......... 435
 Item Method .......... 435
 Parent Property ...... 435

RptDocumentView Class 436
 ActivateDocument Method .... 436
 ActiveReport Method ........ 436
 ActiveSection Method ........ 436
 Begin Method .......... 437
 DownloadReport Method .... 437
 End Method .......... 437
 Next Method .......... 437
 Parent Property ...... 437
 Previous Method .......... 437
 SynchronizeTreeView Method ...

RptDomains Class 440
 Count Property .......... 440
 Item Property .......... 440
 Parent Property ...... 440

RptDomain Class 439
 Name Property .......... 439
 Parent Property ...... 439

RptReport Class 441
 Activate Method .......... 441
 Name Property .......... 441
 Number Property .......... 441
 Parent Property ...... 441
 PrintOut Method .......... 442
 Sections Property .......... 442

RptReports Class 443
 Count Property .......... 443
 Item Method .......... 443
 Parent Property ...... 443

RptSection Class 444
 Activate Method .......... 444
 Level Property .......... 444
 Name Property .......... 444
 Parent Property ...... 445

RptSections Class 446
 Count Property .......... 446
 Item Method .......... 446
 Parent Property ...... 446

RptViewer Class 447
 BrowserComplete Property ...... 447
 BrowserVisible Property ...... 447
 Categories Method .......... 447
 Connect Method .......... 448
 Disconnect Method .......... 448
 Documents Method .......... 448
 DocumentView Method ...... 449
 Domains Method .......... 449
 EnablePopupMenu Property ...... 449
 EnableViewSectionsTree Property .......... 450
 EnableProperty .......... 450
 RefreshBrowser Method .......... 450
 ServerURL Property ...... 450
 ServerVersion Method ...... 451
 ToolBarVisible Property ...... 451
 Version Method .......... 452
Chapter 3

Report Viewer
Component Object Model

In this chapter

- Details of the classes and class members in the Report Viewer Component Object Model.
RptCategories Class

Provides access to ReportViewer categories. Categories are used to filter the documents in your Personal and Corporate document pages.

**Syntax**

```vbnet
Dim var As RptCategories
var is the name of the RptCategories variable that you declare.
```

**Count Property**

Returns the number of categories.

**Definition**

`Count As Long (Read-only)`

**Syntax**

```vbnet
var.Count
var is the name of the RptCategories variable that you declare.
```

**Item Method**

Returns a category based on its index.

**Definition**

`Function Item(Index As Variant) As RptCategory`

**Syntax**

```vbnet
var.Item(ind)
var is the name of the RptCategories variable that you declare.
ind is a Variant that contains either the category index or a string identifier.
```

**Comments**

The lower bound of the index is 1. The upper bound of the index is obtained from the Count property.

**Parent Property**

Returns the creator of the object.

**Definition**

`Parent As Object (Read-only)`

**Syntax**

```vbnet
var.Parent
var is the name of the RptCategories variable that you declare.
```
**RptCategory Class**

Provides access to a ReportViewer category object. A category is used to filter the documents in your Personal and Corporate document pages.

**Syntax**

```
Dim var As RptCategory
```

*var* is the name of the RptCategory variable that you declare.

**Name Property**

Returns the name of the category.

**Definition**

```
Name As String (Read-only)
```

**Syntax**

```
var.Name
```

*var* is the name of the RptCategory variable that you declare.

**Parent Property**

Returns the creator of the object.

**Definition**

```
Parent As Object (Read-only)
```

**Syntax**

```
var.Parent
```

*var* is the name of the RptCategory variable that you declare.
RptDocument Class

Provides access to a REPORTVIEWER document object. You use this class to open, close, print and obtains report sheets from a document.

Syntax

```
Dim var As RptDocument
var is the name of the RptDocument variable that you declare.
```

Close Method

Closes the document.

Definition

```
Sub Close()
```

Syntax

```
var.Close
var is the name of the RptDocument variable that you declare.
```

Name Property

Returns the name of the document.

Definition

```
Name As String (Read-only)
```

Syntax

```
var.Name
var is the name of the RptDocument variable that you declare.
```

Open Method

Opens and displays the document.

Definition

```
Sub Open()
```

Syntax

```
var.Open
var is the name of the RptDocument variable that you declare.
```

Parent Property

Returns the creator of the object.

Definition

```
Parent As Object (Read-only)
```

Syntax

```
var.Parent
var is the name of the RptDocument variable that you declare.
```
PrintOut Method

Print the entire document.

**Definition**
Sub **PrintOut**(bSilent As Boolean)

**Syntax**
```
var.PrintOut(bSilent)
```

*bSilent* is a Boolean which determines whether the Print dialog box is displayed or not. If the value is True it is displayed; if it is False it is not displayed.

Refresh Method

Refresh the document content on the server.

**Definition**
Sub **Refresh**()

**Syntax**
```
var.Refresh
```

*var* is the name of the RptDocument variable that you declare.

Reports Method

Returns the list of Report sheets for the document.

**Definition**
Function **Reports**() As RptReports

**Syntax**
```
var.Reports()
```

*var* is the name of the RptDocument variable that you declare.
Type Property

Returns the type of the document.

**Definition**

Type As RptRepoType (Read-only)

**Syntax**

`var.Type`

`var` is the name of the RptDocument variable that you declare.

**Comments**

RptRepoType is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Values for RptRepoType</th>
</tr>
</thead>
<tbody>
<tr>
<td>rptRepoTypeCorporate (=0)</td>
</tr>
<tr>
<td>rptRepoTypeInbox (=1)</td>
</tr>
<tr>
<td>rptRepoTypePersonal (=2)</td>
</tr>
</tbody>
</table>
RptDocuments Class

Provides access to the list of ReportViewer documents.

Syntax

```vbnet
Dim var As RptDocuments
```

*var* is the name of the RptDocuments variable that you declare.

Count Property

Returns the number of documents.

**Definition**

```
Count As Long (Read-only)
```

**Syntax**

```
var.Count
```

*var* is the name of the RptDocuments variable that you declare.

Item Method

Returns a document based on its index.

**Definition**

```
Function Item(Index As Variant) As RptDocument
```

**Syntax**

```
var.Item(index)
```

*var* is the name of the RptDocuments variable that you declare.

*index* is a Variant that contains either the document index or a string identifier.

**Comments**

The lower bound of the index is 1. The upper bound of the index is obtained from the Count property.

Parent Property

Returns the creator of the object.

**Definition**

```
Parent As Object (Read-only)
```

**Syntax**

```
var.Parent
```

*var* is the name of the RptDocuments variable that you declare.
RptDocumentView Class

Provides access to the document view. You use this class to change the way a document is displayed and to move backwards and forwards through a document.

Syntax
Dim var As RptDocumentView

var is the name of the RptDocumentView variable that you declare.

ActiveDocument Method

Returns the active document.

Definition
Function ActiveDocument() As RptDocument

Syntax
var.ActiveDocument()

var is the name of the RptDocumentView variable that you declare.

ActiveReport Method

Returns the active report.

Definition
Function ActiveReport() As RptReport

Syntax
var.ActiveReport()

var is the name of the RptDocumentView variable that you declare.

ActiveSection Method

Returns the active section.

Definition
Function ActiveSection() As RptSection

Syntax
var.ActiveSection()

var is the name of the RptDocumentView variable that you declare.

Begin Method

Go to the first page in the current report sheet.

Definition
Sub Begin()

Syntax
var.Begin

var is the name of the RptDocumentView variable that you declare.
**DownloadReport Method**

Download all pages in the current report.

**Definition**

Sub *DownloadReport()*

**Syntax**

`var.DownloadReport()`

`var` is the name of the `RptDocumentView` variable that you declare.

---

**End Method**

Go to the last page in the current report.

**Definition**

Sub *End()*

**Syntax**

`var.End`

`var` is the name of the `RptDocumentView` variable that you declare.

---

**Next Method**

Go to the next page in the current report.

**Definition**

Sub *Next()*

**Syntax**

`var.Next`

`var` is the name of the `RptDocumentView` variable that you declare.

---

**Parent Property**

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the `RptDocumentView` variable that you declare.

---

**Previous Method**

Go to the previous page in the current report.

**Definition**

Sub *Previous()*

**Syntax**

`var.Previous`

`var` is the name of the `RptDocumentView` variable that you declare.
SynchronizeTreeView Method

Updates the report map tree according to the document page displayed.

**Definition**
Sub SynchronizeTreeView()

**Syntax**

```vbnet
var. SynchronizeTreeView
```

*var* is the name of the RptDocumentView variable that you declare.

---

Zoom Property

Get/Set the zoom factor (10% to 400%).

**Definition**
Zoom As Single (Read/Write)

**Syntax**

```vbnet
var. Zoom
```

*var* is the name of the RptDocumentView variable that you declare.

---

ZoomIn Method

Zoom in on the active report.

**Definition**
Sub ZoomIn()

**Syntax**

```vbnet
var. ZoomIn
```

*var* is the name of the RptDocumentView variable that you declare.

---

ZoomOut Method

Zoom out from the active report.

**Definition**
Sub ZoomOut()

**Syntax**

```vbnet
var. ZoomOut
```

*var* is the name of the RptDocumentView variable that you declare.
RptDomain Class

 Provides access to a REPORTVIEWER domain object.

 Syntax
 Dim var As RptDomain
 var is the name of the RptDomain variable that you declare.

 Comments
 To create a new RptDomain object, first dimension a variable, using the Dim statement.
 To refer to a method or property of the newly created object, use the syntax: var.property or var.method.

 Name Property

 Returns the name of the domain.

 Definition
 Name As String (Read-only)

 Syntax
 var.Name
 var is the name of the RptDomain variable that you declare.

 Parent Property

 Returns the creator of the object.

 Definition
 Parent As Object (Read-only)

 Syntax
 var.Parent
 var is the name of the RptDomain variable that you declare.
RptDomains Class

Provides access to REPORTVIEWER domains.

Syntax

Dim var As RptDomains

var is the name of the RptDomains variable that you declare.

Count Property

Returns the number of domains.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the RptDomains variable that you declare.

Item Property

Returns a domain based on its index.

Definition

Function Item(Index As Variant) As RptDomain

Syntax

var.Item(index)

var is the name of the RptDomains variable that you declare.

index is a Variant that contains either the domain index or a string identifier.

Comments

The lower bound of the index is 1. The upper bound of the index is obtained from the Count property.

Parent Property

Returns the creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the RptDomains variable that you declare.
**RptReport Class**

Provides access to a REPORTVIEWER report. You can use this class to retrieve sections from a report sheet.

**Syntax**

```
Dim var As RptReport
```

*var* is the name of the RptReport variable that you declare.

**Activate Method**

Go to the first page in the report.

**Definition**

```
Sub Activate()
```

**Syntax**

```
var.Activate()
```

*var* is the name of the RptReport variable that you declare.

**Name Property**

Returns the name of the report.

**Definition**

```
Name As String (Read-only)
```

**Syntax**

```
var.Name
```

*var* is the name of the RptReport variable that you declare.

**Number Property**

Returns the order of the report sheet in the document.

**Definition**

```
Number As Integer (Read-only)
```

**Syntax**

```
var.Number
```

*var* is the name of the RptReport variable that you declare.

**Parent Property**

Returns the creator of the object.

**Definition**

```
Parent As Object (Read-only)
```

**Syntax**

```
var.Parent
```

*var* is the name of the RptReport variable that you declare.
**PrintOut Method**

Print the report.

**Definition**

Sub **PrintOut**(bSilent As Boolean)

**Syntax**

\[ var.PrintOut(bSilent) \]

*var* is the name of the RptReport variable that you declare.

*bSilent* is a Boolean which determines whether the Print dialog box is displayed or not. If the value is True it is displayed; if it is False it is not displayed.

---

**Sections Property**

Returns the list of sections for the report sheet.

**Definition**

**Sections** As RptSections (Read-only)

**Syntax**

\[ var.Sections \]

*var* is the name of the RptReport variable that you declare.
RptReports Class

Provides access to a list of reports.

Syntax

Dim var As RptReports

var is the name of the RptReports variable that you declare.

Count Property

Returns the number of document reports.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the RptReports variable that you declare.

Item Method

Returns a report sheet based on its index.

Definition

Function Item(Index As Variant) As RptReport

Syntax

var.Item(index)

var is the name of the RptReports variable that you declare.

index is a Variant that contains either the report index or a string identifier.

Comments

The lower bound of the index is 1. The upper bound of the index is obtained from the Count property.

Parent Property

Returns the creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the RptReports variable that you declare.
RptSection Class

Provides access to a REPORTVIEWER section object.

**Syntax**

```dim var As RptSection
var is the name of the RptSection variable that you declare.
```

**Activate Method**

Go to the page referenced by the section.

**Definition**

```sub Activate()
```

**Syntax**

```var.Activate
var is the name of the RptSection variable that you declare.
```

**Level Property**

Returns the level of the section.

**Definition**

```Level As Long (Read-only)
```

**Syntax**

```var.Level
var is the name of the RptSection variable that you declare.
```

**Comments**

A section in a report is at level 1. Subsequent sections embedded within this are at levels 2, 3, and so on.

**Name Property**

Returns the name of the section.

**Definition**

```Name As String (Read-only)
```

**Syntax**

```var.Name
var is the name of the RptSection variable that you declare.
```
Parent Property

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the RptSection variable that you declare.
RptSections Class

Provides access to REPORTVIEWER sections.

Syntax

Dim var As RptSections

var is the name of the RptSections variable that you declare.

Count Property

Returns the number of report sheet sections.

Definition

Count As Long (Read-only)

Syntax

var.Count

var is the name of the RptSections variable that you declare.

Item Method

Returns a report sheet section based on its index.

Definition

Function Item(Index As Variant) As RptSection

Syntax

var.Item(index)

var is the name of the RptSections variable that you declare.

index is a Variant that contains either the section index or a string identifier.

Comments

The lower bound of the index is 1. The upper bound of the index is obtained from the Count property.

Parent Property

Returns the creator of the object.

Definition

Parent As Object (Read-only)

Syntax

var.Parent

var is the name of the RptSections variable that you declare.
RptViewer Class

Provides programmed access to the REPORTVIEWER component.

Syntax

Dim var As RptViewer

var is the name of the RptViewer variable that you declare.

BrowserComplete Property

Determines whether the browser display state is complete (True) or partial (False).

Definition

BrowserComplete As Boolean (Read/Write)

Syntax

var.BrowserComplete

var is the name of the RptViewer variable that you declare.

Comments

When this is set to True, only open documents are displayed. When set to false, all available documents are listed on the left of the window. You can only set this value in Design Mode.

BrowserVisible Property

Determines whether the browser is visible (True) or hidden (False).

Definition

BrowserVisible As Boolean (Read/Write)

Syntax

var.BrowserVisible

var is the name of the RptViewer variable that you declare.

Categories Method

Returns the list of categories for a repository type.

Definition

Function Categories(repoType As RptRepoType) As RptCategories

Syntax

var.Connect(repoType)

var is the name of the RptViewer variable that you declare.

repoType is the repository type.

Comments

The repository type must be “Corporate”.
Chapter 3 Report Viewer Component Object Model

Connect Method

Connect to the server using the address in the ServerURL property.

**Definition**
Sub **Connect**([UserName], [UserPassword])

**Syntax**
var .Connect([name], [pass])

*var* is the name of the RptViewer variable that you declare.

*name* is your user name—obtained from your system administrator.

*pass* is your user password—obtained from your system administrator.

Disconnect Method

Disconnect from the server.

**Definition**
Sub **Disconnect**()

**Syntax**
var .Disconnect

*var* is the name of the RptViewer variable that you declare.

Documents Method

Returns the list of documents of a given type in one domain. The document categories may be filtered.

**Definition**
Function **Documents**(repoType As RptRepoType, [sDomain As String], [sFilter As String]) As RptDocuments

**Syntax**
var .Documents(repoType, [sDomain], [sFilter])

*var* is the name of the RptViewer variable that you declare.

*repoType* is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Values for RptRepoType</th>
</tr>
</thead>
<tbody>
<tr>
<td>rptRepoTypeCorporate (=0)</td>
</tr>
<tr>
<td>rptRepoTypeInbox (=1)</td>
</tr>
<tr>
<td>rptRepoTypePersonal (=2)</td>
</tr>
</tbody>
</table>

*sDomain* is the domain in which the document is stored.

*sFilter* specifies a document filter.
Comments

This function does not distinguish between the domains for Inbox and Personal documents. If you set repoType to rptRepoTypeInbox or rptRepoTypePersonal (1 or 2), you must not enter a domain name (sDomain).

DocumentView Method

Returns the document view object.

Definition

Function DocumentView() As RptDocumentView

Syntax

var.DocumentView

var is the name of the RptViewer variable that you declare.

See Also

RptDocumentView Class

Domains Method

Returns the list of domains for a repository type.

Definition

Function Domains(repoType As RptRepoType) As RptDomains

Syntax

var.Domains(repoType)

var is the name of the RptViewer variable that you declare.

repoType is an enumerated type which may take the following values:

<table>
<thead>
<tr>
<th>Values for RptRepoType</th>
</tr>
</thead>
<tbody>
<tr>
<td>rptRepoTypeCorporate   (=0)</td>
</tr>
<tr>
<td>rptRepoTypeInbox       (=1)</td>
</tr>
<tr>
<td>rptRepoTypePersonal    (=2)</td>
</tr>
</tbody>
</table>

EnablePopupMenu Property

Determines whether the pop-up menu is displayed when you right-click on the report window of the Report Viewer. If the value is True it is displayed; if it is False it is not displayed.

Definition

EnablePopupMenu As Boolean (Read/Write)

Syntax

var.EnablePopupMenu

var is the name of the RptViewer variable that you declare.
Chapter 3 Report Viewer Component Object Model

EnableViewSectionsTree Property

Determines whether the View Section Tree command is available on the pop-up menu and whether the View Sections Tree button is activated in toolbar. If the value is True they are activated; if it is False they are not activated.

Definition
EnableViewSectionsTree As Boolean (Read/Write)

Syntax
var.EnableViewSectionsTree

var is the name of the RptViewer variable that you declare.

Comments
Changing the value of this property does not impact the current visibility of the section tree pane. If the section tree pane was displayed it will remain displayed.

Parent Property

Returns the creator of the object.

Definition
Parent As Object (Read-only)

Syntax
var.Parent

var is the name of the RptViewer variable that you declare.

RefreshBrowser Method

Refresh the browser contents using all known information.

Definition
Sub RefreshBrowser()

Syntax
var.RefreshBrowser

var is the name of the RptViewer variable that you declare.

ServerURL Property

Sets or retrieves the server URL address.

Definition
ServerURL As String (Read/Write)

Syntax
var.ServerURL

var is the name of the RptViewer variable that you declare.
## ServerVersion Method

Retrieves the version number and service pack number of a WEBINTELLIGENCE application.

**Definition**

Sub ServerVersion(Major As Integer, Minor As Integer, Maintenance As Integer)

**Syntax**

`var.ServerVersion(Major, Minor, Maintenance)`

`var` is the name of the RptViewer variable that you declare.

*Major* is a pointer to the major part of the version number, for example the “2” in “2.5”.

*Minor* is a pointer to the minor part of the version number, for example the “5” in “2.5”.

*Maintenance* is a pointer to the service pack index, for example “0” for the release version of a product.

**Comments**

Use this method to verify compatibility between versions of BUSINESSOBJECTS and WEBINTELLIGENCE when using the Report Viewer Component.

**See Also**


## ToolBarVisible Property

Determines whether the toolbar is visible (True) or hidden (False).

**Definition**

ToolBarVisible As Boolean (Read/Write)

**Syntax**

`var.ToolBarVisible`

`var` is the name of the RptViewer variable that you declare.
Version Method

Retrieves the version number and service pack number of a BUSINESSOBJECTS application.

**Definition**

Sub **Version**(Major As Integer, Minor As Integer, Maintenance As Integer)

**Syntax**

`var.Version(Major, Minor, Maintenance)`

`var` is the name of the RptViewer variable that you declare.

**Comments**

Use this method to verify compatibility between versions of BUSINESSOBJECTS and WEBINTELLIGENCE when using Report Viewer Component.

**See Also**

Appendix A  Obsolete Properties and Methods

In this appendix

❑ Obsolete for BusinessObjects 5.1.8 456
❑ Obsolete since BusinessObjects 5.0 458
❑ Command Class 460
❑ Commands Class 464
❑ DocumentProvider Class 472
❑ Toolbar Class 473
❑ Toolbars Class 475
❑ Obsolete since BusinessObjects 4.1 477
Appendix A Obsolete Properties and Methods

Obsolete for BusinessObjects 5.1.8

Below are the descriptions of BUSINESSOBJECTS SDK methods that are obsolete for BUSINESSOBJECTS 5.1.8.

ReceiveScript Method from the Application Class

**Obsolete**: Do not use.

Receives a script from the repository.

**Definition**

Sub ReceiveScript([Name As String])

**Syntax**

```vba
var.ReceiveScript([scriptName])
```

*var* is the name of the Application variable that you declare.

*scriptName* is a string that contains the filename of the script without its extension, or the filename with its path and extension. This parameter is optional.

If a script name is not specified, the RetrieveScript dialog box appears that lists the available scripts.

**Comments**

You can modify the domain of the repository with the Application.ExchangeDomain property.

You can modify the import mode with the Application.ExchangeMode property.

This method is maintained for backward compatibility. It may be used in conjunction with Document.ConvertFromReportScript to update ReportScript scripts to VBA macros. A typical workflow might be:

1. Import Script from Repository.
2. Convert Script to VBA.
3. Send Script to Repository.
ConvertFromReportScript Method from the Document class

**Obsolete**: Do not use.

Converting a ReportScript script (.spt file) to a BusinessObjects Version 5 source program.

**Definition**
Sub `ConvertFromReportScript([FileName As String])`

**Syntax**

```vbnet
var.ConvertFromReportScript([fileName])
```

- `var` is the name of the Document variable that you declare.
- `fileName` is a string that contains the name of the file to be converted.
# Appendix A Obsolete Properties and Methods

## Obsolete since BusinessObjects 5.0

Below are the descriptions of ReportScript 4.1 classes, methods and properties that are obsolete since BUSINESSOBJECTS 5.0.

### Commands Property

**Obsolete:** Do not use.

Returns a Commands object that contains the available command identifiers.

**Definition**

Commands As Commands (Read-only)

**Syntax**

`var.Commands`

`var` is the name of the Application variable that you declare.

**Comments**

For a list of command identifiers, refer to ItemID Property.

**Example**

This example displays the menu title that corresponds to the ANALYSISSLICEDICE command.

```vba
Dim coms as Commands
Set coms = Application.Commands
MsgBox coms.Item(boAnalysisSliceDice).Caption
```

### ExchangeRepository Property

**Obsolete:** Do not use. See Application.ExchangeDomain.

**Definition**

ExchangeRepository As String (Read/Write)

**Syntax**

`var.ExchangeRepository`

`var` is the name of the Application variable that you declare.

**Description**

Returns or changes the name of the repository used for the next send/receive procedure.

**Comments**

If the name is omitted, this property returns the current repository name.
Example

This example sets the new repository and sends the active document.

```vbnet
Sub main
    Application.ExchangeRepository = "Financial"
    ActiveDocument.Send("Company")
    MsgBox "The new repository is " & Application.ExchangeRepository
End Sub
```

OpenScript Method

**Obsolete**: Do not use.

**Definition**

Function OpenScript([Name As String]) As Script

**Syntax**

```vbnet
var.OpenScript([scriptName])
```

*var* is the name of the Application variable that you declare.

*scriptName* is a string that contains the filename of the script with its path and extension. If the script is located in the Scripts folder, then *scriptName* should contain just the filename without its extension. This parameter is optional.

**Description**

Opens a BUSINESSOBJECTS script.

**Example**

This example opens the script "Evalkit" and executes it.

```vbnet
Dim scr as Script
Set scr = Application.OpenScript("Evalkit")
scr.Execute
```

Descriptions Property

**Obsolete**: Do not use.

**Definition**

Descriptions As String (Read-only)

**Syntax**

```vbnet
var.Descriptions
```

*var* is the name of the Class variable that you declare.
Command Class

**Obsolete:** Do not use.

**Syntax**

Dim *var* As Command

*var* is the name of the Command variable that you declare.

**Example**

This example executes the command that activates the Slice and Dice analysis. The Command class is used to declare the variable that executes the command.

```vba
Dim com as Command
set com = Application.Commands.Item(ANALYSISSLICEDICE)
com.Execute
```

Application Property

**Obsolete:** Do not use.

**Definition**

*Application* As Application (Read-only)

**Syntax**

*var*.Application

*var* is the name of the Command variable that you declare.

Caption Property

**Obsolete:** Do not use.

**Definition**

*Caption* As String (Read-only)

Returns a string containing the menu that corresponds to the command.
Syntax  
`var.Caption`

`var` is the name of the Command variable that you declare.

'Caption Property Example'

This example displays the menu title that corresponds to the ANALYSISSLICEDICE command.

```vba
dim com as Command
set com = Application.Commands.Item(BOANALYSISSLICEDICE)
MsgBox com.Caption
```

**Execute Method**

**Obsolete**: Do not use.

Executes the command.

**Definition**

Sub `Execute()`

**Syntax**

`var.Execute`

`var` is the name of the Command variable that you declare.

**Example**

This example executes the ANALYSISSLICEDICE command.

```vba
dim com as Command
set com = Application.Commands.Item(boAnalysisSliceDice)
com.Execute
```

**Help Property**

**Obsolete**: Do not use.

Returns the tooltip of the associated command.

**Definition**

`Help As String (Read-only)`

**Syntax**

`var.Help`

`var` is the name of the Command variable that you declare.
Appendix A Obsolete Properties and Methods

Example

This example displays the tooltip associated with the ANALYSISLICEDICE command.

```vbs
'Example
Dim com As Command
Set com = Application.Commands.Item(boAnalysisSliceDice)
MsgBox com.Help
```

ID Property

Obsolete: Do not use.

Returns the ID of the associated command.

Definition

ID As Long (Read-only)

Syntax

`var.ID`

`var` is the name of the Command variable that you declare.

Parent Property

Obsolete: Do not use.

Returns the creator of the object.

Definition

Parent As Object (Read-only)

Syntax

`var.Parent`

`var` is the name of the Command variable that you declare.

State Property

Obsolete: Do not use.

Indicates or changes the status of a command. A command is either enabled, disabled, or removed.

Definition

State As BoCommandState (Read/Write)
### Syntax

```plaintext
var.State
```

*var* is the name of the Command variable that you declare.

**Enum BoCommandState** is an enumerator object that specifies the state of the command. It may be: `boEnabled (=1)`, `boDisabled (=2)`, or `boRemoved (=3)`.

### Comments

Command.state is Read-only for the BusinessObjects Reader and Driller.

### Example

This example changes the status of the Refresh command then refreshes the active document.

```plaintext
dim com as Command
set com = Application.Commands.Item(boDataRefresh)
com.State = Enabled
ActiveDocument.Refresh
com.State = Disabled
```
Appendix A Obsolete Properties and Methods

Commands Class

Obsolete: Do not use.

Provides access to the list of BusinessObjects commands.

Syntax

Dim var As Commands

var is the name of the Commands variable that you declare.

Example

This example displays the name of the command that invokes the Slice and Dice Panel. The Commands class is used to declare the variable.

Dim coms as Commands
Dim msgtext as String
set coms = Application.Commands
msgtext = coms.ItemID(boAnalysisSliceDice).Caption
msgbox msgtext

Application Property

Obsolete: Do not use.

Returns the application object.

Definition

Application As Application (Read-only)

Syntax

var.Application

var is the name of the Commands variable that you declare.

Count Property

Obsolete: Do not use.

Returns the number of commands in the Commands variable.

Definition

Count As Long (Read-only)
Syntax

```
var.Count
```

*var* is the name of the Commands variable that you declare.

**Example**

This example displays the number of available commands in a script.

```
dim coms as Commands
set coms = Application.Commands
MsgBox coms.Count
```

**Item Property**

**Obsolete:** Do not use.

Returns a Command type object based on its identifier.

**Definition**

Property **Item**(Index As Long) As Command (Read-only)

**Syntax**

```
var.Item(ind)
```

*var* is the name of the Commands variable that you declare.

*ind* is an integer that contains the index of the item.

**Comments**

The item number starts at 1.

**ItemID Property**

**Obsolete:** Do not use.

Returns a Commands type object that contains the available command identifiers.

**Definition**

Property **ItemID**(ID As BoCommandID) As Command (Read-only)

**Syntax**

```
var.ItemID(ID)
```

*var* is the name of the Commands variable that you declare.

*ID* is the command identifier.

Enum BoCommandID is an enumerator object that specifies the command identifier. The available command identifiers are:

- boAnalysisBusinessMiner
- boFmtApplyTemplate
- boTBFilePrint
## Appendix A Obsolete Properties and Methods

<table>
<thead>
<tr>
<th>Property</th>
<th>Function</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>boAnalysisCollapse</td>
<td>boFmtBlockStd</td>
<td>boToolConsole</td>
</tr>
<tr>
<td>boAnalysisDrill</td>
<td>boFmtBlockTransformStd</td>
<td>boToolDimension</td>
</tr>
<tr>
<td>boAnalysisDrillDown</td>
<td>boFmtBreak</td>
<td>boToolListOfValues</td>
</tr>
<tr>
<td>boAnalysisDrillUp</td>
<td>boFmtCell</td>
<td>boToolLoginAs</td>
</tr>
<tr>
<td>boAnalysisExpand</td>
<td>boFmtFilter</td>
<td>boToolPassword</td>
</tr>
<tr>
<td>boAnalysisSliceDice</td>
<td>boFmtPageBackground</td>
<td>boToolPreferences</td>
</tr>
<tr>
<td>boAnalysisTakeSnapshot</td>
<td>boFmtReport</td>
<td>boToolUniverses</td>
</tr>
<tr>
<td>boConnectedIt</td>
<td>boFmtSection</td>
<td>boUDOUse</td>
</tr>
<tr>
<td>boDataEdit</td>
<td>boFmtSort</td>
<td>boViewFullScreen</td>
</tr>
<tr>
<td>boDataProviders</td>
<td>boFmtVariables</td>
<td>boViewGridLines</td>
</tr>
<tr>
<td>boDataRefresh</td>
<td>boFormulaCancel</td>
<td>boViewOutline</td>
</tr>
<tr>
<td>boEditDeleteReport</td>
<td>boFormulaDefine</td>
<td>boViewPageDelimiter</td>
</tr>
<tr>
<td>boEditDuplicateReport</td>
<td>boFormulaEdit</td>
<td>boViewPageMode</td>
</tr>
<tr>
<td>boEditLink</td>
<td>boFormulaEditor</td>
<td>boViewSectionDelimiter</td>
</tr>
<tr>
<td>boFileClose</td>
<td>boFormulaValidate</td>
<td>boViewStatusBar</td>
</tr>
<tr>
<td>boFileEditStructure</td>
<td>boFreeHandEdit</td>
<td>boViewStructure</td>
</tr>
<tr>
<td>boFileNew</td>
<td>boFreeHandUse</td>
<td>boViewToolbars</td>
</tr>
<tr>
<td>boFileNewWithPower</td>
<td>boInsReport</td>
<td>boViewZoom</td>
</tr>
<tr>
<td>boFileOpen</td>
<td>boLocalDataEdit</td>
<td>boWinArrangeAll</td>
</tr>
<tr>
<td>boFilePageSetup</td>
<td>boLocalDataUse</td>
<td>boWinNew</td>
</tr>
<tr>
<td>boFilePrint</td>
<td>boLOVEdit</td>
<td></td>
</tr>
<tr>
<td>boFilePrintPreview</td>
<td>boLOVRefresh</td>
<td></td>
</tr>
<tr>
<td>boFileProperties</td>
<td>boLOVUse</td>
<td></td>
</tr>
<tr>
<td>boFileReceiveBatch</td>
<td>boQueryEdit</td>
<td></td>
</tr>
<tr>
<td>boFileReceiveRepository</td>
<td>boQueryEditSQL</td>
<td></td>
</tr>
<tr>
<td>boFileReceiveUsers</td>
<td>boQueryUse</td>
<td></td>
</tr>
<tr>
<td>boFileSave</td>
<td>boQueryView</td>
<td></td>
</tr>
<tr>
<td>boFileSaveAll</td>
<td>boScrUserScript1</td>
<td></td>
</tr>
</tbody>
</table>
Parent Property

**Obsolete**: Do not use.

Returns the creator of the object.

**Definition**

Parent As Object (Read-only)

**Syntax**

`var.Parent`

`var` is the name of the Commands variable that you declare.

Application Property

**Obsolete**: Do not use.

**Definition**

Application As Application (Read-only)

**Syntax**

`objectvar.Application`

`objectvar` is the name of the Script variable that you declare.

**Description**

Returns the application object.
Appendix A Obsolete Properties and Methods

Execute Method

Obsolete: Do not use.

Definition
Sub Execute()
Syntax
objectvar.Execute
objectvar is the name of the Script variable that you declare.
Description
Runs the script.
Example
This example executes the script “Evalkit”.
Sub main
    dim scr as Script
    set scr = Application.OpenScript("evalkit")
    scr.Execute
End Sub

Name Property

Obsolete: Do not use.

Definition
Name As String (Read-only)
Syntax
objectvar.Name
objectvar is the name of the Script variable that you declare.
Description
Returns a string with the name of the script.
Example
This example displays the name of the current script.
Sub main
    MsgBox ActiveScript.Name
End Sub
Parent Property

**Definition**

Parent As Object (Read-only)

**Syntax**

\texttt{objectvar.Parent}

\texttt{objectvar} is the name of the Script variable that you declare.

**Description**

Returns the creator of the object.

---

Script Class

**Definition**

Obsolete: Do not use.

**Syntax**

Dim \texttt{variableName} As Script

\texttt{variableName} is the name of the Script variable that you declare.

**Description**

Provides access to a BusinessObjects script.

**Comments**

To create a new Script object, you must first dimension a variable using the \texttt{Dim} statement.

To refer to a method or property of the newly created object, use the syntax: \texttt{objectvar.property or objectvar.method}.

**Example**

This example opens the script “Evalkit” then executes it.

```
Sub main
    dim scr as Script
    set scr = Application.OpenScript.Item("evalkit")
End Sub
```
Appendix A Obsolete Properties and Methods

Send Method

**Obsolete:** Do not use.

**Definition**

Sub **Send**(User As String)

**Syntax**

`objectvar.Send([userName])`

`objectvar` is the name of the Script variable that you declare.

`userName` is a string that contains the name of the user to whom you plan to address the script.

**Description**

Sends the script to the repository. If a user name is not specified, the Send dialog box appears to choose to whom it should be sent.

**Comments**

You can modify the name of the repository with the ExchangeRepository property (Application class). You can modify the import mode with the ExchangeMode property (Application class).

**Example**

This example exports the script “Evalkit” to the current repository.

```vbs
Sub main
    dim scr as Script
    set scr = Application.Scripts.Open("evalkit")
    Application.ExchangeMode = RepositoryMode
    scr.Send("Company")
End Sub
```

State

**Obsolete:** Use Visible (BOToolbar Class) instead.

**Syntax**

State

**Description**

Indicates or changes the status of a toolbar. A toolbar is either visible or hidden.
Example

This State Property example displays all the toolbars of the BUSINESSOBJECTS module.

Sub main
    for I = 1 to Application.Toolbars.Count
        Application.Toolbars.Item(I).State = Visible
    next
End Sub

See Also

BOToolbar Class
DocumentProvider Class

Activate Method

**Obsolete**: Do not use.

Sets the current data provider as the default data provider.

**Definition**
Sub **Activate**()

**Syntax**
`var.Activate`

`var` is the name of the DataProvider variable that you declare.
### Toolbar Class

**Obsolete:** Do not use.

Provides access to a toolbar.

**Syntax**

Dim `var` As Toolbar
`var` is the name of the Toolbar variable that you declare.

### Application Property

**Obsolete:** Do not use.

Returns the application object.

**Definition**

`Application` As Application (Read-only)

**Syntax**

`var`.Application
`var` is the name of the Toolbar variable that you declare.

### Name Property

**Obsolete:** Do not use.

**Definition**

`Name` As String (Read-only)

**Syntax**

`var`.Name
`var` is the name of the Toolbar variable that you declare.

### Parent Property

**Obsolete:** Do not use.
Appendix A Obsolete Properties and Methods

**Definition**

**Parent** As Object (Read-only)

**Syntax**

```
var.Parent
```

*var* is the name of the Toolbar variable that you declare.

**Visible Property**

**Definition**

**Visible** As Boolean (Read/Write)

**Syntax**

```
var.Visible
```

*var* is the name of the Toolbar variable that you declare.
Toolbars Class

**Obsolete:** Do not use.

Provides access to a Toolbars collection.

**Syntax**

```vbnet
Dim var As Toolbars
```

`var` is the name of the Toolbars variable that you declare.

**Application Property**

**Obsolete:** Do not use.

Returns the application object.

**Definition**

`Application` As Application (Read-only)

**Syntax**

```vbnet
var.Application
```

`var` is the name of the Toolbars variable that you declare.

**Count Property**

**Obsolete:** Do not use.

**Definition**

`Count` As Long (Read-only)

**Syntax**

```vbnet
var.Count
```

`var` is the name of the Toolbars variable that you declare.

**Item Property**

**Obsolete:** Do not use.
Appendix A Obsolete Properties and Methods

**Definition**
*Item*(Index) As Toolbar (Read-only)

**Syntax**
```plaintext
var.Item(ind)
```

*var* is the name of the Toolbar variable that you declare.

**Parent Property**

**Obsolete**: Do not use.

**Definition**
*Parent* As Object (Read-only)

**Syntax**
```plaintext
var.Parent
```

*var* is the name of the Toolbars variable that you declare.
Obsolete since BusinessObjects 4.1

Below are the descriptions of ReportScript methods and properties that are obsolete since BusinessObjects 4.1.

Exit

Obsolete: Use Quit (BOApplication Class) instead.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Quits the BUSINESSOBJECTS module.</td>
</tr>
<tr>
<td>Comments</td>
<td>If you insert this method, the rest of the script is ignored.</td>
</tr>
<tr>
<td>Example</td>
<td>This Exit Method example exits the BUSINESSOBJECTS module using the global variable Application.</td>
</tr>
<tr>
<td></td>
<td>Sub main</td>
</tr>
<tr>
<td></td>
<td>Application.Exit</td>
</tr>
<tr>
<td></td>
<td>End Sub</td>
</tr>
<tr>
<td>See Also</td>
<td>BOApplication Class</td>
</tr>
</tbody>
</table>

ID

Obsolete: Do not use. Refer to ItemID (BOCommands Class).

<table>
<thead>
<tr>
<th>Syntax</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Returns the command identifier. The available identifiers are:</td>
</tr>
</tbody>
</table>
### Appendix A Obsolete Properties and Methods

<table>
<thead>
<tr>
<th>Property/Method</th>
<th>Property/Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYSISCOLLAPSE</td>
<td>ANALYSISDRILL</td>
</tr>
<tr>
<td>ANALYSISDRILLDOWN</td>
<td>ANALYSISDRILLUP</td>
</tr>
<tr>
<td>ANALYSISEXPAND</td>
<td>ANALYSISSSLICEDICE</td>
</tr>
<tr>
<td>ANALYSISISTAKESNAPHT</td>
<td>DATACONVERT</td>
</tr>
<tr>
<td>DATACOPY</td>
<td>DATAEDIT</td>
</tr>
<tr>
<td>DATAFREEZE</td>
<td>DATAOPTIMIZECUBE</td>
</tr>
<tr>
<td>DATAPROVIDERS</td>
<td>DATAREFRESH</td>
</tr>
<tr>
<td>EDITDELETEREPORT</td>
<td>EDITDUPLICATEREPORT</td>
</tr>
<tr>
<td>EDITLINK</td>
<td>FILECLOSE</td>
</tr>
<tr>
<td>FILEDEFAULTPRINT</td>
<td>FILEEXIT</td>
</tr>
<tr>
<td>FILENEW</td>
<td>FILENEWWIZARD</td>
</tr>
<tr>
<td>FILEOPEN</td>
<td>FILEPAGESETUP</td>
</tr>
<tr>
<td>FILEPRINT</td>
<td>FILEPRINTPREVIEW</td>
</tr>
<tr>
<td>FILEPROPERTIES</td>
<td>FILERECEIVEBATCH</td>
</tr>
<tr>
<td>FILERECEIVEMAIL</td>
<td>FILERECEIVEREPOTPOSITORY</td>
</tr>
<tr>
<td>FILERECEIVEUSERS</td>
<td>FILERECENT1</td>
</tr>
<tr>
<td>FILESAVE</td>
<td>FILESAVEALL</td>
</tr>
<tr>
<td>FILESAVEAS</td>
<td>FILESENDMAPI</td>
</tr>
<tr>
<td>FILESENDREPOSITORY</td>
<td>FILESENDSERVER</td>
</tr>
<tr>
<td>FILESENDUSERS</td>
<td>FORMATALERTERS</td>
</tr>
<tr>
<td>FORMATAPPLYTEMPLATE</td>
<td>FORMATBLKSTD</td>
</tr>
<tr>
<td>FORMATBLKTRANSFORMSTD</td>
<td>FORMATBREAK</td>
</tr>
<tr>
<td>FORMATCELL</td>
<td>FORMATFILTER</td>
</tr>
<tr>
<td>FORMATPAGEBACKGROUND</td>
<td>FORMATREPORT</td>
</tr>
<tr>
<td>FORMATSECTION</td>
<td>FORMATSORT</td>
</tr>
<tr>
<td>FORMATVARIABLES</td>
<td>FORMULACANCEL</td>
</tr>
<tr>
<td>FORMULAEDIT</td>
<td>FORMULAEDITOR</td>
</tr>
<tr>
<td>FORMULVALIDATE</td>
<td>INSERTREPORT</td>
</tr>
<tr>
<td>LISTOFVALUES</td>
<td>LISTOFVALUESEDIT</td>
</tr>
<tr>
<td>SCRIPTEXECUTE</td>
<td>SCRIPTEDIT</td>
</tr>
<tr>
<td>SCRIPTRECORD</td>
<td>SCRIPTPAUSE</td>
</tr>
<tr>
<td>SCRIPTSTOP</td>
<td>SCRIPTUSERSCRIPT1</td>
</tr>
<tr>
<td>SCRIPTUSERSCRIPT2</td>
<td>SCRIPTUSERSCRIPT3</td>
</tr>
<tr>
<td>SCRIPTUSERSCRIPT4</td>
<td>SCRIPTUSERSCRIPT5</td>
</tr>
<tr>
<td>SCRIPTEXPORT</td>
<td>SCRIPTEXPORTREPOSITORY</td>
</tr>
<tr>
<td>SCRIPTEXPORTUSER</td>
<td>SCRIPTIMPORT</td>
</tr>
<tr>
<td>TOOLCONSOLE</td>
<td>TOOLDIMENSION</td>
</tr>
<tr>
<td>TOOLCONSOLE</td>
<td>TOOLPASSWORD</td>
</tr>
<tr>
<td>TOOLCONSOLE</td>
<td>TOOLSPECIALS</td>
</tr>
<tr>
<td>TOOLCONSOLE</td>
<td>TOOLUNIVERSES</td>
</tr>
</tbody>
</table>

478 BusinessObjects SDK Developer’s Guide
Obsoleted since BusinessObjects 4.1

VIEWDISPLAY VIEWFULLRESULTS
VIEWFULLSCREEN VIEWGRIDLINES
VIEWNORMALMODE VIEWOUTLINE
VIEWPAGEMODE VIEWPAGEDELIMITER
VIEWPARTIALRESULTS VIEWSECTIONDELIMITER
VIEWSTATUSBAR VIEWSTRUCTURE
VIEWTOOLBARS VIEWZOOM
WINARRANGE WINARRANGEALL
WINARRANGEICONS WINCASCADE
WINNEW WINTILEHORizontaly
WINTILEVERTICALY.

Example
This ID Property example checks the identifier of the first command.

Sub main
    dim com as BOCommand
    set com = Application.Commands.Item(1)
    If com.ID = FILEEXIT
        MsgBox "first command is exit"
    End Sub

See Also
BOCommand Class

Item

Obsoleted: Refer to the Item function (BOCommands Class) instead.

Syntax
Item(ind)

ind is an integer that contains either the index of the list of commands or the command identifier.

Description
Returns a BOCommand type object based on its item number or its identifier. The commands include:
## Appendix A Obsolete Properties and Methods

<table>
<thead>
<tr>
<th>analyzer Collapse</th>
<th>analySisDrill</th>
</tr>
</thead>
<tbody>
<tr>
<td>analySisDrill Down</td>
<td>analySisDrillUp</td>
</tr>
<tr>
<td>analySisExpand</td>
<td>analySisSlicEdIce</td>
</tr>
<tr>
<td>analySisTakesSnapShot</td>
<td>DataConvert</td>
</tr>
<tr>
<td>dataCopy</td>
<td>DataEdit</td>
</tr>
<tr>
<td>dataFreeze</td>
<td>DataOptimizeCube</td>
</tr>
<tr>
<td>dataProviders</td>
<td>DataRefresh</td>
</tr>
<tr>
<td>editDeleteReport</td>
<td>editDuplicateReport</td>
</tr>
<tr>
<td>editLink</td>
<td>FileClose</td>
</tr>
<tr>
<td>fileDefaultPrint</td>
<td>FileExit</td>
</tr>
<tr>
<td>fileNew</td>
<td>FileNewWizard</td>
</tr>
<tr>
<td>fileOpen</td>
<td>FilePageSetup</td>
</tr>
<tr>
<td>filePrint</td>
<td>FilePrintPreview</td>
</tr>
<tr>
<td>fileProperties</td>
<td>FileReceiveBatch</td>
</tr>
<tr>
<td>fileReceiveMail</td>
<td>FileReceiveRepository</td>
</tr>
<tr>
<td>fileReceiveUsers</td>
<td>FileRecent1</td>
</tr>
<tr>
<td>fileSave</td>
<td>FileSaveAll</td>
</tr>
<tr>
<td>fileSaveAs</td>
<td>FileSendMail</td>
</tr>
<tr>
<td>fileSendRepository</td>
<td>FileSendServer</td>
</tr>
<tr>
<td>fileSendUsers</td>
<td>FileSendUsers</td>
</tr>
<tr>
<td>formatApplyTemplate</td>
<td>FormatBlkStd</td>
</tr>
<tr>
<td>formatBlkTransformStd</td>
<td>FormatBreak</td>
</tr>
<tr>
<td>formatCell</td>
<td>FormatFilter</td>
</tr>
<tr>
<td>formatPageBackground</td>
<td>FormatReport</td>
</tr>
<tr>
<td>formatSection</td>
<td>FormatSort</td>
</tr>
<tr>
<td>formatVariables</td>
<td>FormulaCancel</td>
</tr>
<tr>
<td>formulaEdit</td>
<td>FormulaEditor</td>
</tr>
<tr>
<td>formulaValidate</td>
<td>InsertReport</td>
</tr>
<tr>
<td>listofValues</td>
<td>ListOfValuesEdit</td>
</tr>
<tr>
<td>scriptExecute</td>
<td>ScriptEdit</td>
</tr>
<tr>
<td>scriptRecord</td>
<td>ScriptPause</td>
</tr>
<tr>
<td>scriptStop</td>
<td>ScriptUserScript1</td>
</tr>
<tr>
<td>scriptUserScript2</td>
<td>ScriptUserScript3</td>
</tr>
<tr>
<td>scriptUserScript4</td>
<td>ScriptUserScript5</td>
</tr>
<tr>
<td>scriptExport</td>
<td>ScriptExportRepository</td>
</tr>
<tr>
<td>scriptExportUser</td>
<td>ScriptExport</td>
</tr>
<tr>
<td>toolConsole</td>
<td>ToolDimension</td>
</tr>
<tr>
<td>toolLogas</td>
<td>ToolPassword</td>
</tr>
<tr>
<td>toolPreferences</td>
<td>ToolUniverses</td>
</tr>
</tbody>
</table>
**Obsolete since BusinessObjects 4.1**

**Comments**
The item number starts at 1.

**Example**
This Item Method example gets the ExitApplication command and displays the tooltip associated with it.

```vba
Sub main
    dim com as BOCommand
    set com = Application.Commands.Item(FILEEXIT)
    MsgBox com.Help
End Sub
```

**See Also**
BOCommands Class, Caption (BOCommand Class), Execute (BOCommand Class), Help (BOCommand Class), ID (BOCommand Class), State (BOCommand Class)

## Open

**Obsolete**: Use OpenScript (BOApplication Class) instead.

### Syntax

```
Open[(scriptName$)]
```

*scriptName*$ is a string that contains the filename of the script without its extension, or the filename with its path and extension.

### Description
Opens the specified script located in the Scripts folder. If a script name is not specified, the Open dialog box appears.
Appendix A Obsolete Properties and Methods

Example

This Open Method example opens the script "OpenBob" and refreshes it.

Sub main
    dim scr as BOscript
    set scr = Application.Scripts.Open("openbob")
    scr.Execute
End Sub

See Also

BOScripts Class

PosX

Obsolete: Use Left (BOWindow Class) instead.

Syntax

PosX

Description

Returns or changes the horizontal position in pixels.

Comments

You can change the screen definition by the SetScreenWidth and SetScreenHeight properties.

Example

This PosX Property example changes and displays the horizontal position of the main window.

Sub main
    dim wnd as BOWindow
    set wnd = Application.Window
    wnd.PosX = 75
    MsgBox "the new window position is " & wnd.PosX
End Sub

See Also

BOWindow Class
**PosY**

**Obsolete:** Use Top (BOWindow Class) instead.

**Syntax**

```
PosY
```

**Description**

Returns or changes the vertical position in pixels.

**Comments**

You can change the screen definition by the SetScreenWidth and SetScreenHeight properties.

**Example**

This PosY Property example changes and displays the vertical position of the main window.

```
Sub main
    dim wnd as BOWindow
    set wnd = Application.Window
    wnd.PosY = 75
    MsgBox "the new window position is " & wnd.PosY
End Sub
```

**See Also**

BOWindow Class

---

**Print**

**Obsolete:** Use PrintOut (BODocument Class) instead.

**Syntax**

```
Print
```

**Description**

Prints the document using the current print settings.

**Example**

This Print Method example prints the active document.

```
Sub main
    ActiveDocument.Print
End Sub
```

**See Also**

BODocument Class, PrintDialog (BODocument Class)
Appendix A Obsolete Properties and Methods

Print

Obsolete: Use PrintOut (BOReport Class) instead.

Syntax
Print

Description
Prints the report using the current print settings.

Example
This Print Method example prints the active report.

Sub main
    ActiveReport.Print
End Sub

See Also
BOReport Class

Receive

Obsolete: Use ReceiveScript (BOApplication Class) instead.

Syntax
Receive[(scriptName$)]

scriptName$ is a string that contains the filename of the script without its extension, or the filename with its path and extension.

Description
Retrieves the script from the repository. If a script name is not specified, the Retrieve dialog box appears that lists the available scripts.

Comments
You can modify the name of the repository with the ExchangeRepository property (BOApplication class).
You can modify the import mode with the ExchangeMode property (BOApplication class).

Example
This Receive Method example imports the script "OpenBob" from the current repository then executes it.

Sub main
    Application.Scripts.Receive("openbob")
End Sub
ScreenHeight

Obsolete: Do not use.

Syntax
ScreenHeight

Description
Returns or changes the screen height in pixels.

Comments
This screen height is used as a reference to position and size all other windows. You cannot use different screen heights for other windows. If you switch to a larger or smaller screen, the position and size of all windows adjust.

Example
This ScreenHeight Property example changes the height of the main window.

```vbnet
Sub main
    dim wnd as BOWindow
    set wnd = Application.Window
    wnd.ScreenHeight = 600
    wnd.Height = 75
End Sub
```

See Also
BOWindow Class, ScreenWidth (BOWindow Class)

ScreenWidth

Obsolete: Do not use.

Syntax
ScreenWidth

Description
Returns the screen width in pixels.
Appendix A Obsolete Properties and Methods

Comments
This screen width is used as a reference to position and size all other windows. You cannot use different screen widths for other windows. If you switch to a larger or smaller screen, the position and size of all windows adjust.

Example
This ScreenWidth Property example changes the width of the main window.

```vbnet
Sub main
    dim wnd as BOWindow
    set wnd = Application.Window
    wnd.ScreenWidth = 800
    wnd.Width = 75
End Sub
```

See Also
BOWindow Class, ScreenHeight (BOWindow Class)

Scripts

Obsolete: Do not use. See OpenScript (BOApplication Class).

Syntax
Scripts

Description
Returns a BOScripts type object that lets you access the scripts of the BUSINESSOBJECTS module.

Example
This Scripts Method example imports the script "OpenBob" from the current repository.

```vbnet
Sub main
    dim scrs as BOScripts
    set scrs = Application.Scripts
    scrs.Receive("openbob")
End Sub
```

See Also
BOApplication Class, Open (BOScripts Class), Receive (BOScripts Class)